UNIVERSIDADE FEDERAL DE MINAS GERAIS
PROGRAMA INTERUNIDADES DE PÓS-GRADUAÇÃO EM BIOINFORMÁTICA

GUSTAVO SIMÕES CARNIVALI

DESENVOLVIMENTO DE UM PIPELINE DE PREDIÇÃO E ANÁLISE DE REDE DE INTERAÇÃO PROTEINA-PROTEINA COM APLICAÇÃO NO PATOSSISTEMA SOJA-FERRUGEM

Belo Horizonte

2024

GUSTAVO SIMÕES CARNIVALI

DESENVOLVIMENTO DE UM PIPELINE DE PREDIÇÃO E ANÁLISE DE REDE DE INTERAÇÃO PROTEINA-PROTEINA COM APLICAÇÃO NO PATOSSISTEMA SOJA-FERRUGEM

Tese apresentada ao Programa Interunidades de Pós-Graduação em Bioinformática da Universidade Federal de Minas Gerais, como requisito à obtenção do título de Doutor em Bioinformática.

Tiago Antonio de Oliveira Mendes
Edson Mario de Andrade Silva

Belo Horizonte

2024

043 Carnivali, Gustavo Simões.
Desenvolvimento de um pipeline para predição e análise de rede de interação proteína-proteína com aplicação no patossistema soja-ferrugem [manuscrito] / Gustavo Simões Carnivali. – 2024.
132 f. : il. ; 29,5 cm.

Orientador: Dr. Tiago Antônio de Oliveira Mendes. Coorientador: Dr. Edson Mario de Andrade Silva.
Tese (doutorado) – Universidade Federal de Minas Gerais, Instituto de Ciências Biológicas. Programa Interunidades de Pós-Graduação em Bioinformática.

1. Bioinformática. 2. Soja. 3. Ferrugem. 4. Phakopsora Pachyrhizi. 5. Interações Hospedeiro-Patógeno. I. Mendes, Tiago Antônio de Oliveira. II. Silva, Edson Mario de Andrade. III. Universidade Federal de Minas Gerais. Instituto de Ciências Biológicas. IV. Título.

CDU: 573:004

Ficha catalográfica elaborada pela bibliotecária Fabiane C M Reis – CRB 6 - 2680

UNIVERSIDADE FEDERAL DE MINAS GERAIS

FOLHA DE APROVAÇÃO

Gustavo Simões Carnivali

"Desenvolvimento de um pipeline para predição e análise de rede de interação proteína-proteína com aplicação no patossistema soja-ferrugem"

Tese aprovada pela banca examinadora constituída pelos Professores:

Prof. Tiago Antonio de Oliveira Mendes - Orientador
Universidade Federal de Viçosa

Prof. Edson Mario de Andrade Silva- Coorientador
University of Florida

Prof. José Miguel Ortega
Universidade Federal de Minas Gerais

Prof. Carlos Cristiano Hasenclever Borges
Universidade Federal de Juiz de Fora

Prof. José Cleydson Ferreira da Silva
University of Florida

Profª Isabel Samila Lima Castro
Universidade Federal de Viçosa

Belo Horizonte, 24 de junho de 2024.

Documento assinado eletronicamente por **Tiago Antônio de Oliveira Mendes**, **Usuário Externo**, em 26/06/2024, às 14:33, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Jose Cleydson Ferreira da Silva**, **Usuário**

Externo, em 26/06/2024, às 15:09, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Edson Mario de Andrade Silva**, **Usuário Externo**, em 26/06/2024, às 15:35, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Isabel Samila Lima Castro**, **Usuário Externo**, em 26/06/2024, às 15:39, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Carlos Cristiano Hasenclever Borges**, **Usuário Externo**, em 27/06/2024, às 11:06, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Jose Miguel Ortega**, **Servidor(a)**, em 17/07/2024, às 10:28, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufmg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **3296996** e o código CRC **0B7CEF59**.

Referência: Processo nº 23072.231218/2024-41 SEI nº 3296996

Sempre a ela

Agradecimentos

A papai e maninho por tudo.

A todos meus familiares e amigos pelo apoio constante.

A toda equipe de professores, estudantes e técnicos da UFMG por me acompanharem em todo processo.

A todos que de alguma forma contribuíram com este projeto.

A UFMG, Fapemig, Fapesp e CAPES pelos apoios.

"Meu objetivo é simples. Entender completamente o Universo, por que é, como é, e simplesmente seu motivo de existir"

(Stephen Hawking)

Resumo

Redes biológicas moleculares são importantes para propiciar interações entre elementos de uma célula ou comunicação entre dois ou mais organismos. Diferentes abordagens computacionais podem ser utilizadas para realizar predição e agrupamento destas redes. Nesta tese, abordagens de predição e análise de redes foram estabelecidas para estudo das interações entre proteínas de dois diferentes organismos, utilizando integração de diferentes programas de bioinformática. Esta abordagem estabelecida foi aplicada para estudo da interação de um fungo patogênico com soja. O Brasil é o maior produtor mundial de soja ($Glycine\ max$), esta cultura é também a principal contribuinte para o PIB brasileiro associado ao agronegócio, no mundo o comércio de soja movimenta cerca de 200 bilhões de dólares por ano. A ferrugem asiática da soja, causada pelo fungo biotrófico $Phakopsora\ Pachyrhizi$, é a doença de maior importância nesta cultura, estima-se que ela já causou um prejuízo de cerca de 150 bilhões de dólares em todo o mundo. Proteínas desenvolvem um papel chave nos processos biológicos dos organismos, incluindo a interface de interação entre patógeno e seus hospedeiros, propiciando o processo como o de infeção. Desta forma, entender a interação entre proteínas do patógeno e proteínas do hospedeiro pode permitir o desenvolvimento de novos métodos e abordagens como o de controle da ferrugem asiática da soja. Neste projeto foi aplicada abordagem de redes de interação proteína-proteína especificamente na infecção da $Glycine\ max$ pelo $Phakopsora\ pachyrhizi$ para identificação de prováveis proteínas efetoras do fungo. Inicialmente, as sequências de proteínas preditas com base no genoma de referência de G. max e P. $pachyrhizi$ foram recuperadas de bancos de dados públicos e alinhados com sequências de proteínas de portais públicos como o STRING. Scripts in house foram produzidos para identificar os potenciais pares de interação entre proteínas, utilizando as interações de alta confiança descritas em diversos portais, seguindo por cálculo de índice de probabilidade de interação e integração de dados de preditores de localização subcelular de proteínas e dados de expressão gênica gerados por metodologia de RNASeq durante um a interação G. max-P. pachyrhizi. A rede também será expandida com dados de outros bancos de dados públicos de interação proteína-proteína. Até o momento, as interações encontradas foram modeladas no formato de redes complexas, que permite aplicação de técnicas para mineração e identificação de proteínas chaves no processo de infecção. Também é objetivo deste projeto a modelagem estrutural de proteínas selecionadas com base na rede para estudo dos mecanismos moleculares associados à interface de interação.

Palavras-chave: Associações proteicas. Redes de interações proteicas. $Glycine\ max$. $Phakopsora\ Pachyrhizi$.

Abstract

Molecular biological networks are important for providing interactions between elements of a cell or communication between two or more organisms. Different computational approaches can be used to predict and cluster these networks. In this thesis, network prediction and analysis approaches were established to study the interactions between proteins from two different organisms, using integration of different bioinformatics programs. This established approach was applied to study the interaction of a pathogenic fungus with soybeans. Brazil is the world's largest producer of soybeans ($Glycine\ max$), this crop is also the main contributor to Brazilian GDP associated with agribusiness, worldwide soybean trade generates around 200 billion dollars per year. Asian soybean rust, caused by the biotrophic fungus Phakopsora Pachyrhizi, is the most important disease in this crop, it is estimated that it has already caused losses of around 150 billion dollars worldwide. Proteins play a key role in the biological processes of organisms, including the interaction interface between pathogens and their hosts, promoting processes such as infection. In this way, understanding the interaction between pathogen proteins and host proteins can allow the development of new methods and approaches such as controlling Asian soybean rust. In this project, a protein-protein interaction network approach was applied specifically to the infection of $Glycine\ max$ by $Phakopsora\ pachyrhizi$ to identify probable effector proteins of the fungus. Initially, protein sequences predicted based on the reference genome of G. max and P. $pachyrhizi$ were retrieved from public databases and aligned with protein sequences from public portals such as STRING. In-house scripts were produced to identify potential interaction pairs between proteins, using high-confidence interactions described in several portals, followed by calculation of interaction probability index and integration of data from protein subcellular localization predictors and expression data genes generated by RNASeq methodology during a G. max-P. interaction. pachyrhizi. The network will also be expanded with data from other public protein-protein interaction databases. To date, the interactions found have been modeled in the format of complex networks, which allows the application of techniques for mining and identifying key proteins in the infection process. The objective of this project is also the structural modeling of selected proteins based on the network to study the molecular mechanisms associated with the interaction interface.

Keywords: *Protein associations. Protein interaction networks. Glycine max. Phakopsora Pachyrhizi..*

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Proteínas são os principais produtos funcionais das células e dos organismos (DIMITROV, 2012; CORDY, 1976; CHATR-ARYAMONTRI et al., 2017). Elas podem se relacionar ente si formando uma rede complexa que atua de modo direto nos mais diversos processos biológicos, celulares e/ou metabólicos (SZKLARCZYK et al., 2019; CHATR-ARYAMONTRI et al., 2017). Dados em larga escala (proteômica, transcriotômica, genômica...) têm sido empregados em análises computacionais no sentido de identificar e caracterizar as funções de proteínas e até mesmo descrever seus níveis de expressão. Porém, informações sobre o estado nativo de uma proteína e sua sequência de aminoácidos não explicam o grau de complexidade existente em um sistema biológico (FLÓREZ et al., 2010). Essa complexidade pode ser modelada com base na teoria de grafos e estudada empregando redes de interação entre duas biomoléculas (proteínas-proteína, RNA-proteína, DNA-proteína, dentre outras). Essas redes podem representar interações físicas e/ou funcionais, e muitas delas podem ser conservadas ao longo dos diversos grupos filogenéticos (SZKLARCZYK et al., 2019; CHANG et al., 2013).

Atualmente muitos métodos experimentais são empregados para identificar redes IPP (interações proteína-proteína), como, por exemplo, o método de duplo híbrido, cromatografia de afinidade e imunoprecipitação (GAVIN et al., 2002; PHIZICKY; FIELDS, 1995). Em contrapartida, esses métodos possuem limitações, por exemplo, eles não são capazes de identificar as interações para todas as proteínas de um organismo e podem identificar interações que nunca acontecem em um sistema vivo, porque as moléculas envolvidas estão localizadas em compartimentos diferentes e os métodos dependerem de processamento de amostras que expõem as biomoléculas a interações não naturais. Desse modo, métodos computacionais, podem ser utilizados com o intuito de identificar IPP usando as sequências dos aminoácidos de todas as proteínas de uma determinada espécie (SKRABANEK et al., 2008), possibilitando a descoberta de potenciais interações mais rapidamente que os métodos experimentais que podem apresentar mais dificuldades e limitações para detectar.

Ferramentas computacionais para predição de sublocalização celular de proteínas, como TMHMM, SignalP, TargertP, podem auxiliar robustez da identificação dos relacionamentos. Outras abordagens computacionais empregadas no estudo das IPP temos aquelas que empregam o perfil filogenético (HUYNEN; BORK, 1998; MARCOTTE et al., 1999, 1999), genoma Neighborhood (HARRINGTON; JENSEN; BORK, 2008), genes fundidos (ENRIGHT et al., 1999; MARCOTTE et al., 1999), co-evolução de sequências (SATO et al., 2005), dentre outros. Métodos como esses têm sido aplicados com sucesso na identificação de alvos para o controle de fitopatógenos, como no fungo da brusone do arroz, Magnaporthe grisea (HE et al., 2008), bem como na planta, no sentido de selecionar genes candidatos para a resistência a estresse biótico e abiótico (LIU et al., 2017).

Devido a estas aplicações apresentadas, e diversas outras, o estudo sobre relacionamentos proteicos ainda é um problema e uma realidade atual (NASIRI et al., 2021; BERAHMAND et al., 2021). Especificamente, este trabalho, utiliza ferramentas como String, Blast e SignalP (SZKLARCZYK et al., 2019; KORF; YANDELL; BEDELL, 2003; EMANUELSSON et al., 2007) para encontrar novos relacionamentos proteicos. Em especifico, esses portais (String, IntAct, Biogrid), dentre outros (CHATR-ARYAMONTRI et al., 2017; HERMJAKOB et al., 2004; SZKLARCZYK et al., 2019; KORF; YANDELL; BEDELL, 2003; EMANUELSSON et al., 2007), disponibilizam relacionamentos proteicos em milhares de organismos modelo e não modelo. Interações originadas das mais diversas fontes (interações física, funcional, com base em métodos preditivos, etc).

Em especifico, neste trabalho, é estudada as interações que envolvem as proteínas de soja (Glycine max) e o fungo causador da ferrugem asiática Phakopsora pachyrhizi. A cultura da soja foi definida pelo CONAB (Companhia Nacional de Abastecimento), em 2022, como a de maior exportação do Brasil, com produção de 123.829,5 milhões de toneladas. Por outro lado, segundo a EMPRABA, em 2021, a ferrugem asiática foi considerada a doença mais severa desta cultura no Brasil e em outros países do mundo (HELING et al., 2021). Devido a grande relevância da ferrugem asiática para a cultura da soja e o potencial dos estudos de rede de interação proteína-proteína na identificação de elementos chave desse patossitema, este trabalho, utiliza ferramentas computacionais para predizer uma rede entre planta e patógeno para encontrar potenciais efetores/genes de resistência (HELING et al., 2021; SZKLARCZYK et al., 2019, 2019; KORF; YANDELL; BEDELL, 2003).

2 Objetivos

Objetiva-se, com este trabalho, realizar a predição das interações entre as proteínas da $G. max$ e proteínas do fungo $P. pachyrhizi$ associadas ao processo de infecção por integração da rede predita com dados RNA-Seq e localização subcelular de proteínas. Objetivos específicos:

- Realizar predição da rede de interação entre proteínas preditas no genoma de Glycine max e de Phakopsora pachyrhizi.
- Integração de dados de RNA-Seq obtidos durante a infecção de Glycine max por Phakopsora pachyrhizi na rede de interação proteína-proteína.
- Predição de localização subcelular de proteínas e integração na rede de interação proteína-proteína.
- Modelagem molecular da interação de proteínas selecionadas da rede de interação entre Glycine max e de Phakopsora pachyrhizi.
- Disponibilização e analise dos códigos obtidos ao longo de todo trabalho.
- Disponibilização e analise dos dados, como os grafos obtidos ou informações de proteínas da infecção, obtidas ao longo de todo trabalho.

3 Revisão literária

3.1 *Glycine max*

A soja (*Glycine max Merrill*) é uma leguminosa diploide da subfamília Fabaceae (BRAR; JR, 1993). A Soja é indígena para a Ásia Oriental e China, mas agora é amplamente cultivada em muitas regiões temperadas do mundo Brasil ou os Estados Unidos (EUA) (ARORA et al., 2005; WAQAS et al., 2015), em sua origem, na Ásia, a soja se consistia de plantas rasteiras que se desenvolviam ao longo de rios e lagos (soja selvagem). O processo de domesticação da soja ocorreu no século XI a.C., a partir de cruzamentos naturais feitos por cientistas chineses (APROSOJA, 2014). No entanto o Ocidente ignorou seu cultivo até a segunda década do século XX, quando os Estados Unidos (EUA) iniciaram a exploração comercial primeiramente utilizando-a como forrageira e em seguida como grão (EMPRABA, 2014). Foi em 1919, após o final da Primeira Guerra Mundial, que a soja se tornou um importante item do comércio exterior, sendo que o ano de 1921 é considerado como marco da consolidação da cadeia produtiva da soja mundialmente, uma vez que neste ano foi fundada a American Soybean Association (ASA) (APROSOJA, 2014).

A soja é uma cultura versátil, ela é produzida em 50 milhões de hectares, fornece 35% do óleo vegetal do mundo e é uma grande fonte de proteína para mais de um bilhão de asiáticos (BRAR; JR, 1993). Em 1882, a soja chegou ao Brasil via EUA, com registros de cultivos experimentais na Bahia. No entanto o marco de introdução da soja no Brasil foi em 1901, quando começaram os cultivos na Estação Agropecuária de Campinas e a distribuição de sementes para produtores paulistas. Em 1914 a soja foi oficialmente introduzida no Rio Grande do Sul (APROSOJA, 2014). Em meados dos anos 50, com o estabelecimento do programa oficial de incentivo à triticultura nacional, a cultura da soja foi igualmente incentivada e então sua produção tornou-se economicamente importante para o país. A expansão da soja no Brasil começou nos anos 70, quando a indústria de óleo começou a ser ampliada, sendo responsável pela ampliação da fronteira agrícola principalmente nas regiões Centro-Oeste e Norte nas décadas de 80 e 90 (APROSOJA, 2014; EMPRABA, 2014; SILVA et al., 2022).

Na safra brasileira de soja de 2021, segundo a CONAB, a área de plantio de soja correspondia a 3,4% do território brasileiro, a produção corresponde a 21,3% de todas as produções nacionais avaliadas pela CONAB, com uma produção de 3.552 quilos de soja por habitante gerando uma exportação de soja total do Brasil de 78 milhões de toneladas. O Brasil é o maior produtor mundial de soja. Os principais importadores da soja brasileira são: China, União Europeia, Tailândia, Taiwan (Formosa), Coréia do Sul e Vietnã (CONAB, 2014). Estima que o complexo agroindustrial da soja movimenta aproximadamente mais de US$ 200 bilhões/ano (USDA, 2008).

Segundo a EMBRAPA (Empresa Brasileira de Pesquisa Agropecuária), no Brasil, em

2022/2023, a área de plantio da soja correspondia a 44.062,6 milhões de hectares, o que totalizou uma produção de 154.566,3 de toneladas da soja no Brasil. Após o Brasil, em 2020, os EUA é considerado pela EMBRAPA como o segundo maior produtor de soja, com uma produção de 112,549 milhões de toneladas e área plantada de 33,313 milhões de hectares.

Segundo o SuperBack o período de crescimento da soja pode durar cerca de 160 dias. A figura 3.1 apresenta o ciclo da soja com a presença da infecção pelo fungo de interesse deste trabalho (*Phakopsora pachyrhizi*) apresentado abaixo:

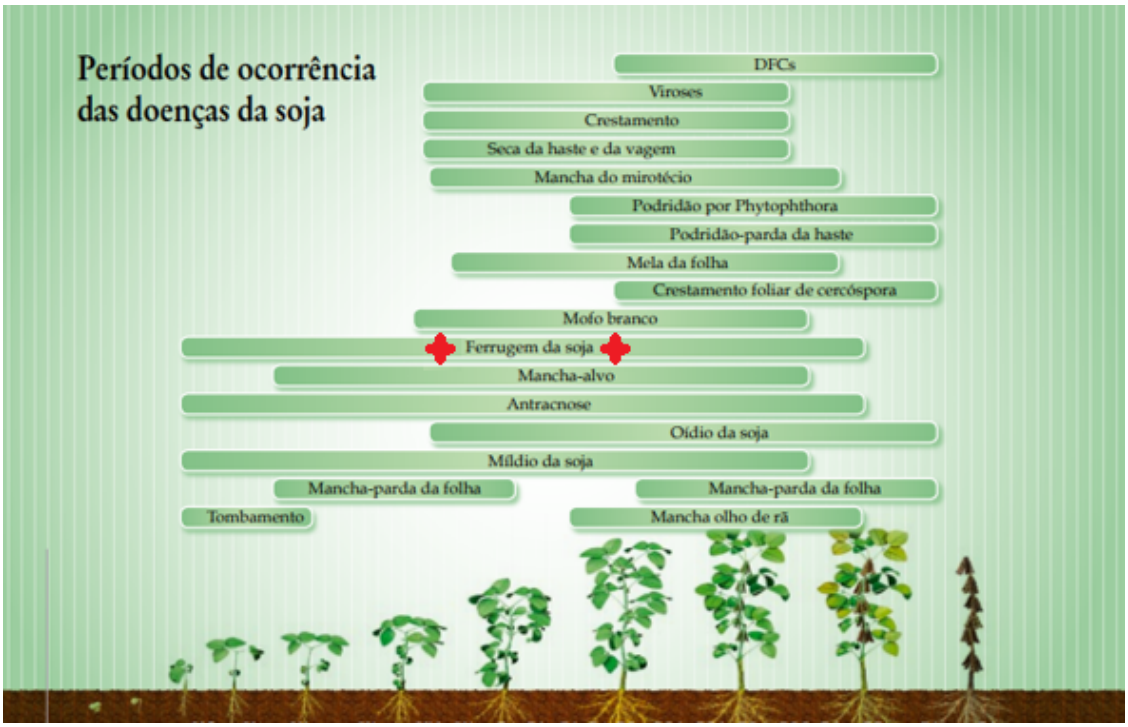


Figura 1 – **Ciclo da soja.** Presença da doença causada pelo fungo *Phakopsora pachyrhizi*. Imagem disponibilizada pela companhia +soja.

3.2 *Phakopsora pachyrhizi*

A ampliação da sojicultura para novas fronteiras agrícolas com diferentes condições climáticas tem aumentado as demandas por tecnologias que deem sustentação ao sistema de produção, como a resistência genética às doenças. Entre os principais fatores que limitam a obtenção de altos rendimentos estão as doenças. Aproximadamente 40 doenças que acometem a cultura da soja causada por fungos, bactérias, nematóides e vírus já foram identificadas no Brasil. Este número continua aumentando com a expansão da soja para novas áreas e como consequência da monocultura. A importância econômica de cada doença varia de ano para ano e de região para região, dependendo das condições climáticas de cada safra. As perdas anuais de produção por doenças são estimadas em cerca de 15% a 20%, entretanto, algumas doenças

podem ocasionar perdas de quase 100% se constituindo deste modo uma ameaça à produtividade e competitividade da soja nacional. (EMBRAPA, 2008; BARROS et al., 2011)

Figura 2 – **Esporos do** $P.\ pachyrhizi$ **em folha de soja.** Imagem disponibilizada pela companhia AgroLink.

A ferrugem asiática da soja, causada por $P.\ pachyrhizi$, possui alto potencial de dano à cultura pois pode causar rápido amarelecimento e queda prematura de folhas, prejudicando a plena formação dos grãos. A infecção do $P.\ pachyrhizi$ em uma folha de soja pode ser vista, como exemplo, na Figura 3.2. Desde o ano de 2001, epidemias da doença têm sido constatadas em algumas regiões do Brasil. Na safra 2001/2002, as lavouras mais atingidas apresentaram queda na produtividade de até 70% (OLIVEIRA; ROSA, 2014), sendo que na região do Planalto do Rio Grande do Sul ocorreram perdas de até 48% (NAVARINI et al., 2007). Na safra de 2002/2003, a ferrugem atingiu as principais áreas produtoras de soja no país e, segundo o estudo (YORINORI; JÚNIOR; LAZZAROTTO, 2004), o custo devido a perdas e aplicações de fungicida foi de pelo menos $US1,126$ bilhão. (SOARES et al., 2004)

O controle da ferrugem da soja compreende diversas medidas conjuntas. Quando a doença já está ocorrendo, o controle químico com fungicidas é, até o momento, o principal método de controle. Ainda não se tem, entre as cultivares recomendadas, materiais com bom nível de resistência. Isto se deve, em parte, à recente ocorrência da doença no país, mas também devido ao fato de o fungo $P.\ pachyrhizi$ possuir diversas raças com genes múltiplos de virulência. (SINCLAIR; HARTMAN, 1995; SOARES et al., 2004)

3.3 Interações proteína-proteína

Desconsiderando raras exceções, os DNAs são moléculas biológicas que sintetizam informações hereditárias passadas de pais para filhos. Essas moléculas biológicas são constituídas por 4 tipos repetidos de nucleotídeos, são eles: adenina (A), guanina (G), citosina (C) e a timina (T). Os diversos emparelhamentos desses nucleotídios geram todos os genes que possuímos. Os DNAs são compostos por fitas duplas compactadas de nucleotídeos. Alguns trechos de DNA podem dar origem à criação de proteínas que vão desempenhar diversos papéis dentro e fora das células. (DEPAMPHILIS et al., 1996; POLLARD et al., 2016)

O início da síntese proteica, ou expressão de um gene, se dá quando um trecho de DNA é separado em duas cadeias por uma enzima celular. Com a molécula de DNA separada, ela se torna guia para a criação de uma molécula de RNA chamada RNA mensageiro (RNAm), no processo de transcrição. O RNAm gerado é complementar ao DNA usado como guia, os nucleotídeos A, G, C e T são trocados respectivamente pelos nucleotídeos U, C, G e A formando a molécula de RNAm. A molécula de RNAm, em conjunto com outras moléculas celulares, produzem as proteínas do corpo no processo de tradução. As proteínas são formadas por moléculas chamadas aminoácidos. Cada conjunto de 3 nucleotídeos do RNAm sintetizam até 20 aminoácidos. Vários outros processos participam da transcrição e tradução, tais processos não serão apresentados pois não constituem o interesse desse estudo. (DEPAMPHILIS et al., 1996; POLLARD et al., 2016)

DNAs, RNAs e proteínas continuamente se relacionam aumentando a complexidade dos sistemas biológicos. Por exemplo, um termo comum na biologia é o de proteínas reguladoras, elas possuem a principal função de participar de processos de expressão genética, ou seja, são proteínas que estimulam ou desestimulam processos de produção de outras proteínas (MORGAN; HARRIS, 1999). Outro termo comum é o de co-expressão gênica, onde se implica um relacionamento gênico quando a expressão de dois ou mais genes se modifica de forma proporcional em organismos diferentes (ZHANG; HORVATH, 2005). Nesse contexto é comum a criação e o estudo de interações biológicas.

Interações biológicas, sejam interações funcionais entre genes ou interações físicas entre proteínas e outras biomoléculas, interações de RNA, DNA, membranas, carboidratos e metabólitos de moléculas pequenas, fornecem a estrutura para entendermos como a célula é estruturada e controlada e servem como uma estrutura para compreender as relações gene-fenótipo e da base mecanicista para algumas funções celulares (YEGER-LOTEM; SHARAN, 2015; MA et al., 2018). A caracterização das interações moleculares e funcionais entre genes, seus produtos e biomoléculas tem sido fundamental na interpretação de associações genéticas relacionadas ao câncer e outras doenças em um conjunto de diferentes contextos. (HOFREE et al., 2013; SAHNI et al., 2015; OUGHTRED et al., 2019)

Associações proteína-proteína provaram ser um conceito útil. O conjunto completo de associações pode ser montado em uma grande rede, que captura o conhecimento atual sobre a

interconectividade de um célula. Além da análise de cada proteína e suas propriedades, pode-se analisar também um conjunto conectado de proteínas. Por este motivo, uma variedade de uso dessas redes de associação funcional surgiu. Por exemplo, (i) As informações de rede proteica podem auxiliar na interpretação dos dados de genômica funcional (SAHNI et al., 2015). (ii) As redes de associação de proteínas também têm se mostrado surpreendentemente úteis para a elucidação de genes de doenças, tanto para mendelianos quanto para doenças complexas (NIK-ZAINAL et al., 2016). Para esta última aplicação, as redes podem ajudar a restringir ou conhecer o espaço de busca, analisando regiões genômicas que abrangem mais de um gene ou proteína, conectados ou não a uma doença. (FRANCESCHINI et al., 2012)

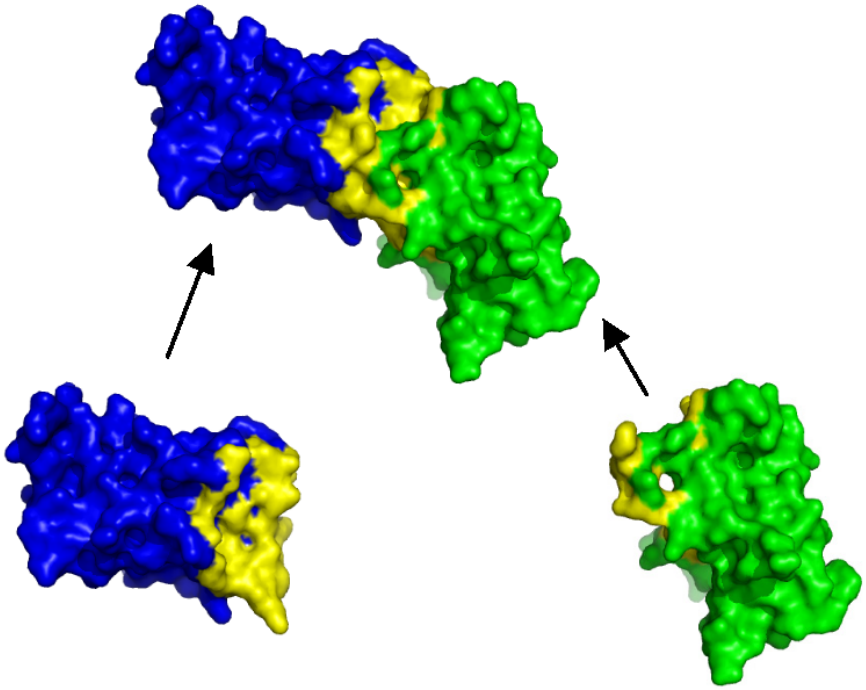


Figura 3 – **Exemplo de interação proteica:** duas proteínas com semelhança estrutural. Imagem retirada do Mundo da Bioquímica.

Duas ou mais proteínas podem se conectar de diferentes formas, neste trabalho, será usada a "associação funcional", ou seja, uma ligação entre duas proteínas que contribuem conjuntamente para uma função biológica específica (STUDHAM et al., 2014; SNEL; BORK; HUYNEN, 2002; ENRIGHT; OUZOUNIS, 2001). Para que duas proteínas sejam associadas dessa forma, elas não precisam interagir fisicamente, em vez disso, é suficiente que pelo menos alguma de suas funções ocorra na mesma região celular. Por essa definição, mesmo proteínas que antagonizam umas às outras podem ser associadas funcionalmente, como um inibidor e um ativador no mesmo caminho. (SZKLARCZYK et al., 2019). A Figura 3 apresenta um exemplo de interação proteica, nela duas proteínas com alguma semelhança estrutural são pareadas.

3.3.1 Bases de dados: interações proteicas

Diversos portais apresentam e disponibilizam interações entre proteínas. Neste trabalho será especificado abaixo 3 portais de alta relevância pelo número de conexões, proteínas e espécies considerado:

- **String:** "STRING é um banco de dados de interações proteína-proteína conhecidas e previstas. As interações incluem associações diretas (físicas) e indiretas (funcionais); elas decorrem da previsão computacional, da transferência de conhecimento entre organismos e de interações agregadas a partir de outras bases de dados (primárias)". O String, em especifico, oferece dois conjuntos de interações proteicas: um primeiro conjunto com as interações de 9 espécies consideradas de alta significância e confiabilidade pelo String; outro conjunto considerando todas as espécies do String, os dois conjuntos de dados serão usados neste trabalho. Na figura 3.3.1 um exemplo visual das interações proteicas da String pode ser visto.(<https://string-db.org/>) (SZKLARCZYK et al., 2019) [1]

- **BioGrid:** "BioGRID é um repositório de interação biomédica com dados compilados por meio de esforços abrangentes de curadoria. Sua atual versão é a 4.4.213 com pesquisa de 80.848 publicações e 2.537.592 interações proteicas ou genéticas, 29.417 interações químicas e 1.128.339 modificações pós-translacionais de espécies principais do organismo modelo."(<https://thebiogrid.org/>) (**??**) [2]

- **Intact:** "O IntAct fornece um sistema de banco de dados gratuito e de código aberto e ferramentas de análise para dados de interação molecular. Todas as interações são derivadas da curadoria da literatura ou de submissões diretas de usuários."(<https://www.ebi.ac.uk/intact/>) (HERMJAKOB et al., 2004) [3]

3.3.2 Bases de dados: genomas da $Glycine\ max$ e $Phakopsora\ pachyrhizi$

Segundo o artigo (STRATTON; CAMPBELL; FUTREAL, 2009) o sequenciamento do genoma fornece a coleção mais abrangente da variação genética de um indivíduo. Especificamente portais como o Phytozome (<https://phytozome-next.jgi.doe.gov/>) fornecem a sequência completa de nucleotídeos contidos no DNA de diversas espécies. Segundo o próprio site: o Phytozome é um portal de genômica comparativa vegetal do Instituto De Genoma Conjunto do Departamento de Energia, fornece aos usuários do JGI ($Joint\ Genome\ Institute$) e a comunidade científica vegetal mais ampla um hub para acessar, visualizar e analisar genomas vegetais sequenciados por JGI, bem como genomas e conjuntos de dados selecionados que foram sequenciados em outros lugares.

[1] Dados baixados e obtidos em Agosto de 2022
[2] Dados baixados e obtidos em Agosto de 2022
[3] Dados baixados e obtidos em Agosto de 2022

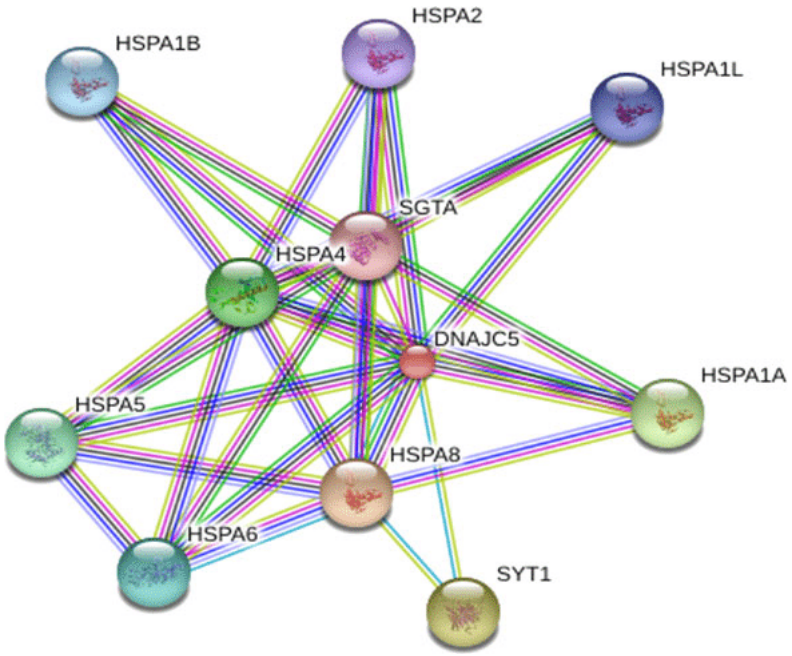


Figura 4 – **Exemplo das interações proteicas do String**. (BARBERA et al., 2017)

Deseja-se, para este trabalho, as sequências proteicas da $G.\ max$ e do $P.\ pachyrhizi$. Devido a alta relevância, principalmente econômica, das duas espécies. O sequenciamento completo e seguro das duas espécies pode e será facilmente obtidos neste trabalho.

3.3.3 Ferramentas: determinadores da sub-localização celular

Diversos formatos de relacionamentos proteicos podem e são usados neste trabalho. Porém, sabe-se que relacionamentos proteicos mais confiáveis são de proteínas com relevante similaridade funcional ou estrutural. Portanto, será usado um conjunto de 5 programas que, a partir da sequência proteica das proteínas usadas, são capazes de supor o possível local de funcionalidade de uma proteína. A saída desses programas será usada, neste trabalho, para supor que proteínas com atuação em semelhantes regiões celulares tenham maiores chances de se relacionarem.

Possíveis programas de alta qualidade são listados abaixo:

- **SignalP:** segundo o site do SignalP (<https://services.healthtech.dtu.dk/service .php?SignalP-5.0>) o servidor SignalP prevê a presença de peptídeos de sinal e a localização de seus locais de decote em proteínas. Como saída o SignalP informa se a proteína de entrada possui: um peptídeo sinal Sec (saída: Sec/SPI), um peptídeo sinal de lipoproteína (saída: Sec/SPII), um peptídeo sinal Tat (saída: Tat/SPI) ou nenhum peptídeo sinal (saída: Other). [4]

[4] Dados baixados e obtidos em Agosto de 2022

- **TMHMM:** segundo o site do TMHMM (<https://services.healthtech.dtu.dk/ service.php?TMHMM-2.0>) o servidor TMHMM prevê hélices transmembranares em proteínas. O TMHMM foi classificado como melhor em uma comparação independente de programas para previsão de hélices TM. O TMHMM informa se a proteína de entrada é uma: proteína transmembrana alfa-helicoidal sem um peptídeo sinal (saída: TM); proteína alfa-helicoidal transmembranar com peptídeo sinal (saída: SP + TM); proteína transmembranar beta-barril (saída: Beta); proteínas globulares com peptídeo sinal (saída: SP + Globular) ou uma proteína globular sem peptídeo sinal (saída: Globular). [5]

- **TargetP:** segundo o site do TargetP (<https://services.healthtech.dtu.dk/service. php?TargetP-2.0>) o servidor TargetP-2.0 prevê a presença de algumas sequencias: peptídeo de sinal (SP), peptídeo de trânsito mitocondrial (mTP), peptídeo de trânsito de cloroplasto (cTP) ou peptídeo de trânsito luminal de timkokoide (lTP). Para as sequências previstas contendo uma sequencia n-terminal também é previsto um potencial local de decote. [6]

- **WoLF PSORT:** segundo o site do WoLF PSORT (<https://wolfpsort.hgc.jp/>) WoLF PSORT prevê os locais de localização subcelular de proteínas com base em suas sequências de aminoácidos. O método faz previsões baseadas em motivos de sinal de classificação conhecidos e em algumas características de sequência correlativa, como o conteúdo de aminoácidos. Assim o PSORT exibe algumas informações sobre os sinais de classificação detectados, o que é útil para ajudar os usuários a determinar a confiabilidade da previsão em casos específicos. [7]

- **Phobius:** segundo o site do Phobius (<https://phobius.sbc.su.se/poly.html >) o servidor phobius é para previsão da topologia transmembrana e peptídeos de sinal da sequência de aminoácidos de uma proteína. Ele identifica o número de segmentos transmembrana previstos, se um peptídeo sinal foi previsto ou não e a possível topologia prevista da proteína. [8]

3.4 Grafos e Redes Complexas

O mundo é repleto de sistemas intrinsecamente complexos. Por exemplo, o funcionamento corporal humano requer a existência de milhares de vias metabólicas funcionando em conjunto. A comunicação entre a sociedade é possível graças às infraestruturas de comunicação que integram bilhões de computadores e satélites. Esses sistemas são chamados coletivamente de sistemas complexos, caracterizados por serem compostos de muitas partes interconectadas, por arranjos complexos de peças ou unidades. Por trás de cada sistema complexo existe uma

[5] Dados baixados e obtidos em Agosto de 2022
[6] Dados baixados e obtidos em Agosto de 2022
[7] Dados baixados e obtidos em Agosto de 2022
[8] Dados baixados e obtidos em Agosto de 2022

rede que codifica as interações entre os componentes do sistema, sendo essa chamada rede complexa (BARABÁSI et al., 2016).

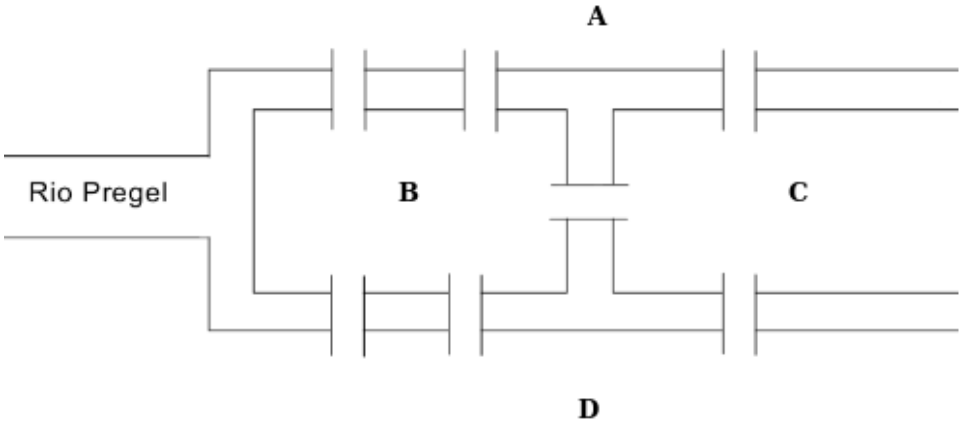


Figura 5 – **Diagrama das sete pontes de Königsberg.**

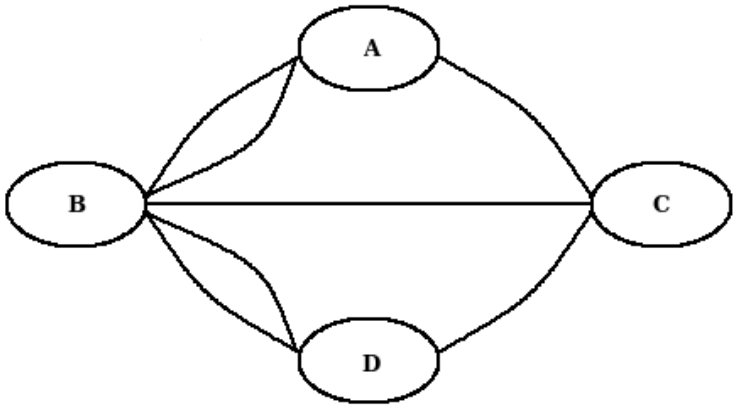


Figura 6 – **Sete pontes de Königsberg em grafo:** grafo gerado a partir da configuração de pontes expressas no diagrama das sete pontes de Königsberg.

O primeiro grafo conhecido foi desenvolvido no século XVII por Leonard Euler no seu estudo do problema das Sete Pontes de Konigsberg I (EULER, 1953). O problema consistia em, a partir da configuração de pontes da cidade de Konigsberg apresentada na Figura 5, encontrar um caminho que passe uma única vez por todas as pontes. Para a resolução do problema, Leonhard Euler modelou caminhos como arestas de um grafo e as interseções entre esses caminhos como vértices, gerando o grafo apresentado na Figura 6. Um grafo $G = (V, E)$ é uma estrutura formada por um conjunto não vazio de vértices V e um conjunto de arestas $E \subseteq P(V)$ com $P(V) = \{\{x, y\} : x, y \in V\}$ que é o conjunto de todos os pares não ordenados e não necessariamente distintos gerados a partir de V. Cada aresta $\{x, y\} \in E$ é formada por um par de vértices distintos ($x \neq y$). Para cada par de vértices, existe no máximo uma aresta associada a eles (FEOFILOFF; KOHAYAKAWA; WAKABAYASHI, 2011). Um exemplo de um grafo pode ser visto na Figura 7 que apresenta as possíveis adjacências entre estados do Brasil.

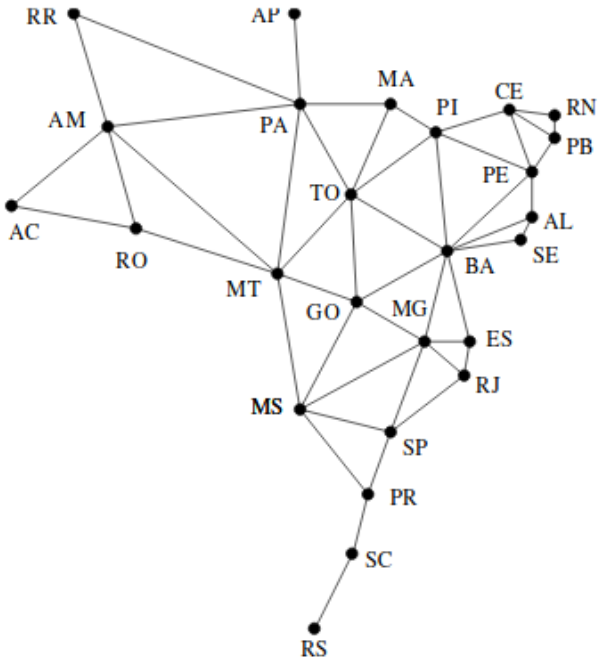


Figura 7 – **Grafo:** possíveis adjacências entre estados do Brasil. (FEOFILOFF; KOHAYAKAWA; WAKABAYASHI, 2011)

O grafo apresentado na Figura 7 é classificado como simples (i.e., regular), não valorado, não direcionado e conexo. Porém, um grafo pode assumir várias propriedades que podem ser usadas para modificá-lo a fim de modelar os relacionamentos de problemas reais, como apresentado a seguir

Grafo valorado ou ponderado: um grafo valorado é aquele nos quais valores são atribuídos às suas arestas ou aos seus vértices. Tendo como exemplo um grafo que modela estradas conectando cidades, com cidades sendo vértices e as estradas sendo arestas, um grafo valorado poderia apresentar a distância entre as cidades em suas arestas (em quilômetros por exemplo), ou o número de habitantes por cidade em cada vértice. Um exemplo de grafo valorado pode ser visto na Figura 8.

Grafo direcionado: um grafo é direcionado se cada aresta $e = \{x, y\}$ conecta o vértice x ao vértice y mas não necessariamente conecta o vértice y ao x. Um exemplo de grafo direcionado pode ser visto na Figura 8, no qual, há uma via de mão única conectando Cambridge a Stanford e outra via de mão única conectando Stanford a Cambridge. Na figura a referência à direcionalidade se dá unicamente pelas setas conectando as cidades.

Computacionalmente, um grafo é recorrentemente representado por uma entre duas estruturas de dados: lista de adjacência e matriz de adjacência. Uma matriz de adjacência é uma matriz $A \in \mathbb{R}^{|V|\times|V|}$ onde o elemento $A_{ij} = 1$ caso exista ligação entre os vértices i e j e $A_{ij} = 0$

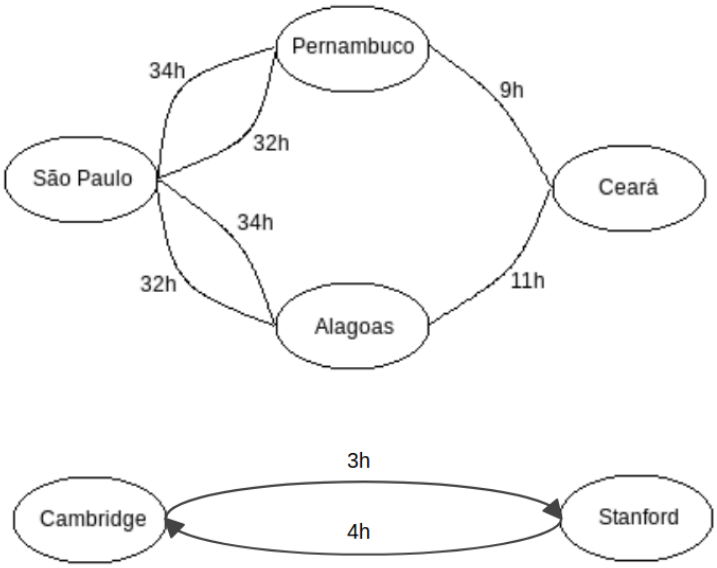


Figura 8 – **Grafo ponderado e direcionado:** exemplo de grafo apresentando possíveis custos temporais de rotas de carro entre cidades e estados.

caso contrário, para grafos não valorados. Em grafos valorados, A_{ij} pode assumir o valor da aresta, com $A_{ij} = A_{ji}$, se o grafo for não direcionado e $A_{ij} \neq A_{ij}$, se o grafo for direcionado. A lista de adjacência é uma lista de tamanho E onde cada posição da lista contém uma tupla (i, j) que representa uma conexão entre os vértices i e j (KHULLER; RAGHAVACHARI, 1996). A lista de adjacência é geralmente preferida para grafos de menor grau médio, porque o espaço de armazenamento necessário é linear em relação ao número de arestas do grafo. A matriz de adjacência pode possuir um custo alto de armazenamento em relação à lista, porém possui um custo linear de acesso aos dados. Assim, tal representação é geralmente preferida no caso de grafos densos (KHULLER; RAGHAVACHARI, 1996).

Redes complexas surgem dos relacionamentos intrinsecamente existentes nos sistemas complexos, representando relacionamentos reais contidos nestes sistemas, possivelmente através de grafos (BARABÁSI et al., 2016). Em (NEWMAN, 2003) é dito que as redes complexas se originaram de aplicações oriundas da teoria de grafos e de conceitos provenientes da mecânica estatística, física não-linear e sistemas complexos.Assim, o estudo de redes complexas possui um caráter multidisciplinar e cobre várias áreas do conhecimento. Já em (BASSETT et al., 2011) é definido que teoria de redes é uma parte da teoria dos grafos. Uma rede pode ser definida como um grafo no qual os vértices ou arestas possuem atributos que representam algum fator real. Um grafo é definido pela informação estrutural contida em sua matriz de adjacência. Uma rede complexa pode ter uma quantidade arbitrária de informações auxiliares tornando-a capaz de melhor representar vários sistemas complexos.

3.5 Ferrugem asiática

A ferrugem asiática, uma doença que acomente a soja, causada pelo fungo $P.\ pachyrhizi$ e é o principal objetivo de estudo deste trabalho. A infecção do $P.\ pachyrhizi$ desde o ano de 2001, tem causado diversos prejuizos economicos, e ainda é um problema para o Brasil e outros paises do mundo. (ANDRADE; ANDRADE, 2002)

Algumas estrategias de combate a Ferrugem asisticaestão sugindo. Como diversos fungicidas que já foram desenvolvidos para a Ferrugem asiática, dois podem ser vistos na imagem 9. Na imagem 9 pode se ver dois fungicidas com indicação de uso em diferentes fases de vida da soja, segundo a Embrapa (Empresa Brasileira de Pesquisa Agropecuária), o periodo de vida da soja é algo relevante e que deve ser estudado. (SEIXAS et al., 2018)

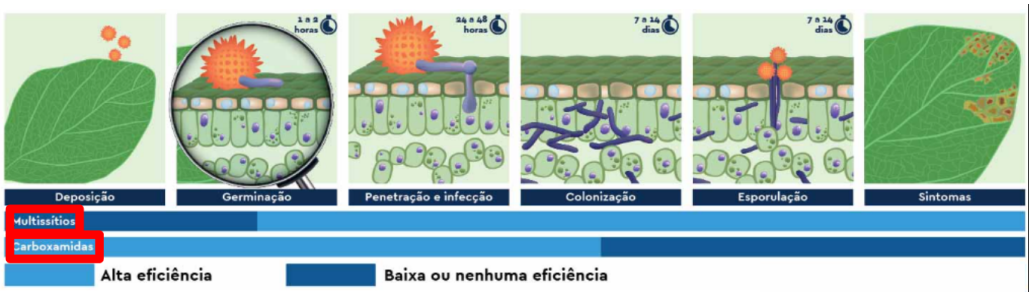


Figura 9 – **Fungicidas para soja com ferrugem asistica**

A figura 10 apresenta uma indicação da Embrapa de plantio da soja em diferentes estados do Brasil. Isso pois o Brasil possui estados com diferentes climas, seguindo este calendario, espera-se que quando ocorrer a contaminação, a soja tenha idade especifica para melhor combater a ferrugem asiatica e outras infecções.

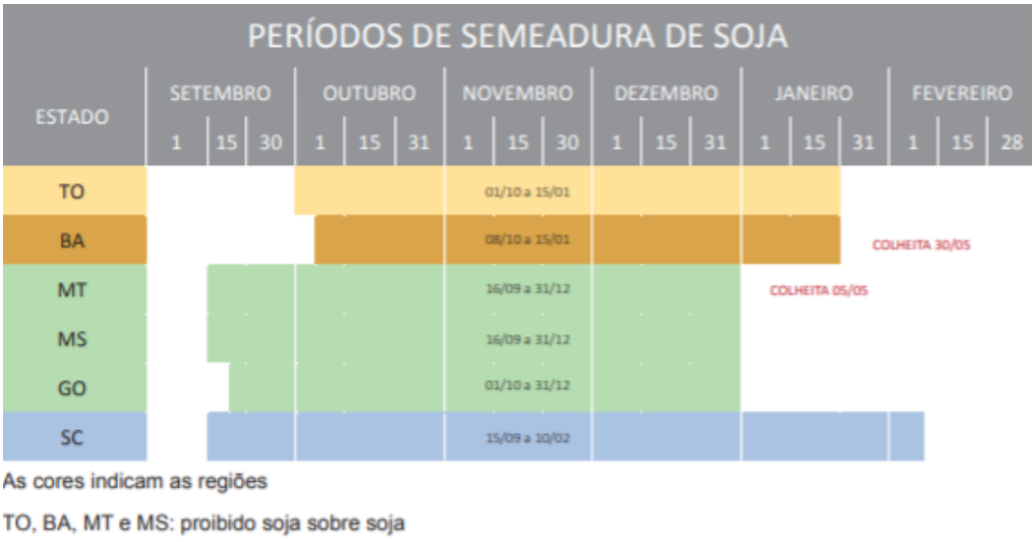


Figura 10 – **Indicação de perioo de plantio da soja no Brasil.**

Um possível exemplo de contaminação do fungo pode ser visto na figura 11. Nela pode ser visto diversas proteinnas atuadoras que ainda não foram identificadas, este trabalho pode ajudar a identificar essas protínas. É comum a existencia de estudos como (COSTA, 2010; BUENO, 2021) que estudam possiveis proteínas que participam da infecção. Apresentando a relevancia deste estudo.

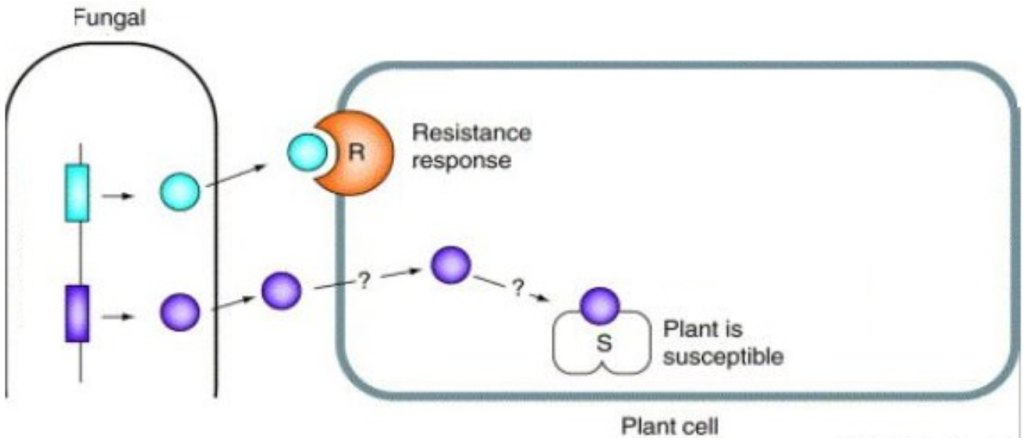


Figura 11 – **Exemplo de contaminação de uma planta por um fungo.** Diversas proteinas envolvidas na contaminação podem ser vistas.

3.6 Trabalhos relacionados

Diversos trabalhos apresentam a metodologia próxima à proposta. Alguns destes trabalhos serão apresentadas nesta seção.

Várias ferramentas estabelecem possíveis relação proteicas como: String ((SZKLARCZYK et al., 2019)); BioGrid ((**??**)); Intact ((HERMJAKOB et al., 2004)); Mint ((ZANZONI et al., 2002)); DIP ((XENARIOS et al., 2000)). Alguma dessas bases de dados serão usadas por este trabalho. Os 5 trabalhos especificados acima apenas estabelecem conexões proteicas de uma mesma espécie, a metodologia apresentada e o estudo realizado aqui permite conectar proteínas de espécies diferentes.

Trabalhos como (VASCONCELOS; CAMPOS; REZENDE, 2018; FLÓREZ et al., 2010; REZENDE et al., 2012; DIAS, 2020) utilizam uma metodologia parecida com a proposta por este, também com o objetivo de determinar novas conexões entre conjuntos diferentes de células. Porém, os 4 trabalhos apresentados não encontram relações proteicas entre a $G.\ max$ e o $P.\ pachyrhizi$.

Alguns trabalhos apresentam metodologias com limitações em relação ao aqui proposto e, por estudarem a $G.\ max$ e o $P.\ pachyrhizi$ podem ser usados em conjunto a este. Por exemplo, o trabalho (PIEROZZI et al., 2008; RODRIGUES et al., 2015), encontra apenas proteínas de defesa da $G.\ max$, em relação ao $P.\ pachyrhizi$ e outros causadores de doenças na planta. O estudo (FURTADO et al., 2009) analisou o efeito da idade da $G.\ max$ em ataques do $P.$

$pachyrhizi$. O estudo (TSUKAHARA; HIKISHIMA; CANTERI, 2008) analisa a influencia do clima na defesa da $G.\ max$ em ataques do $P.\ pachyrhizi$.

4 Metodologia

Para sintetizar a metodologia proposta por este trabalho, inicialmente, serão apresentadas as bases de dados que foram escolhidas e serão utilizadas na seção 4.2. Por seguinte, nas seções 4.3 e 4.4, serão apresentadas as ferramentas utilizadas para analisar os dados obtidos anteriormente. Por fim, na seção 4.1 e seção 4.6, será apresentado o pipeline que usa as base de dados e as ferramentas expostas para sintetizar os objetivos deste trabalho. O Pipeline que será apresentado será aplicado aos dados apresentados.

4.1 Pipeline

Este trabalho executa diversos programas em sequência. A sequência de programas pode ser vista na imagem 12 e foi chamada de pipeline. A sequência do pipeline explicada textualmente pode ser vista abaixo. A descrição completa dos programas e dados usados são descritas em outras seções deste mesmo capitulo.

- **Sequência proteica da $G.\ max$ e $P.\ pachyrhizi$:** as sequências de proteínas especificadas na sessão 3.3.2.
- **Sequência proteica da String, BioGrid e Intact:** as sequências de proteínas especificadas.
- **Blast versão DIAMOND:** o DIAMOND é aplicado às 5 sequências proteicas, procurando e gerando as 6 sequências de dados entre as bases de dados (String, BioGrid e Intact) e as sequências proteicas das duas espécies ($G.\ max$ e $P.\ pachyrhizi$).
- **Grafo de interações:** o grafo objetivado por este trabalho. União dos relacionamentos obtidos pelo DIAMOND pela semelhança entre as proteínas da $G.\ max$ e do $P.\ pachyrhizi$ com as relações proteicas obtida pelos bancos de dados String, BioGrid e Intact. Nesta etapa também é gerado o score final das conexões encontradas, este score é baseado nos scores de confiança do String, BioGrid e Intact, ele é calculado a partir de cálculos clássicos da estatística e de um trabalho citado.
- **Análises topológicas:** o grafo obtido passa por diversas análises, as que objetivam avaliar características topológicas dos vértices contidos nesse grafo são realizadas neste momento. Estas análises foram descritas na sessão 4.4.
- **Análises de localização celular:** o grafo obtido passa por diversas análises, as que objetivam avaliar características das funcionalidades proteicas das espécies especificadas, são realizadas neste momento. Estas análises foram descritas na sessão 3.3.3.

- **Dados experimentais:** novos dados apresentados na seção 4.5.

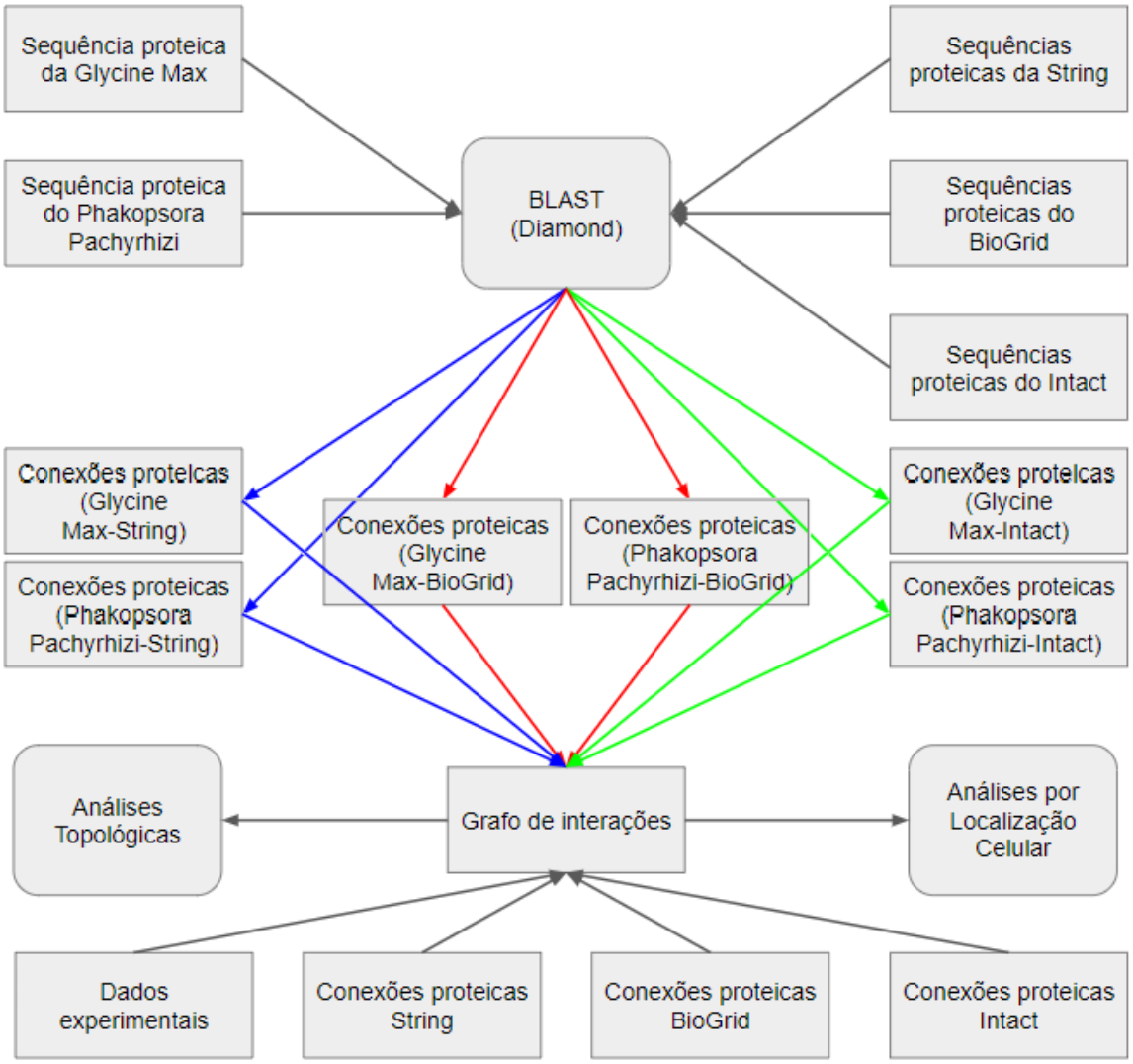


Figura 12 – **Pipeline metodológico**. Quadrados representam dados. Quadrados com quina arredondada representam programas.

Este pipeline pode ser obtido e executado. Ele foi totalmente implementado como um simples arquivo de texto que pode ser executado em terminal do Linx. O pipeline disponibilizado executa diversos programas como o DIAMOND, estes programas podem ser encontrados e compreendidos em suas respectivas páginas na internet. Todos os programas feitos para completar os objetivos deste trabalho foram implementados em C++, os mesmos são automaticamente executados pelo pipeline em texto disponibilizado.

A sequência de programas pode ser vista na Imagem 13. Inicialmente o DIAMOND é executado, após ele é executado um programa de filtragem nos dados de saída do DIAMOND, com o objetivo de selecionar as conexões devidamente substanciais, um gerador de grafos é executado, ele conecta a saída do programa de filtragem com outros dados, programas de análise

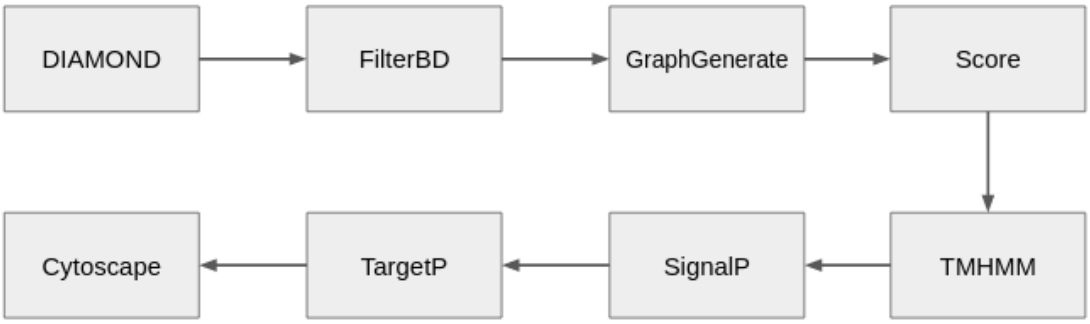


Figura 13 – **Sequência de Programas**

funcional proteica (SignalP, TMHMM e TargetP) são executados para ajudar a analisar o grafo gerado. Por fim são executados algoritmos no grafo obtido com o objetivo de analisa-lo.

4.2 Bases de dados

Diversos portais apresentam e disponibilizam interações entre proteínas. Neste estudo, em especifico, pela relevância dos mesmos previamente citada, será usados 3 portais (String, BioGrid e Intact). Os 3 portais oferecem um valor entre 0 e 1 de score, valores próximos de 1 representam relações altamente confiáveis. Inicialmente todas as relações são filtradas e apenas relacionamentos com score maior que 0,9 são mantidos. Após isso, as 3 bases de dados são usadas em conjunto para sintetizar o grafo estudado.

Neste trabalho também será usado os potais SignalP, TMHMM e TargeP. Deseja-se, no futuro, encontrar proteínas de espécies diferentes ou semelhantes, e que essas proteínas sejam de alguma sub-localidade apresentada acima semelhante. Assim, tem-se semelhanças funcionais encontradas pelo conjunto de programas String, Biogrid, Intact ou DIAMOND e, em conjunto, semelhanças funcionais encontradas pelo segundo conjunto de programas SignalP, TMHMM ou TargeP. Espera-se maior semelhança em conexões proteicas encontradas pelos dois conjuntos de programas em paralelo.

Para a completude deste trabalho será necessário as sequências proteicas da $G.\ max$ e do $P.\ pachyrhizi$.

O sequenciamento da $G.\ max$ foi obtido no próprio Phytozome. O Phytozome porém, na data em que foi acessado (Agosto de 2022), possuía cinco diferentes sequenciamentos proteicos. O Phytozome para cada sequência oferece um conjunto de informações. Para concluir os propósitos deste trabalho, a sequência com maiores valores nas propriedades "Percentage of Eukaryote BUSCO Genes"(segundo o guia do usuário BUSCO: "a BUSCO tenta fornecer uma avaliação quantitativa da completude em termos de conteúdo genético esperado"), "Percentage of Embroyphyte BUSCO Genes" e "Number of Protein-coding Transcripts" foi escolhida, espera-se que a sequência com maior valor dessas propriedades seja a que apresente maiores informação

da espécie escolhida. Especificamente, no Phytozome, o genoma escolhido possui o nome "$Glycine\ max\ Fiskeby\ v1.1$"[1] [2]

Para o sequenciamento do $P.\ pachyrhizi$ o portal da MycoCosm foi usado (em Agosto de 2022). Segundo a nota técnica (EMPRABA, 2019): "foi estabelecido um consórcio internacional único, representando 12 instituições públicas e privadas. Este esforço conjunto congrega a Fundação 2Blades, a Bayer, a Empresa Brasileira de Pesquisa Agropecuária (Embrapa), as Universidade Alemãs de Hohenheim e de RWTH Aachen, o Instituto Nacional da Pesquisa Agronômica (INRA-França) e a Universidade de Lorraine (França), além do Joint Genome Institute (JGI, EUA), da KeyGene, do Laboratório Sainsbury (Reino Unido), da Syngenta e a Universidade Federal de Viçosa (Brasil). Hoje, esse consórcio anuncia a conclusão do sequenciamento e montagem", por fim, o consorcio depositou o sequenciamento no portal MycoCosm. [3]

Em especifico, neste trabalho, duas sequencias de proteínas do $P.\ pachyrhizi$ foram usadas. Ambas as sequencias foram obtidas no MycoCosm porém, a primeira representa a sequencia inteira do $P.\ pachyrhizi$, a segunda sequencia do $P.\ pachyrhizi$ representa apenas as proteínas de superfície determinadas pelos programas de sublocalização (TMHMM, SignalP, TargetP, Phobius e WoLF PSORT,) apresentados na Seção 3.3.3.

4.3 Alinhamento local

Segundo o estudo (KORF; YANDELL; BEDELL, 2003) o BLAST ($Basic\ Local\ Alignment\ Search\ Tool$) é um algoritmo para comparar informações de sequências biológicas primárias, tais como sequências de aminoácidos de diferentes proteínas ou nucleotídeos de sequências de DNA. Segundo o site do BLAST (<https://blast.ncbi.nlm.nih.gov/ Blast.cgi>), o BLAST encontra regiões de semelhança entre sequências biológicas. O programa compara sequências de nucleotídeos ou proteínas a bancos de dados sequênciais e calcula a significância estatística.

Devido ao tamanho dos dois genomas em que o BLAST deve ser aplicado neste trabalho (genomas da $G.\ max$ e do $P.\ pachyrhizi$), será usada uma versão implementada e acelerada do BLAST chamada DIAMOND (BUCHFINK; REUTER; DROST, 2021; BUCHFINK; XIE; HUSON, 2015). O DIAMOND obtém a mesma saída do BLAST porém, o mesmo gasta um menor valor temporal. Segundo a página do GitHub em que o DIAMOND é disponibilizado o DIAMOND analisa proteínas a uma velocidade de 100-10.000 vezes mais rápido que o BLAST clássico. [4]

[1] Dados baixados e obtidos em Agosto de 2022
[2] O genoma $Glycine\ max\ Fiskeby\ v1.1$ esta disponibilizado na data que este artigo foi escrito no site do Phytozome.
[3] Dados baixados e obtidos em Agosto de 2022
[4] Dados do DIAMOND foram baixados e obtidos em Agosto de 2022

Para executar o Diamond o comando utilizado foi: diamond blastp –query ... –db ... –outfmt 6 qseqid sseqid qstart qend pident length slen evalue sstart send stitle qlen –evalue 1e-10 –query-cover 50 –subject-cover 40 –out ....

O Diamond oferece para cada conexão encontrada uma pontuação entre 0 e 1 que define a confiabiilidade da mesma. Após a execução do Diamond, para o uso de apenas conexões confiáveis, todas as conexões do Diamond com pontuaç ão menor que 0.8 foram removidas.

O Blast ou o Diamond também oferece diversas informações sobre as conexões encontradas (que podem ser encontradas em seu site). Para evitar o uso de conexões fracas, diversos filtros foram aplicados as conexões encontradas, todas as várias filtragens podem ser vistas no codigo disponibilizado no algporitmo "GraphGenerate". Em especfico, foi analisado os tamanhos das sequencias comparadas e, o número de incompatibilidade nucleica encontrada pelo Diamond.

4.4 Analisador e visualizador de grafos

O grafo gerado pelos métodos propostos acima é analisado por um conjunto de ferramentas que serão apresentadas nesta seção. O objetivo em aplicar essas ferramentas e obter mais informações sobre o grafo gerado e o que ele representa.

Inicialmente será usado o Cytoscape no grafo. Segundo o site do Cytoscape (<https://cytoscape.org/>, acessado em 2022), o Cytoscape é uma plataforma de software de código aberto para visualizar redes de interação molecular e caminhos biológicos. Neste trabalho o Cytoscape será usado principalmente para visualizar o grafo gerado. Porém, o Cytoscape também será usado para obter algumas propriedades do grafo, essas propriedades serão uteis para a analise e a aplicação do grafo. As propriedades que serão usadas são apresentadas abaixo:

- **Analise do grau:** o grau de um vértice é o numero de outros vértices conectados ao inicial (NETTO, 2003). Neste trabalho, vértices do alto grau representam proteínas altamente conectas a outros. Caso sejam proteínas de uma mesma espécie, pode-se supor que, a proteína estudada, possui grande semelhança com diversas outras em um organismo; se forem proteínas de espécies distintas, proteínas com alto grau representam possíveis conexões entre espécies e, possivelmente, podem ser foco de produção de, por exemplo, fungicidas.

- **Distribuição de graus:** é analisado o grau de cada vértice em um grafo. A analise da distribuição de graus pode apresentar propriedades do grafo. Especificamente e existência de hubs, hubs são considerados vértices com um alto grau em relação aos outros (AMARAL; LADEIRA, 2021), neste grafo, são vértices com grandes semelhanças funcionais a outros ou, que participam de grandes grupos proteicos funcionais. Por exemplo, caso um produto químico atue sobre um hub, em paralelo ele também atuará sobre várias outas proteínas.

- **Betweenness centrality:** a centralidade de $betweenness$ é a soma do numero de caminhos mínimos que passam por determinado vértice (LULLI et al., 2015). Neste trabalho, vértices com alta centralidade de $betweenness$, representam proteínas com alta relevância para conexão de outras proteínas. Assim como anteriormente, um químico que atua sobre uma proteína com alto valor de $betweenness$, possivelmente, atuará em paralelo sobre várias outras proteínas.

Pelo Cytoscape será usado o método de Louvain (MEO et al., 2011) no grafo. O método de Louvain objetiva encontrar comunidades em um grafo, ou seja, encontrar vértices com alguma semelhança estrutural. Segundo alguns estudo o método de Louvain se caracteriza por detectar comunidades de forma rápida e eficaz (CARNIVALI et al., 2018; LESKOVEC; LANG; MAHONEY, 2010; LANCICHINETTI; FORTUNATO, 2009). O algoritmo de Louvain não considera informações funcionais biológicas, porém compreender a anatomia gráfica gerada também pode ser util pra os objetivos descritos. Para tornar a aplicação mais simples. será usada uma versão implementado do Algoritmo de Louvain disponibilizada pelo Cytoscape, chamado de CyCommunityDetection (SINGHAL et al., 2020)

Após a detecção de comunidades, em posse das comunidades do grafo, será aplicado a análise de enriquecimento funcional. Para os estudos de funções gênicas e processos biológicos associados às diferentes comunidades, um arquivo de anotação de termos GO será obtido com base nas ferramentas $PANNZR2$ e $EggNogg$ (CANTALAPIEDRA et al., 2021; TÖRÖNEN; MEDLAR; HOLM, 2018). Para isso serão utilizados, separadamente, os arquivos fasta de proteínas de $P.\ pachyrhizi$ e $G.\ max$. A partir da anotação do $PANNZR2$ e $EggNogg$, um arquivo de anotação global no padrão do plugin $BiNGO$ será obtido e utilizado para identificar os processos enriquecidos nos diferentes clusters de proteínas. Para isso, o teste hipergeométrico, com correção múltipla de $Benjamini\ E\ Hochberg\ False\ Discovery\ Rate$ (FDR) foi empregado (MAERE; HEYMANS; KUIPER, 2005).

4.5 Dados e analises baseadas em $High\ Throughput\ Sequencing$

Um estudo de expressão gênica utilizando dados de $High\ Throughput\ Sequencing$ (HTS) será realizado para identificar prováveis genes efetores em $P.\ pachyrhizi$ e prováveis do $G.\ max$. Para isso, dados robustos da interação soja-$P.\ pachyrhizi$ serão utilizados, baseado no $bioproject\ PRJNA565800$ (ELMORE et al., 2020) e $PRJNA294234$ (CARVALHO et al., 2017). O $bioproject\ PRJNA565800$ é um dos conjuntos de dados mais completos para estudo de transcriptoma dessa interação e que tem foco na identificação de genes efetores do patógeno, porém, esse estudo deu foco em genes não-soja e não-plantas, buscando compreender apenas os potenciais efetores do patógeno.

Estes confiáveis dados serão comparados aos dados utilizados por este trabalho, o objetivo desta comparação é verificar se os dados obtidos por este trabalho estão de acordo com dados

existentes e clássicos da infecção já estudada.

4.6 Algoritmos desenvolvidos

Os programas produzidos independentes à este trabalho (DIAMOND, TMHMM, SignaP, TargetP) não serão estudados e analisados. Entre os programas desenvolvidos por este trabalho tem-se apenas o "FilterBD", "GraphGenerate"e "Score", como mostra a imagem 13.

O algoritmo "FilterBD" é simples. O algoritmo apenas lê os dados dos bancos de dados de conexões proteicas e grava as melhores conexões encontradas em arquivos separados, com o objetivo de acelerar o algoritmo seguinte "GraphGenerate". Este algoritmo ("FilterBD") não possui qualquer otimização de velocidade.

O algoritmo "GraphGenerate" lê os arquivos de saída do algoritmo "FilterBD" e as conexões produzidas pelo DIAMOND. Após as leituras, é analisado se, entre espécies semelhantes, há proteínas da saída do DIAMOND conectadas por algum dos bancos de dados String, BioGrid ou Intact ou pelo DIAMOND. O que é buscado pelo algoritmo pode ser visto em um exemplo na imagem 14. O algoritmo, por fim, retorna um grafo para cada banco de dados usado (String, BioGrid e Intact).

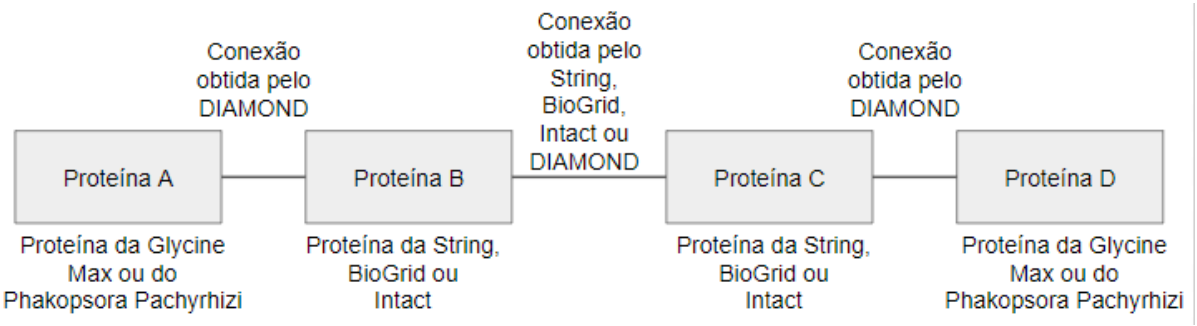


Figura 14 – **Exemplo do algoritmo gerador de grafos**

No exemplo da imagem 14 uma nova conexão entre as proteínas A e D é criada. Pelo String, BioGrid ou Intact é encontrada uma conexão entre as proteínas B e C, como há conexões entre A e B e entre C e D, pode-se implicar uma nova conexão entre as proteínas A e D. Veja que as proteínas A e D podem ser de espécies diferentes, portanto, este modelo é capaz também de encontrar novas conexões proteicas entre espécies diferentes.

Veja também que as proteínas A e B (ou C e D), no exemplo da imagem 14, não são conectadas pelo String, BioGrid ou Intact pois A e B (ou C e D) não necessariamente são da mesma espécie.

O algoritmo "GraphGenerate" pode ser lento, devido a diversidade de dados utilizados, por este motivo foi proposto uma paralelização do mesmo, uma forma de reduzir seu tempo de processamento gasto. Esta otimização sera apresentada e discutida na seção 5.2.

Após a criação do grafo um algoritmo de "Score" percorre as conexões encontradas calculando uma confiança da conexão, devido a sua complexidade este algoritmo será apresentado na seção seguinte 4.6.1.

4.6.1 Score

Ó grafo sintetizado pode possuir um conjunto de 3 tipos de arestas: arestas entre proteínas de vértices do $G.\ max$; arestas entre proteínas de vértices do $P.\ pachyrhizi$; arestas entre proteínas do $G.\ max$ e do $P.\ pachyrhizi$. É esperado que este trabalho, seja capaz de falar qual destas conexões possuem maior relevância, baseado nos dados que foram usados para desenvolver cada aresta.

Felizmente o Blast, String, BioGrid e Intact oferecem um score que será usado por este trabalho. Inicialmente será filtrado as arestas, pouco confiáveis, que foram geradas por estas ferramentas (Blast, String, BioGrid e Intact) após isso, o valor de pontuação oferecido por cada uma das 3 (String, BioGrid e Intact) ferramentas, será usado para sintetizar o valor de Score final aqui disponibilizado.

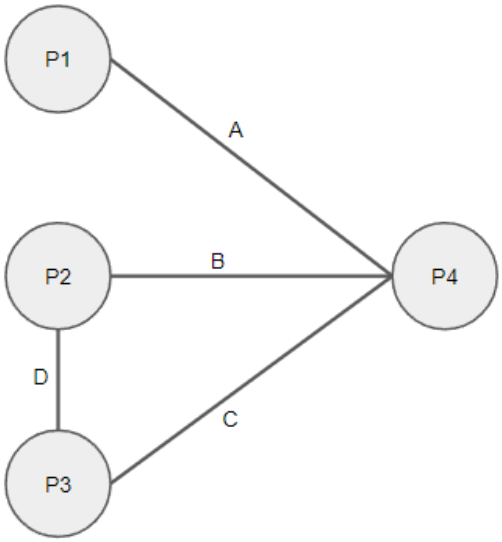


Figura 15 – **Exemplo de um possível grafo gerado**: esferas representam proteínas e traços conexões proteicas.

A figura 15 apresenta um possível grafo gerado por este programa. O valor das arestas (A, B, C e D) é oferecido pelos programas usados (Blast, String, BioGrid e Intact). Porém, neste exemplo, é possível ver que há uma conexão entre os vértices P1 e P4 e, duas conexões entre os vértices P2 e P4 (uma passando pela aresta A e outro passando pelas arestas D e C). Portanto, neste exemplo, o Score entre P1 e P4 é o somatório de 1 valor já, o Score entre P2 e P4 é o somatório de 2 valores. Portanto, o Score oferecido por este programa será o somatório de um conjunto de valores, estabelecidos pelo número de caminhos entre dois vértices do grafo.

Veja que ciclos podem ocorrer neste grafo, podendo gerar erros como o apresentado na figura 16 aonde, por exemplo, a pontuação de P1 para P2 pode ser um somatório de infinitos termos. Para contornar este erro, o programa que calcula o Score possui uma distância máxima entre dois vértices, um número máximo de vértices é predefinido, se um caminho possui um número maior de vértices do que o predefinido, automaticamente, o caminho é desconsiderado.

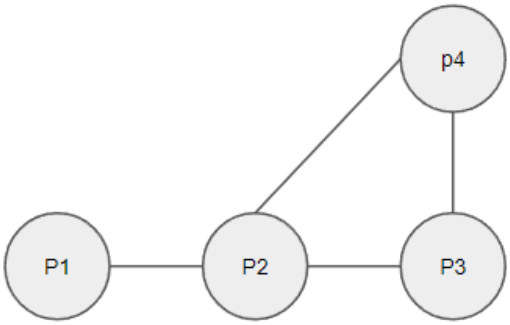


Figura 16 – **Exemplo de um possível grafo gerado com ciclo**: esferas representam proteínas e traços conexões proteicas.

Também é possível a ocorrência de vários caminhos conectando dois vértices. Por exemplo, na figura 15 existe dois caminhos conectando P2 à P4, o caminho que passa pela aresta B e o caminho que passa em conjunto pelas arestas D e C. Os portais usados oferecem os valores das arestas (no exemplo A, B, C e D) mas não o valor do caminho que passa por D e C.

Para encontrar o valor do caminho que passa pelas arestas C e D no exemplo da figura 15, será usado o calculo interseção. A interseção de dois conjuntos A e B é conjunto formado por todos os elementos que pertencem simultaneamente a A e B (KIRILOV, 2017). Basicamente a interseção calcula a probabilidade de duas ao mais arestas ocorrerem em conjunto. A interseção possui o simbolo $\cap$ e é calculada como a multiplicação dos conjuntos, no exemplo: $B \cap C = B \cdot C$.

Todos os caminhos conectando vértices, que passam por apenas uma aresta ou que passam por mais de uma, com o calculo da interseção, possuem um valor entre 0 e 1. Porém deseja-se saber o valor final da conexão entre dois vértices, para isso, após o cálculo do número de caminhos entres todos pares de vértices do grafo, a equação 4.1 será usada. A equação é derivada do trabalho (FLÓREZ et al., 2010) que possui semelhantes objetivos a este. Como o score dos três métodos é um valor entre 0 e 1, o retorno da equação 4.1 também será sempre um valor entre 0 e 1.

$$Score = 1 - \prod_{i \in E} (1 - R_i)^n \tag{4.1}$$

Onde R_i é o score do método (String, BioGrid ou Intact) na aresta i, n e E é o número de interações previstas entre dois vértices pelos distintos bancos de dados.

Foi analisado em conjunto o uso da união de conjuntos. A união de conjuntos corresponde a junção dos elementos dos conjuntos dados (KIRILOV, 2017). Porém sobre testes realizados no grafo gerado, a equação gerada pela equação 4.1 obteve melhores resultados, portanto, a equação 4.1 será a usada por este trabalho.

Afim de evidenciar as arestas com maior significância, um valor mínimo de pontuação foi estabelecido. Após todo o calculo, arestas com um valor menor que o limite são simplesmente ignoradas. A tabela 1 apresenta o número de arestas com pontuação em diversos intervalos, esta tabela foi usada para definir que apenas valores de pontuação maiores que $0,7$ serão usados.

Tabela 1 – **Número de arestas com pontuação entre os intervalos estabelecidos.**

Valor	N Arestas
0.0 - 0.1	83
0.1 - 0.2	52
0.2 - 0.3	112
0.3 - 0.4	37
0.4 - 0.5	13
0.5 - 0.6	28
0.6 - 0.7	16
0.7 - 0.8	21
0.8 - 0.9	34
0.9 - 1.0	5203

4.7 Metodologia de testes

Todos os códigos usados e desenvolvidos serão disponibilizados, assim como todos os dados obtidos por esse estudo. Para validar a ferramenta proposta dois grafos foram desenvolvidos, ambos com os dados do $G.\ max$ e do $P.\ pachyrhizi$. O primeiro grafo foi usado para interpretar a ferramenta enquanto o segundo para avaliar a infeção do $P.\ pachyrhizi$. Os dois grafos desenvolvidos são descritos abaixo em subseções especificas.

Além do TMHMM, SignalP, TaregtP, Wolf Psort, Phobius e do próprio programa analisado, será apresentado outros programas que serão usados e podem ser usados em outros contextos para determinar a possível funcionalidade das proteínas encontradas. Neste estudo, portanto, é usado o eggNOG, segundo o próprio site do eggNOG é "um hierárquico, funcional e filogeneticamente anotado recurso ortológico baseado em 5090 organismos e 2502 vírus". Basicamente o eggNOG encontra possíveis funcionalidades proteicas comparando as proteínas de entrada a proteínas com funcionalidade conhecida.

Também pode e foi usado um detector de comunidade, um detector de comunidades encontra grupos de vértices bem conectados em um grafo (CARNIVALI et al., 2020). Especificamente foi usado o CyCommunityDetection, o CyCommunityDetection é um aplicativo

do Cytoscape e segundo o mesmo,para detecção de comunidades, é utilizado o algoritmo de Louvain, um clássico e eficiente algoritmo de detecção de comunidades (CARNIVALI et al., 2020). Algoritmos de detecção de comunidades podem ajudar a identificar vértices que possuem funções parecidas ou, que participam em conjunto de uma mesma função.

4.7.1 Metodologia para união dos bancos de dados

Grafo obtido com os dados do $G.\ max$ e do $P.\ pachyrhizi$ com o uso dos bancos de dados String, Biogrid e Intact. Espera-se com este estudo de caso maior interpretação da ferramenta, por isso foi utilizado bases de dados pequenas e facilmente acessíveis, todas as bases de dados utilizadas (proteínas do $G.\ max$, proteínas do $P.\ pachyrhizi$, dados do String, dados do Biogrid e dados do Intact) estão disponibilizadas para download.

Para esse grafo foi utilizado apenas 9 conjunto de dados de todo String, o próprio String selecionou esses 9 dados, os dados selecionadas são: Homo sapiens, Mus muscle, Arabidopsis thaliana, Saccharomyces cerevisiae, Drosophila melanogaster, Danio rerio, Caenorhabditis elegans, Caenorhabditis elegans, Escherichia coli e Pseudomonas aeruginosa. Segundo o String estas especies selecionadas são seus dados mais confiáveis (FRANCESCHINI et al., 2012; SZKLARCZYK et al., 2019).

Em conjunto, para a criação desse grafo, também foi utilizado outras duas bases de dados completas, o Intact e o Biogrid. Detalhes sobre as 3 bases de dados podem ser lidos em mais detalhes neste trabalho. O grafo que será apresentado portanto representa a união das conexões de 3 bancos de dados (String, Biogrid e Intact) com os dados completos do $G.\ max$ e do $P.\ pachyrhizi$.

4.7.2 Metodologia para String completo

O grafo produzido nesta etapa objetiva a analise mais detalhada da infecção, devido aos objetivo deste trabalho acreditamos que este seja um cenário esperado. Para isto será usado o maior conjunto de conexões proteicas encontradas o String completo, com mais especies além das 9 anteriormente usadas. Também será usado um conjunto restrito de proteínas do $P.\ pachyrhizi$ objetivando analisar apenas proteínas mais prováveis de atuarem na infecção, determinadas como de superfície por um dos cinco programas especificados na seção 3.3.3.

String completo

O banco de dados String possui dois conjuntos de dados, um com dados de 9 especies previamente especificadas e seu banco de dados total, nesta parte o banco de dados total do String será usado, com 12535 especies. O banco de dados completo do String naturalmente possui mais conexões e proteínas (FRANCESCHINI et al., 2012; SZKLARCZYK et al., 2019), ao usa-lo, espera-se o encontre de mais proteínas da infecção do $P.\ pachyrhizi$ sobre a $G.\ max$.

Glycine Max

Neste trabalho apenas um conjunto de dados do $G.\ max$ será usado. Assim como o grafo que será produzido na seção 4.7.1 como o que será aqui produzido possuem as mesmas informações proteicas.

Phakopsora Pachyrhizi

O grafo que será especificado nesta seção foi produzido com os dados do $G.\ max$ e do $P.\ pachyrhizi$, porém os dados do $P.\ pachyrhizi$ foram previamente filtrados como especificado na seção 4.2, este filtro objetivou a especificação de proteínas que possivelmente participam da infecção estudada. Para esse grafo foi usado todo o String com todas as suas espécies.

Para visualizar qual dos 5 programas determinou quais as proteínas do $P.\ pachyrhizi$ são de superfície, um Diagrama de Venn foi feito e pode ser visto na imagem 17, o diagrama foi feito pela ferramenta que pode ser acessada pelo link: https://bioinformatics.psb.ugent.be/webtools/Venn/. Os números na imagem 17 representam as proteínas de superfície especificadas por cada ferramenta. Foram usadas as 11587 proteínas do $P.\ pachyrhizi$ especificadas na imagem 17 no grafo produzido.

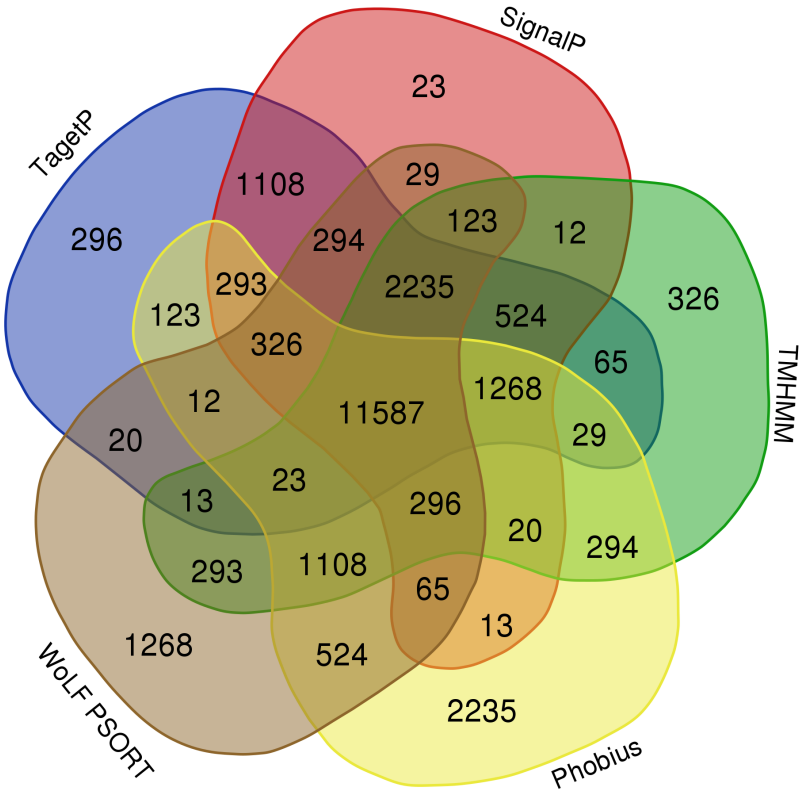


Figura 17 – **Diagrama de Venn do** $P.\ pachyrhizi$**.**

Para comparação, neste trabalho, também foi usado um conjunto de genomas da $G.\ max$ infectado pelo $P.\ pachyrhizi$. Esses dados foram otídios e podem ser acessados pelo NCBI com o código PRJEB44222 (<https://www.ncbi.nlm.nih.gov/bioproject/PRJEB44222>). No estudo citado o genoma do $G.\ max$ foi sequenciado em diversas horas, neste estudo foi usado apenas as horas 0 hora, 12 horas e 24 horas, acredita-se que os dados mais relevantes da infecção estejam nessas horas, segundo trabalhos anteriores (TREMBLAY et al., 2010; TREMBLAY et al., 2011; ALVES et al., 2015). Os programas utilizados nesta etapa são listados abaixo:

- **Trimmomatic-039:** segundo seu site "o Trimmomatic executa uma variedade de tarefas úteis de corte para iluminar dados de extremidade emparelhada e de extremidade única." (BOLGER; LOHSE; USADEL, 2014)
- **MagicBlast:** segundo seu site o "Magic-BLAST é uma ferramenta para mapear grandes sequenciamentos de RNA ou DNA de próxima geração corre contra um genoma inteiro ou transcriptoma." (BORATYN et al., 2018)
- **Santools:** segundo seu site o "Samtools é um conjunto de programas para interagir com dados de sequenciamento de alto rendimento." (PETR et al., 2021)
- **Kallisto:** segundo seu site o "kallisto é um programa para quantificar abundâncias de transcritos de dados de RNA-Seq em massa e de célula única, ou mais geralmente de sequências alvo usando leituras de sequenciamento de alto rendimento. Baseia-se na nova ideia de pseudoalinhamento para determinar rapidamente a compatibilidade de leituras com alvos, sem a necessidade para alinhamento." (BRAY et al., 2016)

Neste trabalho o Trimmomatic foi usado para filtrar as bases de dados, remover os dados não adequados para serem interpretados pelos outros programas; O MagicBlast para separar as proteínas que se assemelham a do banco e não se assemelham a ela; O Santools apenas para converter o arquivo de saída do MagicBlast para uma saída aceita pelo Kallisto; Por fim o Kallisto para realmente contar as semelhaças encontradas. As tabelas 2, 3, 4 e 5 apresentem os resultados encontrados.

Baseado nos programas citados acima (Trimmomatic, MagicBlast, Santool e Kallisto) e os dados usados por este trabalho as tabelas 2, 3, 4 e 5 foram criadas. Espera-se com esses experimentos maior compreensão dos dados estudados e uma melhor compreensão do fungo estudado.

A tabela 3 apresenta as leituras do fungo utilizadas pelo trabalho (nos dois experimentos) em comparação as leituras da infecção, as leituras presentes na infecção se demonstra pequena em relação a todas as leituras do fungo. O numero de leituras alinhadas também é menor para as proteínas de superfície (realmente usadas no grafo), porém apesar de pequeno, isto é o esperado. A tabela 3 apresenta as leituras que realmente participam da infecção da soja pelo fungo, é esperado que em número essas leituras sejam de fato menores que todas as leituras do fungo.

Tabela 2 – **Total de conexões do grafo.** Coluna 2 representa o numero total de proteínas encontradas no experimento. A coluna 3 o numero de proteínas semelhantes entre as encontradas no experimento e as proteínas usadas pele trabalho apresentado. A coluna 4 o numero de proteínas não semelhantes as encontradas no experimento e as proteínas usadas pele trabalho apresentado. Ultima linha representa a média de toda coluna superior.

	Total de leituras	Leituras alinhadas entre Soja e Experimento	Leituras não alinhadas entre Soja e Experimento
0h1	21067447	18273730	2214734
0h2	20559348	16375727	1815352
0h3	20334256	11135451	1707848
12h1	21258948	5746621	264066
12h2	20447013	6785939	1025102
12h3	21107512	5697552	1225357
24h1	20304069	16221161	3485863
24h2	20823517	17713955	2467726
24h3	20775820	17881510	2414992
Média	20.741.992,22	12.870.182,89	1.846.782,22

Tabela 3 – **Comparação com dados experimentais.** Proteínas encontradas no experimento semelhantes e não semelhantes as proteínas do fungo usadas nas suas especificações.

	Leituras alinhadas entre as do experimento e fungo	Leituras alinhadas entre as do experimento e fungo de superfície
0h1	188410	8
0h2	238208	13
0h3	243456	15
12h1	191936	11
12h2	509014	6
12h3	476977	15
24h1	242931	15
24h2	238942	21
24h3	136059	12
Média	273.992,56	12,89

	Leituras não alinhadas entre as do experimento e fungo	Leituras não alinhadas entre as do experimento e fungo superfície
0h1	20212533	20488391
0h2	17932299	18191023
0h3	19177086	19842831
12h1	19799532	20604259
12h2	19762986	20060036
12h3	19014079	20543022
24h1	18272313	19706954
24h2	19621461	20181564
24h3	19484356	20296434
Média	19.252.960,5	19.990.501,55

Tabela 4 – **Comparação com dados experimentais.** Leituras não alinhadas entre o experimento e a soja comparadas as leituras usadas por este trabalho.

	Leituras não alinhadas entre experimento e soja e alinhados a fungo	Leituras não alinhadas entre experimento e soja e alinhados a fungo de superfície
0h1	40976041	40976043
0h2	36380855	36380860
0h3	39684863	39684865
12h1	41207202	41207203
12h2	40118049	40118051
12h3	41085004	41085006
24h1	39411991	39411995
24h2	39521798	39331972
24h3	40591157	40591159
Média	39.886.328,8	39.865.239,3
	Leituras não alinhadas entre experimento e soja e não alinhados a fungo	Leituras não alinhadas entre experimento e soja e não alinhados a fungo de superfície
0h1	23	25
0h2	21	26
0h3	23	25
12h1	23	24
12h2	20	22
12h3	19	21
24h1	27	31
24h2	26	30
24h3	27	29
Média	23,2	25,8

A tabela 4 apresenta leituras da contaminação comparados ao dados do fungo usado nos dois experimentos. Pode-se ver que de fato os dados escolhidos representam quase a totalidade dos dados da infecção, a terceira coluna na tabela 4 possui um pequeno decaimento em relação a segunda coluna, dentre os dados alinhados, apresentando que de fato a maioria dos dados da infecção são de leituras do fungo de superfície.

5 Resultados

5.1 Nomenclatura dos dados

É comum os diversos portais usados usarem nomenclaturas diferentes para proteínas semelhantes. Por exemplo, a proteína keratin 84 pelo portal NCBI possui o simbolo KRT84. No String (portal utilizado) é feito uma busca inicial, mesmo que se utilize qualquer dos dois nomes proteicos (keratin 84 ou KRT84) o String retornará a mesma saída.

No software oferecido, as nomenclaturas utilizadas nos diversos portais não são relevantes na busca. O Diamond ou Blast utiliza apenas a sequência de nucleotídeos das proteínas, irrelevante ao nome dado a elas. O algoritmo gerador de grafos realiza comparações de nomes apenas entre proteínas de um mesmo portal, portanto, o portal pode utilizar qualquer nomenclatura. Os algoritmos usados após o algoritmo gerador de grafos usam apenas as nomenclaturas iniciais (neste estudo as proteínas da $G.\ max$ e do Phakospsora Pachyrhizi).

A irrelevância a nomenclatura utilizada permite que as bases de dados (sequências proteicas da $G.\ max$ e do Phakospsora Pachyrhizi, interações proteicas da String, BioGrid e Intact) sejam facilmente modificadas, atualizadas ou que outras bases de dados sejam adicionadas ao mesmo software.

5.2 Paralelismo e complexidade

Com o objetivo de acelerar o pipeline disponível, foi proposta uma paralelização do mesmo. Basicamente para realizar a paralelização, os arquivos de entrada do algoritmo de filtragem e do algoritmo gerador de grafo (os algoritmos mais lentos do pipeline) são divididos no número especificado de threads, para cada thread o algoritmo é executado em paralelo.

O algoritmo de filtragem e o algoritmo gerador de grafos ambos possuem apenas tarefas independentes, ou seja, suas tarefas internas não dependem de outras tarefas internas executadas, por isso eles podem ser paralelizados, como uma tarefa independe da outra elas podem ser executadas em paralelo, em threads distintas.

Para testar a implementação com threads, o pipeline foi testado em Human (dados obtidos no site String [https://string-db.org/] em 2023) e Phakopsora Pachyrhizi (dados obtidos no site String [https:// string-db.org/] em 2023), e conexões com o banco de dados String resumido (dados obtidos no site String [https://string-db.org/] em 2023) e em um computador Aspire 3 com 8 núcleos, processador Ryzen 7 e 8 GB de memória RAM. A figura 18 apresenta o resultado desta aplicação em relação a redução do tempo de processamento gasto.

A figura 18 mostra o tempo gasto pelo pipeline com o aumento do número de threads, cada ponto representa a média de 3 execuções, como pode ser visto o tempo diminui com o aumento de threads conforme o esperado, o tempo, entretanto, não diminui de forma constante, possivelmente porque o pipeline também executa outros programas sem implementação da paralelização.

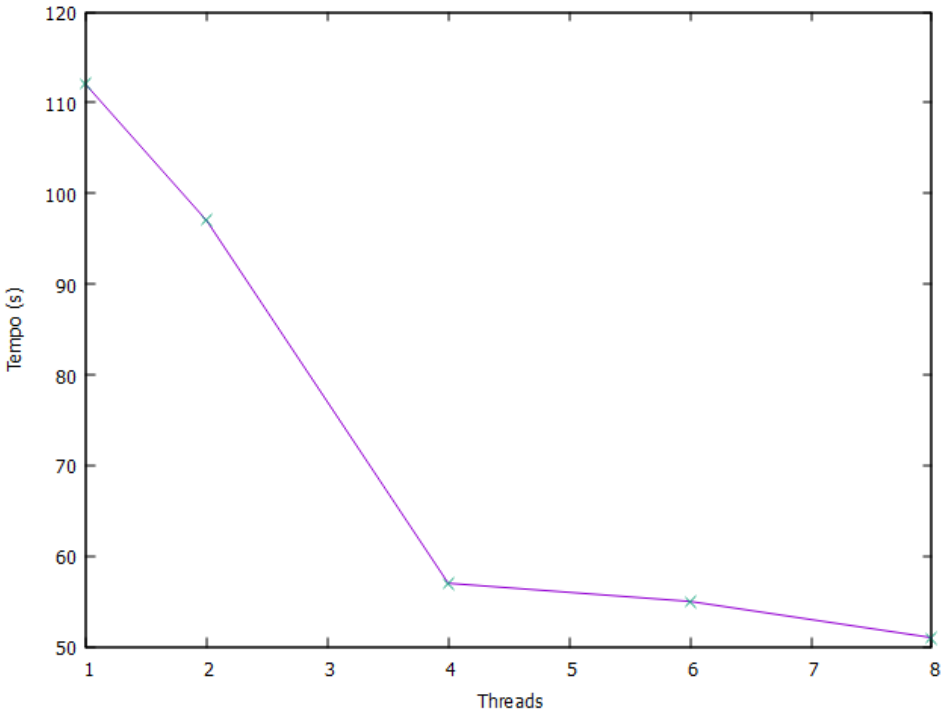


Figura 18 – **Aumento do número de Threads:** eixo x representa o número de threads e eixo y representa o tempo de processamento gasto.

5.3 Características do grafo gerado

Para este estudo foram gerados 5 grafos apresentados e listados abaixo:

1. **Intact:** para todas as proteínas obtidas do $G.\ max$ e do $P.\ pachyrhizi$, foi criado um grafo utilizando a metodologia apresentada usando o banco de dados Intact. O grafo obtido com o Intact possui 428 vértices e 922 arestas. Um exemplo visual do grafo gerado pelo Cytoscape pode ser visto na figura 19.

2. **Biogrid:** para todas as proteínas obtidas do $G.\ max$ e do $P.\ pachyrhizi$, foi criado um grafo utilizando a metodologia apresentada usando o banco de dados Biogrid. O grafo obtido com o Biogrd possui 5172 vértices e 23073 arestas. Um exemplo visual do grafo gerado pelo Cytoscape pode ser visto na figura 20.

3. **String resumido:** para todas as proteínas obtidas do $G.\ max$ e do $P.\ pachyrhizi$, foi criado um grafo utilizando a metodologia apresentada usando o banco de dados String

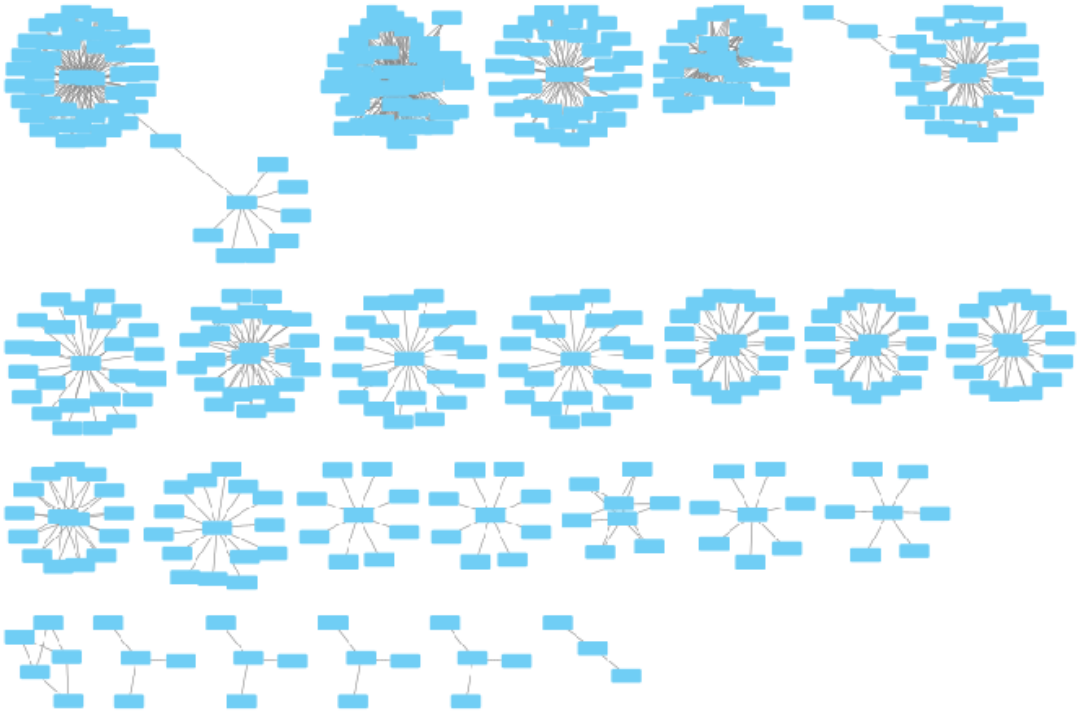


Figura 19 – **Grafo gerado com o banco de dados Intact**

com as 9 espécies mais relevantes do mesmo. O grafo obtido com o String resumido possui 6879 vértices e 700343 arestas. Um exemplo visual do grafo gerado pelo Cytoscape pode ser visto na figura 21.

4. **União dos 3 bancos de dados:** para todas as proteínas obtidas do $G.\ max$ e do $P.\ pachyrhizi$, foi criado um grafo utilizando a metodologia apresentada usando o banco de dados Intact, Biogrig e String com as 9 espécies mais relevantes. O grafo obtido com a união dos bancos de dados possui 7213 vértices e 72211 arestas. Um exemplo visual do grafo gerado pelo Cytoscape pode ser visto na figura 25.

5. **String completo:** para todas as proteínas obtidas do $G.\ max$ e apenas as proteínas consideradas de superfície do $P.\ pachyrhizi$, como foi apresentado na seção 4.2, foi criado um grafo utilizando a metodologia apresentada usando o banco de dados String com todas as suas espécies, como foi apresentado na seção **??**. O grafo obtido com a união dos bancos de dados possui 815 vértices e 3403 arestas. Um exemplo visual do grafo gerado pelo Cytoscape pode ser visto na figura 28.

Neste trabalho, nesta seção, será apresentado e discutido apenas os grafos 4 (união dos bancos de dados) e 5 (String completo).

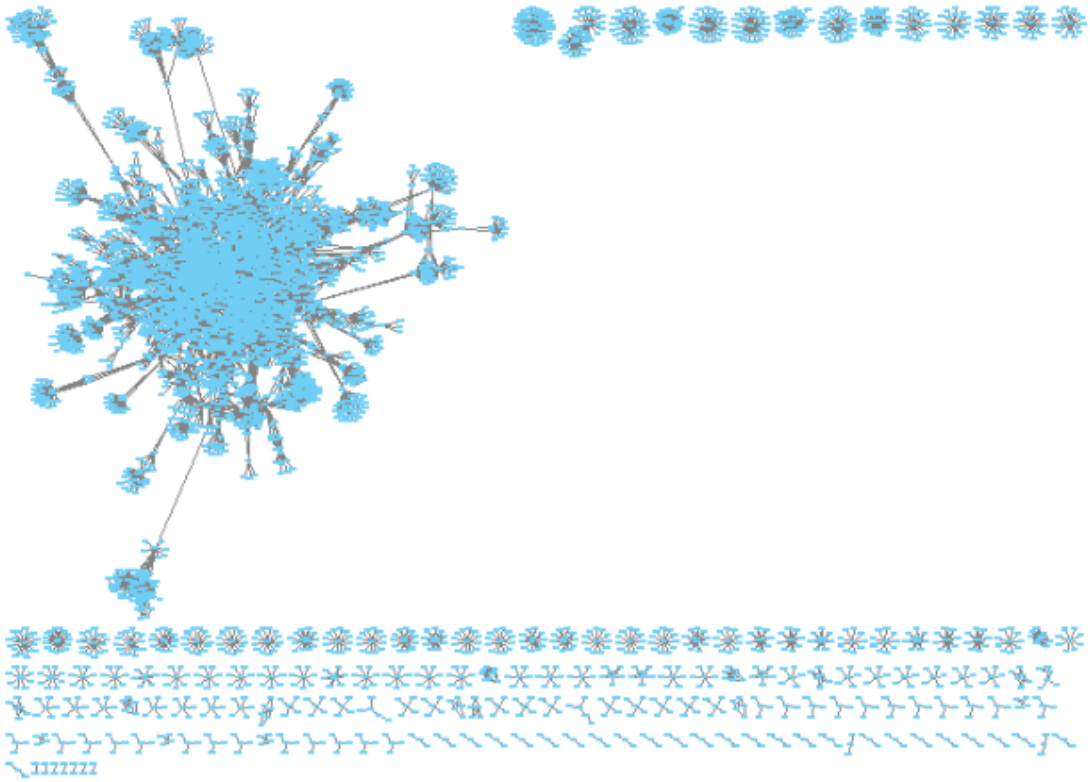


Figura 20 – **Grafo gerado com o banco de dados Biogrid.**

5.3.1 Grafo com a união dos bancos de dados

Como este estudo de caso envolve contaminação, pode ser interessante conhecer a funcionalidade das proteínas encontradas pela rede, para permitir, por exemplo, a criação de novas estratégias de defesa. Além de TMHMM, SignalP e TargetP, foi utilizado outro programa, EggNOG, segundo seu site: "EggNOG-mapper é uma ferramenta para anotação funcional rápida de novas sequências. Ele usa grupos ortólogos (OGs) pré-computados e filogenias do banco de dados EggNOG para transferir informações funcionais apenas de ortólogos refinados."

A saída do EggNOG pode ser baixada no link: https://github.com/gustavocarnivali /Files.git

As informações obtidas sobre o grafo de acordo com o Cytoscape podem ser vistas nas imagens 22, 23 e 24. Especificamente, na imagem imagem 22 podem ser vistas diversas informações sobre o grafo, na imagem 23 a distribuição de graus do grafo, onde um padrão de redes livres de escala pode ser visto, na imagem 24 você pode ver a distribuição de intermediação. Outras informações também produzidas pelo Cytoscape podem ser vistas na imagem 27.

Conforme informado ao longo do trabalho, após a criação do grafo, TMHMM, SignalP e TargetP são implementados ao grafo, especificamente o grafo que representa a união dos bancos de dados foi utilizado nestes programas, a saída para download dos 3 programas pode ser obtida por este link: https://github.com/gustavocarnivali/Files.git. No entanto, as saídas serão descritas abaixo:

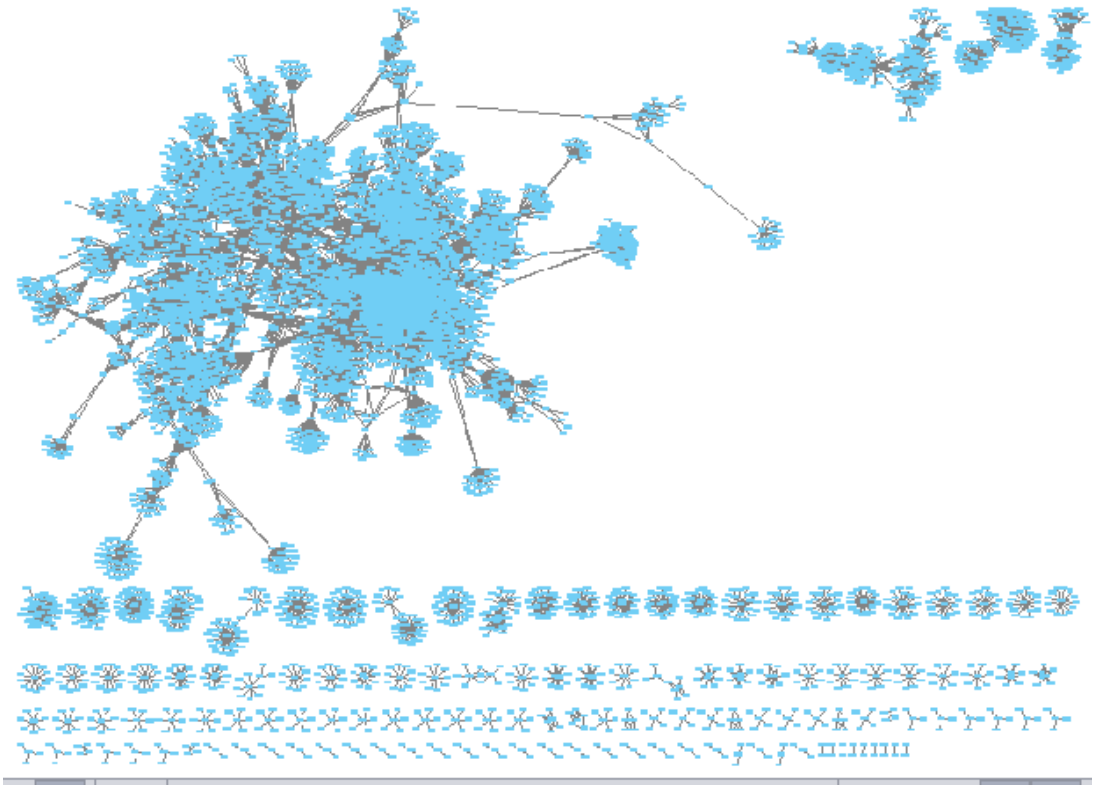


Figura 21 – **Grafo gerado com o banco de dados String**

Summary Statistics	
Number of nodes	7310
Number of edges	72211
Avg. number of neighbors	21,443
Network diameter	17
Network radius	9
Characteristic path length	5,768
Clustering coefficient	0,000
Network density	0,003
Network heterogeneity	2,220
Network centralization	0,067
Connected components	83
Analysis time (sec)	26,457

Figura 22 – **Informações sobre o grafo.** Na língua padrão do Cytoscape.

- **SignalP:** de acordo com SignalP 7111 proteínas possuem OUTRA localização e 115 proteínas possuem localização SP(Sec/SPI). (legenda no mesmo site).
- **TMHMM:** de acordo com SignalP 6606 proteínas possuem topologia o e 607 proteínas possuem topologia i. (legenda no mesmo site).
- **TargetP:** de acordo com TargetP 6523 proteínas têm localização noTP, 444 proteínas têm

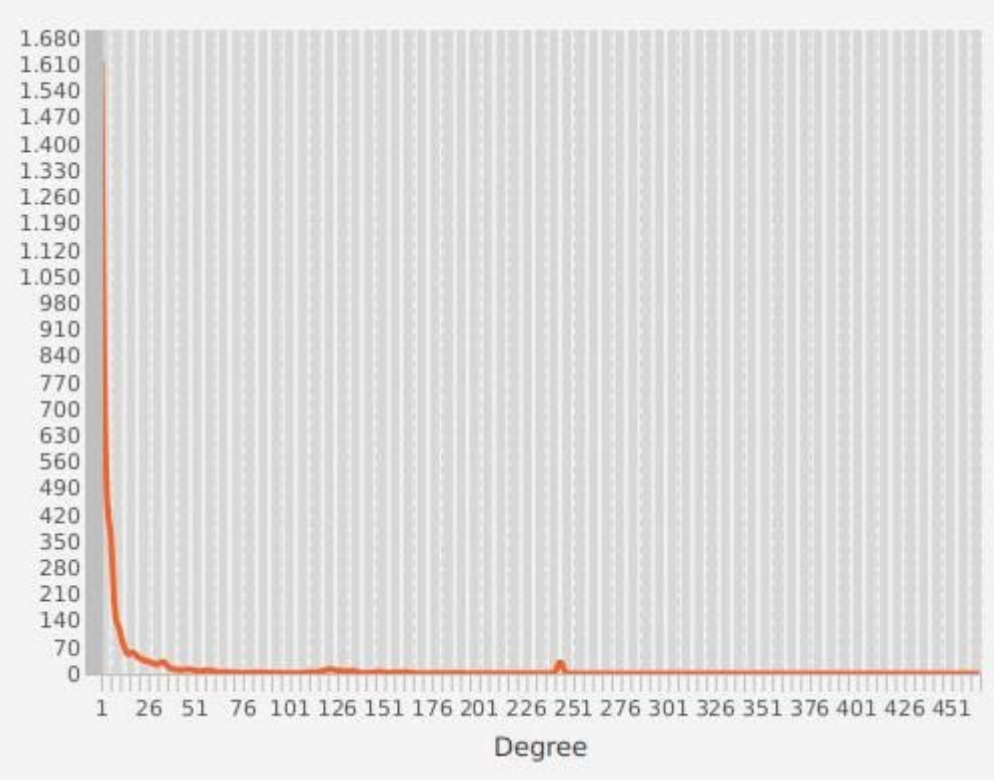


Figura 23 – **Distribuição de grau do grafo produzido com a união dos 3 bancos de dados.**

localização mTP e 263 proteínas têm localização SP. (legenda no mesmo site).

O algoritmo de Louvain a partir do Cytoscape foi aplicado a este grafo. O resultado gráfico pode ser visto na Figura 26. O algoritmo de louvain para esse grafo gerou 444 comunidades de vértices altamente conectados.

O arquivo de comunidades está disponibilizado. Mas por exemplo a comunidade 444 possui esta lista de vértices: "$GlymaFiskIII.20G160400.1.p$, $GlymaFiskIII.20G160400.2.p$, $GlymaFiskIII.20G160400.3.p$, $jgi|PhapaK8108|3147559|estExt_Genewise1$", é possível ver vértices da soja ($GlymaFiskIII$) e do fungo ($jgi|PhapaK$) na mesma comunidade, indicando possíveis proteínas semelhantes ou que participão da contaminação.

5.3.2 Grafo com String completo

O grafo produzido com o String completo será discutido e apresentado nesta seção. O grafo completo gerado pode ser visto na imagem 28

O EggNOG também foi aplicado a este grafo. Como anteriormente, sua saida para este grafo também pode ser baixada pelo link: https://github.com/gustavocarnivali/Files.git.

Informações diversas sobre o grafo podem ser vistas nas imagens 29, 30, 31. È possível ver uma distribuição parecida com a do grafo anterior, considerando que os dois grafos foram

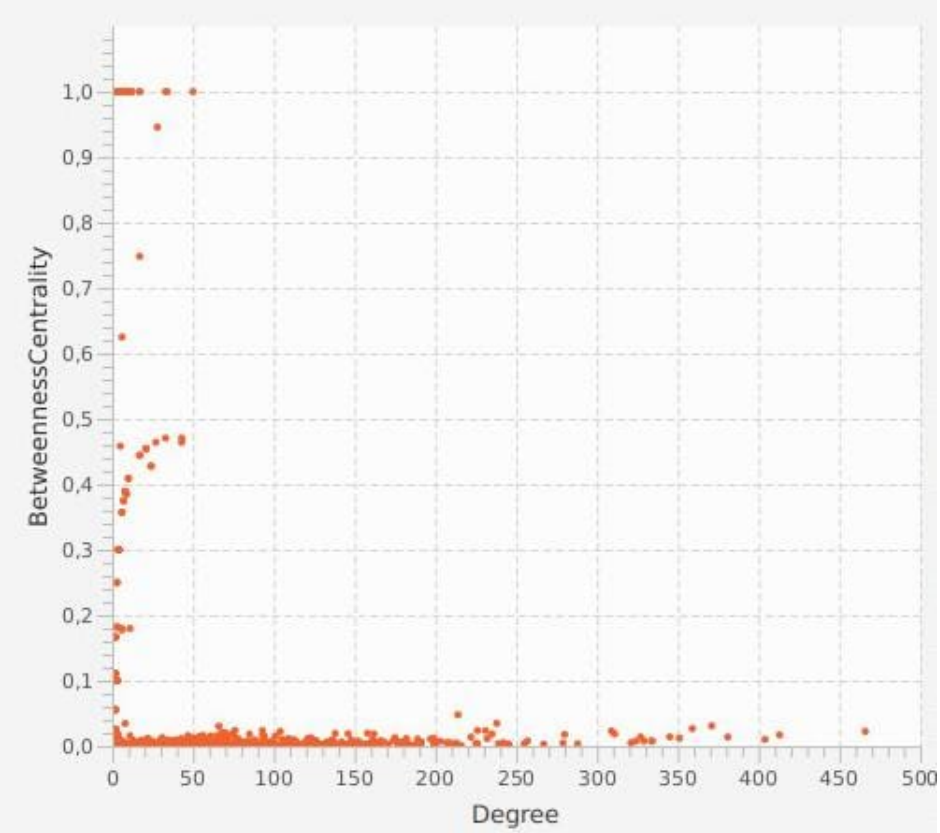


Figura 24 – **Betweenness do grafo produzido com a união dos 3 bancos de dados.**

feitos com dados semelhantes. Outras informações também produzidas pelo Cytoscape podem ser vistas na imagem 33.

Após a criação do grafo, como anteriormente, para melhor conhecer também este grafo, o algoritmo de detecção de comunidades Louvain foi aplicado a este grafo. A saída do algoritmo de Louvain pode ser visto na figura 32. A primeira e maior comunidade possui 697 vértices, a segunda maior possui 167 vértices e a ultima maior apenas 2 vértices, mostrando que o grafo pode apresentar particularidades que devem ser analisadas.

Outra comunidade de exemplo possui os vértices: $GlymaFiskIII.01G191500.2.p$, $GlymaFiskIII.04G108900.2.p$, $GlymaFiskIII.07G045700.1.p$, $GlymaFiskIII.13G110100.1.p$, $jgi|PhapaK8108|7409588|fgenesh1_pm.tig00032074_6309$. Com vértices da soja e do fungo, portanto eles podem ser mais estudados, considerando que eles demonstram uma possível relação entre as especies estudadas.

A tabela 5 apresenta uma relação entre os dados do grafo gerado e, aqui aprentado, e os dados experimentais do genoma da soja especoficados na seção anterior. Pode-se ver um alto valor entre as leituras não alinhadas, ou seja, ah muitas leituras encontradas no grafo que não participam da infecção, podem ser leituras de funcionalidades corporais parecidas entre as duas especies por exemplo. Porém a tabela 5 não apresenta os dados alinhados zerados, isso

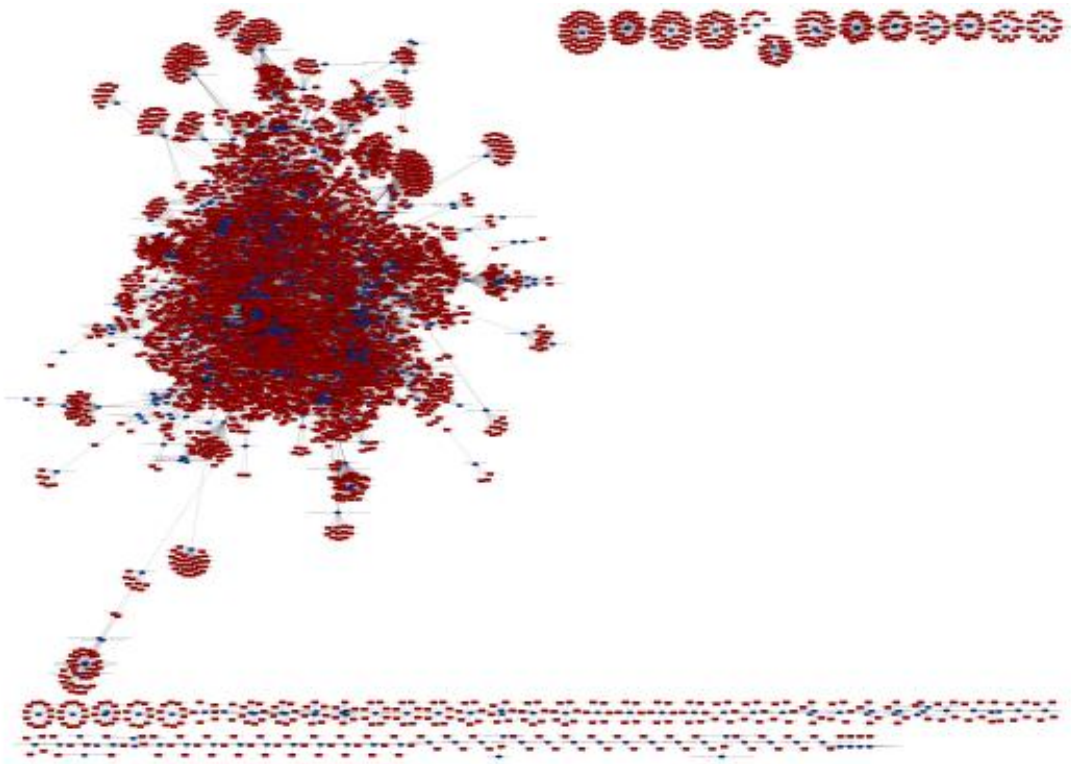


Figura 25 – **Grafo com os três bancos de dados**: gerado pelo Cytoscape, quadrados vermelhos representam proteínas da soja e círculos azuis proteínas do fungo.

Tabela 5 – **Relação grafo e fungo.** Número de leituras semelhante e definidos como não semelhantes, pelos programas utilizados neste trabalho, entre o grafo encontrado pelo nosso trabalho e a totalidade de leituras do fungo.

	Leituras não alinhadas entre experimento e soja e	
	alinhadas ao grafo	não alinhadas ao grafo
0h1	7	20488446
0h2	3	18191058
0h3	5	19842866
12h1	11	20604298
12h2	13	20060064
12h3	5	20543045
24h1	11	19707001
24h2	14	20181652
24h3	6	20296471
Média	8,33	19.990.544,5

representa que de fato, nesse contexto, o grafo apresentou leituras da infecção mas o grafo como um todo deve ser melhor estudado caso se deseje apenas esses dados.

Figura 26 – **Comunidades com o grafo dos três bancos de dados**: gerado pelo Cytoscape.

5.4 Disponibilidade do código

Todas as bases de dados usadas podem ser acessadas via intenet e estão disponíveis para download (no momento em que este artigo foi escrito). Todo pipeline está contido em um único arquivo executável de linux $.sh$. O mesmo pode ser facilmente baixado, executado e modificado se assim o usuário desejar. Também todos os programas criados assim como o ".sh"podem ser modificados e usados em outros estudos.

O código com os programas implementados (FilterBD, GraphGenerate, Score e SubTitle) pode ser baixado no link: https://github.com/gustavocarnivali/PPINP. O pipeline completo com os programas implementados, os programas utilizados (DIAMOND, TMHMM, SignalP e TargetP), bem como os exemplos de inputs podem ser baixados no link: https://drive.google.com/file/d/1iCsUqDoIvrgzX
5a2T405/view?usp=compartilhamento. Todos os programas utilizados são licenciados para uso gratuito, suas licenças estão disponíveis em seus respectivos sites.

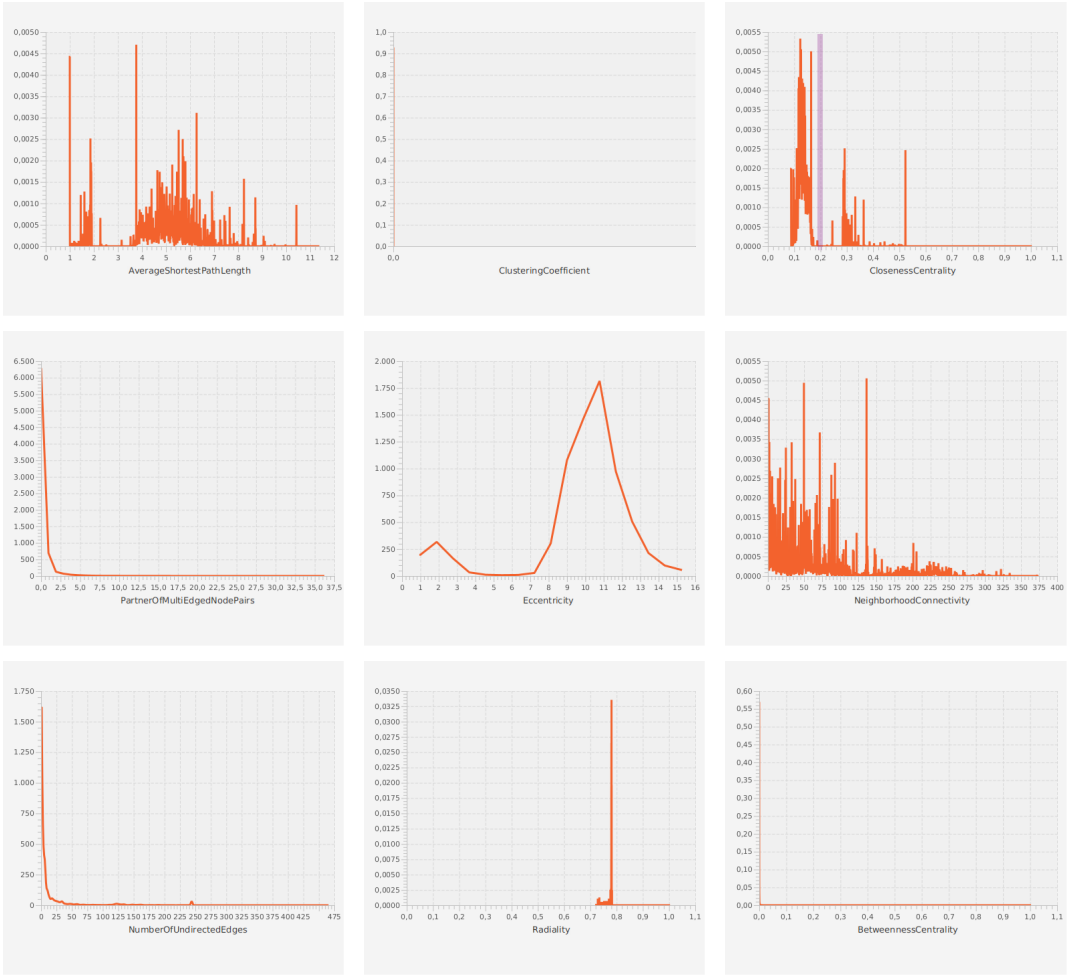


Figura 27 – **Outras informações sobre o grafo produzido com união três bancos de dados.** Na língua padrão do Cytoscape. Informações sobre o que representa cada grafo esta descrito abaixo do mesmo.

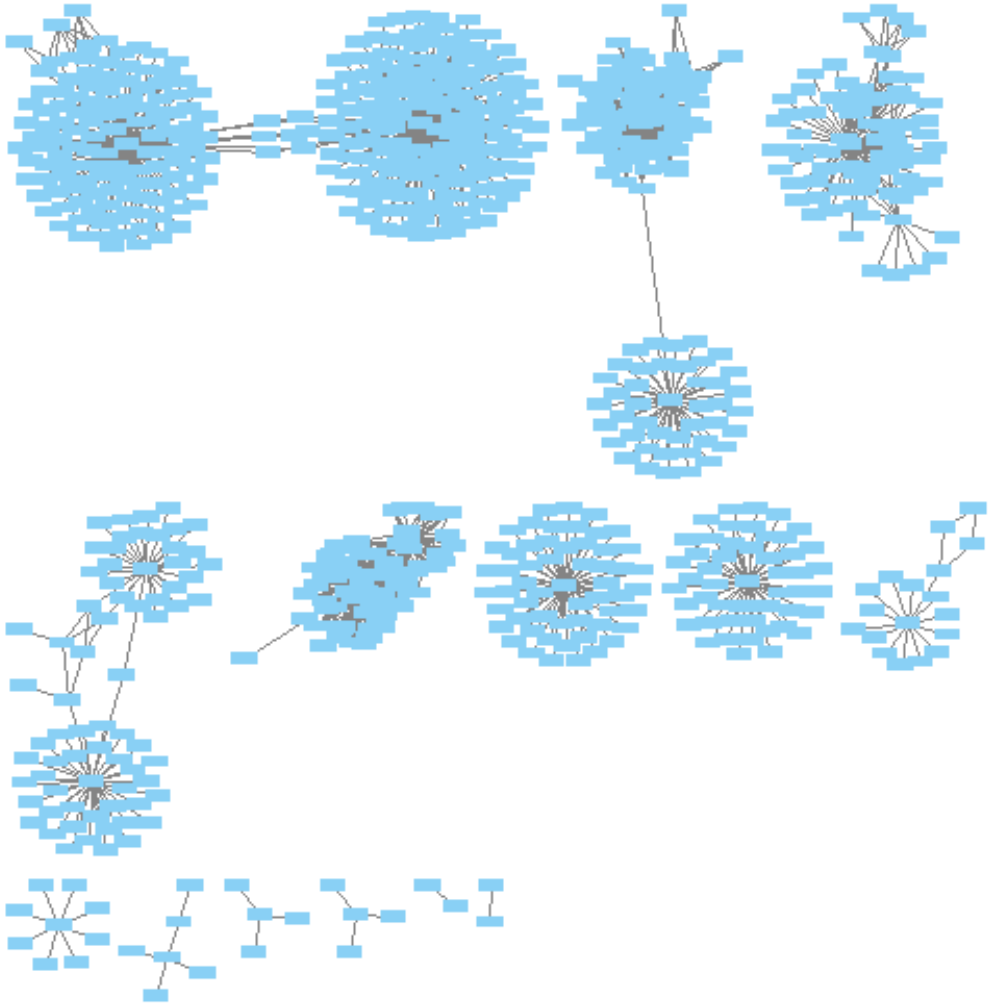


Figura 28 – **Grafo com o String completo**: gerado pelo Cytoscape, quadrados vermelhos representam proteínas da soja e círculos azuis proteínas do fungo.

Summary Statistics	
Number of nodes	815
Number of edges	3403
Avg. number of neighbors	2,094
Network diameter	5
Network radius	3
Characteristic path length	2,941
Clustering coefficient	0,000
Network density	0,007
Network heterogeneity	5,872
Network centralization	0,581
Connected components	14
Analysis time (sec)	0,152

Figura 29 – **Informações sobre o grafo produzido com o String completo.** Na língua padrão do Cytoscape.

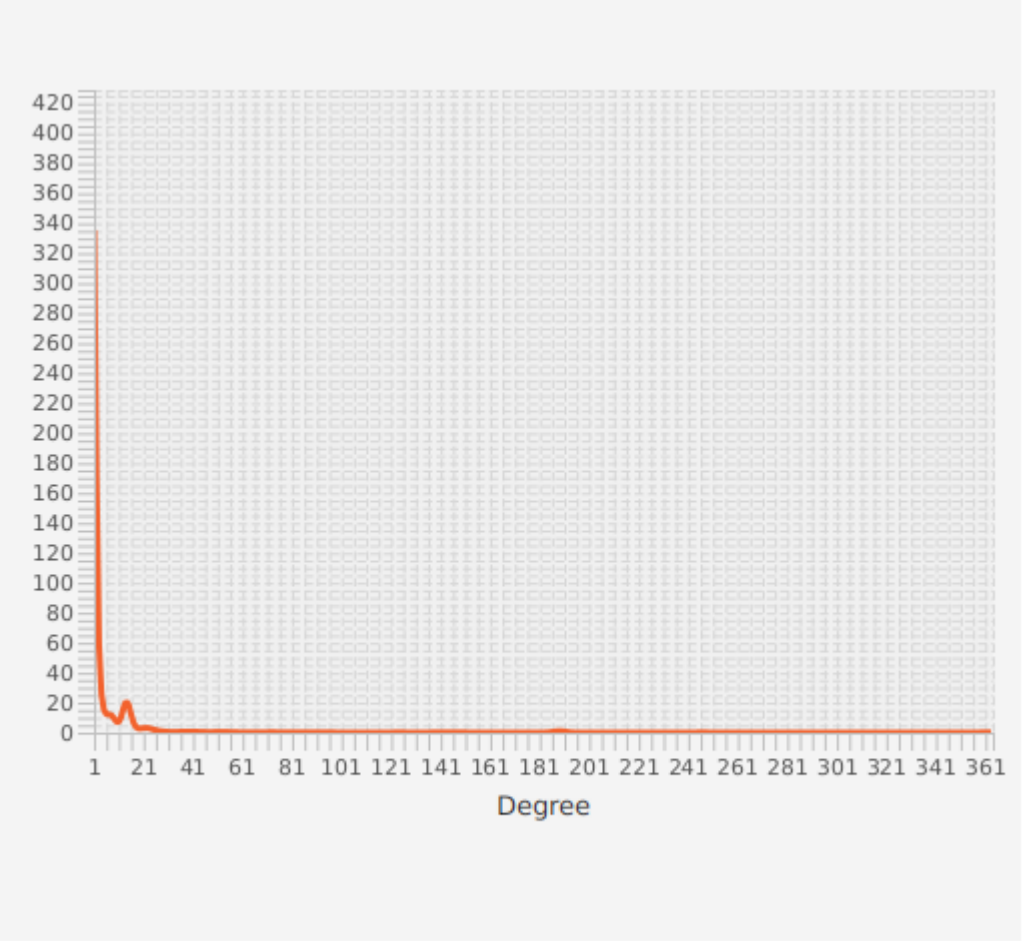


Figura 30 – **Distribuição de grau do grafo produzido com o String completo.**

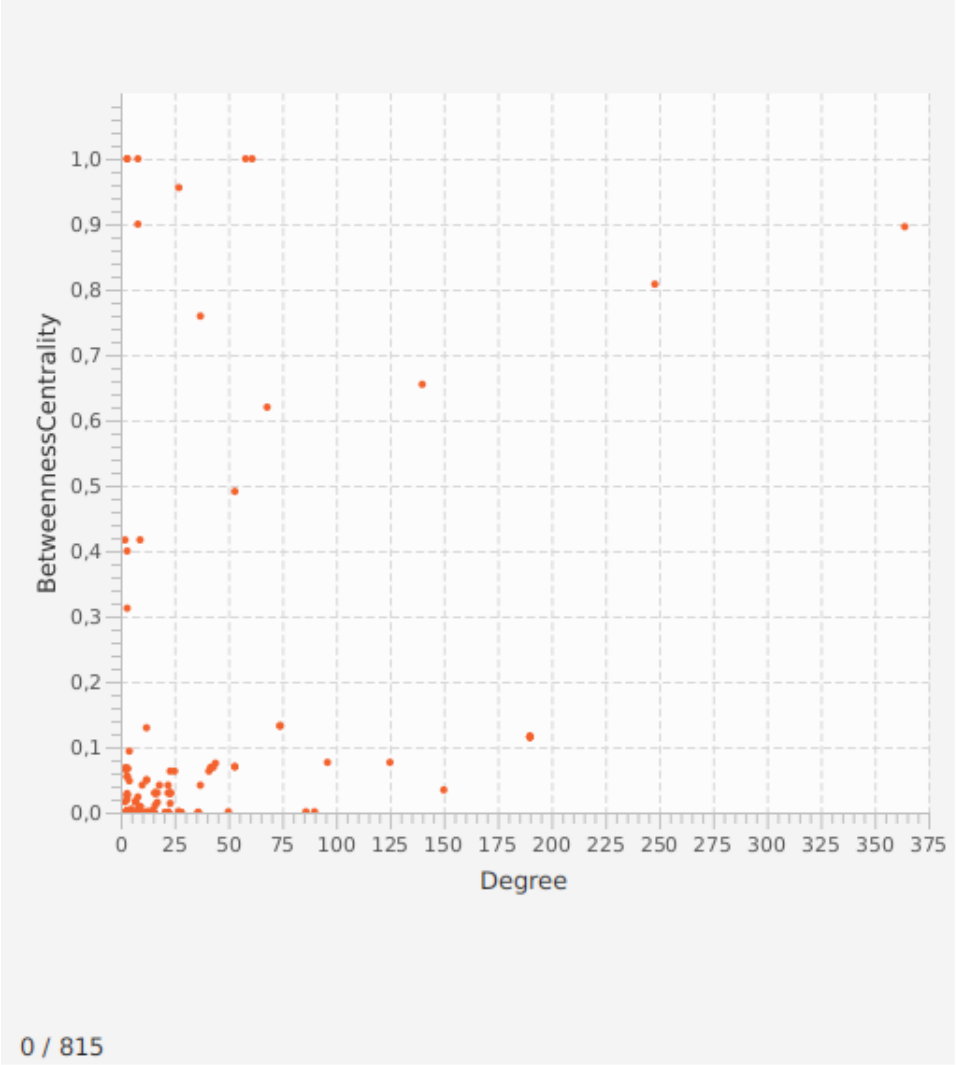


Figura 31 – **Betweenness do grafo produzido com o String completo.**

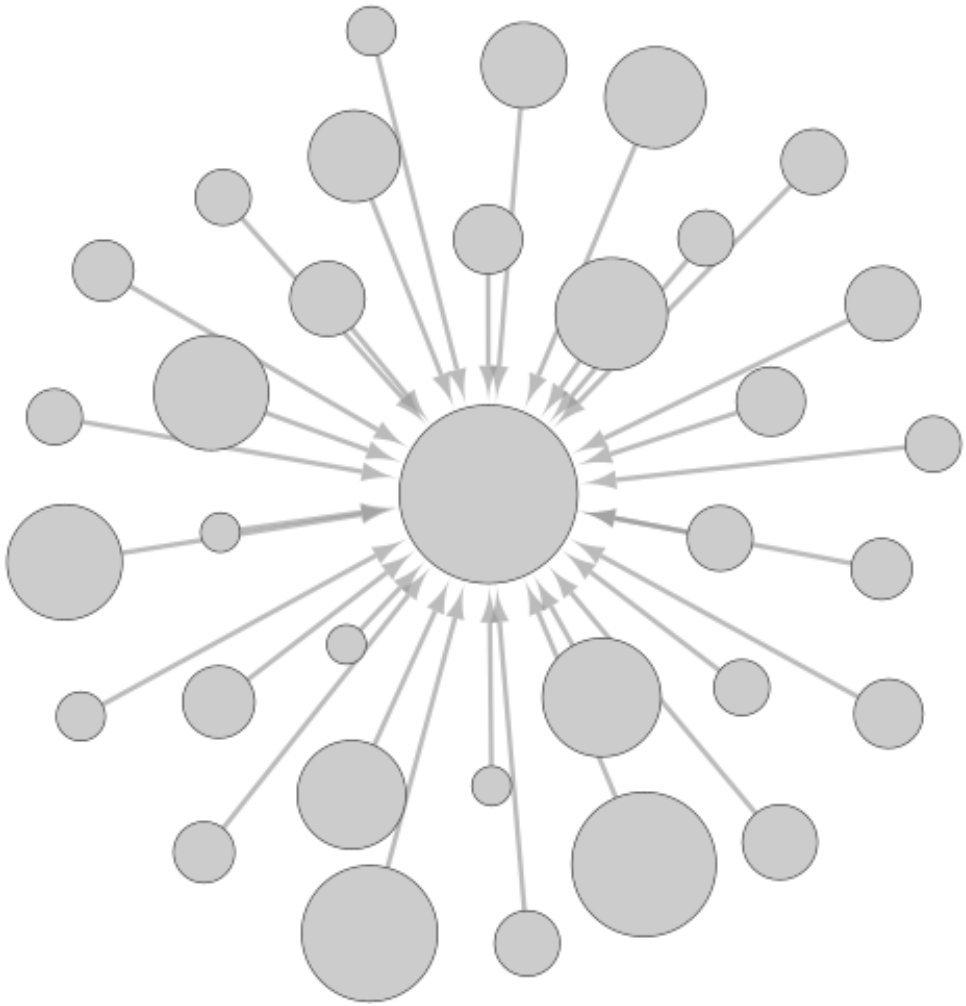


Figura 32 – **Informações sobre o grafo produzido com o String completo.** Na língua padrão do Cytoscape.

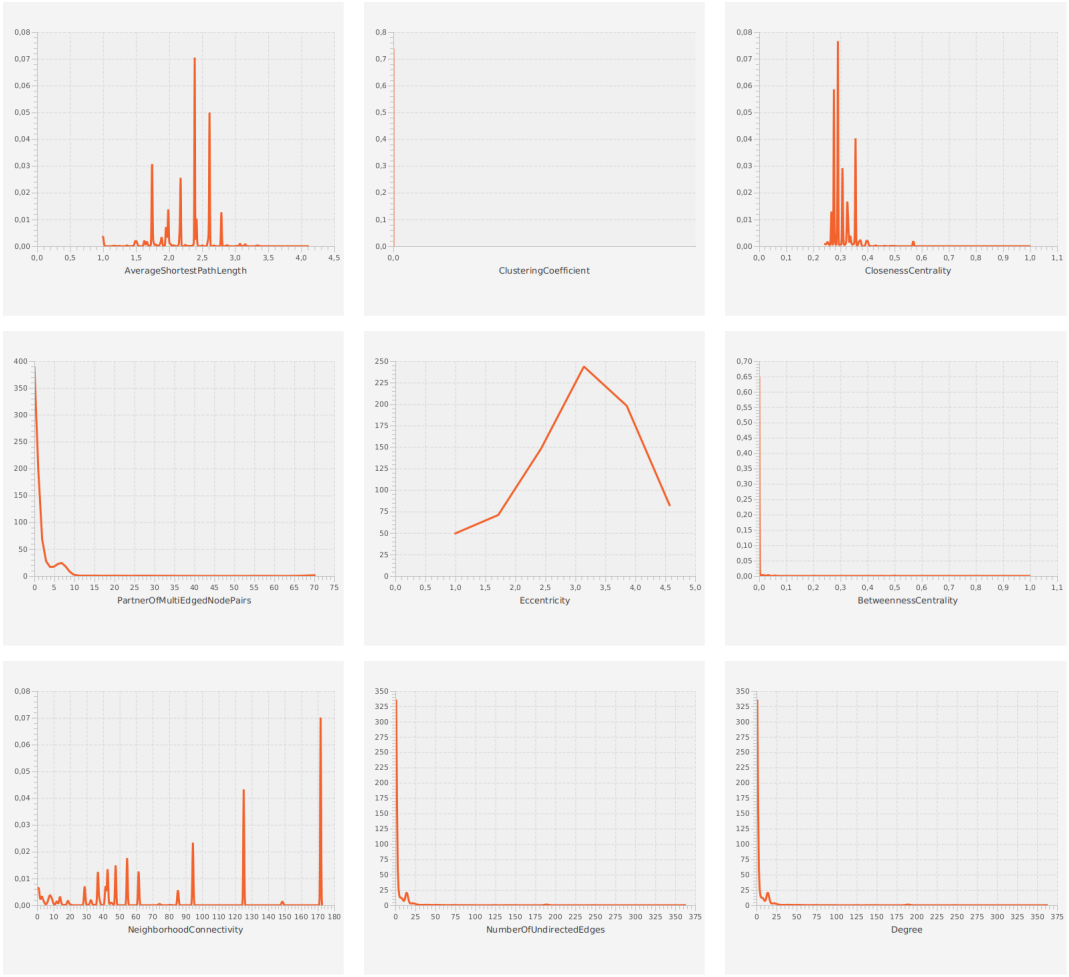


Figura 33 – **Outras informações sobre o grafo produzido com o String completo.** Na língua padrão do Cytoscape. Informações sobre o que representa cada grafo esta descrito abaixo do mesmo.

6 Discussão

Foi proposto uma ferramenta que descobre novos relacionamentos proteicos entre duas espécies diferentes. Para isso, inicialmente, foi apresentado as espécies escolhidas, e também, os portais em que os dados sobre essas espécies foram obtidos. Por serem dados comuns (sequenciamento gênico), de duas espécies comuns ($G.\ max$ e Phakospsora Pachyrhizi), podeu-se usar portais habituais e conhecidos.

Após isso, foi apresentado ferramentas que foram usadas para analisar as conexões geradas. Foi possível uma otimização da velocidade em diversas ferramentas usadas. Para acelerar o analisador de sequências proteicas foi usado o Diamond; Para acelerar o detector de comunidades foi usado o COVEC; Para acelerar o algoritmo mais lento produzido por este trabalho, foi usado apenas a paralelização discutida na seção 5.2 e devido a características deste algoritmo, foi obtido um ganho substancial de velocidade apresentada na seção 5.2.

Com o método desenvolvido foi possível obter o desejável: proteínas diferentes, de espécies diferentes, que se conectaram ou, pelo menos, possuem substancial semelhança ou conexão funcional. Especificamente, neste trabalho, foi analisado proteínas da $G.\ max$ e do $P.\ pachyrhizi$ mas, outras proteínas das mesmas ou de outras espécies podem ser usadas no mesmo método.

Toda metodologia usando o String, Biogrid, Intact e DIAMOND conseguiu encontrar semelhanças em proteínas diversas, mas, após a obtenção dessas conexões, foi aplicado a essas proteínas o SignalP, o TMHMM e o TargetP. Foram encontradas proteínas conectadas por esses dois conjuntos de ferramentas (String, Biogrid, Intact ou DIAMOND e SignalP, TMHMM ou TargetP), isto apresenta uma alta conexão ou semelhança entre essas proteínas.

Utilizando o detector de comunidades Louvain, comunidades com proteínas da soja e do fungo foram encontradas, em especifico a comunidade com as proteínas: GlymaFiskIII.01G191500, GlymaFiskIII.04G108900, GlymaFiskIII.13G110100, jgi|PhapaK8108| 7409588|fgenesh1pm.tig000320746309. Um detalhamento sobre á funcionalidade de duas proteínas será apresentado.

- GlymaFiskIII.01G191500: dominio DNAJ (segundo o phytozome)
- GlymaFiskIII.04G108900: dominio MATH (segundo o phytozome)

Portanto o grafo gerado apresenta, além das interações proteicas do soja e fungo, uma aproximação entre os dominios proteicos: "DNAJ"e "MATH".

Referências

ALVES, M. S. et al. Differential expression of four soybean bzip genes during phakopsora pachyrhizi infection. *Functional & integrative genomics*, Springer, v. 15, p. 685–696, 2015. Citado na página 44.

AMARAL, F.; LADEIRA, R. Barabási e as redes: Uma infraestrutura dinâmica para a complexidade. *Revista Inteligência Empresarial*, 2021. Citado na página 36.

ANDRADE, P. J. M.; ANDRADE, D. d. A. Ferrugem asiática: uma ameaça à sojicultura brasileira. Dourados: Embrapa Agropecuária Oeste; Chapadão do Sul: Fundação Chapadão, 2002., 2002. Citado na página 29.

APROSOJA, B. *Associação dos Produtores de Soja do Brasil*. 2014. Urlhttp://aprosojabrasil.com.br/2014/sobre-a-soja/a-historia-da-soja/. Citado na página 18.

ARORA, S. et al. Floating drug delivery systems: a review. *Aaps PharmSciTech*, Springer, v. 6, n. 3, p. E372–E390, 2005. Citado na página 18.

BARABÁSI, A.-L. et al. *Network science*. [S.l.]: Cambridge University Press, 2016. Citado 2 vezes nas páginas 26 e 28.

BARBERA, G. L. et al. Proteomic analysis and bioluminescent reporter gene assays to investigate effects of simulated microgravity on caco-2 cells. *Proteomics*, Wiley Online Library, v. 17, n. 15-16, p. 1700081, 2017. Citado 2 vezes nas páginas 10 e 24.

BARROS, L. et al. Genótipos de soja de ciclo semi precoce/médio quanto à doenças fungicas foliares e caracteres agronômicos. *ENCICLOPÉDIA BIOSFERA*, v. 7, n. 12, 2011. Citado na página 20.

BASSETT, D. S. et al. Dynamic reconfiguration of human brain networks during learning. *Proceedings of the National Academy of Sciences*, National Academy of Sciences, 2011. Citado na página 28.

BERAHMAND, K. et al. Spectral clustering on protein-protein interaction networks via constructing affinity matrix using attributed graph embedding. *Computers in Biology and Medicine*, Elsevier, v. 138, p. 104933, 2021. Citado na página 16.

BOLGER, A. M.; LOHSE, M.; USADEL, B. Trimmomatic: a flexible trimmer for illumina sequence data. *Bioinformatics*, Oxford University Press, v. 30, n. 15, p. 2114–2120, 2014. Citado na página 44.

BORATYN, G. M. et al. Magic-blast, an accurate dna and rna-seq aligner for long and short reads. *BioRxiv*, Cold Spring Harbor Laboratory, p. 390013, 2018. Citado na página 44.

BRAR, G. S.; JR, T. E. C. Soybean: Glycine max (l) merrill. *Genetic improvement of vegetable crops*, Elsevier, p. 427–463, 1993. Citado na página 18.

BRAY, N. L. et al. Near-optimal probabilistic rna-seq quantification. *Nature biotechnology*, Nature Publishing Group US New York, v. 34, n. 5, p. 525–527, 2016. Citado na página 44.

BUCHFINK, B.; REUTER, K.; DROST, H.-G. Sensitive protein alignments at tree-of-life scale using diamond. *Nature methods*, Nature Publishing Group, v. 18, n. 4, p. 366–368, 2021. Citado na página 35.

BUCHFINK, B.; XIE, C.; HUSON, D. H. Fast and sensitive protein alignment using diamond. *Nature methods*, Nature Publishing Group, v. 12, n. 1, p. 59–60, 2015. Citado na página 35.

BUENO, T. V. Caracterização de efetores de phakopsora pachyrhizi e de seus alvos de interação na soja. Universidade Federal de Viçosa, 2021. Citado na página 30.

CANTALAPIEDRA, C. P. et al. eggnog-mapper v2: functional annotation, orthology assignments, and domain prediction at the metagenomic scale. *Molecular biology and evolution*, Oxford University Press, v. 38, n. 12, p. 5825–5829, 2021. Citado na página 37.

CARNIVALI, G. S. et al. Método rápido de agrupamento de vértices para detecçao de comunidades em redes complexas de larga-escala. In: SBC. *Anais do XVII Workshop em Desempenho de Sistemas Computacionais e de Comunicação*. [S.l.], 2018. Citado na página 37.

CARNIVALI, G. S. et al. Covec: coarse-grained vertex clustering for efficient community detection in sparse complex networks. *Information Sciences*, Elsevier, v. 522, p. 180–192, 2020. Citado 2 vezes nas páginas 41 e 42.

CARVALHO, M. C. d. C. de et al. Prediction of the in planta phakopsora pachyrhizi secretome and potential effector families. *Molecular plant pathology*, Wiley Online Library, v. 18, n. 3, p. 363–377, 2017. Citado na página 37.

CHANG, J.-M. et al. Efficient and interpretable prediction of protein functional classes by correspondence analysis and compact set relations. *PloS one*, Public Library of Science San Francisco, USA, v. 8, n. 10, p. e75542, 2013. Citado na página 15.

CHATR-ARYAMONTRI, A. et al. The biogrid interaction database: 2017 update. *Nucleic acids research*, Oxford University Press, v. 45, n. D1, p. D369–D379, 2017. Citado 2 vezes nas páginas 15 e 16.

CONAB. *Acompanhamento da Safra Brasileira de Grãos.* 2014. Urlhttp://www.conab.gov.br/OlalaCMS/uploads/arquivos/$14_0 6_1 0_1 2_1 2_3 7_b oletim_g raos_j unho_2 014.pdf.Citado$

CORDY, D. *The protein book*. [S.l.]: Naturegraph Books, 1976. Citado na página 15.

COSTA, P. M. Caracterização e expressão transiente de genes de phakopsora pachyrhizi que codificam proteínas secretadas. Universidade Federal de Viçosa, 2010. Citado na página 30.

DEPAMPHILIS, M. L. et al. *DNA replication in eukaryotic cells.* [S.l.]: Cold Spring Harbor Laboratory, 1996. Citado na página 21.

DIAS, R. X. L. Rede de interação proteína-proteína na relação entre rickettsia amblyommatis e células do intestino e ovário de amblyomma sculptum (berlese, 1888). Universidade Federal de Viçosa, 2020. Citado na página 30.

DIMITROV, D. S. Therapeutic proteins. *Therapeutic Proteins*, Springer, p. 1–26, 2012. Citado na página 15.

ELMORE, M. G. et al. De novo transcriptome of phakopsora pachyrhizi uncovers putative effector repertoire during infection. *Physiological and Molecular Plant Pathology*, Elsevier, v. 110, p. 101464, 2020. Citado na página 37.

EMANUELSSON, O. et al. Locating proteins in the cell using targetp, signalp and related tools. *Nature protocols*, Nature Publishing Group, v. 2, n. 4, p. 953–971, 2007. Citado na página 16.

EMBRAPA. *Embrapa Soja divulga balanço sobre a ferrugem na safra de 2007/08*. 2008. Urlhttp://www.cnpso.embrapa.br/noticia/ver$_noticia.php?cod_noticia = 469.Citadonapágina20$.

EMPRABA. O genoma altamente complexo do fungo p. pachyrhizi causador da mais devastadora doença da soja foi decifrad por um consórcio único de empresas públicas e privadas. *Nota Técnica*, Emoraba, 2019. Citado na página 35.

EMPRABA, S. *Empresa Brasileira de Pesquisa Agropecuári*. 2014. Urlhttp://www.cnpso.embrapa.br/index.php?op$_page = 22cod_pai = 16.Citadonapágina18$.

ENRIGHT, A. J. et al. Protein interaction maps for complete genomes based on gene fusion events. *Nature*, Nature Publishing Group, v. 402, n. 6757, p. 86–90, 1999. Citado na página 15.

ENRIGHT, A. J.; OUZOUNIS, C. A. Functional associations of proteins in entire genomes by means of exhaustive detection of gene fusions. *Genome biology*, BioMed Central, v. 2, n. 9, p. 1–7, 2001. Citado na página 22.

EULER, L. Leonhard euler and the königsberg bridges. *Scientific American*, JSTOR, v. 189, n. 1, p. 66–72, 1953. Citado na página 26.

FEOFILOFF, P.; KOHAYAKAWA, Y.; WAKABAYASHI, Y. Uma introdução sucinta à teoria dos grafos. 2011. Citado 3 vezes nas páginas 10, 26 e 27.

FLÓREZ, A. F. et al. Protein network prediction and topological analysis in leishmania major as a tool for drug target selection. *BMC bioinformatics*, BioMed Central, v. 11, n. 1, p. 1–9, 2010. Citado 3 vezes nas páginas 15, 30 e 40.

FRANCESCHINI, A. et al. String v9. 1: protein-protein interaction networks, with increased coverage and integration. *Nucleic acids research*, Oxford University Press, v. 41, n. D1, p. D808–D815, 2012. Citado 2 vezes nas páginas 22 e 42.

FURTADO, G. Q. et al. Influência do estádio fenológico e da idade dos trifólios de soja na infecção de phakopsora pachyrhizi. *Tropical Plant Pathology*, SciELO Brasil, v. 34, p. 118–122, 2009. Citado na página 30.

GAVIN, A.-C. et al. Functional organization of the yeast proteome by systematic analysis of protein complexes. *Nature*, Nature Publishing Group, v. 415, n. 6868, p. 141–147, 2002. Citado na página 15.

HARRINGTON, E. D.; JENSEN, L. J.; BORK, P. Predicting biological networks from genomic data. *FEBS letters*, Elsevier, v. 582, n. 8, p. 1251–1258, 2008. Citado na página 15.

HE, F. et al. The prediction of protein-protein interaction networks in rice blast fungus. *BMC genomics*, BioMed Central, v. 9, n. 1, p. 1–12, 2008. Citado na página 15.

HELING, A. et al. Monitoramento de phakopsora pachyrhizi na safra 2020/2021 para tomada de decisão do controle da ferrugem-asiática da soja. Londrina: Embrapa Soja, 2021., 2021. Citado na página 16.

HERMJAKOB, H. et al. Intact: an open source molecular interaction database. *Nucleic acids research*, Oxford University Press, v. 32, n. suppl_1, p. D452–D455, 2004. Citado 3 vezes nas páginas 16, 23 e 30.

HOFREE, M. et al. Network-based stratification of tumor mutations. *Nature methods*, Nature Publishing Group, v. 10, n. 11, p. 1108–1115, 2013. Citado na página 21.

HUYNEN, M. A.; BORK, P. Measuring genome evolution. *Proceedings of the National Academy of Sciences*, National Acad Sciences, v. 95, n. 11, p. 5849–5856, 1998. Citado na página 15.

KHULLER, S.; RAGHAVACHARI, B. Graph and network algorithms. *ACM Computing Surveys (CSUR)*, ACM, v. 28, n. 1, p. 43–45, 1996. Citado na página 28.

KIRILOV, A. *Introdução a Teoria de Conjuntos*. [S.l.]: 2017, 2017. Citado 2 vezes nas páginas 40 e 41.

KORF, I.; YANDELL, M.; BEDELL, J. *Blast*. [S.l.]: "O'Reilly Media, Inc.", 2003. Citado 2 vezes nas páginas 16 e 35.

LANCICHINETTI, A.; FORTUNATO, S. Community detection algorithms: a comparative analysis. *Physical review E*, APS, v. 80, n. 5, p. 056117, 2009. Citado na página 37.

LESKOVEC, J.; LANG, K. J.; MAHONEY, M. Empirical comparison of algorithms for network community detection. In: *Proceedings of the 19th international conference on World wide web*. [S.l.: s.n.], 2010. p. 631–640. Citado na página 37.

LIU, S. et al. A computational interactome for prioritizing genes associated with complex agronomic traits in rice (oryza sativa). *The Plant Journal*, Wiley Online Library, v. 90, n. 1, p. 177–188, 2017. Citado na página 15.

LULLI, A. et al. Distributed current flow betweenness centrality. In: IEEE. *2015 IEEE 9th International Conference on Self-Adaptive and Self-Organizing Systems*. [S.l.], 2015. p. 71–80. Citado na página 37.

MA, J. et al. Using deep learning to model the hierarchical structure and function of a cell. *Nature methods*, Nature Publishing Group, v. 15, n. 4, p. 290–298, 2018. Citado na página 21.

MAERE, S.; HEYMANS, K.; KUIPER, M. Bingo: a cytoscape plugin to assess overrepresentation of gene ontology categories in biological networks. *Bioinformatics*, Oxford University Press, v. 21, n. 16, p. 3448–3449, 2005. Citado na página 37.

MARCOTTE, E. M. et al. Detecting protein function and protein-protein interactions from genome sequences. *Science*, American Association for the Advancement of Science, v. 285, n. 5428, p. 751–753, 1999. Citado na página 15.

MEO, P. D. et al. Generalized louvain method for community detection in large networks. In: IEEE. *2011 11th international conference on intelligent systems design and applications*. [S.l.], 2011. p. 88–93. Citado na página 37.

MORGAN, B. P.; HARRIS, A. L. *Complement regulatory proteins*. [S.l.]: Academic Press, 1999. Citado na página 21.

NASIRI, E. et al. A novel link prediction algorithm for protein-protein interaction networks by attributed graph embedding. *Computers in Biology and Medicine*, Elsevier, v. 137, p. 104772, 2021. Citado na página 16.

NAVARINI, L. et al. Controle químico da ferrugem asiática (phakopsora pachyrhizi sidow) na cultura da soja. *Summa phytopathologica*, SciELO Brasil, v. 33, p. 182–186, 2007. Citado na página 20.

NETTO, P. O. B. *Grafos: teoria, modelos, algoritmos*. [S.l.]: Editora Blucher, 2003. Citado na página 36.

NEWMAN, M. E. The structure and function of complex networks. *SIAM review*, SIAM, v. 45, n. 2, p. 167–256, 2003. Citado na página 28.

NIK-ZAINAL, S. et al. Landscape of somatic mutations in 560 breast cancer whole-genome sequences. *Nature*, Nature Publishing Group, v. 534, n. 7605, p. 47–54, 2016. Citado na página 22.

OLIVEIRA, A. d.; ROSA, A. Indicações técnicas para a cultura da soja no rio grande do sul e em santa catarina, safras 2014/2015 e 2015/2016. *Pelotas: Embrapa Clima Temperado*, 2014. Citado na página 20.

OUGHTRED, R. et al. The biogrid interaction database: 2019 update. *Nucleic acids research*, Oxford University Press, v. 47, n. D1, p. D529–D541, 2019. Citado na página 21.

PETR, D. et al. Twelve years of samtools and bcftools. *Gigascience10*, 2021. Citado na página 44.

PHIZICKY, E. M.; FIELDS, S. Protein-protein interactions: methods for detection and analysis. *Microbiological reviews*, Am Soc Microbiol, v. 59, n. 1, p. 94–123, 1995. Citado na página 15.

PIEROZZI, P. H. B. et al. New soybean (glycine max fabales, fabaceae) sources of qualitative genetic resistance to asian soybean rust caused by phakopsora pachyrhizi (uredinales, phakopsoraceae). *Genetics and Molecular Biology*, SciELO Brasil, v. 31, p. 505–511, 2008. Citado na página 30.

POLLARD, T. D. et al. *Cell Biology E-Book*. [S.l.]: Elsevier Health Sciences, 2016. Citado na página 21.

REZENDE, A. M. et al. Computational prediction of protein-protein interactions in leishmania predicted proteomes. *PloS one*, Public Library of Science San Francisco, USA, v. 7, n. 12, p. e51304, 2012. Citado na página 30.

RODRIGUES, B. et al. Correlations between traits in soybean (glycine max l.) naturally infected with asian rust (phakopsora pachyrhizi). *Genet. Mol. Res*, v. 14, n. 4, p. 17718–17729, 2015. Citado na página 30.

SAHNI, N. et al. Widespread macromolecular interaction perturbations in human genetic disorders. *Cell*, Elsevier, v. 161, n. 3, p. 647–660, 2015. Citado 2 vezes nas páginas 21 e 22.

SATO, T. et al. The inference of protein–protein interactions by co-evolutionary analysis is improved by excluding the information about the phylogenetic relationships. *Bioinformatics*, Oxford University Press, v. 21, n. 17, p. 3482–3489, 2005. Citado na página 15.

SEIXAS, C. D. S. et al. Monitoramento de phakopsora pachyrhizi na safra 2017/2018 para tomada de decisão do controle da ferrugem-asiática da soja. *Londrina: Embrapa Soja*, 2018. Citado na página 29.

SILVA, F. et al. *Soja: do plantio à colheita*. [S.l.]: Oficina de Textos, 2022. Citado na página 18.

SINCLAIR, J.; HARTMAN, G. Management of soybean rust. In: COLLEGE OF AGRICULTURAL, CONSUMER AND ENVIRONMENTAL SCIENCES URBANA. *Soybean Rust Workshop*. [S.l.], 1995. p. 6–10. Citado na página 20.

SINGHAL, A. et al. Multiscale community detection in cytoscape. *PLoS computational biology*, Public Library of Science San Francisco, CA USA, v. 16, n. 10, p. e1008239, 2020. Citado na página 37.

SKRABANEK, L. et al. Computational prediction of protein–protein interactions. *Molecular biotechnology*, Springer, v. 38, n. 1, p. 1–17, 2008. Citado na página 15.

SNEL, B.; BORK, P.; HUYNEN, M. A. The identification of functional modules from the genomic association of genes. *Proceedings of the National Academy of Sciences*, National Acad Sciences, v. 99, n. 9, p. 5890–5895, 2002. Citado na página 22.

SOARES, R. M. et al. Fungicidas no controle da ferrugem asiática (phakopsora pachyrhizi) e produtividade da soja. *Ciência rural*, SciELO Brasil, v. 34, p. 1245–1247, 2004. Citado na página 20.

STRATTON, M. R.; CAMPBELL, P. J.; FUTREAL, P. A. The cancer genome. *Nature*, Nature Publishing Group, v. 458, n. 7239, p. 719–724, 2009. Citado na página 23.

STUDHAM, M. E. et al. Functional association networks as priors for gene regulatory network inference. *Bioinformatics*, Oxford University Press, v. 30, n. 12, p. i130–i138, 2014. Citado na página 22.

SZKLARCZYK, D. et al. String v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. *Nucleic acids research*, Oxford University Press, v. 47, n. D1, p. D607–D613, 2019. Citado 6 vezes nas páginas 15, 16, 22, 23, 30 e 42.

TÖRÖNEN, P.; MEDLAR, A.; HOLM, L. Pannzer2: a rapid functional annotation web server. *Nucleic acids research*, Oxford University Press, v. 46, n. W1, p. W84–W88, 2018. Citado na página 37.

TREMBLAY, A. et al. Transcriptome analysis of a compatible response by glycine max to phakopsora pachyrhizi infection. *Plant Science*, Elsevier, v. 179, n. 3, p. 183–193, 2010. Citado na página 44.

TREMBLAY, A. et al. Gene expression in leaves of susceptible glycine max during infection with phakopsora pachyrhizi using next generation sequencing. *Sequencing*, Hindawi, v. 2011, 2011. Citado na página 44.

TSUKAHARA, R. Y.; HIKISHIMA, M.; CANTERI, M. G. Relações entre o clima e o progresso da ferrugem asiática (phakopsora pachyrhizi) em duas micro-regiões do estado do paraná. *Semina: Ciências Agrárias*, Universidade Estadual de Londrina, v. 29, n. 1, p. 47–52, 2008. Citado na página 31.

USDA. *Projeções*. 2008. Urlhttp://www.fas.usda.gov. Citado na página 18.

VASCONCELOS, C. R. dos S.; CAMPOS, T. de L.; REZENDE, A. M. Building protein-protein interaction networks for leishmania species through protein structural information. *BMC bioinformatics*, BioMed Central, v. 19, n. 1, p. 1–13, 2018. Citado na página 30.

WAQAS, M. K. et al. Dermatological and cosmeceutical benefits of glycine max (soybean) and its active components. *Acta Pol Pharm*, v. 72, n. 1, p. 3–11, 2015. Citado na página 18.

XENARIOS, I. et al. Dip: the database of interacting proteins. *Nucleic acids research*, Oxford University Press, v. 28, n. 1, p. 289–291, 2000. Citado na página 30.

YEGER-LOTEM, E.; SHARAN, R. Human protein interaction networks across tissues and diseases. *Frontiers in genetics*, Frontiers Media SA, v. 6, p. 257, 2015. Citado na página 21.

YORINORI, J. T.; JÚNIOR, J. N.; LAZZAROTTO, J. J. Ferrugem"asiática"da soja no brasil: evolução, importância econômica e controle. *Embrapa Soja-Fôlder/Folheto/Cartilha (INFOTECA-E)*, Londrina: Embrapa Soja, 2004., 2004. Citado na página 20.

ZANZONI, A. et al. Mint: a molecular interaction database. *FEBS letters*, Wiley Online Library, v. 513, n. 1, p. 135–140, 2002. Citado na página 30.

ZHANG, B.; HORVATH, S. A general framework for weighted gene co-expression network analysis. *Statistical applications in genetics and molecular biology*, De Gruyter, v. 4, n. 1, 2005. Citado na página 21.

Information Sciences 522 (2020) 180–192

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

CoVeC: Coarse-grained vertex clustering for efficient community detection in sparse complex networks☆

Gustavo S. Carnivali[a], Alex B. Vieira[c,*], Artur Ziviani[b], Paulo A.A. Esquef[b]

[a] Universidade Federal de Minas Gerais (UFMG), Brazil
[b] National Laboratory for Scientific Computing (LNCC), Brazil
[c] Universidade Federal de Juiz de Fora (UFJF), Brazil

ARTICLE INFO

Article history:
Received 11 July 2019
Revised 4 January 2020
Accepted 2 March 2020
Available online 2 March 2020

Keywords:
Graphs
Community detection
Louvain algorithm

ABSTRACT

This paper tackles the problem of community detection in large-scale graphs. In the literature devoted to this topic, an iterative algorithm, called Louvain Method (LM), stands out as an effective and fast solution for this problem. However, the first iterations of the LM are the most costly. To overcome this issue, this paper introduces CoVeC, a *Co*arse-grained *Ve*rtex *C*lustering for efficient community detection in sparse complex networks. CoVeC pre-processes the original graph in order to forward a graph of reduced size to the LM. The subsequent group formation, including the maximization of group quality, as per the modularity metric, is left to the LM. We evaluate our proposal using real-world and synthetic networks, presenting distinct sizes and sparsity levels. Overall, our experimental results show that CoVeC can be a way faster option than the first iterations of the LM, yet similarly effective. In fact, for sparser graphs, the combo CoVeC+LM outperforms the standalone LM and its variations, attaining a mean processing time reduction of 47% and a mean modularity reduction of only 0.4%.

1. Introduction

In a simple way, a graph is a structure composed of vertices that can be connected by edges, indicating a relation between a pair of vertices. Conveniently, graphs can represent various real-world relationships [1]. Actually, large-scale graphs can represent complex systems or large-scale networked services, such as Facebook and Twitter, or other common services in domains such as WWW [2,3].

In several real complex networks, the distribution of edges between the vertices is highly heterogeneous. This leads, in certain cases, to a high concentration of edges within groups of vertices and a low concentration of edges between distinct groups. This characteristic of real networks is typically called a community structure. Communities are groups of vertices that probably share some common proprieties or perform similar positions in the studied graph. In this context, considering the analysis of graphs representing real complex networks, a common problem is community detection [4–7], which is the main focus of this article.

☆ This work was funded in part by research project grants from CAPES, CNPq, FAPERJ, and FAPESP.

* Corresponding author.
E-mail addresses: carnivali@bioinfo.dout.ufmg.br (G.S. Carnivali), alex.borges@ufjf.edu.br (A.B. Vieira), ziviani@lncc.br (A. Ziviani), pesquef@lncc.br (P.A.A. Esquef).

https://doi.org/10.1016/j.ins.2020.03.004
0020-0255/

The detection of communities plays a very important role in analyzing the structure of complex networks. For instance, it helps in identifying and visualizing the internal structure of the network, detecting potentially useful information, and mine the relationships between individuals [8]. Note that many complex networks tend to be organized into communities, i.e., tightly-knit modules of nodes. Identifying these communities by only using the information encoded in the network topology is a challenge and an important task [9].

Considering examples from different disciplines, we can observe that communities often play important roles in the organization of networks. For example, in online social networks, communities correspond to groups of friends who attended the same school, or neighborhood, or even share common interests. In a protein-protein interaction network, groups (or communities) of proteins are those that functionally interact with each other, possibly contributing to the same cellular function. In brain networks of interconnected neurons, communities can correspond to specialized functional components, such as visual and auditory systems. In this sense, detecting network communities allows us to discover functionally related objects study interactions between modules, infer missing attribute values, and predict unobserved connections [10].

Currently, the scale of complex networks is incredibly large. For example, the WWW has an estimated size of over one trillion documents (10^{12} pages), being the Web the largest network humanity has ever built [11]. It exceeds in size even the human brain (10^{11} neurons). In other words, a high-speed and high-quality community detection algorithm is crucial for the analysis of the current large-scale networks.

In this sense, the community detection task in graphs consists of finding groups of vertices that have one or more characteristics in common. For example, the neighborhood vertices share may be used as a characteristic to define a community. In this case, intuitively, a community will have a number of vertices that are well connected with each other. When talking about community detection, clustering may be used as a synonym for it. Nevertheless, clustering is a more general concept. As expected, there are several methodologies—and variations—to detect communities in graphs. Each methodology presents its strengths and applicability. In the same way, there is a number of metrics to evaluate how a community is a well representative [12] (i.e., how the members of a community are similar). In this work, we consider community detection methodologies based on *modularity*. More precisely, *modularity* is an important metric, commonly used to evaluate the quality of a community structure in a graph [13,14]. Intuitively, the modularity is a metric that shows how densely connected are the vertices of a given group or community. The higher/lower the value of the modularity, the more/less connected are the vertices of a given community, when compared to a random distribution of edges interconnecting these vertices.

The community detection problem can then be formalized as a modularity maximization problem from the partitions of a graph. It has been proved that modularity optimization is an NP-complete problem [15]. In other words, it is probably impossible to find an optimal solution in a time growing polynomially with the size of the graph. However, there are currently several methods to detect fairly good approximations of the modularity maximum in a reasonable time [16,17]. Among these methods, the Louvain Method (LM) [18] stands out for quickly and efficiently identifying communities (i.e, with high modularity) in large-scale graphs. The LM has been widely adopted in practice because of its speed and high quality of results [19]. In fact, until the present day, the LM continues to be one of the most widely used tools for serial community detection [20].

LM thus employs an iterative heuristic based on modularity maximization. Initially, the LM assigns each vertex of the graph to a different community. In its first stage, for all vertices of the graph, the LM checks, in a greedy way, whether there is any gain in modularity by changing the membership of vertex i from the community it is actually belonging (c_i) to a neighbor community (c_j). Vertex i is then moved to c_j in the case there is a gain on the modularity. This process is repeated for each vertex until there is no gain of modularity.

The second stage of the LM simply generates a new (reduced) graph where the new single vertices are the communities discovered by the previous stage. Here, a run of the first and second stages will be called an LM cycle. Starting a new LM cycle, the reduced graph is fed to the first stage, so that new communities that maximize modularity are detected. LM cycles continue while there is a gain of modularity.

The LM has computational complexity $O(n \log n)$ [18], where n is the total number of vertices in the graph. The LM initiates with a graph of n vertices and, after each LM cycle, the trend is towards graphs with a smaller number of vertices (communities). Therefore, the first LM cycle accounts for most of the overall computational cost. In the literature, preprocess the original graph via fast algorithms is often used as a way to speed up this method. For example, one may eliminate redundant edges to offload the first step of the LM. However, most of the existing methods may not be efficient under certain conditions (e.g. in sparse networks) or may reflect the real communities of the network, as we will discuss in Section 2.

In this paper, we propose CoVeC, a Coarse-Grained Vertex Clustering method. CoVeC pre-processes the original graph in order to forward a graph of reduced size to the LM. CoVeC does a similar job as the first costly round of the LM, i.e., to generate a reduced graph with communities still close to the original graph structure. The subsequent maximization of quality, as per the modularity metric, is left to the LM. As a consequence, we show that the hybrid community detector CoVeC+LM tends to outperform LM in terms of execution time, at the cost of a slight reduction in the modularity of the found communities.

We evaluate the performance of the proposed combo CoVeC+LM by comparing it with the original LM, as well as with two other enhancements of the LM process: *Fire-Forest* and *Local Sparsification* methods (related work is discussed in Section 2). Our evaluations rely on large-scale real-world, as well as synthetic, networks. Networks we evaluate present distinct sizes and sparsity. Overall, our experimental results show that CoVeC can be a way faster option than the first iter-

ations of the LM, yet similarly effective. In fact, for sparser graphs, CoVeC outperforms the LM and its variations, attaining a mean time reduction of 47% and mean modularity (quality) reduction of only 0.4%.

The remainder of this article is organized as follows: first, we present the related work in Section 2. Then, in Section 3, we present a detailed description of the CoVeC and the necessary modifications to the LM. In Section 4, we present our evaluation methodology and the performance of the CoVeC. Moreover, we evaluate the sensibility of the CoVeC to the choice of its parameters. Finally, in Section 5, we conclude our work.

2. Related work

Despite the low complexity of the LM, using it to detect communities in graphs with a very large number of vertices can be computationally costly. To alleviate this problem, there are several ways to reduce the runtime of the LM in the literature. Some of these methods are quite simple. For example, Ozaki et al. [21] have analyzed the LM processes that most contribute to slow down its runtime. Authors, then, simply randomly choose the community of the neighbor node—instead of considering all neighboring communities–during the LM phase 1. Even though it is likely to lower the quality, it makes almost no effect on the quality since many neighbors are likely to belong to the same community in the initial partition [21]. Moreover, according to the authors, their method retains almost the same quality as the original LM. Similarly, Traag [22] also proposes a scheme where a vertex, from a given community, is moved to another community chosen at random. This minor adjustment to the original LM process leads to reductions in running time, without losing much quality.

Fortunato and Barthelemy [23] question the modularity maximization strategy which the LM employees. According to their work, the original LM modularity maximization strategy tends to miss small communities. In this same direction, Aldecoa and Marín [24] have analyzed the performance of 18 methods of community detection and have shown the sub-optimality of the results when the modularity is maximized. They had proposed a new global parameter to evaluate the quality of graph partition, called surprise [25], whose maximization, obtained by the combination of multiple algorithms, yields more meaningful results as compared with modularity. Until the present time, authors still dealing with the modularity of the LM method. For example, Chaudhary et al. [8] introduce an agglomerative hierarchical community detection approach, called enhanced Louvain method. Their proposal is not to speedup the LM. Authors, on the other hand, propose a modularity and similarity measure-based approach that does not need the information of the communities as input. Their approach can find community structure in as fast as the original LM. Gutierrez et al. [9] also present a modification of the LMalgorithm for crisp network that allows us to deal with fuzzy information in the network. In particular, authors incorporate to the classical LM the use of fuzzy measures for the nodes of the graph. Their proposal does not focus on the speedup of the LM. In this sense, authors could also incorporate their proposal to our current algorithm. In this sense, they could also have a faster-method to deal with fuzzy information in the network.

Some methods accelerate the LM through the use of parallel computing [20,26,27]. For example, Fazlali et al. [27] present an adaptive parallel thread assignment for the calculation of adding qualified neighbor nodes to a community, when using the LM. Their new parallel approach obtains a better load balancing execution of threads. The authors evaluated their parallel version of the LM on an AMD system with 64 cores, and they have shown a reduction in the execution time by 50% in comparison with the previous fastest parallel algorithms. Ghosh et al. [20] also present a parallel version of the LM. However, the authors present the design of a distributed memory implementation of the Louvain algorithm for parallel community detection. Their approach begins with an arbitrarily partitioned distributed graph input and employs several heuristics to speed-up the computation of the different steps of the Louvain algorithm. Their parallel implementation demonstrated speedups up to 46x when using up to 4K processes (relative to the baseline LM version) for a wide variety of real-world networks. Moreover, authors have shown that the parallel algorithm results are, in most cases, comparable to the best modularities obtained by a state-of-the-art multithreaded Louvain implementation.

A common approach to speed up the LM is to preprocess the original graph via fast algorithms. In this case, the LM works on a smaller graph, with fewer vertices. For example, in [28], the original graph is reduced by eliminating edges that possibly could be part of a given community. Authors have shown their approach reduces the LM execution time, with no substantial change to the quality of communities found. However, this method is inefficient in sparse networks, where the ratio between the number of edges and vertices of the graph is small. Peng et al. [29] split the original graph based on its k-core (i.e., subgraphs of the original graph where all vertices have k connections). Authors later submit these subgraphs to a community detection algorithm (like the LM). Finally, they properly reconnect the subgraphs. Despite the simplicity, subgraphs may not reflect communities and, as a consequence, final communities may not reflect the overall network reality.

Finally, we highlight that when talking about community detection, normally clustering is used as a synonym for it. Nevertheless, clustering is a more general concept. For example, clustering methods, as [30] are mostly based on a procedure for updating cluster prototypes and requires calculating the centroids of each cluster and its boundaries. In the case of [30], it calculates the lower and upper of the limit of the cluster interval using the fuzzy and possibilistic membership matrices. This method differs from the method we present in this paper. The goodness of partitioning into communities is typically measured using a well-known measure called modularity. Modularity optimization is an NP-complete problem and because of its speed and relatively high quality of output in practice, the LM has been widely adopted by practitioners [19].

Moreover, currently LM and its variations, as *Fire-Forest* and *Local Sparsification* methods, are one of the most widely used tools for community detection [20]. Therefore, in this work, we further evaluate the performance of the combo

CoVeC+LM by comparing our proposal results against the original LM and these two well-known variations. We refer readers to Appendix A for further detail about both LM variations.

3. CoVeC: a coarse-grained vertex clustering

In this section, we first formalize the CoVeC. We present a pseudo-algorithm and an illustrative example of its functioning. Then, we present a temporal complexity analysis. Finally, we present a space complexity analysis of the CoVeC.

3.1. CoVeC formalization

In this section, we present CoVeC: a new method to preprocess the original graph which turns the execution of the LM faster. In short, the CoVeC takes as input a graph and returns a set of communities, i.e., partitions of the input graph into a group of vertices. This initial coarse-grained community detection alleviates the initial step of the LM, which accounts for most of the overall computational cost. Once we define the initial communities, we can leverage the remainder of the process to the second stage of the LM, i.e., each detected community is processed as a single vertice from a simplified graph. Then, the reduced graph is fed to the first stage, so that new communities that maximize modularity are detected. LM cycles continue while there is a gain of modularity.

Algorithm 1 depicts the method we propose to preprocess a graph. First, the CoVeC takes as an input a graph (i.e., variable *Graph*[]). As expected, *Graph*[] is a set of vertices. Each vertex is a tuple containing a label (state) to the weather the graph belongs to a community (or not) and a control variable *Score* which will further be specified. Initially, all vertices are stated as *Ungrouped* (Line 2).

Algorithm 1: (CoVeC) a coarse-grained vertex clustering method.

```
1  Vertex Graph[ ] = Input Graph ;   % Vertex is a tuple with: State = {Grouped, Ungrouped}, Score = Real Number %
2  Community C[ ] = ∅ ;   % Return of the algorithm. Communities vector %
3  For All (Vertex X ∈ Graph) {X.State = Ungrouped};   % Label all vertices as Ungrouped %
4  while ( ∃ (Vertex X ∈ Graph where X.State == Ungrouped)) do
5      Vertex G[ ];   % Empty community starts %
6      For All (Vertex X ∈ Graph) {X.Score = 0};
7      Limit = 0;
8      control = true;
9      while (control==true) do
10         Vertex Vs[ ] = Ungrouped vertices of Input Graph;
11         Vertex V'[ ] = argMax (Vs.Score)
12         Vertex V = Random (V');
13         if (V.Score ≥ Limit) then
14             G ← V;
15             V.Estate = Grouped;
16             Limit ← Limit + K;
17             for (I neighbor of V) do
18                 I.Score ← I.Score + (Neighbor(I,V)+1)/Degree(I);   % Updates the Score of all the neighbors of V %
19             end
20         else
21             C ← G;
22             control = false;
23         end
24     end
25 end
26 Return C;
```

While the method does not group all vertices, we repeat the following procedure: first, we create a new empty group $G[]$ (Line 4), where $G[]$ is a set of vertices. We also initialize the score (the control variable) for each vertex as zero (Line 5). Then, we set up $Limit = 0$ (Line 6). At this point, the main idea is to group vertices with a *Score* greater than a current *Limit*. In this case, *Limit* is a variable that indirectly defines an approximation limit required for a vertex to enter a community. Intuitively, all vertices with a *Score* greater than the current *Limit* will belong to the same group. Note that, the higher the limit, the harder it will be vertices grouping. On the other hand, lower values to *Limit* will turn easier vertices grouping. Further, in this section, we will discuss how to update a vertex *Score* and the grouping *Limit* threshold.

Let $V'[\,]$ (Lines 8 and 9) be a variable which contains a set of all vertices of input graph, tagged as ungrouped, with the highest score among all vertices which are neighbors (i.e., connected) to the last vertex added to the group G. In case of

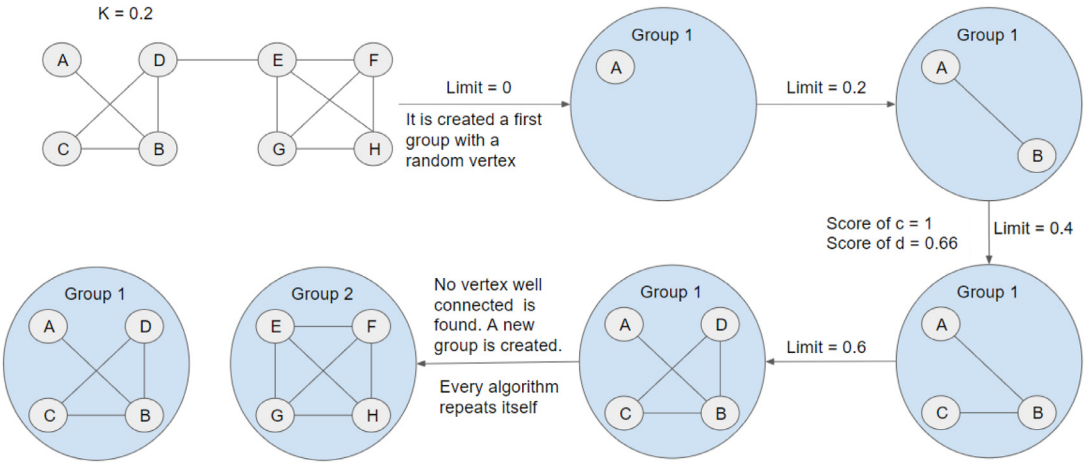


Fig. 1. Example of CoVeC phases with $K = 0.2$.

a tie, we randomly select one single vertex and assign it to the variable *V*. Clearly, during the first round of the loop, we randomly select one vertex, as all vertices *Score* are set up as zero and the group *G* is empty.

Case *V.Score* does not achieve the current *Limit* (Line 11), we *close* the current group and cease the current loop round (Line 18). Otherwise, case *V.Score* $\geq$ *Limit* (Line 11), we assign *V* to the current group *G*. Again, the randomly chosen vertex at the first round will be automatically added to the group, initializing it properly. At this point, we label *V* as *Grouped* and then we update the value of the limit, incrementing it by a given constant *k*.

At this point, it is timely to clarify the rationale behind comparing *Score* and *Limit*, and justify the lines 14 and 16 of the pseudocode. A possible metric for the connectivity level between an ungrouped vertex *B* and a neighbor vertex *A* in group *G* can be given by:

$$\mathcal{C}(B,A) = \frac{Nei(A,B)+1}{Degree(B)}, \tag{1}$$

where *Nei*(*A, B*) is the number of common neighbors between *A* and *B*, with *Degree*(*B*) being the total number of neighbors of *B*. The sum of one unit in the numerator of Eq. (1) represents the direct connection between *A* and *B*. In this case, $0 \leq Nei(A,B)+1 \leq Degree(B)$, making $\mathcal{C}(B,A)$ a percentage of the neighbors of *B*. Consider the set of values of $\mathcal{C}(B,A_j)$, with $j = 1,2,\ldots,N$, computed for each grouped vertex A_j up to iteration *N*. If the arithmetic mean of $\mathcal{C}(B,A_j)$, i.e., $\bar{\mathcal{C}} = N^{-1}\sum_{j=1}^{N}\mathcal{C}(B,A_j)$, is greater or equal to a limit *K*, arbitrarily defined for the level of connectivity, then *B* is qualified to be member of group *G*. Therefore, $\mathcal{C}(B,A)$ can be thought of as a measure of the network density around the edge connecting a node pair (*A, B*).

Due to the iterative nature of the algorithm, by which the group is gradually populated with a vertex at time, instead of comparing $\bar{\mathcal{C}}$ with *K*, which involves computing averages at each iteration, one can compare $N\bar{\mathcal{C}} = \sum_{j=1}^{N}\mathcal{C}(B,A_j)$ with *NK*, at iteration *N*. The sum of $\mathcal{C}(B,A_j)$ is calculated recursively (Eq. (2)) and stored in the variable *Score* (Line 16). Similarly, *NK* is obtained by a recursive form (Eq. (3)) and stored in the variable *Limit* (Line 14).

From the rationale above, the score of an ungrouped vertex *B*, a candidate to join the group *G* is generated by the following recursive dynamics:

$$Score(B) \leftarrow Score(B) + \frac{Nei(A,B)+1}{Degree(B)}, \tag{2}$$

with initial condition $Score(B) = 0$, forced after the creation of a group (Line 5). Equivalently, the recursive dynamic of variable *Limit* is given by the initial condition $Limit = 0$ (Line 6) and also by:

$$Limit \leftarrow Limit + K, \;\; 0 \leq K \leq 1, \tag{3}$$

The vertex that last gets into the group (Line 12) becomes the reference vertex (vertex *A* in the above formulation), for the purpose of setting the new candidate vertices to join at the group (i.e., ungrouped vertices that are neighbors of the reference vertex). The score of those candidate vertices is updated through the Eq. (2) (Line 16), from which the candidate with the highest score is selected (Lines 9 and 10) and evaluated if it qualifies to enter the group (Line 11). The rationale behind this scheme is to quickly group node pairs whose neighborhood is dense into a single community, whereas connected nodes with a sparse or empty common neighborhood (e.g., a bridge) are placed in different connected communities.

Fig. 1 presents a visual example of the CoVeC iterations (with $K = 0.2$) over a simple graph. Initially, a new group is initiated (Group 1) to which a random vertex is allocated (Vertex *A*). Vertex *B* is the only neighbor of *A* and, thus, is the sole

candidate, with $Score = 0$, to possibly join Group 1. As the current $Limit = 0$, the membership criterion is met and B joins Group 1.

Now, it is possible to add C or D to the group, once both vertices are neighbors of the most recent vertex of Group 1. When evaluating Eq. (2), the *Score* of C and D turns to 1 and 0.66, respectively. At the next round of the loop, C will be chosen as a candidate to join the group once it presents the highest score among all neighbors of B. The *Score* of C exceeds the current *Limit*, which is 0.2, and as a consequence, C joins the current group. At this point, vertex D is the only neighbor of C. As D *Score* is larger than the current *Limit* (i.e., $1.32 \geq = 0.4$), D joins Group 1. The single available neighbor of D is E. The *Score* of E is smaller than the current *Limit*, therefore, Group 1 is closed and a new group is created (Group 2). In the following, vertices E, F, G and H will be added to Group 2. Finally, the CoVeC will return two distinct groups.

3.2. Time complexity analysis

The complexity of the algorithm can be inferred from its two-loop structure: (i) the main *loop*, where the algorithm visits all the vertices of the graph; and (ii) the secondary *loop*, where the algorithm visits all the neighbors of each vertex. Let D_a be the average degree of a vertex (i.e., the average number of neighbors of a vertex):

$$D_a = \frac{2 \cdot |Edges|}{|Vertices|} = \frac{2 \cdot |E|}{|V|}. \tag{4}$$

The complexity of CoVeC is thus $O(|V| \cdot D_a)$. We can then rewrite the CoVeC time complexity replacing D_a, using Eq. (4). Thus, CoVeC time complexity is $O(2 \cdot |E|)$, where $|E|$ is the number of graph edges (links). Finally, we can state that CoVeC time complexity is $\approx O(|E|)$.

The time complexity of the LM is well-known and has been previously presented by Blondel et al. [18]. In this case, the entire LM complexity is $O(|V|log(|V|))$. The time complexity of the combo CoVeC+LM is then given by $O(|E| + |V'|log(|V'|)$, where $O|E|$ refers to the CoVeC complexity and $|V'|$ is the number of vertices (i.e., communities) in the new simplified graph that CoVeC generates. Overall, one can expect that $|V'| < |V|$ since CoVeC groups vertices into communities. In fact, we have previously evaluated CoVeC using eight real networks and we have verified that $|V'|$ is 37% smaller than $|V|$ [31] on average for these networks.

In this sense, the time complexity of the combo CoVeC+LM depends mostly on the performance of CoVeC, i.e., mainly on the reduced number of groups it bypasses to the second phase of the LM method. As a consequence, the use of the combo CoVeC+LM will outperform the original LM method in the cases that satisfy the inequality (5),

$$|E| + |V'|log(|V'|) < |V|log(|V|). \tag{5}$$

The left-side expression represents the complexity of CoVeC ($O|E|$) summed to the LM complexity after the application of CoVeC. The right-side expression represents the complexity of LM. The objective is that the left-side expression is smaller than the right-side expression, i.e., that the complexity of CoVeC+LM is smaller than the complexity of LM.

Clearly, to satisfy the inequality (5), $|E| < |E| + |V'|log(|V'|)$ since we may assume $|V'| > 0$. As a consequence, we can rewrite this inequality as $|E| < |V|log(|V|)$. Replacing $|E|$ as in Eq. (4), we have:

$$\frac{|V| \cdot D_a}{2} < |V|log(|V|). \tag{6}$$

Thus, the combo CoVeC+LM outperforms the original LM method when:

$$D_a < 2 \cdot log(|V|). \tag{7}$$

In other words, the combo CoVeC+LM is worth using when the average density of the complex networks is relatively low, i.e. the complex network is sparse. Beyond the indicated limit, the complex network is dense enough to make the complexity of applying CoVeC+LM not interesting, thus the CoVeC proposal is suitable for sparse complex networks.

3.3. Space complexity analysis

The LM main data structure is an adjacency list. This method uses this list in all its internal procedures and no other more complex structure is needed to perform the method. In other words, let $|E|$ be the number of edges of a graph, the space complexity of the LM is $O(|E|)$.

Note that CoVeC only interacts with vertices and its neighbors. As a consequence, one of the adjacency list of the original graph needs to be stored. Indeed, we have developed CoVeC to act as the first step of the LM. In this sense, both methods space complexities are equivalent, i.e., CoVeC has $O(|E|)$ space complexity.

4. Evaluation methodology and analysis

In this section, we first present our evaluation methodology (Section 4.1). We then evaluate the CoVeC sensibility to the choice of its parameters (Section 4.2). We also compared the performance of CoVeC, comparing it with the original LM and two other methods that simplify the LM input graph (Section 4.3). Finally, we evaluate the CoVeC+LM performance in the face of the network density (Section 4.4).

Table 1
Real networks used in the experiments.

Name	# vertices	# Edges	D_a
Amazon	334,862	925,872	5.53
Road Net	1,088,092	1,541,898	2.834
California	1,965,205	2,766,607	2.816
Youtube	3,223,642	9,296,202	5.768
Skitter	1,696,414	11,095,298	13.08
Wiki Topcats	1,791,489	28,511,807	31.83
Flickr	2,302,924	33,140,018	28.78
Patents	3,774,767	16,518,948	8.752
Live Journal	5,204,174	49,174,464	18.898

4.1. Evaluation methodology

We first evaluate the CoVeC sensibility to the choice of its parameters (Section 4.2). Note that, the only parameter that may influence its performance is K, as seen in Eq. (3). In fact, K defines a threshold to tell if a vertex belongs to a community or not. For low values of K, CoVeC decreases the connectivity-level requirements and, as a consequence, a vertex will be easily added to a community. The method, then, may create fewer—but more populous—groups. Fewer groups may turn the remainder of the process (LM second step) faster. However, it may also reduce the modularity of the communities' CoVeC+LM forms. On the other hand, higher values to K may produce a high number of communities. The overall execution time of CoVeC+LM may be worse, but the modularity of communities may be better.

To evaluate the CoVeC sensibility to the choice of its parameter, we perform tests over nine graphs that are commonly used in community detection works [21,32,33]. Table 1 presents the dimension of these graphs by showing, for each graph, its number of vertices, its number of edges, and its average degree.

For each graph in Table 1, we evaluate the performance of CoVeC+LM by varying the value of K from 0.01 to 0.9 (i.e., $K \in \{0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. For each configuration, we repeat 50 times the experiment, randomly choosing the initial vertex. Unless indicated otherwise, the presented results have a 99% confidence interval.

Then, we evaluate the performance of the combo CoVeC+LM by comparing it with the original LM method, as well as with two other enhancements of the LM process. In this sense, we also compare the combo CoVeC+LM with the *Fire-Forest* and *Local Sparsification* methods. We further detail both methods in Appendix A. Once more, we use the nine real graphs to analyze the performance of each method. For each configuration, we repeat the experiment 50 times, randomly choosing the initial vertex. We then compare the execution time of methods and the modularity of the resulting communities.

Finally, we further evaluate the CoVeC+LM performance in the face of the network density. In this sense, we use a set of artificial graphs where we vary the average vertex degree D_a—and, as a consequence, the network density—from low to high values. The synthetic network generator we adopt [16,34] has been widely used. This tool generates networks in which one can observe group structures and, as a consequence, these graphs serve as a reference to test the efficiency of community detection methods.

Each network we evaluate presents 1 million vertices. We vary the average vertex degree D_a to cover the range of average degree we observe to the previously evaluated real networks. More in depth, we cover 19 levels of D_a and, for each level, we generate five distinct networks. In total, we have generated 95 networks. For each network, we evaluate the performance of LM and CoVeC+LM methods.

We have performed all experiments using an Intel Xeon Dual Quad-Core (2.27 GHz) processor, 48 GB of RAM and 1 TB disk storage. All datasets and codes used in our analyses are publicly available the following at trace repository: http://netlab.ice.ufjf.br/index.php/covec/.

4.2. CoVeC sensibility to parameters

We evaluate the sensibility of CoVeC to the choice of its parameters by varying the value of K. In this sense, we evaluate the performance of the CoVeC+LM accessing its execution time and quality of the communities it provides (i.e., modularity). Unless indicated otherwise, in the following graphs, the abscissa axis represents the value of K. Solid curves present mean values of all tests and the colored region that surrounds it represents the confidence interval for each measure. Note that, in some cases, confidence intervals are negligible and, as a consequence, not visible in the figure.

One could expect to increase the number of groups, and also the execution time while increasing the value of K. In fact, as we previously discussed in Section 3, high values to K only allow groups where their members share a high number of neighbors. In case a vertex does not present a considerable number of common neighbors (links) to the members of a group, it will not be added to this group. As a consequence, a new group will be created, increasing the number of groups. Recall that the time complexity of the LM is $O(n\log_2 n)$, where n will be the number of groups. A higher number of groups, during the first stage, implies a higher execution time of the combo CoVeC+LM.

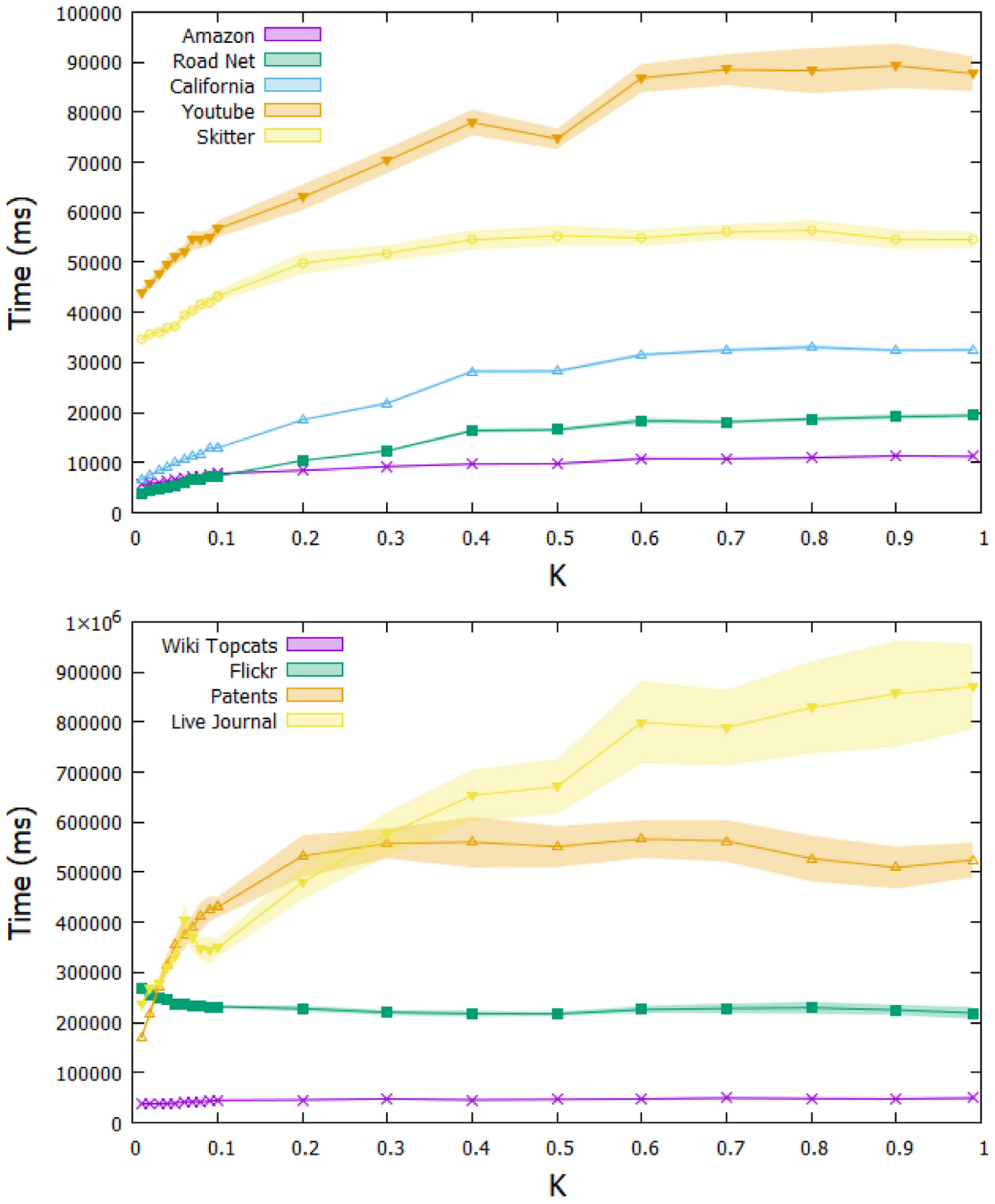


Fig. 2. Execution time of *CoVeC + LM* in relation to *LM* as a function of *K*.

Fig. 2 shows the execution time of the combo CoVeC+LM as a function of *K*. In this figure, we evaluate nine real networks. For almost all real networks, it is possible to note the growing trend of the average execution time of the CoVeC+LM, with an increase of *K*. More precisely, we note a rapid growth for smaller values of *K* and then, a roughly stable execution time for larger values of *K*. The Flickr network is the only which presents a non-expected behavior, remaining roughly stable for all values of *K*. This roughly stable execution time may be closely related to the internal network structure and deserves further investigation in the future.

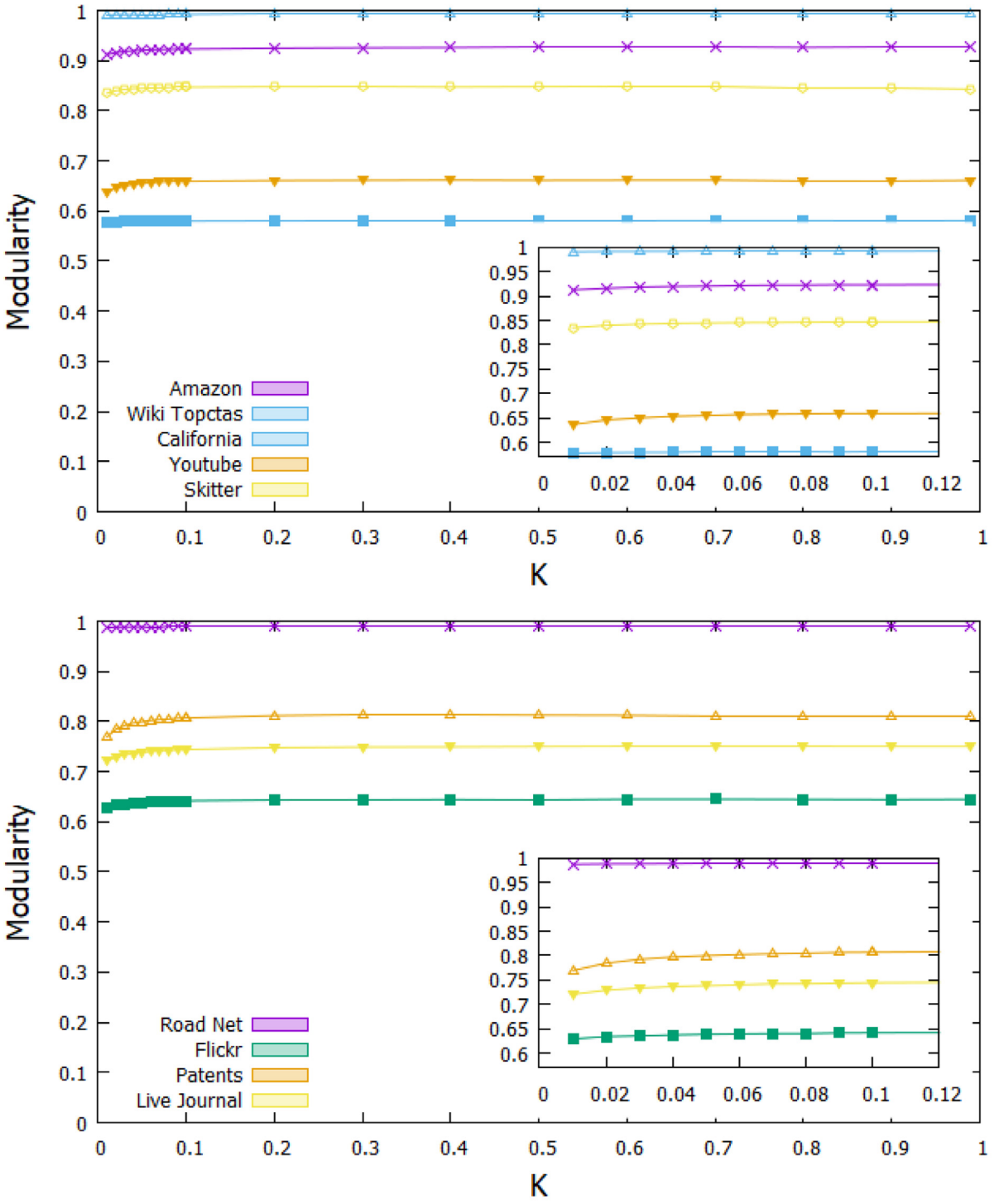


Fig. 3. Modularity of communities found by *CoVeC + LM*, in relation to *K*.

Fig. 3 presents the modularity of the identified groups (communities) as a function of *K*. Overall, the choice of the parameter *K* does not present a significant impact on modularity, for all real networks we evaluate. In fact, as one can see in the zoomed figures, the modularity for a given graph presents a very slight growing tendency for $K < 0.2$, and then, it turns practically constant for $0.2 < K \leq 1$.

In sum, according to Figs. 2 and 3, reducing the *K* parameter tends to decrease the average execution time of CoVeC+LM, at the cost of a slight drop in the average modularity of the communities detected.

Table 2
Comparative performances among LM, CoVeC+LM, Fire+LM, and LS+LM.

Graphs	D_a	Time (s)				Modularity			
		LM	CoVeC +LM	Fire +LM	LS +LM	LM	CoVeC +LM	Fire +LM	lS +LM
California	2.816	30.61	**9.08**	25.08	110.41	**0.992**	**0.992**	**0.992**	0.924
Road Net	2.834	15.66	**5.07**	13.05	58.02	**0.989**	**0.989**	**0.989**	0.913
Amazon	5.53	9.83	**6.29**	10.96	15.04	**0.926**	0.919	0.924	0.842
Youtube	5.768	58.06	**49.61**	92.72	101.82	**0.658**	0.653	0.649	0.605
Pantents	8.752	445.15	316.34	**272.13**	387.33	**0.812**	0.797	0.788	**0.812**
Skitter	13.08	**27.39**	36.74	40.03	43.86	0.841	0.843	**0.844**	0.793
Live Journal	18.898	494,64	307.68	**232.34**	240.25	**0.750**	0.736	0.730	0.715
Flickr	28.78	**52.82**	245.71	66.43	80.86	**0.643**	0.637	0.637	0.629
Wiki Topcats	31.83	101.97	263.59	**59.16**	90.77	0.635	0.617	0.624	**0.741**

Initially, it is possible to speculate that the value of K that optimizes the execution time and the loss of modularity is contained in the aforementioned range. In fact, the choice of the parameter practically does not impact on groups modularity and, the lower the value of K, the lower of the execution time.

4.3. CoVeC performance evaluation

In this section, we evaluate the performance of the combo CoVeC+LM by comparing it with the original LM method, as well as with two other enhancements of the LM process: *Fire-Forest* and *Local Sparsification* methods. We refer interested readers to further detail about both methods to Appendix A.

Once more, we use the nine real graphs to verify the performance of each method. We also assume the CoVeC parameter $K = 0.04$. As we previously mentioned, for each configuration, we repeat 50 times the experiment, randomly choosing the initial vertex. We then compare the execution time of methods and their final communities' modularity.

Table 2 presents the execution time of each evaluated method, for all real networks we use in our experiments. We also present the network density, in terms of vertex mean degree (D_a) (i.e., as networks present qualitative similar size, the higher the mean degree, the higher will be the network density). In this table, we omit the confidence intervals as they are negligible for all methods (i.e., $< 0.1\%$). Finally, we emphasize in this table (bold text) the best result for each network.

As expected, the original LM method produces the best modularity among the four methods, for all networks but *Skitter*. However, the difference in the modularity, when compared with the other methods, is relatively small. In special, when compared with the combo CoVeC+LM, the LM is less than 3% better in the worst case (*Wiki Topcats* network).

As discussed in Section 3.2, CoVeC+LM presents a better time complexity, when compared with the original LM method, for sparser graphs (i.e., graphs where vertices present a lower average degree D_a). The results in Table 2 corroborate with the previous analysis. In this case, for networks with lower D_a—as California, Road Net, Amazon, and Youtube—, the CoVeC+LM presents better execution times with respect to LM and its two other modifications. Again, note that the difference in modularity among these methods can be negligible, which reinforces the argument that the method we propose enhances execution time, without a cost in the quality of the groups it generates. More precisely, the application of CoVeC+LM for these networks achieves a mean reduction in the execution time of 47% and a mean modularity penalty of only 0.4%. For these same networks, the combo Fire+LM presents an execution time only 9% better than the original LM method. On the other hand, the combo LS+LM presents an execution time 265% worse.

For highly dense networks, where D_a is as high as 10, the combo CoVeC+LM execution time tends to be worse than the other methods. In fact, *Fire-Forest* and *Local Sparsification* reduce the total number of edges before the second stage of the LM, turning both more efficient to the *Patents, Live Journal*, and *Wiki Topcats* networks. Finally, note that the original LM method presents better execution time for the *Skitter* and *Flickr* networks, which are also highly dense networks.

We have further verified that the modularity values for both the original LM and CoVeC+LM pass the Kolmogorov-Smirnov normality test. Comparing the modularity results for both methods, for a 90% confidence level, CoVeC+LM and the original LM are statistically equivalent, according to the t-test (p-value 0.92). In other words, our proposed approach CoVeC+LM for accelerating the original LM does not imply statistically significant changes in modularity.

The execution times, presented in Table 2, do not follow a normal distribution. Then, we have applied the Wilcoxon-Mann-Whitney Test to evaluate if the combo CoVeC+LM presents a lower execution time than the original LM, and its variations. According to this test, the execution time of the combo CoVeC+LM is statistically lower (p-value 0.7), when compared with all other methods.

4.4. Evaluating the impact of network density on CoVeC performance

Fig. 4 shows the ratio between the execution time of the LM and the CoVeC+LM methods. In this figure, we plot the ratio for the synthetic networks we evaluate.

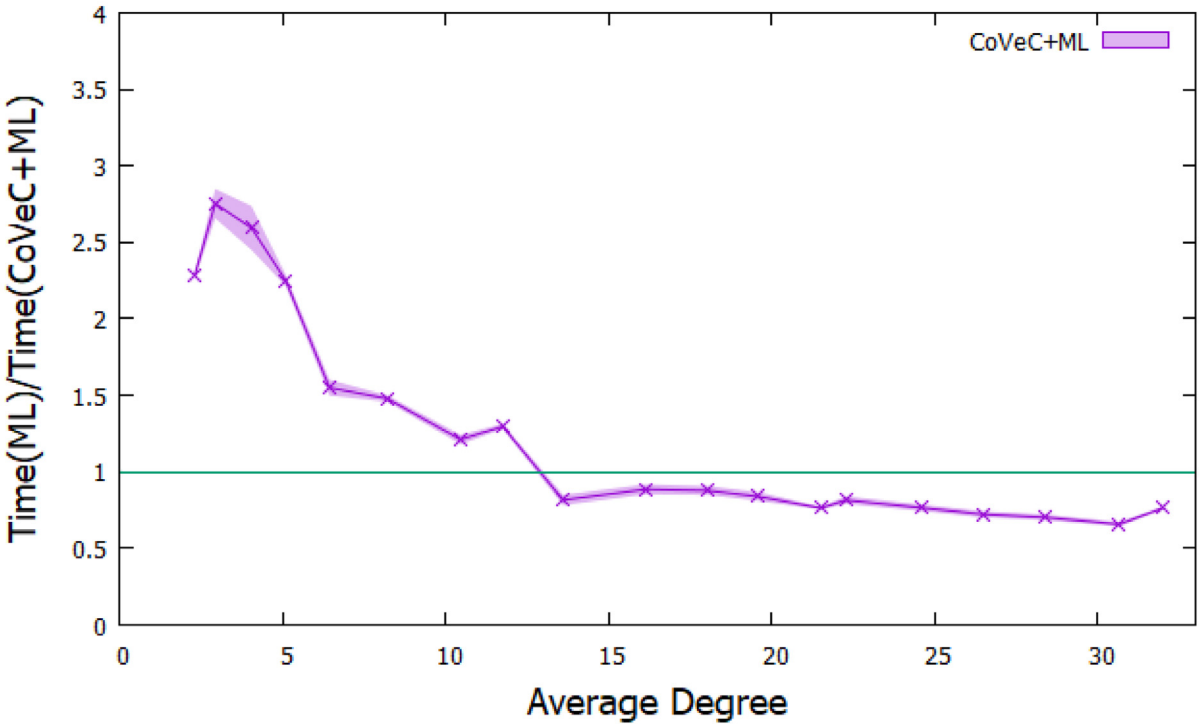


Fig. 4. Reason between the LM time by the CoVeC+LM time, for artificial graphs with different values of density.

As shown in Fig. 4, the CoVeC+LM presents clearly better results than LM for networks where vertices have a low average degree (D_a), i.e. more sparse networks. In this case, the CoVeC+LM method is up to 3 times better than the original LM. On the other hand, for networks with high node density, where the mean degree is high (i.e., $D_a > 20$), the original LM method is better, as expected. However, the difference between our new approach and the original method is not as high as for sparse networks. Finally, we point out that both methods are not clearly different from each other for networks where $11 < D_a < 20$. In this case, the performance of CoVeC+LM is close to LM performance. We believe that, in some cases, topological factors may influence the temporal improvement generated by the proposed method. However, there is a clear influence of the mean degree (i.e., the graph density) in the performance of the CoVeC.

We have also evaluated the Spearman correlation coefficient of the network density and the relative execution time between LM and CoVeC+LM execution times (i.e., the results presented in Fig. 4). In this case, the Spearman coefficient is about -0.8, i.e., the relative gain in execution time of the proposed CoVeC+LM with respect to the LM decreases as the network density increases, further highlighting the suitability of the proposal for sparse complex networks.

On the other hand, no significant correlation is perceived between the network density and the relative execution times of Fire-LM or LS-LM executions times as compared with the LM execution time. This means that, in contrast to our proposal, the performance of these LM variants show unpredictable behavior as respect to network density/sparsity change.

According to Eq. (7), the combo CoVeC+LM is better than the original LM method in networks where $D_a < 2 \cdot log(|V|)$. Given the topology size we evaluate in this section, the combo CoVeC+LM will outperform LM for networks where $D_a \approx 12$, which corroborates to the asymptotic time complex analysis we present in Section 3.2.

5. Conclusion and future work

Community detection is a key problem in graph analytics. Basically, this problem consists of finding groups of vertices that have one or more characteristics in common. The Louvain Method (LM) is a well-known method that stands out for quickly and efficiently identifying communities in large-scale graphs. However, the LM method presents a costly first round that may impose a high execution time on the method.

To overcome the costly first round of the LM method, in this paper, we propose CoVeC, a Coarse-Grained Vertex Clustering method. CoVeC pre-process the original graph in order to forward a graph of reduced size to the LM, as a replacement for the costly first round of the LM method. The subsequent group formation is left to the original LM method.

Overall, our new method is well adequated to sparse graphs and is not sensible to the choice of its parameters. Moreover, our experimental results show that the new method can be a way faster option than the first iteration of the LM, yet

similarly effective. In fact, according to our evaluation, the CoVeC outperforms the LM and its variations, attaining a mean execution time reduction of 47% and a mean modularity reduction of only 0.4%.

Future work includes applying the CoVeC to other community detection algorithms as well as investigating the application of CoVeC+LM to a wide range of real and artificial complex networks. Moreover, we intend to parallelize CoVeC, allowing for an enhancement in execution times and the processing of even larger complex networks.

Declaration of Competing Interest

The authors whose names are listed immediately below certify that they have NO affiliations with or involvement in any organization or entity with any financial interest (such as honoraria; educational grants; participation in speakers' bureaus; membership, employment, consultancies, stock ownership, or other equity interest; and expert testimony or patent-licensing arrangements), or non-financial interest (such as personal or professional relationships, affiliations, knowledge or beliefs) in the subject matter or materials discussed in this manuscript.

Gustavo S. Carnivali, Alex B. Vieira, Artur Ziviani, Paulo A. A. Esquef

Appendix A. Sampling methods

In this appendix, we present two fast methods that simplify the Louvain method (LM). These two sparsification methods simplify the original graph by removing edges from it. The modified graph is then processed by the original LM. We name these two process as Fire+LM (Section A.1) and LS+LM (Section A.2), respectively.

In [28], the authors studied the application of sampling algorithms as pre-processing for methods of community detection. The sampling algorithms used in the paper reduce the total number of edges in the graph, maintaining the number of vertices. It is expected that the methods of community detection are faster acting on a graph with a reduced number of edges. Two methods presented in [28] are used in the comparative evaluation with the proposed method. The choice of these two methods is due to the quality presented by them in relation to the acceleration of methods of community detection and the maintenance of the quality of the answers (modularity) in comparison with other methods present in the literature. Among the existing methods, they appear as widely used for their efficiency in various contexts.

The paper [35] makes a study using the methods of community detection used in [28] in addition to LM. The paper [35] then compares the methods taking into account various properties, such as the scalability and speed of the method in very large graphs and the community quality found for these graphs. In the performed experiments, LM was better than the other methods in terms of execution time.

This paper, besides analyzing the performance of CoVeC+LM, also evaluates and compares the application of the two sampling methods that are presented in the following and that were used in [28] as LM pre-processing. These analyses serve to assess the efficiency of using sampling methods together with LM and to test the efficiency of CoVeC+LM in relation to other already developed methods.

A1. Fire-forest

The Fire-Forest was chosen as a comparative baseline due to its satisfactory results (in relation to modularity and execution time) in the experiments we conduct in the current work. Moreover, Fire-Forest is a classic well-studied method. The Fire-Forest simulates a forest fire. Initially, a randomly chosen tree (i.e., a vertex) is burned. Trees close to a burning tree (i.e., the neighboring vertices) can be burned with a probability α. If a neighboring tree is burned, the edge that connects these two trees is removed. The process is repeated recursively until no tree is set on fire [36].

As in ([28]), a fire will start at all vertices of the graph. The α parameter was previously selected and the same value will be used in all the tests in this work, with the value $\alpha = 80\%$. The specific value used considered the best performance in edge removal and quality maintenance (modularity) generated in the tests performed in this work.

A2. Local sparsification (LS)

The Local Sparsification (LS) method was developed for by Satuluri et al. [28]. The purpose of the method is to remove edges that possibly will belong to the same community, preserving the graph original structure. Considering that the graph has a community structure, it is expected that most of edges of a vertex are contained within the same community, increasing the effectiveness of the method.

To find the edges that possibly will belong to the same community, the method uses the Similarity measure shown in Eq. (A.1):

$$Sim(i, j) = \frac{|Adj(i) \cap Adj(j)|}{|Adj(i) \cup Adj(j)|}. \tag{A.1}$$

This equation is equivalent to the Jaccard Similarity [37], thus it analyzes the similarity between the neighbors of vertices i and j. The idea is that an edge (i,j) is probably (not) within a community if the vertices i and j have adjacency lists with

high (low) overlap. $Adj(i)$ is the set of neighbors of i. The method will remove edges that have a $Sim(i, j)$ below a pre-specified threshold. The LS code is made available by its authors. In this work, this code was used with its default settings that specify the similarity limit being equal to 0.5.

References

[1] J.A. Bondy, U.S.R. Murty, et al., Graph Theory with Applications, Elsevier, 1976.
[2] H. Kwak, C. Lee, H. Park, S. Moon, What is twitter, a social network or a news media? in: Proc. 19th Int. Conf. on World Wide Web, ACM, 2010, pp. 591–600.
[3] A. Mislove, M. Marcon, K.P. Gummadi, P. Druschel, B. Bhattacharjee, Measurement and analysis of online social networks, in: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, ACM, 2007, pp. 29–42.
[4] S. Fortunato, Community detection in graphs, Phys. Rep. 486 (3–5) (2010) 75–174.
[5] S. Harenberg, G. Bello, L. Gjeltema, S. Ranshous, J. Harlalka, R. Seay, K. Padmanabhan, N. Samatova, Community detection in large-scale networks: a survey and empirical evaluation, Wiley Interdiscip. Rev. 6 (6) (2014) 426–439.
[6] B.S. Khan, M.A. Niazi, Network community detection: a review and visual survey, 2017, arXiv:1708.00977.
[7] Y. Zhao, A survey on theoretical advances of community detection in networks, Wiley Interdiscip. Rev. 9 (e1403) (2017) 1–13.
[8] L. Chaudhary, B. Singh, Community detection using an enhanced Louvain method in complex networks, in: International Conference on Distributed Computing and Internet Technology, Springer, 2019, pp. 243–250.
[9] I. Gutiérrez, D. Gómez, J. Castro, R. Espínola, A new community detection algorithm based on fuzzy measures, in: International Conference on Intelligent and Fuzzy Systems, Springer, 2019, pp. 133–140.
[10] J. Yang, J. McAuley, J. Leskovec, Community detection in networks with node attributes, in: 2013 IEEE 13th International Conference on Data Mining, IEEE, 2013, pp. 1151–1156.
[11] A.-L. Barabási, et al., Network Science, Cambridge university press, 2016.
[12] T. Chakraborty, A. Dalmia, A. Mukherjee, N. Ganguly, Metrics for community analysis: a survey, ACM Comput. Surv. (CSUR) 50 (4) (2017) 54.
[13] M. Girvan, M.E. Newman, Community structure in social and biological networks, Proc. Natl. Acad. Sci. 99 (12) (2002) 7821–7826.
[14] M.E. Newman, Modularity and community structure in networks, Proc. Natl. Acad. Sci. 103 (23) (2006) 8577–8582.
[15] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski, D. Wagner, Maximizing modularity is hard, arXiv:0608255(2006).
[16] A. Lancichinetti, S. Fortunato, F. Radicchi, Benchmark graphs for testing community detection algorithms, Phys. Rev. E 78 (4) (2008) 46110.
[17] J. Leskovec, K.J. Lang, M. Mahoney, Empirical comparison of algorithms for network community detection, in: Proc. 19th Int. Conf. on World Wide Web, ACM, 2010, pp. 631–640.
[18] V.D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, J. Stat. Mech. 2008 (10) (2008) P10008.
[19] D. Hric, R.K. Darst, S. Fortunato, Community detection in networks: structural communities versus ground truth, Phys. Rev. E 90 (6) (2014) 62805.
[20] S. Ghosh, M. Halappanavar, A. Tumeo, A. Kalyanaraman, H. Lu, D. Chavarria-Miranda, A. Khan, A. Gebremedhin, Distributed Louvain algorithm for graph community detection, in: 2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS), IEEE, 2018, pp. 885–895.
[21] N. Ozaki, H. Tezuka, M. Inaba, A simple acceleration method for the Louvain algorithm, Int. J. Comput. Electr.Eng. 8 (3) (2016) 207.
[22] V.A. Traag, Faster unfolding of communities: Speeding up the Louvain algorithm, Phys. Rev. E 92 (3) (2015) 32801.
[23] S. Fortunato, M. Barthelemy, Resolution limit in community detection, Proc. Natl. Acad. Sci. 104 (1) (2007) 36–41.
[24] R. Aldecoa, I. Marín, Surprise maximization reveals the community structure of complex networks, Sci. Rep. 3 (2013) 1060.
[25] R. Aldecoa, I. Marín, Deciphering network community structure by surprise, PloS one 6 (9) (2011) e24195.
[26] S. Bhowmick, S. Srinivasan, A template for parallelizing the Louvain method for modularity maximization, in: Dynamics on and of Complex Networks, Volume 2, Springer, 2013, pp. 111–124.
[27] M. Fazlali, E. Moradi, H.T. Malazi, Adaptive parallel Louvain community detection on a multicore platform, Microprocess. Microsyst. 54 (2017) 26–34.
[28] V. Satuluri, S. Parthasarathy, Y. Ruan, Local graph sparsification for scalable clustering, in: Proc. ACM SIGMOD Int. Conf. on Management of Data, 2011, pp. 721–732.
[29] C. Peng, T.G. Kolda, A. Pinar, Accelerating community detection by using k-core subgraphs, 2014, arXiv:1403.2226.
[30] E. Rubio, O. Castillo, F. Valdez, P. Melin, C.I. Gonzalez, G. Martinez, An extension of the fuzzy possibilistic clustering algorithm using type-2 fuzzy logic techniques, Adv. Fuzzy Syst. 2017 (2017).
[31] G.S. Carnivali, A.B. Vieira, P.A. Esquef, A. Ziviani, Método rápido de agrupamento de vértices para detecção de comunidades em redes complexas de larga-escala, CSBC Wperformance, 2018.
[32] J. Kunegis, Konect: the koblenz network collection, in: Proceedings of the 22nd International Conference on World Wide Web, ACM, 2013, pp. 1343–1350.
[33] J. Leskovec, A. Krevl, snap datasets: stanford large network dataset collection, 2014, (http://snap.stanford.edu/data).
[34] A. Lancichinetti, S. Fortunato, Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities, Phys. Rev. E 80 (1) (2009) 16118.
[35] Y. Ruan, D. Fuhry, J. Liang, Y. Wang, S. Parthasarathy, Community discovery: simple and scalable approaches, in: User Community Discovery, Springer, 2015, pp. 23–54.
[36] J. Leskovec, C. Faloutsos, Sampling from large graphs, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2006, pp. 631–636.
[37] S. Niwattanakul, J. Singthongchai, E. Naenudorn, S. Wanapu, Using of Jaccard coefficient for keywords similarity, in: Proceedings of the International Multiconference of Engineers and Computer Scientists, 2013, pp. 380–384.

Development of an optimized pipeline for prediction and analysis of genome scale protein-protein interaction network

Gustavo Simões Carnivali, Edson Mario de Andrade Silva, Tiago Antonio de Oliveira Mendes .

***Abstract*—Protein-protein interactions are associated with important mechanisms of biological processes in cells, including interactions between different species such as pathogens and hosts. The prediction of these interactions is complex and requires the integration of different computational tools. In this work, we proposed a pipeline for predicting and visualizing protein-protein interaction networks on a genome-wide scale for interactions between proteins from two different species. This pipeline considers different protein attributes to predict these interactions, including the presence of interaction domains, subcellular localization and experimental data.**

***Index Terms*—Protein-protein interaction, interatome, molecular interaction, graphs, molecular network**

I. Introduction

Proteins are the main functional products of cells and organisms [1]–[3]. They can relate to each other forming a complex network that acts directly on the most diverse biological, cellular and/or metabolic processes [3], [4]. Large scale data (proteomics, transcriptomics, genomics...) have been used in computational analyzes in order to identify and characterize protein functions and even describe their expression levels. However, information about the native state of a protein and its amino acid sequence does not explain the degree of complexity existing in a biological system [5]. These complexities can be modeled based on graph theory and studied using interaction networks between two biomolecules (protein-protein, RNA-protein, DNA-protein, among others). These networks can represent physical and/or functional interactions, and many of them can be conserved across different phylogenetic groups [4], [6].

Currently, many experimental methods are employed to identify IPP networks (protein-protein interactions), such as, for example, the double hybrid method, affinity chromatography and immunoprecipitation [7], [8]. These methods however have limitations, for example, they are not able to identify interactions for all proteins in an organism and can identify interactions that never happen in a living system, due to the fact that the molecules involved are located in different compartments and methods rely on sample processing that exposes biomolecules to unnatural interactions. Thus, computational methods can be used in order to identify IPP using the amino acid sequences of all proteins of a given species [9], enabling the discovery of potential interactions faster than the experimental methods that may present more difficulties and limitations to detect.

Computational tools for predicting the cellular sublocalization of proteins, such as TMHMM, SignalP, TargertP, can help with the robustness of the identification of relationships. Other computational approaches used in the study of IPPs include those that employ the phylogenetic profile [10]–[12], Neighborhood genome [13], fused genes [12], [14], co - sequence evolution [15], among others. Methods like these have been successfully applied in the identification of targets for the control of phytopathogens, such as in the rice blast fungus, Magnaporthe grisea [16], as well as in the plant, in the sense of selecting candidate genes for resistance to Biotic and abiotic stress [17].

Due to these presented applications, and several others, the study of relationships between proteins is still a problem and a current reality [18], [19]. Specifically, this work uses tools like String, Blast and SignalP [3], [4], [20]–[22] to find new protein relationships, mainly between different species, these portals provide protein ratios in thousands of model and non-model organisms. Interactions from the most diverse sources (physical, functional interactions, based on predictive methods, etc.). [23]–[25].

Portals such as String or the BLAST tool are famous and reliable, which is why they were chosen, but for other applications in which other protein relationships are useful, other portals can be used [26]–[28]. For example, if you are interested in the 3D structure of proteins, the article [9] can be used, or the article [29] can be used if you are interested in studying chemical compounds connected to the proteins studied.

The objective of this work is to develop a method that predicts the interactions between proteins of two previously defined species. Performing the prediction of the interaction network between proteins of the two species from RNA-Seq data.

II. Materials and Methods

A. protein-protein interactions

DNAs, RNAs and proteins are continually interrelated, increasing the complexity of biological systems. For example,

The authors are with the Interunit Graduate Program in Bioinformatics, Universidade Federal de Minas Gerais, Belo Horizonte, Minas Gerais, Brazil and Department of Biochemistry and Molecular Biology, Universidade Federal de Viçosa, Viçosa, Minas Gerais, Brazil. E-mail: gustavocarnivali@gmail.com and tiagoaomendes@ufv.br

...

a common term in biology is regulatory proteins, they have the main function of participating in gene expression processes, that is, they are proteins that stimulate or discourage production processes of other proteins [30]. Another common term is gene co-expression, where a gene relationship is implied when the expression of two or more genes changes proportionally in different organisms [31]. In this context, the creation and study of biological interactions is common.

Biological interactions, whether functional interactions between genes or physical interactions between proteins and other biomolecules, interactions of RNA, DNA, membranes, carbohydrates, and small molecule metabolites, provide the framework for understanding how the cell is structured and controlled, and serve as a framework for understand gene-phenotype relationships and the mechanistic basis for some cellular functions [32], [33]. The characterization of molecular and functional interactions between genes, their products and biomolecules has been fundamental in the interpretation of genetic associations related to cancer and other diseases in a set of different contexts. [34]–[36]

Protein-protein associations have proven to be a useful concept. The complete set of associations can be assembled into a large network, which captures current knowledge about the interconnectivity of a cell. In addition to the analysis of each protein and its properties, a connected set of proteins can also be analyzed. For this reason, a variety of uses of these functional association networks have emerged. For example, (i) Protein network information can aid in the interpretation of functional genomics data [35]. (ii) Protein association networks have also proved to be surprisingly useful for elucidating disease genes, both for Mendelian and for complex diseases [37]. For this last application, the networks can help to narrow or know the search space, analyzing genomic regions that cover more than one gene or protein, connected or not to a disease. [38]

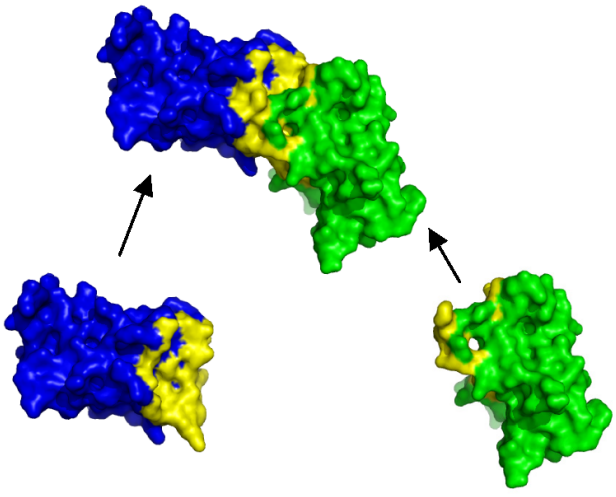


Fig. 1. **Example of protein interaction:** two proteins with structural similarity. Image taken from the World of Biochemistry.

Two or more proteins can connect in different ways, in this work, "functional association" will be used, that is, a connection between two proteins that jointly contribute to a specific biological function [39]–[41]. For two proteins to be associated in this way, they need not physically interact, rather it is sufficient that at least some of their functions occur in the same cellular region. By this definition, even proteins that antagonize each other can be associated functionally, as an inhibitor and an activator in the same way. [4]. Figure 1 presents an example of protein interaction, in which two proteins with some structural similarity are paired.

1) Databases: protein interactions: Several portals present and provide interactions between proteins. In this work, 3 portals of high relevance will be specified below by the number of connections, proteins and species considered:

- **String:** "STRING is a database of known and predicted protein-protein interactions. Interactions include direct (physical) and indirect (functional) associations; they result from computational prediction, knowledge transfer between organisms and interactions aggregated from other (primary) databases." In the figure II-A1 a visual example of String protein interactions can be seen.(¡https://string-db.org /¿) [4] [1]
- **BioGrid:** "BioGRID is a biomedical interaction repository with data compiled through comprehensive curation efforts. Its current version is 4.4213 with search of 80,848 publications and 2,537,592 protein or genetic interactions, 29,417 chemical interactions and 1,128,339 post-translational modifications of major model organism species." (¡https://thebiogrid.org/¿) [36] [2]
- **Intact:** "IntAct provides a free and open source database system and analysis tools for molecular interaction data. All interactions are derived from curated literature or direct user submissions." (¡https://www.ebi.ac.uk/intact/¿) [22] [3]

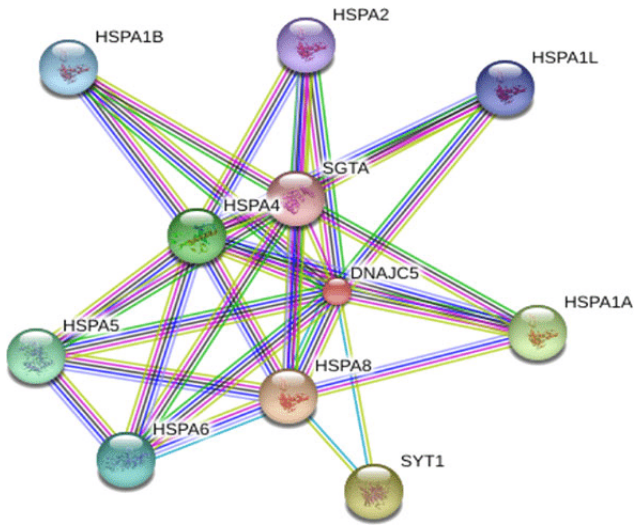


Fig. 2. **Example of String protein interactions**: random proteins from the String database connected. [42]

[1]Data downloaded and retrieved in August 2022
[2]Data downloaded and retrieved in August 2022
[3]Data downloaded and retrieved in August 2022

B. Graphs and Complex Networks

The world is full of intrinsically complex systems. For example, human bodily functioning requires the existence of thousands of metabolic pathways working together. These systems are characterized by being composed of many interconnected parts and are called graphs [43]. A graph $G = (V, E)$ is a structure formed by a non-empty set of vertices V and a set of edges $E \subseteq P(V)$ with $P(V) = \{\{x, y\} : x, y \in V\}$ which is the set of all unordered and not necessarily distinct pairs generated from V. Each edge $\{x, y\} \in E$ is formed by a pair of distinct vertices ($x \neq y$). For each pair of vertices, there is at most one edge associated with them [44].

A graph can have several modifications in order to model several scenarios. In this work a graph with variations is used. Below are two common modifications to graphs:

Valued or weighted graph: A valued graph is one in which values are assigned to its edges or vertices. Taking as an example a graph that models roads connecting cities, with cities being vertices and roads being edges, a valued graph could present the distance between cities on their edges (in kilometers for example), or the number of inhabitants per city in each vertex. An example of a valued graph can be seen in Figure 3.

Directed graph: A graph is directed if each edge $e = \{x, y\}$ connects vertex x to vertex y but does not necessarily connect vertex y to x. An example of a directed graph can be seen in Figure 3, in which there is a one-way street connecting Cambridge to Stanford and another one-way street connecting Stanford to Cambridge. In the figure, the reference to directionality is given only by the arrows connecting the cities.

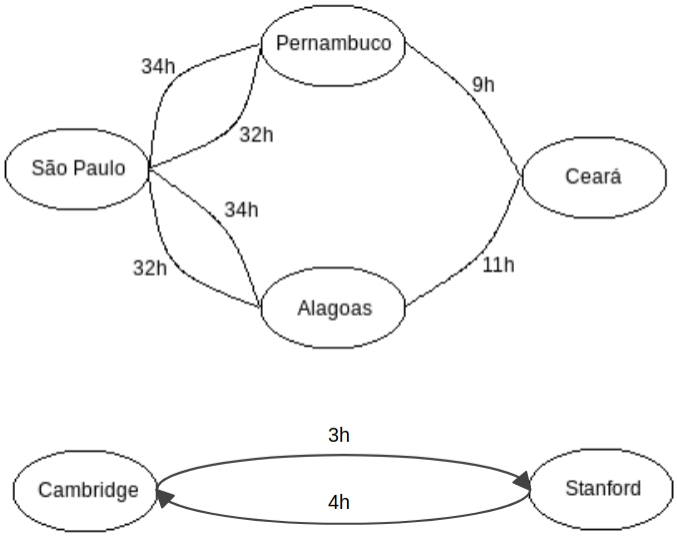


Fig. 3. **Sample graphs**; directed graph connecting real cities by travel time.

C. Pipeline

This job runs several programs in sequence. The sequence of programs can be seen in the figure 4 and was called a pipeline. The pipeline sequence explained verbatim can be seen below. The complete description of the programs and data used are described in other sections of this same chapter.

- **Protein sequence of the chosen species:** your protein sequences.
- **String, BioGrid and Intact protein sequence:** the protein sequences specified in the section II-A1.
- **Blast DIAMOND version:** DIAMOND is applied to the protein sequences (databases plus chosen species), searching and generating the comparison data sequences between the original data.
- **Interaction graph:** the graph objectified by this work. Union of the relationships obtained by DIAMOND by the similarity between the specified proteins. In this step, the final score of the connections found is also generated, this score is based on the confidence scores offered by the databases, it is calculated from classic statistical calculations and cited works.
- **Topological analysis:** the obtained graph undergoes several analyses, those that aim to evaluate topological characteristics of the vertices contained in this graph are carried out at this moment. These analyzes were described in the section II-E.

This pipeline can be obtained and executed. It was fully implemented as a simple text file that can be executed in a Linx terminal. The available pipeline runs several programs like DIAMOND, these programs can be found and understood in their respective pages on the internet. All programs made to complete the objectives of this work were implemented in C++, they are automatically executed by the pipeline in available text.

D. local alignment

According to the [20] study, BLAST ($Basic\ Local\ Alignment\ Search\ Tool$) is an algorithm for comparing primary biological sequence information, such as amino acid sequences from different proteins or nucleotides from DNA sequences. According to the BLAST website (¡https://blast.ncbi.nlm.nih.gov/Blast.cgi¿), BLAST finds regions of similarity between biological sequences. The program compares nucleotide or protein sequences to sequence databases and calculates statistical significance.

Due to the diversity of data that can be used in this work, an implemented and accelerated version of BLAST called DIAMOND [45], [46] will be used. DIAMOND obtains the same output as BLAST, however, it spends a smaller amount of time. According to the GitHub page where DIAMOND is available, DIAMOND analyzes proteins 100-10,000 times faster than classic BLAST. [4]

E. Graph analyzer and viewer

The graph generated by the methods proposed above is analyzed by a set of tools that will be presented in this section. The objective in applying these tools is to obtain more information about the generated graph and what it represents.

[4]DIAMOND data was downloaded and retrieved in August 2022

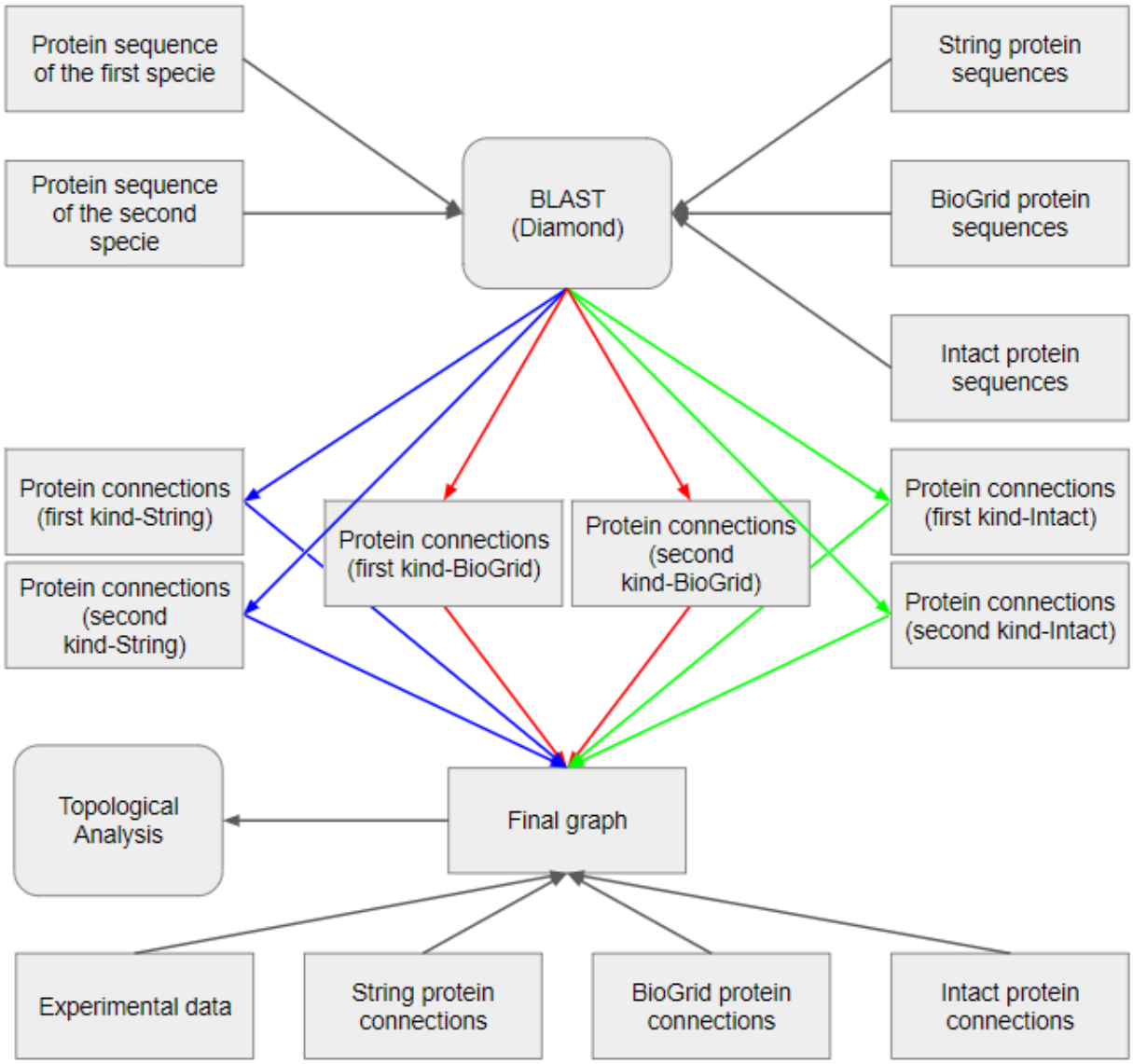


Fig. 4. **Methodological pipeline**: programs and processes executed in the sequence presented. Squares represent data. Squares with rounded corners represent programs.

Initially Cytoscape will be used in the graph. According to the Cytoscape website (¡https://cytoscape.org/¿, accessed 2022), Cytoscape is an open source software platform for visualizing molecular interaction networks and biological pathways. In this work Cytoscape will be used mainly to visualize the generated graph. However, Cytoscape will also be used to obtain some properties of the graph, these properties will be useful for the analysis and application of the graph. The properties that will be used are presented below:

- **Degree analysis:** the degree of a vertex is the number of other vertices connected to the initial one [47]. In this work, high-degree vertices represent proteins highly connected to each other. If they are proteins of the same species, it can be assumed that the studied protein has great similarity with several others in an organism; if they are proteins from different species, proteins with a high degree represent possible connections between species and, possibly, could be the focus of production of, for example, fungicides.
- **Distribution of degrees:** The degree of each vertex in a graph is analyzed. The analysis of the distribution of degrees can show properties of the graph. Specifically and the existence of hubs, hubs are considered vertices with a high degree in relation to others [48], in this graph, they are vertices with great functional similarities to others or, which participate in large functional protein groups. For example, if a chemical product acts on a hub, in parallel it will also act on several other proteins.
- **Betweenness centrality:** the centrality of $betweenness$ is the sum of the number of shortest paths passing through a given vertex [49]. In this work, vertices with high centrality of $betweenness$ represent proteins with high relevance for connecting other proteins. As before, a chemical that acts on a protein with a high $betweenness$

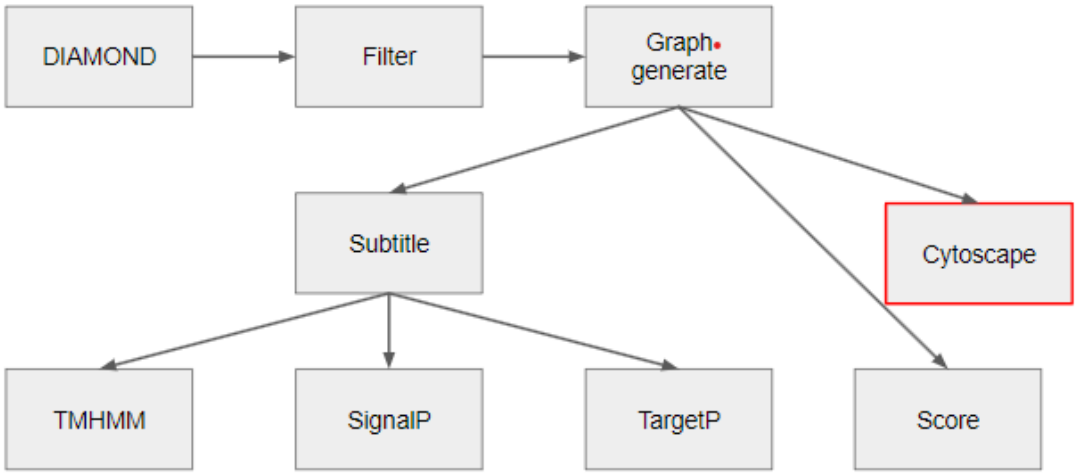


Fig. 5. **Sequence of programs executed by the developed pipeline.**

value will possibly act in parallel on several other proteins.

Due to the size of the graphs used, as well as DIAMOND was chosen, in this work another version of the Louvain method called Covec [50], [51] will be used. Covec, according to his work, manages to find communities with similar quality to the Louvain method, but with a lower cost in processing time.

F. Algorithms developed

Programs produced independently of this work will not be studied and analyzed. Among the programs developed in this work, there are only the Graph Generator, the filtering algorithm and the adding sub-title algorithm, as shown in the figure 5.

The filtering algorithm is simple. The algorithm just reads the output of String (or other database) and writes the best connections found, in a suitable file format to speed up the Graph Generator (algorithm following in figure 5). The algorithm reads the connections file and, it generates a new file for each first protein of the pair of proteins offered by the connections file, each file will have all the protein connections.

The legend addition algorithm is also simple, it adds the nucleotides to the protein names found by the graph. Putting the output in the proper format for the following programs.

The scoring algorithm just adds the score to the found connections. How the algorithm calculates scores will be described in the next section.

The Cytoscape in red in the figure 5 is just the program chosen to analyze the generated graph. It will be used several times in this work, however, it is not available together with all the programs used in the pipeline.

The graph generating algorithm reads the output files of the filtering algorithm and the connections produced by the databases. After the readings, it is analyzed whether, among similar species, there are DIAMOND output proteins connected by any of the databases or by DIAMOND. What the algorithm looks for can be seen in an example in the figure 7. The algorithm finally returns a graph for each database used.

With the objective of accelerating the available pipeline, a parallelization of it was proposed. Basically to perform the parallelization, the input files of the filtering algorithm and the graph generator algorithm are divided into the specified number of threads, for each thread the algorithm is executed in parallel.

The filtering algorithm and the graph generator algorithm are independent, that is, their internal tasks do not depend on other internal tasks performed in the algorithm, which is why they can be parallelized, as one task is independent of the other, they can be executed in parallel.

To test the implementation with threads, the pipeline was tested on Human (data obtained from the String website [https://string-db.org/] in 2023) and Phakopsora Pachyrhizi (data obtained from the String website [https://string-db.org/] in 2023) files, String database and on an Aspire 3 computer with 8 cores, Ryzen 7 processor and 8 GB of RAM memory. The figure 6 presents the time spent by the pipeline with the increase in the number of threads, each point represents the average of 3 executions, as can be seen the time decreases with the increase of threads as expected, the time however, it does not decrease steadily, possibly because the pipeline also executes other programs without implementing parallelization.

In the example of the figure 7 a new connection between proteins A and D is created. By String, BioGrid or Intact a connection is found between proteins B and C, as there are connections between A and B and between C and D, a new connection between proteins A and D can be implied. See that proteins A and D can be from different species, so this model is also able to find new protein connections between different species.

See also that proteins A and B (or C and D), in the example of figure 7, are not connected by String, BioGrid or Intact because A and B (or C and D) are not necessarily of the same species.

The code with the implemented programs (Filter,

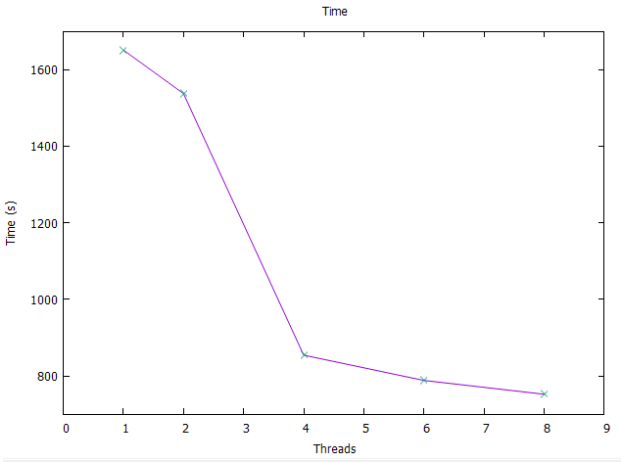


Fig. 6. **Threads by Time**: Y axis represents the total time in seconds spent by the number of Threads specified on the X axis.

GraphGenerate, Score and SubTitle) can be downloaded from the link: https://github.com/gustavocarnivali/PPINP. The complete pipeline with the implemented programs, the programs used (DIAMOND, TMHMM, SignalP and TargetP), as well as the example inputs can be downloaded from the link: https://drive.google.com/file/d/1iCsUqDoIvrgzXlAcdAmvNr6N-5a2T405/view?usp=sharing. All programs used are licensed for free use, their licenses are available on their respective websites.

G. Score

Fortunately Blast, String, BioGrid and Intact provide a score that will be used by this work. Initially, the unreliable edges that were generated by these tools (Blast, String, BioGrid and Intact) will be filtered. the final Score value available here.

The figure 8 presents a possible graph generated by this program. The value of the edges (A, B, C and D) is provided by the programs used (Blast, String, BioGrid and Intact). However, in this example, you can see that there is a connection between vertices P1 and P4, and two connections between vertices P2 and P4 (one passing through edge A and the other passing through edges D and C). Therefore, in this example, the Score between P1 and P4 is the sum of 1 value, the Score between P2 and P4 is the sum of 2 values. Therefore, the Score offered by this program will be the sum of a set of values, established by the number of paths between two vertices of the graph.

See what cycles can occur in this graph, which can generate errors like the one shown in the figure 9 where, for example, the score from P1 to P2 can be a sum of infinite terms. To work around this error, the program that calculates the Score has a maximum distance between two vertices, a maximum number of vertices is predefined, if a path has a greater number of vertices than the predefined one, the path is automatically disregarded.

It is also possible for multiple paths connecting two vertices to occur. For example, in figure 8 there are two paths connecting P2 to P4, the path that passes through edge B and the path that passes together through edges D and C. The portals used provide the values of the edges (in the example A, B, C and D) but not the value of the path that goes through D and C.

To find the value of the path that passes through the edges C and D in the example of figure 8, the intersection calculation will be used. The intersection of two sets A and B is a set consisting of all the elements that belong simultaneously to A and B [52]. Basically the intersection calculates the probability that two or more edges occur together. The intersection has the symbol $\cap$ and is calculated as the multiplication of sets, in the example: $B \cap C = B \cdot C$.

All paths connecting vertices, which pass through only one edge or which pass through more than one, with the intersection calculation, have a value between 0 and 1. However, we want to know the final value of the connection between two vertices, for that , after calculating the number of paths between all pairs of vertices in the graph, the equation 1 will be used. The equation is derived from work [5] which has similar objectives to this one. As the score of the three methods is a value between 0 and 1, the return of the equation 1 will also always be a value between 0 and 1.

$$Score = 1 - \prod_{i \in E} (1 - R_i)^n \tag{1}$$

Where R_i is the score of the method (String, BioGrid or Intact) on the edge i, n and E is the number of predicted interactions between two vertices by the different databases.

The use of the union of sets was analyzed together. The union of sets corresponds to the joining of the elements of the given sets [52]. However, regarding tests carried out on the generated graph, the equation generated by equation 1 obtained better results, therefore, equation 1 will be used in this work.

In order to highlight the edges with greater significance, a minimum score value was established. After all the calculation, edges with a value less than the limit are simply ignored. The table I presents the number of edges with scores in different intervals, this table was used to define that only score values greater than 0.7 will be used.

TABLE I
NUMBER OF EDGES WITH SCORES BETWEEN THE ESTABLISHED INTERVALS.

Value	N Edges
0.0 - 0.1	83
0.1 - 0.2	52
0.2 - 0.3	112
0.3 - 0.4	37
0.4 - 0.5	13
0.5 - 0.6	28
0.6 - 0.7	16
0.7 - 0.8	21
0.8 - 0.9	34
0.9 - 1.0	5203

III. Results

A. Code

The code for the entire pipeline is made available. Its use is currently exclusive to Linux, it only requires code execution in

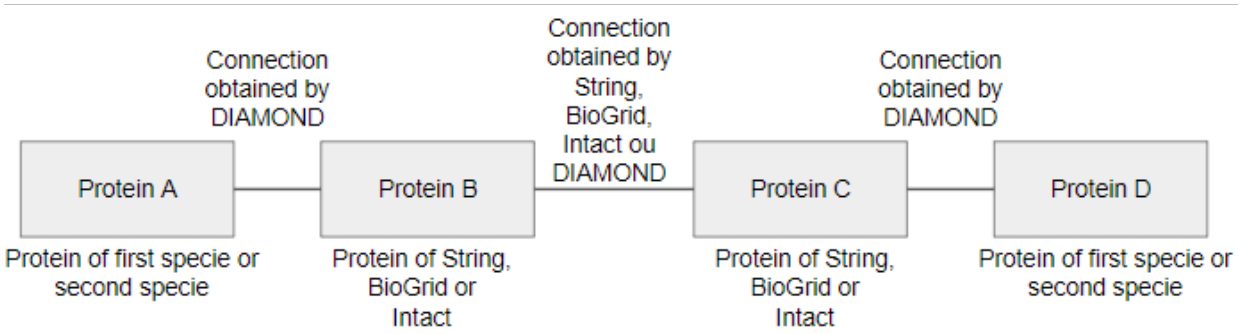


Fig. 7. **Connection of two random proteins by others.**

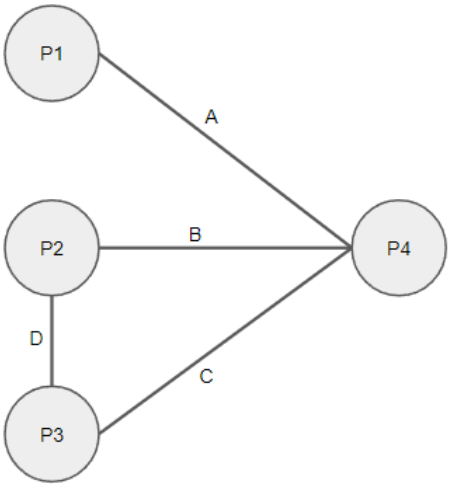


Fig. 8. **Example of a possible generated graph**: spheres represent proteins and traces protein connections.

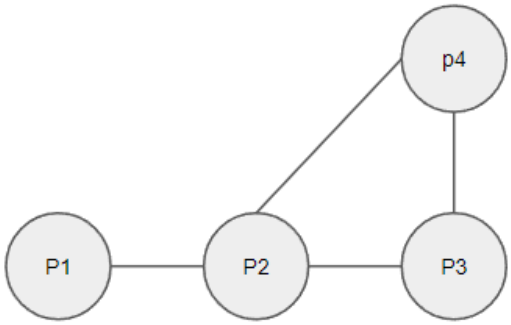


Fig. 9. **Example of a possible graph generated with cycle**: spheres represent proteins and traces protein connections.

a shell. All programs developed by this work were developed in C++ so it is not necessary to install them, however the pipeline comes with Blast (Diamond version), TMHMM, SignalP and TaregtP, to execute them it is necessary to install two programs . The pipeline accompanies a textual document where all the information for its execution is detailed.

The pipeline was called PPINP (Protein Protein Interaction Network Pipeline). The pipeline can be downloaded via the link: https://github.com/gustavocarnivali/PPINP, in this link only the programs developed by this work are available. The complete pipeline can be downloaded via the link: https://drive.google.com/file/d/1iCsUqDoIvrgzXlAcdAmvNr6N-5a2T405/view?usp=share_link, in this link all programs are available as well as input examples.

B. Data naming

It is common for the various portals used to use different nomenclatures for similar proteins. For example, the keratin 84 protein from the NCBI portal has the symbol KRT84. An initial search is performed on the String (portal used), even if you use either of the two protein names (Keratin 84 or KRT84) the String will return the same output.

In the offered software, the nomenclatures used in the different portals are not relevant in the search. Diamond or Blast uses only the nucleotide sequence of the proteins, irrelevant to the name given to them. The graph generator algorithm performs name comparisons only between proteins from the same portal, therefore, the portal can use any nomenclature. The algorithms used after the graph generator algorithm use only the nomenclatures obtained by the algorithm.

The irrelevance of the nomenclature used allows databases of species and protein interactions in databases to be easily modified, updated or other databases to be added to the same software.

C. Algorithm complexity

As reported, the slowest part of the pipeline is the algorithm that generates graphs. This algorithm, however, performs only three tasks: (1) data reading; (2) find new connections; (3) record data. Tasks 1 and 3 listed above have linear complexity. But task 2 does not have linear complexity, so it will be discussed in this section.

The process of finding new connections finds connections between proteins of the two specified species. Therefore, for each protein of the first species, all proteins of the other species must be analyzed, so this process has a complexity O(N*M), where N is the number of proteins from the first species and M is the number of proteins from the second species. .

Note that such a process has three sets of data: interactions between proteins from the first species and from the databases; interactions between proteins from the second species and from the databases; interactions between proteins from the databases. If you choose to use these three sets of data in main memory, the algorithm can be parallelized. Such a choice was not made in the available algorithm due to the diversity of the amount of data that the species or databases may have. However, the algorithm can be easily modified if this structure is considered.

To make the algorithm faster two strategies explained below were used:

Splitting into multiple connection files: Database connection files can be very large. In our program it needs to be accessed every time we want to check if there is a potential connection between all proteins of the first species with all proteins of the second species. The connections file is a list with 3 terms: the proteins of the first species; proteins of the second kind; score. This list was split into several sub-list files, one file for each output protein. Once you have the name of the first protein, you can go straight to its file, avoiding reading connections to other proteins.

OS Parallelism: the offered program only requires 2 inputs: the protein file of the first species; the second species protein archive. You can choose to split the two files into several subfiles with the same content and run the program for each set of files created. This is possible due to the independence of the internal terms of each file.

D. Characteristics of the generated graph

In this study, Glycine Max and Phakopsora Pachyrhizi will be used to test pilenine. According to IBAMA (Instituto Brasileiro do Meio Ambiente), $G.\ max$ represents the largest production in Brazil in 2022, with 123,829.5 million tons produced, in 2021 the soybean planting area corresponded to 3.4% of the Brazilian territory with the production of 21.3% of all Brazilian productions. Also according to IBAMA, $P.\ pachyrhizi$ is a biotrophic fungus that causes Asian rust, the fungus that caused the most damage to soybean production in 2022

Asian rust is still a problem for soybean cultivation in Brazil and in other countries, so finding proteins that can arise from the relationship between the two is essential and will be done in the example and test of this pipeline. The String database will still be used due to its greater number of data in relation to the other mentioned databases.

The graph of $G.\ max$ and $P.\ pachyrhizi$ was generated with the String, Biogrid and Intact databases, A graph was also generated from the union of the three databases mentioned. Quick information about the 4 graphs developed is described below:

- **String:** the graph generated by the String database has 6879 vertices and 700343 edges.
- **Biogrid:** the graph generated by the Biogrid database has 5172 vertices and 23073 edges.
- **Intact:** the graph generated by the Intact database has 429 vertices and 922 edges.
- **Union:** the graph generated by the union of the databases has 7213 vertices and 72211 edges, 6291 vertices of $G.\ max$ and 922 vertices of $P.\ pachyrhizi$. This graph of the 4 will be presented and detailed in this section.

All can be downloadable obits via the link: https://github.com/gustavocarnivali/Files.git

As this case study involves contamination, it may be interesting to know the functionality of the proteins found by the network, to allow, for example, the creation of new defense strategies. In addition to TMHMM, SignalP and TargetP, another program was used, EggNOG, according to its website: ”EggNOG-mapper is a tool for fast functional annotation of novel sequences. It uses precomputed Orthologous Groups (OGs) and phylogenies from the EggNOG database to transfer functional information from fine-grained orthologs only.”

The EggNOG output can be downloaded from the link: https://github.com/gustavocarnivali/Files.git

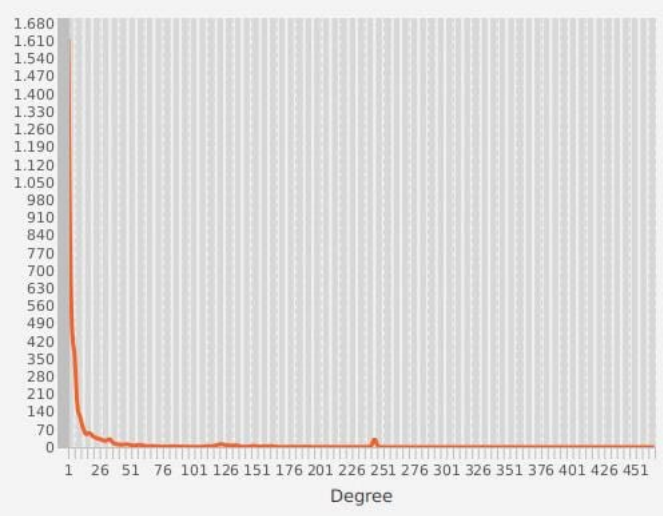


Fig. 10. **Graph degree distribution**: Y axis represents the number of vertices with the specified degree on the X axis. Information obtained by the graph presented.

The information obtained about the graph according to Cytoscape can be seen below and in the figure 10 and the figure11. Specifically, in the list below you can see various information about the graph, in the figure 10 the degree distribution of the graph, where a pattern of scale-free networks can be seen, in the figure 11 you can see the betweenness distribution.

- **Number of nodes:** 7310
- **Number of edges:** 72211
- **Avg. number of neighbors:** 21.443
- **Network diameter:** 17
- **Network radius:** 9
- **Path lenght:** 5.768
- **Clustering coeficiente:** 0.000
- **Network density:** 0.003
- **Network heterogeneity:** 2.220
- **Network centralization:** 0.067
- **Connected components:** 83

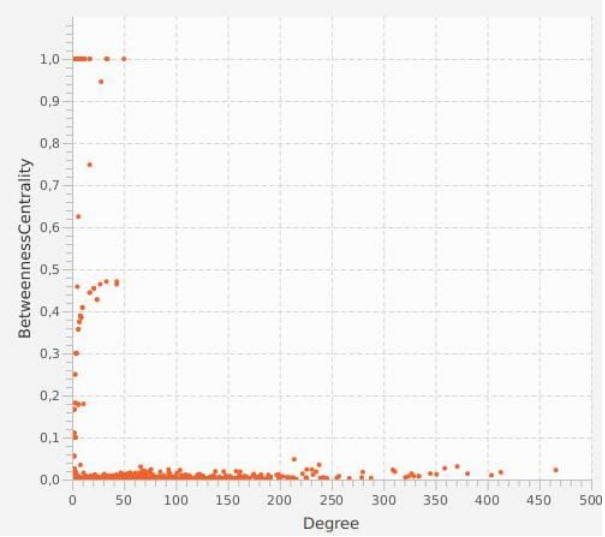


Fig. 11. **Betweenness of the graph**: Y axis represents the degree of vertices betweenes with the specified degree on the X axis. Information obtained by the graph presented.

As informed throughout the work, after creating the graph, TMHMM, SignalP and TargetP are implemented to the graph, specifically the graph that represents the union of the databases was used in these programs, the output for downloading the 3 programs can be obtained by this link: https://github.com/gustavocarnivali/Files.git. However, the outputs will be described below:

- **SignalP:** according to SignalP 7111 proteins have OTHER localization and 115 proteins have SP(Sec/SPI) localization. (caption on the same website).
- **TMHMM:** according to SignalP 6606 proteins have topology o and 607 proteins have topology i. (caption on the same website).
- **TargetP:** according to TargetP 6523 proteins have noTP localization, 444 proteins have mTP localization and 263 proteins have SP localization. (caption on the same website).

IV. Discussion

A tool that discovers new protein relationships between two different species has been proposed. For this, initially, the databases that should be used and that were presented were presented, and tools that were used to analyze the generated connections were presented.

Several tools used had an optimization in their speed. To speed up the protein sequence analyzer Diamond was used; To accelerate the slower algorithm produced by this work, a set of previously discussed strategies were used, which obtained a substantial gain in speed of the produced algorithms.

With the case study, it was possible to better understand the tool as well as its possible applications. It was possible to analyze and propose suggestions for analyzing the output offered by the program.

With the developed method, it was possible to obtain what was desirable: different proteins, from different species, that connected or, at least, have substantial similarity or functional connection.

References

[1] D. S. Dimitrov, Therapeutic proteins, Therapeutic Proteins (2012) 1–26.
[2] D. Cordy, The protein book, Naturegraph Books, 1976.
[3] A. Chatr-Aryamontri, R. Oughtred, L. Boucher, J. Rust, C. Chang, N. K. Kolas, L. O'Donnell, S. Oster, C. Theesfeld, A. Sellam, et al., The biogrid interaction database: 2017 update, Nucleic acids research 45 (D1) (2017) D369–D379.
[4] D. Szklarczyk, A. L. Gable, D. Lyon, A. Junge, S. Wyder, J. Huerta-Cepas, M. Simonovic, N. T. Doncheva, J. H. Morris, P. Bork, et al., String v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets, Nucleic acids research 47 (D1) (2019) D607–D613.
[5] A. F. Flórez, D. Park, J. Bhak, B.-C. Kim, A. Kuchinsky, J. H. Morris, J. Espinosa, C. Muskus, Protein network prediction and topological analysis in leishmania major as a tool for drug target selection, BMC bioinformatics 11 (1) (2010) 1–9.
[6] J.-M. Chang, J.-F. Taly, I. Erb, T.-Y. Sung, W.-L. Hsu, C. Y. Tang, C. Notredame, E. C.-Y. Su, Efficient and interpretable prediction of protein functional classes by correspondence analysis and compact set relations, PloS one 8 (10) (2013) e75542.
[7] A.-C. Gavin, M. Bösche, R. Krause, P. Grandi, M. Marzioch, A. Bauer, J. Schultz, J. M. Rick, A.-M. Michon, C.-M. Cruciat, et al., Functional organization of the yeast proteome by systematic analysis of protein complexes, Nature 415 (6868) (2002) 141–147.
[8] E. M. Phizicky, S. Fields, Protein-protein interactions: methods for detection and analysis, Microbiological reviews 59 (1) (1995) 94–123.
[9] L. Skrabanek, H. K. Saini, G. D. Bader, A. J. Enright, Computational prediction of protein–protein interactions, Molecular biotechnology 38 (2008) 1–17.
[10] M. A. Huynen, P. Bork, Measuring genome evolution, Proceedings of the National Academy of Sciences 95 (11) (1998) 5849–5856.
[11] I. Xenarios, D. W. Rice, L. Salwinski, M. K. Baron, E. M. Marcotte, D. Eisenberg, Dip: the database of interacting proteins, Nucleic acids research 28 (1) (2000) 289–291.
[12] E. M. Marcotte, M. Pellegrini, H.-L. Ng, D. W. Rice, T. O. Yeates, D. Eisenberg, Detecting protein function and protein-protein interactions from genome sequences, Science 285 (5428) (1999) 751–753.
[13] E. D. Harrington, L. J. Jensen, P. Bork, Predicting biological networks from genomic data, FEBS letters 582 (8) (2008) 1251–1258.
[14] A. J. Enright, I. Iliopoulos, N. C. Kyrpides, C. A. Ouzounis, Protein interaction maps for complete genomes based on gene fusion events, Nature 402 (6757) (1999) 86–90.
[15] T. Sato, Y. Yamanishi, M. Kanehisa, H. Toh, The inference of protein–protein interactions by co-evolutionary analysis is improved by excluding the information about the phylogenetic relationships, Bioinformatics 21 (17) (2005) 3482–3489.
[16] F. He, Y. Zhang, H. Chen, Z. Zhang, Y.-L. Peng, The prediction of protein-protein interaction networks in rice blast fungus, BMC genomics 9 (1) (2008) 1–12.
[17] S. Liu, Y. Liu, J. Zhao, S. Cai, H. Qian, K. Zuo, L. Zhao, L. Zhang, A computational interactome for prioritizing genes associated with complex agronomic traits in rice (oryza sativa), The Plant Journal 90 (1) (2017) 177–188.
[18] E. Nasiri, K. Berahmand, M. Rostami, M. Dabiri, A novel link prediction algorithm for protein-protein interaction networks by attributed graph embedding, Computers in Biology and Medicine 137 (2021) 104772.
[19] K. Berahmand, E. Nasiri, Y. Li, et al., Spectral clustering on protein-protein interaction networks via constructing affinity matrix using attributed graph embedding, Computers in Biology and Medicine 138 (2021) 104933.
[20] I. Korf, M. Yandell, J. Bedell, Blast, " O'Reilly Media, Inc.", 2003.
[21] O. Emanuelsson, S. Brunak, G. Von Heijne, H. Nielsen, Locating proteins in the cell using targetp, signalp and related tools, Nature protocols 2 (4) (2007) 953–971.
[22] H. Hermjakob, L. Montecchi-Palazzi, C. Lewington, S. Mudali, S. Kerrien, S. Orchard, M. Vingron, B. Roechert, P. Roepstorff, A. Valencia, et al., Intact: an open source molecular interaction database, Nucleic acids research 32 (suppl_1) (2004) D452–D455.
[23] G. Weber, G. Weber, B. Biochemiker, G. Weber, B. Biochemist, G. Weber, B. Biochimiste, Protein interactions, no. 574.19245 WEBp, Springer, 1992.

[24] I. M. Nooren, J. M. Thornton, Diversity of protein–protein interactions, The EMBO journal 22 (14) (2003) 3486–3492.
[25] B. K. Shoichet, W. A. Baase, R. Kuroki, B. W. Matthews, A relationship between protein stability and protein function., Proceedings of the National Academy of Sciences 92 (2) (1995) 452–456.
[26] T. Tang, X. Zhang, Y. Liu, H. Peng, B. Zheng, Y. Yin, X. Zeng, Machine learning on protein–protein interaction prediction: models, challenges and trends, Briefings in Bioinformatics 24 (2) (2023) bbad076.
[27] X. Hu, C. Feng, T. Ling, M. Chen, Deep learning frameworks for protein–protein interaction prediction, Computational and structural biotechnology journal 20 (2022) 3223–3233.
[28] M. Khatun, W. Shoombuatong, M. M. Hasan, H. Kurata, et al., Evolution of sequence-based bioinformatics tools for protein-protein interaction prediction, Current Genomics 21 (6) (2020) 454–463.
[29] S. Lim, Y. Lu, C. Y. Cho, I. Sung, J. Kim, Y. Kim, S. Park, S. Kim, A review on compound-protein interaction prediction methods: data, format, representation and model, Computational and Structural Biotechnology Journal 19 (2021) 1541–1556.
[30] B. P. Morgan, A. L. Harris, Complement regulatory proteins, Academic Press, 1999.
[31] B. Zhang, S. Horvath, A general framework for weighted gene co-expression network analysis, Statistical applications in genetics and molecular biology 4 (1) (2005).
[32] E. Yeger-Lotem, R. Sharan, Human protein interaction networks across tissues and diseases, Frontiers in genetics 6 (2015) 257.
[33] J. Ma, M. K. Yu, S. Fong, K. Ono, E. Sage, B. Demchak, R. Sharan, T. Ideker, Using deep learning to model the hierarchical structure and function of a cell, Nature methods 15 (4) (2018) 290–298.
[34] M. Hofree, J. P. Shen, H. Carter, A. Gross, T. Ideker, Network-based stratification of tumor mutations, Nature methods 10 (11) (2013) 1108–1115.
[35] N. Sahni, S. Yi, M. Taipale, J. I. F. Bass, J. Coulombe-Huntington, F. Yang, J. Peng, J. Weile, G. I. Karras, Y. Wang, et al., Widespread macromolecular interaction perturbations in human genetic disorders, Cell 161 (3) (2015) 647–660.
[36] R. Oughtred, C. Stark, B.-J. Breitkreutz, J. Rust, L. Boucher, C. Chang, N. Kolas, L. O'Donnell, G. Leung, R. McAdam, et al., The biogrid interaction database: 2019 update, Nucleic acids research 47 (D1) (2019) D529–D541.
[37] S. Nik-Zainal, H. Davies, J. Staaf, M. Ramakrishna, D. Glodzik, X. Zou, I. Martincorena, L. B. Alexandrov, S. Martin, D. C. Wedge, et al., Landscape of somatic mutations in 560 breast cancer whole-genome sequences, Nature 534 (7605) (2016) 47–54.
[38] A. Franceschini, D. Szklarczyk, S. Frankild, M. Kuhn, M. Simonovic, A. Roth, J. Lin, P. Minguez, P. Bork, C. Von Mering, et al., String v9. 1: protein-protein interaction networks, with increased coverage and integration, Nucleic acids research 41 (D1) (2012) D808–D815.
[39] M. E. Studham, A. Tjärnberg, T. E. Nordling, S. Nelander, E. L. Sonnhammer, Functional association networks as priors for gene regulatory network inference, Bioinformatics 30 (12) (2014) i130–i138.
[40] B. Snel, P. Bork, M. A. Huynen, The identification of functional modules from the genomic association of genes, Proceedings of the National Academy of Sciences 99 (9) (2002) 5890–5895.
[41] A. J. Enright, C. A. Ouzounis, Functional associations of proteins in entire genomes by means of exhaustive detection of gene fusions, Genome biology 2 (9) (2001) 1–7.
[42] G. La Barbera, A. L. Capriotti, E. Michelini, S. Piovesana, M. M. Calabretta, R. Zenezini Chiozzi, A. Roda, A. Laganà, Proteomic analysis and bioluminescent reporter gene assays to investigate effects of simulated microgravity on caco-2 cells, Proteomics 17 (15-16) (2017) 1700081.
[43] A.-L. Barabási, et al., Network science, Cambridge University Press, 2016.
[44] P. Feofiloff, Y. Kohayakawa, Y. Wakabayashi, Uma introdução sucinta à teoria dos grafos (2011).
[45] B. Buchfink, K. Reuter, H.-G. Drost, Sensitive protein alignments at tree-of-life scale using diamond, Nature methods 18 (4) (2021) 366–368.
[46] B. Buchfink, C. Xie, D. H. Huson, Fast and sensitive protein alignment using diamond, Nature methods 12 (1) (2015) 59–60.
[47] P. O. B. Netto, Grafos: teoria, modelos, algoritmos, Editora Blucher, 2003.
[48] F. Amaral, R. Ladeira, Barabási e as redes: Uma infraestrutura dinâmica para a complexidade, Revista Inteligência Empresarial (2021).
[49] A. Lulli, L. Ricci, E. Carlini, P. Dazzi, Distributed current flow betweenness centrality, in: 2015 IEEE 9th International Conference on Self-Adaptive and Self-Organizing Systems, IEEE, 2015, pp. 71–80.
[50] G. S. Carnivali, A. B. Vieira, A. Ziviani, P. A. Esquef, Covec: coarse-grained vertex clustering for efficient community detection in sparse complex networks, Information Sciences 522 (2020) 180–192.
[51] G. S. Carnivali, A. B. Vieira, P. A. Esquef, A. Ziviani, Método rápido de agrupamento de vértices para detecçao de comunidades em redes complexas de larga-escala, in: Anais do XVII Workshop em Desempenho de Sistemas Computacionais e de Comunicação, SBC, 2018.
[52] A. Kirilov, Introdução a teoria de conjuntos (2017).

.

7.3 Artigo em produção: Predição de interação entre o fungo causador da ferrugem asiática e a soja

IJAMML 15:1 (2021) 53-67 September 2021
ISSN: 2394-2258
Available at http://scientificadvances.co.in
DOI: http://dx.doi.org/10.18642/ijamml_710012230

MACHINE LEARNING METHOD TO DIFFERENTIATE ATAXIAS

Gustavo Simões Carnivali

Universidade de Minas Gerais - Belo Horizonte - Brazil, Av. Pres. Antônio Carlos, 6627 - Pampulha, Belo Horizonte - MG, 31270-901, Brazil

Abstract

Spinocerebellar ataxias or SCAs, are a group of more than 37 genetically and clinically heterogeneous known neurodegenerative diseases. This work analyzes the level of genetic similarity between several ataxias, we identified proteins that are associated with more than one ataxia. A decision tree was trained to identify ataxias by identifying whether a new entry disease not yet identified and not classified can be grouped as an ataxia. Altogether 12 proteins from different ataxias were verified, all 12 proteins were analyzed in 500 different trees to better evaluate the method used. Of the 12 proteins tested, the method was correct for 10 different proteins or 83% of correct results. This identifier and the results obtained in the experiments allow a greater characterization of the diseases addressed, it also allows applications such as the reuse of treatments for similar diseases.

Keywords: machine learning, ataxias, decision tree.

*Corresponding author.
E-mail address: gustavocarnivali@gmail.com (G. S. Carnivali).

2020 Mathematics Subject Classification: 68.
Submitted by Jianqiang Gao.
Received September 21, 2021; Revised September 30, 2021

1. Introduction

Spinocerebellar ataxias or SCAs are a group of more than 37 known neurodegenerative diseases that are genetically and clinically heterogeneous. The most common type among SCAs has an occurrence of 1 to 5 cases per 100,000 people. They commonly affect the nervous system, causing loss of coordination [1].

This work analyzes the level of genetic similarity between several ataxias by identifying proteins that are associated with more than one of them. Once, it has been shown that ataxias have many proteins in common [2]. In the study of [3], it was shown that ataxias also have more genes related to each other compared to other diseases. This study intends to evolve the studies carried out in [2, 3], using machine learning tools to characterize the ataxias presented.

Studies have shown that genes can be related to other genes, that is, the increase in the action of one gene can lead to an increase or decrease in the phenotypic effects of another gene [4]. From this feature a network of gene or protein interactions can be generated. Thus, a decision tree was trained to identify ataxias in order to establish whether or not an as-yet unidentified and unclassified new entry disease can be grouped as an Ataxia [5].

Some studies such as [16] already use machine learning methods to better analyze diseases, the study presented here can also be used for this purpose, but its main objective, not addressed in [16], as well as not in related works, is the comparison between already known diseases. Studies such as [17, 18, 19] uses machine learning methods to compare diseases and can and were used as a basis for the composition of this study, but they do not present comparative studies on the studied diseases (Ataxias), as well as, do not use genetic characteristics in comparisons. This study therefore generates a new method that can be applied to other diseases because it is precise and uses a quick and simple data structure, but specifically, in this study, it was applied to a set of diseases not yet studied by machine learning methods.

This classification allows for greater understanding of the group of diseases studied and also allows applications such as the reuse of drugs/treatments for similar diseases that were or will be analyzed by the generated tree, as well as an understanding of why this indication is functional based on genetic factors of the disease [6].

2. Methodology

2.1. Protein networks

Protein relationships can occur in different ways in our organism, a common way, which can be mentioned, is the gene co-expression [7]. Gene expression can be interpreted as the process by which DNA nucleotide sequences are transcribed into RNA used by the cell or RNAs that are translated into proteins [8]. A change in the expression of one gene can increase or decrease the expression of other genes [9]. This correlation between genes can be represented by an interaction graph. A graph is a mathematical structure G = (V, E) where V is a non-empty set of vertices that can represent genes or proteins for example and E are edges that connect two vertices and indicate a relationship between them, just as they can represent a co-expression relationship between two genes [1]. An edge, in a genetic or protein network, may have associated with it a weight that corresponds to the confidence of the co-expression of the gene [7].

In this work we use the String-DB database to generate the biological networks. String-DB is a database containing thousands of protein-protein interactions. It also includes a score that associates a degree of confidence in its occurrence to each interaction. String-DB calculates this score, a value between 0 and 1, using different prediction approaches and different databases such as NCBI Gene Expression Omnibus, ProteomeHD, PubMed, Ensembl, SwissProt [7]. We consider this confidence value as the weight of the edges of our graph G. Only connections with a confidence above the threshold defined by 0.5 were considered to increase reliability.

In Figure 1, a biological network obtained by the string platform for the connections of the protein that causes squamous cell ataxia type 1 (ATXN1) is represented.

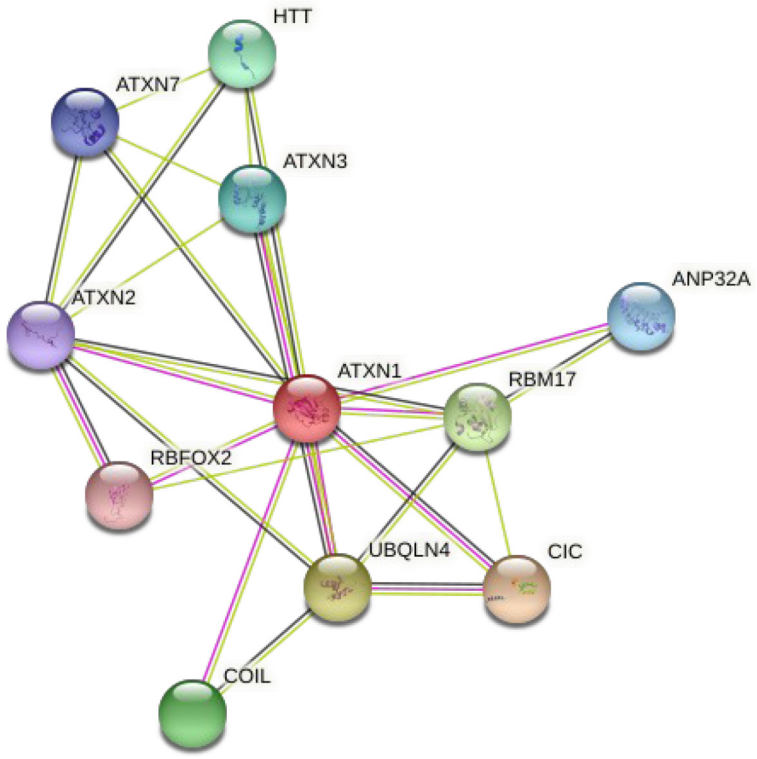


Figure 1. Main connections of protein caused of the Spinocerebellar ataxia type 1 (ATXN1). Edge scores are not shown in the figure, but for example, score between ATXN7 protein and HTT explained by PubMed ([11]): "score 0.810. Putative homologs are mentioned together in other organisms (score 0.092)".

2.2. Representation: Graph as a vector

The machine learning methods used require the input data to be vectors with characteristics. Unfortunately a graph is not naturally synthesized by this format. There are programs that use the graph

structure to create feature vectors that can be used as input to machine learning methods [11]. In this study, the program presented in [11] called NBNE (Neighborhood Based Node Embeddings) was used. The NBNE in its own study is compared to other current methods and surpasses them. NBNE creates a vector with 128 features for each vertex of the graph.

Unfortunately, NBNE does not allow an easy interpretation of the created vectors, that is, NBNE creates a vector for each edge of the graph with values that imply complex properties of the network topology, these properties, however, do not allow an easy biological interpretation of the values found, its main focus is on the quality of the representation and not on the interpretation of the results generated. However, NBNE was chosen in this work because it is a current method and presents degrees of efficiency in similar applications that are more suitable than other existing methods [11].

2.3. Machine learning methods

The method used in this work is classified as supervised. This is also possible thanks to the representation presented in Subsection 2.2, which allows the representation of a graph as a numerical vector of features. This class of methods features a training phase and a testing phase. In the training phase, in general, the obtained data are divided and, based on classical methods, a machine is taught to classify new data based on the teaching generated by the previous data [12]. In the case of this work, we want to develop a machine capable of telling whether a new input protein causes an ataxia or not. For this, the training set will contain genes from the input graph that are linked or not to an ataxia. Therefore, the objective is to know whether the data used to develop the network are sufficient to identify this class of diseases and, in parallel, identify the characteristics that differentiate this group of diseases from others.

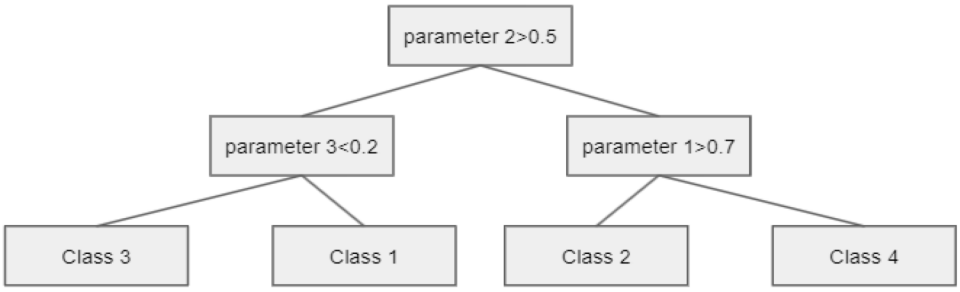


Figure 2. Example decision tree.

A recurrent problem that can occur in machine learning is that of overfitting. In our context, a method that produces overfitting would generate a machine that would memorize the input data, being unable to recognize other ataxias, besides those used as input, which have few significant differences. Therefore, overfitting should be avoided in this study, aiming to reduce this problem in this work, a method that classically generates little overfitting was chosen [12].

The machine learning method used is the decision tree. This method was chosen over the others because it presents an easy biological interpretation of the results obtained, that is, from the characteristics of the tree it can be more easily implicated in the characteristics of the studied diseases. Therefore, in addition to the main objective of this study of classifying diseases, deep interpretations and the generation of the tree will allow for a better understanding of this class of diseases that is still poorly studied and known (i.e., ataxias) [13].

In addition, the tree used has already obtained several improvements. Comparing it to other methods by the accuracy metric that will be described in the future, the tree managed to obtain significant quality results in relation to the other tested methods.

2.4. Decision tree

The decision tree is a tree of choices about stacked decisions. At each node the tree asks a question according to the input data parameters, if the answer is positive the user goes to the left of the tree and if the answer is negative the user goes to the right of the tree. The process is repeated until the user arrives at a node that no longer has children called a leaf [13].

A visual example of a tree can be seen in Image 2. For example, if the input data has a value of parameter 2 greater than 0.5 it will go to the node to the right of the current node, if the value of this parameter is smaller that 0.5 it will go left. As an application example, for this tree, if a data has a parameter 2 value less than 0.5 and a parameter 3 value greater than 0.2 it will be classified as class 1.

The continuous growth of the tree with many decision nodes can generate overfitting because the tree would decorate the input data, preventing its generalizability [13]. An adequate height of the tree must be chosen by the user from the knowledge of his data [13]. It will be seen that in fact the tree generated by this study does not have a great height.

For the creation of the decision tree (i.e., correct positioning of the nodes) a classic method used in this work can be used. Basically, the method calculates the entropy of each question and gives preference to questions with higher entropy to assume higher positions in the tree. Entropy, among other things, considers the number of data that will be directed to the right or left child in a tree [13].

The accuracy metric will be used in this work, it serves to identify the efficiency of the method used. After the method is trained, some test data are delivered to the method, if it manages to classify these data correctly its accuracy increases, if its classification is wrong its accuracy decreases linearly, that is, for example, 100% accuracy means that the algorithm got all the possibilities right, 50% accuracy means that the algorithm got half of all the possibilities right.

2.5. Cross validation

In order to improve probabilistic results about the efficiency of a method on a dataset, cross-validations can be used. Owned input data can be divided into 2 groups: training data and testing data, but how does this division occur? From iterative methods, training and testing groups are created based on the input data and different at each iteration, thus allowing the evaluation of the method in a more independent way from the data used.

In this study, we want to know which data will be present in each set, in order to be able to characterize the Ataxias exclusively. Therefore, no classical cross-validation method will be used, but your idea, for bringing benefits to the experiments, will be used to assemble the test and training sets used. How the sets were assembled will be better described in the next section of Experiments (Section 4).

3. Algorithm

The algorithm that implements the project was entirely made in Python. Python has libraries that allow easy use of machine learning methods, including the decision tree. From sklearn, the code presented by Algorithm 1 was produced and it manages to synthesize all the methodology proposed by this work.

The algorithm (exposed by Algorithm 1) basically reads and stores the input data in lines 1 to 3 using the NBNE. Lines 4 to 6 create and use the tree based on functions provided by the sklearn library. Finally, line 7 calculates the final result. Naturally, Algorithm 1 has been simplified by hiding unnecessary secondary functions for understanding the method.

Algorithm 1: Decision tree

1. X train = <reading the data> % input data read

2. Y train = <reading the data>

3. Y pred = <reading the data>

4. model = Decision TreeClassifier() %tree creation

5. model = model.fit (Xtrain, Ytrain)

6. Y pred = model.predict(Xtest) %creation of the prediction variable

7. result = result + (accurancy_score(Ytest, Ypred))% accuracy calculation

Algorithm 1: Proposed method.

4. Experiments and Results

In this study, we used a set of 12 proteins directly linked to some ataxia. The types of ataxia used are considered monogenetic diseases that are caused by the mutation of a single gene, for each ataxia used, at least one protein is known to cause or influence it. The proteins used are described in Table 1 [15]. Another set of 24 proteins not directly bound to ataxias and chosen at random from all the existing set of proteins in the human body was used. Initially, a training group was created containing 9 proteins from the first group and 18 proteins from the second, chosen at random from the whole set, the random subdivision was done 30 times with the aim of bringing more confidence to the results found.

The decision tree was applied to this iterated set of 27 proteins as described above. The remaining pool of 9 proteins was used as a test to evaluate the method used. To assess tree size, 500 trees with sizes ranging from 1 to 500 were generated, one tree for each specified size. For

the 500 trees with 30 iterations each, the highest average accuracy obtained for the specified data was 0.89.[1]

Each tree height level obtains a distinct accuracy value. The variance of the calculated accuracy of all 500 trees generated in the algorithm was only 0.0016 showing that: the tree quality varied little with the change of its height, the tree quickly obtained good quality even with few nodes and, finally, the tree did not significantly present the overfitting problem. To facilitate viewing the result, it is shown in Figure 3.

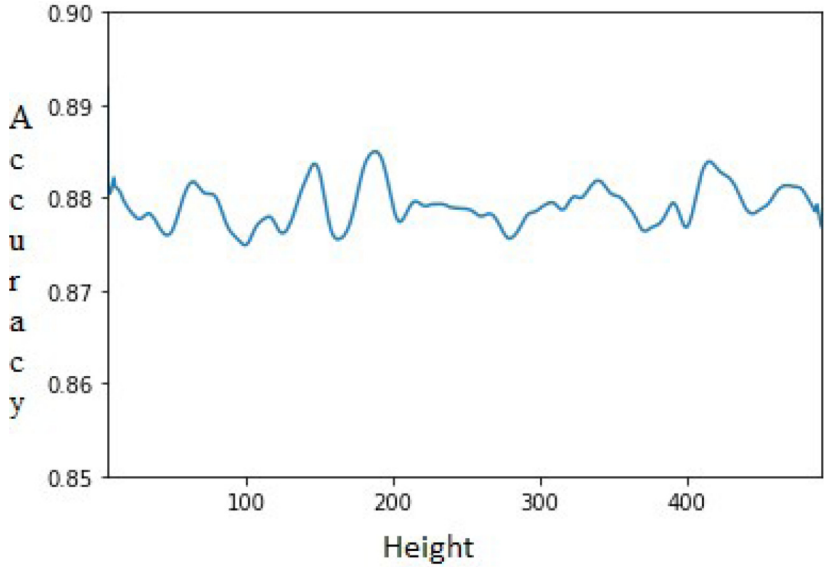


Figure 3. Accuracy value variation in a decision tree with height variation from 1 to 500.

[1]The average was used because each tree generated 30 results, one for each iteration. Measures such as variance or confidence interval will not be presented throughout the work as they obtained low and irrelevant values for the results found.

The characteristic of getting a good result even with low tree height can be a property of ataxias. If the ataxias have more similarities as described in studies [2, 3] the decision tree, even with few vertices, would achieve a good result. Furthermore, based on this result, one can assume the existence of a strong characteristic that approximates the Ataxias that was found by the decision tree and that was used by it, thus, even with the addition of new vertices, new choices did not generate other significant separations. However, all results can also be explained by technical factors such as the low dimensionality and diversity of training and test data.

A second set of tests was performed. In order to determine which of the tested ataxias differs the most from the others in the test set, 12 sets with all 24 chosen proteins plus 12 – 1 = 11 ataxias proteins were created. Twelve experiments were carried out removing in each experiment a protein related to an ataxia. The tree was trained with the height that obtained the best accuracy performance in the previous test. The result for the 12 proteins can be seen in Table 1.

Table 1. As predicted by the previous experiment, this experiment achieved a high hit rate of 83%. Among the 12 ataxias tested, only 2 had misclassification, Ataxias of type 10 and 11 represented by proteins ATXN10 and TTBK2

Table 1. Search results

Disease	SCA2	SCA2	SCA3	SCA17	SCA3	SCA7
Protein	ATXN2	ATXN2L	ATXN3L	TBP	ATXN3	ATXN7L2
Result	Right	Right	Right	Right	Right	Right

Disease	SCA36	SCA7	SCA10	SCA11	SCA1	SCA7
Protein	NOP56	ATXN7	ATXN10	TTBK2	ZNF674	ATXN7L3
Result	Right	Right	Missed	Missed	Right	Right

5. Conclusion

This work presented a way to classify, from a machine learning method, a set of neurodegenerative and genetic diseases from a set of a biological interaction network, specifically, the set of diseases called ataxias was studied.

The machine learning method used was the decision tree. This method was chosen mainly for its easy interpretability and explainability, but also for obtaining adequate results in relation to other existing methods.

This work performed two sets of experiments on a set of 12 proteins acting on several ataxias. From the first experiment, it was possible to conclude that the ataxias are well separated even by decision trees with few vertices, showing the existence of a possible strong similarity between the Ataxias. As a result of the second set of experiments, a possible lesser similarity was seen between ataxias of types 10 and 11 in relation to the other ataxias tested considering the proteins used.

In the future, the data and methodology presented may support comparative studies of other diseases. The classifier generated, despite having good experimental quality, can be retrained using more proteins not bound to ataxias or, if new studies are carried out that analyze this class of diseases, other proteins also bound to ataxias can be used, as well, to other proteins from diseases close to the ataxias.

6. Acknowledgement

Capes, Fapemig and UFMG.

References

[1] H. L. Paulson, The spinocerebellar ataxias, Journal of Neuro-Ophthalmology: The Official Journal of the North American Neuro-Ophthalmology Society 29(3) (2009), 227-237.

DOI: https://doi.org/10.1097/WNO0b013e3181b416de

[2] J. Lim, T. Hao, C. Shaw, A. J. Patel, G. Szabó, J.-F. Rual, C. J. Fisk, N. Li, A. Smolyar, D. E. Hill, A.-L. Barabási, M. Vidal and H. Y. Zoghbi, A protein-protein interaction network for human inherited ataxias and disorders of Purkinje cell degeneration, Cell 125(4) (2006), 801-814.

DOI: https://doi.org/10.1016/j.cell.2006.03.032

[3] G. S. Carnivali and S. V. A. Campos, Does the ataxia group have genetic similarities?, Anais do XIII Encontro Academico de Modelagem Computacional (2020), 135.

[4] B. Snel, G. Lehmann, P. Bork and M. A. Huynen, STRING: A web-server toretrieve and display the repeatedly occurring neighbourhood of a gene, Nucleic Acids Research 28(18) (2000), 3442-3444.

DOI: https://doi.org/10.1093/nar/28.18.3442

[5] M. De Souto, A. Lorena, A. Delbem and A. de Carvalho, Tecnicas de aprendizado de maquina para problemas de biologia molecular, Sociedade Brasileira de Computacao 1(2) (2003).

[6] M. Fatima and M. Pasha, Survey of machine learning algorithms for disease diagnostic, Journal of Intelligent Learning Systems and Applications 9(1) (2017), 1-16.

DOI: https://doi.org/10.4236/jilsa.2017.91001

[7] D. Szklarczyk, A. Franceschini, S. Wyder, K. Forslund, D. Heller, J. Huerta-Cepas, M. Simonovic, A. Roth, A. Santos, K. P. Tsafou, M. Kuhn, P. Bork, L. J. Jensen and C. Von Mering, STRING v10: protein–protein interaction networks, integrated over the tree of life, Nucleic Acids Research 43(D1) (2015), 447-452.

DOI: https://doi.org/10.1093/nar/gku1003

[8] A. Brazma and J. Vilo, Gene expression data analysis, FEBS Letters 480(1) (2000), 17-24.

DOI: https://doi.org/10.1016/s0014-5793(00)01772-5

[9] S. Horvath and J. Dong, Geometric interpretation of gene coexpression network analysis, PLoS Computational Biology 4(8) (2008); e1000117.

DOI: https://doi.org/10.1371/journal.pcbi.1000117

[10] G. Goeckenjan, H. Sitter, M. Thomas, D. Branscheid, M. Flentje, F. Griesinger, N. Niederle, M. Stuschke, T. Blum, K.-M. Deppermann, J. H. Ficker, L. Freitag, A. S. Lübbe, T. Reinhold, E. Späth-Schwalbe, D. Ukena, M. Wickert, M. Wolf, S. Andreas, T. Auberger, R. P. Baum, B. Baysal, J. Beuth, H. Bickeböller, A. Böcking, R. M. Bohle, I. Brüske, O. Burghuber, N. Dickgreber, S. Diederich, H. Dienemann, W. Eberhardt, S. Eggeling, T. Fink, B. Fischer, M. Franke, G. Friedel, T. Gauler, S. Gütz, H. Hautmann, A. Hellmann, D. Hellwig, F. Herth, C. P. Heußel, W. Hilbe, F. Hoffmeyer, M. Horneber, R. M. Huber, J. Hübner, H.-U. Kauczor, K. Kirchbacher, D. Kirsten, T. Kraus, S. M. Lang, U. Martens, A. Mohn-Staudner, K.-M. Müller, J. Müller-Nordhorn, D. Nowak, U. Ochmann, B. Passlick, I. Petersen, R. Pirker, B. Pokrajac, M. Reck, S. Riha, C. Rübe, A. Schmittel, N. Schönfeld, W. Schütte, M. Serke, G. Stamatis, M. Steingräber, M. Steins, E. Stoelben, L. Swoboda, H. Teschler, H. W. Tessen, M. Weber, A. Werner, H.-E. Wichmann, E. Irlinger Wimmer, C. Witt and H. Worth, Prävention, Diagnostik, Therapie und Nachsorge des Lungenkarzinoms, Pubmed Results, Pneumologie 65(8) (2011), e51-e75.

DOI: https://doi.org/10.1055/s-0030-1256562

[11] T. Pimentel, A. Veloso and N. Ziviani, Fast node embeddings: Learning egocentric representations, 2018.

[12] M. C. Monard and J. A. Baranauskas, Conceitos sobre aprendizado de maquina, Sistemas Inteligentes: Fundamentos e Aplicacoes 1(1) (2003), 32.

[13] J. Gama, Arvores de decisao, Palestra ministrada no Nucleo da Ciencia de Computacao da Universidade do Porto, Porto, 2002.

[14] G. Keijzers, D. Bakula and M. Scheibye-Knudsen, Monogenic diseases of DNA repair, New England Journal of Medicine 377(19) (2017), 1868-1876.

DOI: https://doi.org/10.1056/NEJMra1703366

[15] J. M. Stuart, E. Segal, D. Koller and S. K. Kim, A gene-coexpression network for global discovery of conserved genetic modules, Science 302(5643) (2003), 249-255.

DOI: https://doi.org/10.1126/science.1087447

[16] A. Dagliati, S. Marini, L. Sacchi, G. Cogni, M. Teliti, V. Tibollo, Pasquale de Cata, Luca Chiovato, and Riccardo Bellazzi, Machine learning methods to predict diabetes complications, Journal of Diabetes Science and Technology 12(2) (2018), 295-302.

DOI: https://doi.org/10.1177/1932296817706375

[17] S. Uddin, A. Khan, M. E. Hossain and M. A. Moni, Comparing different supervised machine learning algorithms for disease prediction, BMC Medical Informatics and Decision Making 19(1) (2019), 1-16.

DOI: https://doi.org/10.1186/s12911-019-1004-8

[18] S. Pouriyeh, S. Vahid, G. Sannino, G. De Pietro, H. Arabnia and J. Gutierrez, A comprehensive investigation and comparison of machine learning techniques in the domain of heart disease, In 2017 IEEE Symposium on Computers and Communications (ISCC) (2017), pp. 204-207.

DOI: https://doi.org/10.1109/ISCC.2017.8024530

[19] M. Brahimi, K. Boukhalfa and A. Moussaoui, Deep learning for tomato diseases: Classification and symptoms visualization, Applied Artificial Intelligence 31(4) (2017), 299-315.

DOI: https://doi.org/10.1080/08839514.2017.1315516

■

GSC Advanced Research and Reviews

eISSN: 2582-4597 CODEN (USA): GARRC2

Cross Ref DOI: 10.30574/gscarr

Journal homepage: https://gsconlinepress.com/journals/gscarr/

(CASE REPORT)

The use of antibiotic drugs associated with the onset of symptoms of Spinocerebellar Ataxia disease

Gustavo Simões Carnivali *

Department of Biology, Universidade Federal de Minas Gerais (UFMG), Brazil.

GSC Advanced Research and Reviews, 2022, 12(01), 145–147

Publication history: Received on 14 June 2022; revised on 20 July 2022; accepted on 22 July 2022

Article DOI: https://doi.org/10.30574/gscarr.2022.12.1.0189

Abstract

This case report reports the onset of symptoms of a disease. Spinocebellar Ataxia (unknown type) is studied, a genetic, neurodegenerative disease that commonly starts in adulthood. A case is reported in which the onset of symptoms of this disease, in the patient, possibly starts after the ingestion of a set of antibiotics in a short period of time. Several works in the literature report the onset of symptoms of this disease by the use of antibiotics. This study reaffirms previous studies.

Keywords: Spinocerebellar Ataxia; Antibiotics; Start; Case report

1. Introduction

This case study studies the onset of symptoms in a patient with Spinocerebellar Ataxia. The disease in question (Spinocerebellar Ataxia) is one in a group of diseases named Ataxias, they are all characterized by the gradual loss of muscular abilities, such as the ability to walk or write, and can also influence secondary systems such as vision or the digestive system [1 - 3]. It is estimated that there are 1 to 5 cases of ataxia per 100,000 population worldwide and 37 different types of known ataxias [4, 5]. The symptoms of Ataxia commonly start in adulthood [3]. In the family of the patient studied at close to 30 years of age. However, because it is a genetic disease, symptoms can occur at any time in the carrier's life. For this reason, it is common to try to find the reason, in addition to age, for which symptoms of this and other diseases start. In this context, one hypothesis is that the use of drugs classified as antibiotics may be associated with the onset of symptoms of this disease [6 - 10]. In this study, a specific case is analyzed. The symptoms of Spinocerebellar Ataxia are initiated, in the studied patient, by taking a large set of antibiotics, in a short and unusual period of time. This reinforces the idea mentioned above, that the symptoms of Spinocerebellar Ataxias diseases can start with the use of antibiotics.

2. Case Report

The patient studied was, at the time of writing this article, 28 years old, defined as white, male and Brazilian of Brazilian origin. The same reports that he does not have any disease that can resemble Ataxia, also reports that he does not have a disease associated with the onset of Ataxia symptoms. The patient's mother, who died, also had the disease, as did his deceased maternal uncle and maternal grandfather. The patient's family members did not perform the genetic exam, making it impossible to specify the type of ataxia present in the family. However, all had clinical confirmation of the case, presenting the symptoms to specialist professionals. Specifically, the patient studied performed the tests named: Magnetic Resonance and Computed Tomography. Both exams indicated the possible existence of Spinocerebellar Ataxia, both exams presented the common origin of the symptoms (i.e., slight reduction of the cerebellum). The results

* Corresponding author: Gustavo Simões Carnivali
epartment of Biology, Universidade Federal de Minas Gerais (UFMG), Brazil.

of the specified exams in Portuguese can be found attached to this article. The patient, specified, reports that he did not feel the symptoms common to the disease for more than 3 years, in relation to the date of writing of this article. The patient also reports the onset of symptoms after treatment for an unspecified illness. Among other drugs taken, the patient took, not together, in the period of 2 months (i.e., approximately 60 days), the 3 drugs, characterized as antibiotics, reported below. The patient took the 3 drugs in the order they are presented.

2.1. Amoxicillin

Amoxicillin is a penicillin derivative used for the treatment of infections caused by gram-positive bacteria, in particular streptococcal bacteria that cause upper respiratory tract infections [11]. Amoxicillin was taken by the patient for a total period of 10 days, 2 tablets of 500 mg per day.

2.2. Azithromycin

Azithromycin is a macrolide antibiotic used to treat a variety of bacterial infections [11]. Azithromycin was taken by the patient for a total period of 3 days, 1 tablet of 500 mg per day.

2.3. Amoxicillin + Potassium Clavulanate

Amoxicillin variation. Amoxicillin + Potassium Clavulanate was taken by the patient for a total period of 10 days, 3 tablets of 500 mg per day.

3. Conclusion

This case report presented the specific case of a man with Spinocerebellar Ataxia disease. The disease, common to other members of his family, had a different onset in him. This study therefore reports how the onset of symptoms occurred in this patient. This report can support and help to reaffirm the hypothesis that reports how the symptoms of this disease can start. Naturally, this report may also allow a better understanding of this disease, indirectly supporting several other studies that created new hypotheses about it. The author of this study indicates that new case reports, with the same or different objective, be written and published. It is also indicated that other studies, in addition to case studies, be carried out on this disease, in order to understand different processes in it and in other diseases.

Compliance with ethical standards

Acknowledgments

We acknowledged to FAPEMIG, Capes and UFMG.

Statement of informed consent

The consent was obtained from participant included in the study.

Permissions

This report as the publication of the attached exams were authorized by the patient studied.

References

[1] Bhandari, Jenish, Pawan K. Thada, Debopam Samanta. Spinocerebellar ataxia. StatPearls [Internet]. StatPearls Publishing. 2020; © 2020 [cited 2020 July 26]. Available from https://www.ncbi.nlm.nih.gov/books/NBK557816/

[2] Lloyd, Deanah. At-risk individual's perspectives of Spinocerebellar Ataxia (SCA) Presymptomatic Testing (PT) [MS Thesis]. Faculty of Health Sciences. 2021.

[3] Brooker, Sarah M., et al. Spinocerebellar ataxia clinical trials: opportunities and challenges." Annals of clinical and translational neurology. 2021; 8(7): 1543-1556.

[4] Paulson, Henry L. The spinocerebellar ataxias. Journal of neuro-ophthalmology: the official journal of the North American Neuro-Ophthalmology Society. 2009; 29(3): 227.

[5] Carnivali, Gustavo. Machine Learning Method to Differentiate Ataxias. International Journal of Applied Mathematics and Machine Learning. 2021; 15: 53-67.

[6] Matibag JL, CC Diesta, E Uematsu. Correlation of trinucleotide repeat with SARA score, age of onset and disease duration of genetically-positive spinocerebellar ataxia 2 Filipino family." Parkinsonism & Related Disorders. 2020; 79: e111.

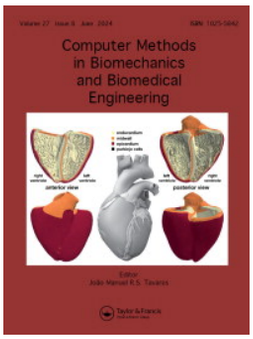
[7] Wantaneeyawong C, S Tanprawate, A Nudsasarn. Metronidazole induced cerebellar ataxia: first 2 case reports from Thailand. Parkinsonism & Related Disorders. 2020; 79: e111.

[8] Chittilappilly, Jomal Mathew, Jerry Jose Kalampadan, and Sholy Vareed Kaitharath. Cerebellar ataxia following antitubercular therapy. Neurology and Clinical Neuroscience. 2022; 160-162.

[9] Snavely S, Hodges G. The neurotoxicity of antibacterial agents. Ann Intern Med. 1984; 101: 92–104.

[10] Grill MF, Maganti RK. Neurotoxic effects associated with antibiotic use: management considerations. British Journal of Clinical Pharmacology. 2011; 72: 381-393.

[11] Wishart, David S., et al. DrugBank 5.0: a major update to the DrugBank database for 2018. Nucleic acids research. 2018; 46(D1): D1074-D1082.



Taylor & Francis
Taylor & Francis Group

Computer Methods in Biomechanics and Biomedical Engineering

ISSN: (Print) (Online) Journal homepage: www.tandfonline.com/journals/gcmb20

Method to link medicines to diseases using multiplex networks

Gustavo Simões Carnivali & Carlos Cristiano Borges

To cite this article: Gustavo Simões Carnivali & Carlos Cristiano Borges (22 Jun 2024): Method to link medicines to diseases using multiplex networks, Computer Methods in Biomechanics and Biomedical Engineering, DOI: 10.1080/10255842.2024.2362860

To link to this article: https://doi.org/10.1080/10255842.2024.2362860

Published online: 22 Jun 2024.

COMPUTER METHODS IN BIOMECHANICS AND BIOMEDICAL ENGINEERING
https://doi.org/10.1080/10255842.2024.2362860

Method to link medicines to diseases using multiplex networks

Gustavo Simões Carnivali[a] and Carlos Cristiano Borges[b]

[a]Department of Biology, UFMG, Belo Horizonte, Brazil; [b]Department of Computer Science, UFJF, Juiz de Fora, Brazil

ABSTRACT

The reuse of well-established medicines using computational modeling has gained a lot of attention due to its tremendous benefits. Based on this perspective, a new method for linking known medicines to diseases is proposed. The creation of a new treatment or medicine can be financially and temporally costly and the reuse of medicines is one possibility to accelerate this process efficiently. The main purpose of the reuse of medicines is to reduce some stages of the development of new medicines, motivating the proposition of several methods nowadays. In this work, a new method is developed aiming to connect known medicines to diseases based on available networks of protein interactions and available lists of medicines that affect protein action. The concepts of multiplex networks are used to connect subgraphs of vertices that represent medicines and proteins. The core of the procedure is determined by a weighting strategy constructed to define precisely the more relevant connections. The method was compared to other network link methods in the literature and a case study was presented and evaluated by the proposed method.

ARTICLE HISTORY
Received 25 January 2024
Accepted 28 May 2024

KEYWORDS
New method; medicine reuse; biological networks; multiplex networks

1. Introduction

The development of a new medicine or the creation of a new treatment for a disease is temporally and financially expensive (da et al. 2006; Calixto and Siqueira Junior 2008; I. F. of Pharmaceutical Manufacturers and Associations 2014). According to INTERFARMA (Pharmaceutical Research Industry Association) and BNDES (National Bank for Economic and Social Development) (https://www.bndes.gov.br/wps/portal/site/home) the availability of a new medicine in Brazil it takes an average of 14 years. The average investment to enter a new medicine on the market, is around US$1.2 to US$1.4 billion dollars (Pinto 2017; Carvalho 2021).

Another possibility, faster and cheaper, for the development of a treatment is to carry out the repositioning of known medicines (Muthyala 2011). Medicine repositioning is an indication of a medicine for a disease for which it was not initially designed or developed (McRae et al. 2016). According to the study (Farha and Brown 2019), the classic stages of medicine development for the USA can be seen in Figure 1(A), in parallel, the Figure 1(B) presents the classic steps common to medicine reuse. You can see a significant reduction in the total time spent.

At the beginning of a medicine repositioning project, pre-clinical information (pharmacological, toxicological, etc.), clinical efficacy and safety information is already available, making it possible to reduce or remove these stages of medicine development (Rudrapal et al. 2020; Gil and Martinez 2021).

There are several examples of medicines that have been repositioned. For example, the medicine Thalidomide was created in 1954 to act as a hypnotic-sedative. In 1965, a dermatologist used it to treat a patient's insomnia, and observed an improvement in the inflammatory condition. Later, the anti-inflammatory action of Thalidomide was proven. In 1998, the Food and medicines Administration (FDA) approved the use of Thalidomide for the treatment of ENL (Erythema Nodosum Leprosum) (Silva 2015). Other examples include Duloxetine, which was originally developed for depression and is now in the US FDA as a first-class therapy for stress urinary incontinence. Dapoxetine, initially developed as an antidepressant, is undergoing Phase III clinical trials as a first-class therapy for premature ejaculation (Ashburn and Thor 2004).

The objective of this work is: to create a score indicative of the effect of a medicine on one or more human proteins, based on probabilistic values offered

CONTACT Gustavo Simões Carnivali gustavocarnivali@gmail.com

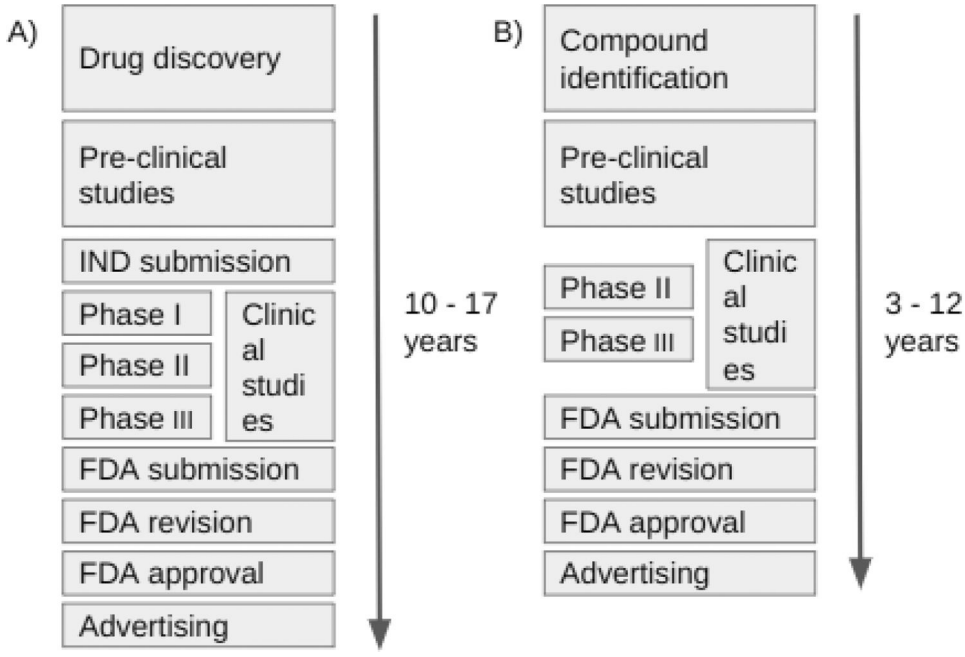


Figure 1. Stages of medicine reuse: (A) stages of classic medicine development, common to different regions of the world and (B) common steps for medicine reuse. FDA: Food and Medicine Administration; IND: investigational New Medicine.

by other databases, on proteins that are directly or indirectly linked to a medicine. Specifically, a graph of protein interactions with values between 0 and 1 was downloaded, which is intended to be connected to another graph downloaded of medicines that act on proteins. It is expected, based on the portals used, that medicines with higher scores are more likely to act on the studied proteins. Finally, we want to develop and suggest a program that achieves the listed objectives. The program will be analyzed in relation to classical computational properties.

1.1. Complex networks

The world is full of intrinsically complex systems. For example, human bodily functioning requires the existence of thousands of metabolic pathways working together. These systems are characterized by being composed of many interconnected parts and are called graphs (Barabási et al. 2016). A graph $G = (V, E)$ is a structure formed by a non-empty set of vertices V and a set of edges $E \subseteq P(V)$ with $P(V) = \{\{x, y\} : x, y \in V\}$ which is the set of all unordered and not necessarily distinct pairs generated from V. Each edge $\{x, y\} \in E$ is formed by a pair of distinct vertices ($x \neq y$). For each pair of vertices, there is at most one edge associated with them (Feofiloff et al. 2011).

A graph can undergo several modifications to model different scenarios. In this work a graph is

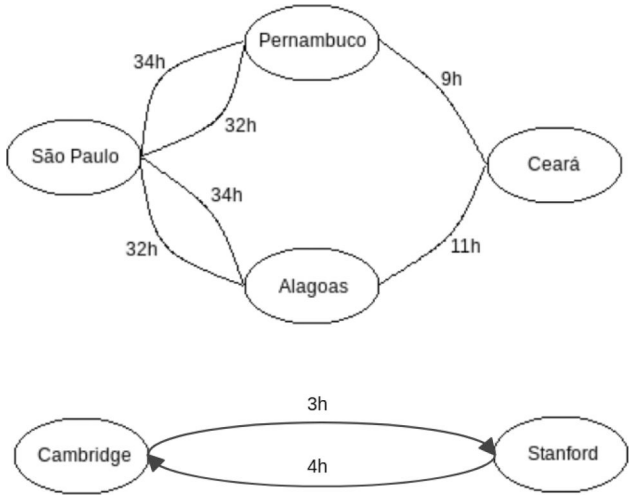


Figure 2. Graph example: a valued and directed graph.

modeled as a multiplex biological network, for this, some modifications were made to it, below are two common modifications that were made in this work.

1.1.1. Valued or weighted graph

A valued graph is one in which values are assigned to its edges or vertices. Taking as an example a graph that models roads connecting cities, with cities being vertices and roads being edges, a valued graph could present the distance between cities on their edges (in kilometers for example), or the number of inhabitants per city in each vertex. An example of a valued graph can be seen in Figure 2.

1.1.2. Directed graph

A graph is directed if each edge $e = \{x, y\}$ connects vertex x to vertex y but does not necessarily connect vertex y to x. An example of a directed graph can be seen in Figure 2, in which there is a one-way street connecting Cambridge to Stanford and another one-way street connecting Stanford to Cambridge. In the figure, the reference to directionality is given only by the arrows connecting the cities.

1.2. Biological networks

Disregarding rare exceptions, DNAs are biological molecules that store information to synthesize functional products. DNAs are formed by double compacted strands of nucleotides namely: adenine (A), guanine (G), cytosine (C) and thymine (T). Some stretches of DNA can give rise to the creation of proteins that will play different roles inside and outside cells (DePamphilis et al. 1996; Pollard et al. 2016).

The beginning of protein synthesis, or gene expression, occurs when a stretch of DNA is separated into two strands by a cellular enzyme. With the DNA molecule separated, it becomes the guide for the creation of an RNA molecule called messenger RNA (mRNA) in the transcription process. The mRNA molecule, together with other cellular molecules, produces the body's proteins in the process of translation. Several other processes participate in transcription and translation, such processes will not be presented as they are not the interest of this study (DePamphilis et al. 1996; Pollard et al. 2016).

DNAs, RNAs and proteins continually relate to each other increasing the complexity of biological systems. For example, a common term in biology is that of regulatory proteins, they have the main function of participating in gene expression processes, that is, they are proteins that stimulate or discourage production processes of other proteins (Morgan and Harris 1999). Another common term is that of gene co-expression, where a gene relationship is implied when the expression of two or more genes changes proportionally in different organisms (Zhang and Horvath 2005). In this context, the creation and study of biological networks is common. Biological networks may have vertices with biological instances, such as the aforementioned DNAs, RNAs and proteins, and edges representing relationships between these instances such as gene co-expression.

This study will use networks in which the vertices are proteins and their edges represent different relationships between these proteins. Specifically, in this work, the protein co-expression relationship will be used, but it is possible to use other protein relationships, using part of the methodology proposed in this work. The user may use other protein relationships if this is his interest. String (Szklarczyk et al. 2015) is an online tool (<https://string-db.org/>) that provides its protein interaction database, which will be used in this work.

The String offers edges with probabilistic values between 0 and 1. As biological relationships are hardly observed and tested in the entire world population, it is expected that some relationships, even if they occur in several individuals, do not occur in all. Therefore, a probabilistic value is ideal. When two proteins have more than one score of the parameters presented above, the String performs a calculation that keeps the new value between 0 and 1.

According to the study (Kinsley et al. 2020):" the power of multilayer networks lies in their flexibility to characterize multiple types of interactions not possible using a traditional monolayer network approach. Multilayer networks also consist of nodes and edges, but the nodes exist in separate layers, representing different forms of interactions, which connect to form an aspect. Aspects can be used to represent different types of contacts, spatial locations, subsystems, or points in time."

This study use multiplex networks one because its vertices can represent medicines or proteins and its edges can represent protein relationships or relationships between medicines and proteins. In this work, the types presented will also have a different structure and application. For example, only protein interactions have a statistical value.

1.3. Database: medicines acting on genes or proteins

The aim of this work is to value connections between medicines and diseases caused by a genetic or protein variation. For this, it is reasonable to use a database of medicines with already known links to genes and proteins. This work uses DrugBank as an information base (Wishart et al. 2018).

The DrugBank database is a medicine repository widely used in other works (Re and Valentini 2013) due to its extension and quality, it is continuously updated and its conclusions are based on published scientific works. DrugBank, among other things, offers two classes of medicines that will be used. Initially it provides a database of 102,031 known iterations between medicines and mRNA. DrugBank also

provides another list with 1049 known interactions between medicines and proteins. The DrugBank uses interactions previously described in other published articles and its website provides an access code to the work in which this association is presented for all iterations.

The use of medicines that affect mRNA and not just proteins (as expected by the graph used), represents a great advance. This use allows two distinct classes of medicines to be used, this choice increases the number of possible medicines and the forms of treatment of diseases.

1.4. Related works

Among studies of medicine repositioning from biological networks, a recurrent methodology is based on bipartite graphs (Yamanishi et al. 2008; Bleakley and Yamanishi 2009; Fakhraei et al. 2014; Chen et al. 2015; Wang and Loscalzo 2016). Bipartite graphs are subdivided into two groups of vertices with empty intersections between these groups, however, these graphs allow edges between vertices from one set to the other (Asratian et al. 1998). In this context, the vertex groups are exclusive, they only accept medicines, molecular targets such as proteins or genes, diseases, proteins or non-target genes. Figure 3 presents an application example for this methodology. Due to the simplicity of the data structure used, these models are fast and accurate, but may have generalization flaws. Considering only connections between two instances, works that use bipartite graphs may fail to consider long paths that, despite being reliable, walk on more than one edge. Bipartite graphs also prevent paths between more than two different entities.

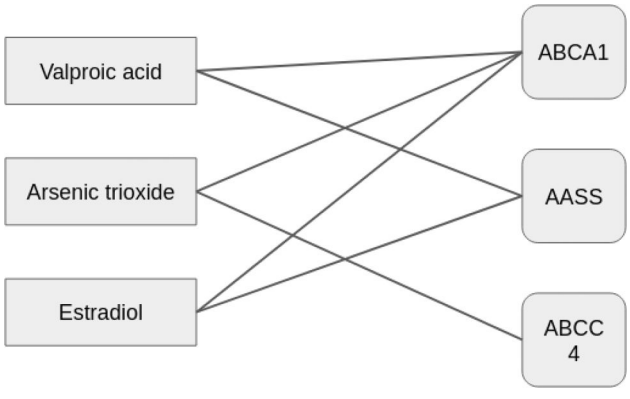


Figure 3. Reuse of medicines by bipartite graphs: left side medicines, right side mRNA, data obtained by DrugBank (Wishart et al. 2018), because it is a bipartite graph, there are connections only between different instances. As an application example, if valproic acid is used to treat a disease caused by the expression of AASS mRNA, the medicine estradiol possibly has the same effect.

Other studies use a structure similar to the one shown in Figure 4 (van Laarhoven et al. 2011; Wang and Loscalzo 2016; Cheng et al. 2018). By using non-bipartite graphs, the problems presented above are overcome. However, by using only inter-protein connections, these studies only use medicines with direct effects on known proteins. DrugBank, for example, provides a list of 102,031 medicines that act by reducing or increasing the expression of mRNAs. Lists of mRNAs can also be used in systems like these. Because there is a biological relationship between mRNA and proteins (Gygi et al. 1999; Koussounadis et al. 2015), lists of mRNAs can also be used and have been used in systems like this. It is also possible to find some medicines that act mutually on the expressions of proteins and mRNAs (for example, acetaminophen that acts on the AADAT mRNA (Heard et al. 2014) and the protein REN (Wishart et al. 2018; Cayir 2022)) generating relevant side effects for the interpretation of the effects of a medicine.

The relationships between proteins can be represented by valued graphs. In this case, the edges have a value between 0 and 1 that determine the probability of a protein actually modifying the action of another. Currently, it is common to ignore the valued part of these networks, so medicines with more pathways to a protein are more likely to act on it. Edge values are often used to filter the graph, so that edges with values less than a certain threshold are removed

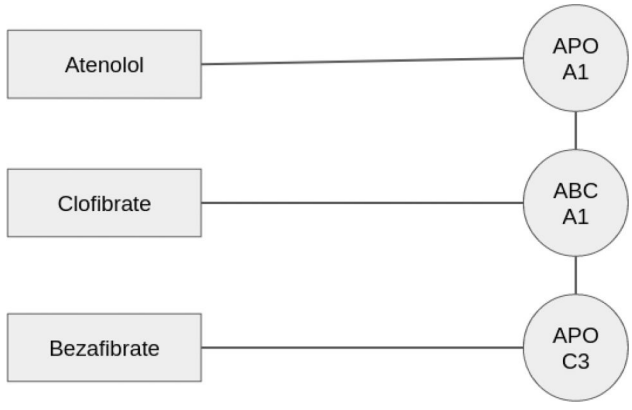


Figure 4. Reuse of drugs acting on proteins: squares on the left medicines obtained by DrugBank (Wishart et al. 2018), spheres on the right proteins obtained by String (Szklarczyk et al. 2015), connection between medicine and protein also obtained by DrugBank citewishart2018DrugBank and connection between proteins obtained by String (Szklarczyk et al. 2015). as an example of application, if atenolol is used to treat a disease caused by the APOA1 protein, the medicines clorifibrate or bezafibrate possibly have the same effect.

from the final graph, while edges with a value greater than that threshold are kept. This strategy, however, may not consider small variations in probability. Therefore, one must consider the precision that one wants to have in the study. Greater precision would imply greater computational time. This is the case of the studies (Van Laarhoven and Marchiori 2013; Zheng et al. 2013; Yan et al. 2016). An example of this form of application can be seen in Figure 5.

Some works like (Cheng et al. 2018, 2019) initially create a subgraph with vertices not far from the input vertex (i.e. few edges connecting them) and then calculate the proximity of these vertices with different medicines. The main basis of this change is to reconsider that proteins do not act in isolation in the body. These works seek to modify the action of several other proteins, in addition to the one that causes the studied disease. This model, however, does not allow the differentiation of the medicines found. Medicines that act on the protein initially causing the disease will be equivalently indicated with medicines that act on proteins only bound to the initial protein. This methodology is represented in Figure 6.

The studies presented above, despite still having some limitations, allowed to obtain significant results in the repositioning of medicines. As an example in the work (Wang and Loscalzo 2016) that studied the reuse of medicines for myocardial infarction, using bipartite graphs, 12 potential medicines were found that still need to be experimentally tested. In the study (Dias et al. 2020), medicine reuse based on data mining techniques was used to study psychiatric and neurological disorders, with 63 medicines targeting 31 proteins causing 8 different anomalies studied. With the aim of discovering efficient medicines to combat the new Covid-19 disease, some studies were created using various techniques of medicine reuse, such as (Gordon et al. 2020; Zhou et al. 2020):

2. Methodology

The aim of this work is to find connections from a protein, participant of a body variation, to a medicine, previously known and studied. Informally, Figure 7 represents some steps of this program. It will link genes or proteins to medicines, from known protein connections. It is possible to know an anomaly connected to the genetic or protein variation studied,

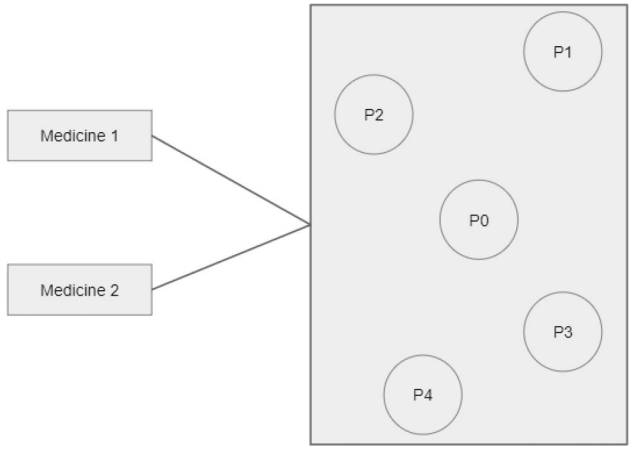


Figure 6. Reuse of medicines with clustering of vertices: squares on the left medicines, square on the right grouping, spheres represent proteins. As an application example, protein 0 causes a disease, the other proteins are bound to it. The medicines modify one or more of the group's proteins.

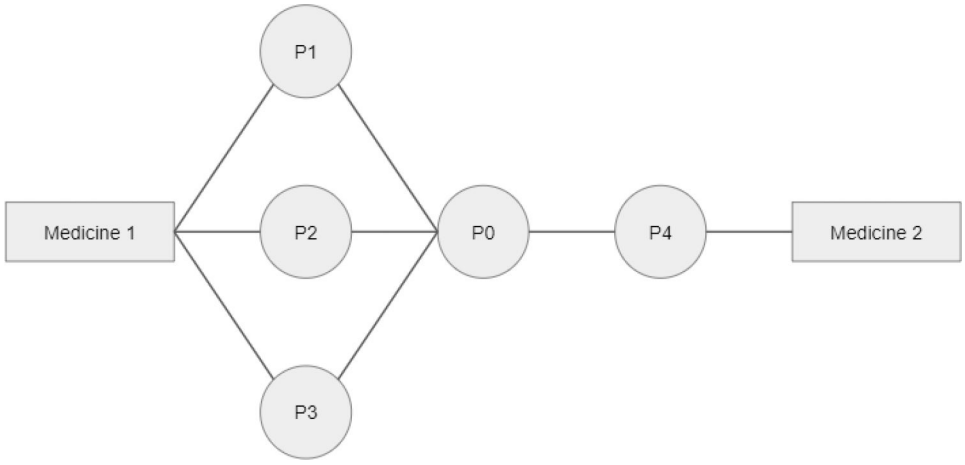


Figure 5. Reuse of medicines without probabilistic relationships: squares represent medicines, spheres represent proteins. As an example of application, medicine 1 possibly influences protein 0 more because it has more paths connecting them than medicine 2.

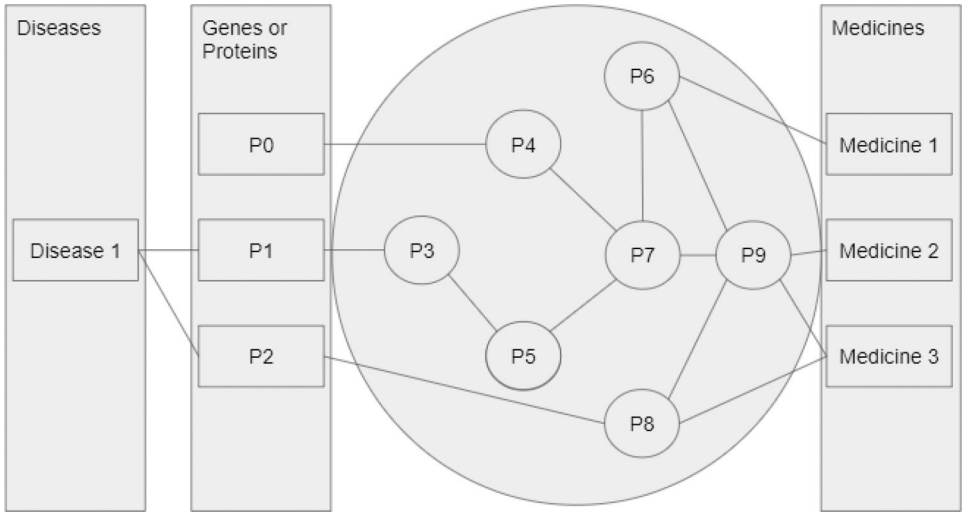


Figure 7. Protein-medicine connection: it is exemplified that several medicines can affect the protein causing a disease, although the effect of a medicine on a disease may involve biological pathways not yet described.

thus, this method can also find new connections between anomalies and medicines.

The structures that will fill in the data of the proposed program (anomalies, input offered by the user, graph of protein and medicine connections), are described below:

- **Anomalies:** this box is not represented in the program. Its existence is based on application of the method, and it will not be automatically populated by the method.
- **Genes or Proteins:** the input provided by the user. It may contain names of proteins or genes connected to bodily anomalies or connections not yet reported in the literature.[1]
- **Graph:** protein relationships in graph format. Obtained by the String database. The String has several types of protein connections, specifically, throughout the work, only the co-expression relationship will be used. If you are interested in studying other relationships, the program offered allows you to easily switch between them. The String provides all relationships with values that are structurally similar to the co-expression, so the calculations presented can be easily applied to the other relationships.
- **Medicines:** known medicines that, directly or indirectly, act on the entry. They are the output offered by the program.

Figure 8 summarizes the evolution suggested and carried out by the program. The user must indicate the entry of interest (genes or proteins). The program will return a set of medicines. It will also return, for each connection between a medicine and an input protein, a value between 0 and 1, indicated in the figure by xa and xb. This value tries to represent and is based on the reliability of the medicine in fact acting on the input, and this value is calculated from values, of equal intention, existing in the protein relationships.

The total steps of the program can be seen in Figure 9.[2]

The steps taken in the program can also be seen in the list below, in the order in which they are done. Process 1 is done by the user the others by the program.

1. The user adds the name of the protein or gene to be searched.
2. The program finds input-related proteins. Currently done with the String database, the protein connections found are valued. The String database offers seven different ways to connect two different proteins. In this work only the co-expression relationship is used. The other unused relationships can be easily integrated into the methodology presented, but due to the lack of knowledge of the implications of these forms of connection in the methodology, it will not be used in this work.
3. The program finds medicines related to the found proteins. Currently done with the DrugBank database, Connections between medicines and proteins found are not valued.
4. The program weights (with a value between 0 and 1) the input (genes or proteins) with the medicines found.

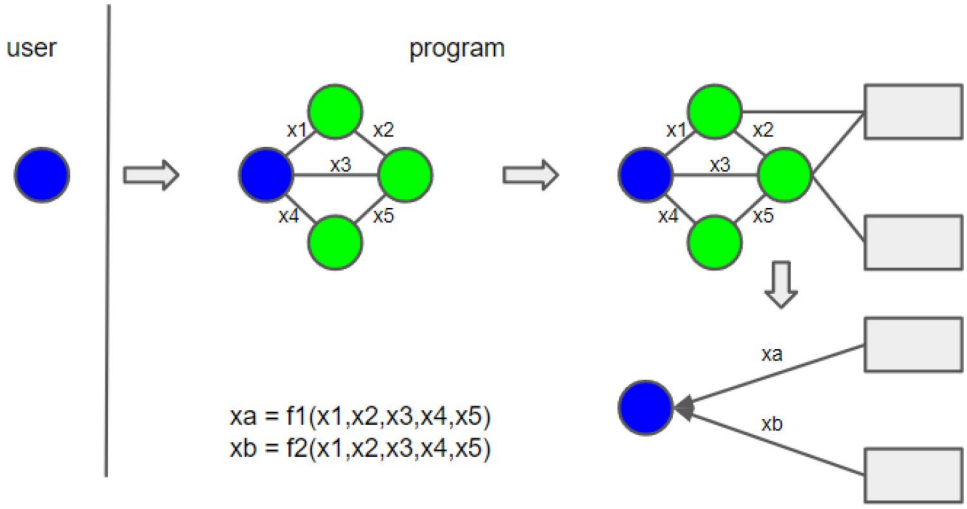


Figure 8. Program evolution: blue sphere represents the input, green sphere represents proteins added by the program, rectangles represent medicines that potentially act on the input.

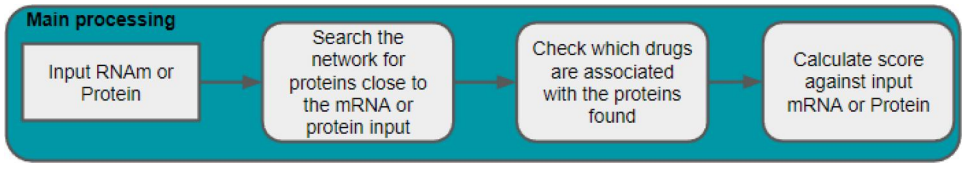


Figure 9. Program: theoretical steps of the methodology proposed by this work.

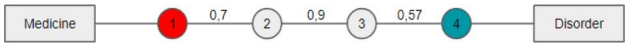


Figure 10. Simple connection: example of a simple connection between a medicine and a disease.

2.1. Calculating the confidence of a path

The edges of the network used have a statistical value that expresses the possible probability of a protein influencing another. In this section, based on the probability offered, a way to perform a ranking calculation will be developed, the probability of a medicine being functional to functionally alter a studied protein will be ranked.

To develop the desired calculation, initially, it will be exposed how the statistical reliability of a single path connecting a medicine to a disease by two distinct proteins is calculated. In the example of Figure 10, what is the confidence in indicating the medicine to control the exemplified disease?

Analyzing only Figure 10, it can be stated that assuming that protein 1 has its expression altered, there is a 70% chance that this change will culminate in a proportional change in the expression of protein 2 (it will be called of $M(1,2) = 0.7$). Also from the figure, the probability of the expression of protein 3 being altered, if initially the expression of protein 1 is altered, is the combination of the first two probabilities, the two must occur together, i.e. $M(1,2) \cap M(2,3) = M(1,2) \cdot M(2,3)$, so the probability will be $M(1,2) \cdot M(2,3) = 0.7 \cdot 0.9 = 0.63$.

In this study, the calculation of the path between a medicine and a protein will be as exemplified in Figure 10 (i.e. the intersection between the values of the edges that make up the path). The sections following this one are intended to analyze specific cases that may occur.

2.1.1. Probability that the path does not occur

The probability that a pathway does not occur occurs when varying the expression of one protein does not change the expression of another protein. In the example of Figure 10 denoted by $(P^{-1}(N_1, N_4))$, the fact occurs when simultaneously protein 4 does not have its expression increased and also does not have its expression decreased, even if other proteins of the system (1, 2 or 3) have their expression modified. Logo $P^{-1}(N_1, N_4) = 1 - P(N_1, N_4)$

2.1.2. Medicine affecting more than one protein

A medication can act on more than one biological pathway through polypharmacology; in our specific study, a medication can act on several proteins in parallel. Specifically, the database used offers medicines that can act on the properties of more than one protein at the same time. In the example in

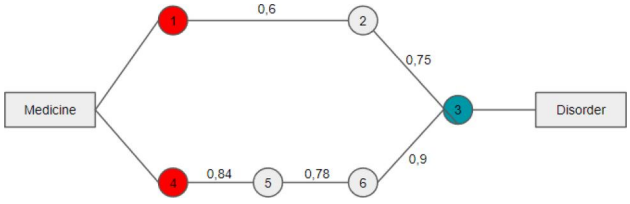


Figure 11. Complete connection: example of a connection between a medicine and a disease.

Table 1. Connection possibilities.

Possibility	Protein 1	Protein 4
1	Modify protein 3	Modify protein 3
2	Modify protein 3	Do not modify protein 3
3	Do not modify protein 3	Modify protein 3
4	Do not modify protein 3	Do not modify protein 3

Note: Possibilities 1 and 3: only one path is activated; possibility 2: both paths are activated; possibility 4: both paths are inactive.

Figure 11 we have a medicine directly affecting the expression of two proteins. In this case, how to calculate confidence in the drug's action?

In this example the medicine can modify the disease by protein 1 and protein 4. Interest occurs in three scenarios. All 4 possible scenarios are specified in Table 1. Interest is in possibilities 1, 2, and 3 where somehow protein 3 is reached. Disinterest occurs only in scenario 4, where proteins 1 and 4 do not reach protein 3 together. The calculation of the three possibilities of interest are specified in the equation below:

$$\begin{aligned}&((P(N_1,N_3))\cdot(P(N_4,N_3)))\\&+((P(N_1,N_3))\cdot(1-P(N_4,N_3)))\\&+((1-P(N_1,N_3))\cdot(P(N_4,N_3)))\\&=\end{aligned}\tag{1}$$

Line 1, 2 and 3 with of equation 1 represents line 1, 2 and 3, respectively, of Table 1.

The distributive is done in terms of line 3 of the previous equation, obtaining the following equation:

$$\begin{aligned}&((P(N_1,N_3))\cdot(P(N_4,N_3)))\\&+((P(N_1,N_3))\cdot(1-P(N_4,N_3)))\\&+(P(N_4,N_3))-((P(N_4,N_3))\cdot P(N_1,N_3)))\\&=\end{aligned}\tag{2}$$

The first and second lines of the previous equation are manipulated, keeping the equivalent term $(P(N_1,N_3)))$ and adding the other terms.

$$\begin{aligned}&((P(N_1,N_3))\cdot 1)\\&+(P(N_4,N_3))-((P(N_4,N_3))\cdot P(N_1,N_3)))\\&=\end{aligned}\tag{3}$$

Removing number 1.

$$P(N_1,N_3)+P(N_4,N_3)-(P(N_4,N_3)\cdot P(N_1,N_3))=\tag{4}$$

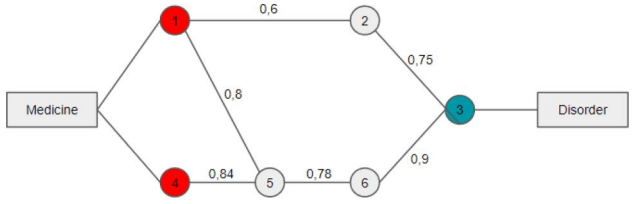


Figure 12. Connection with variation: example of a connection between a medicine and a disease with more than one path.

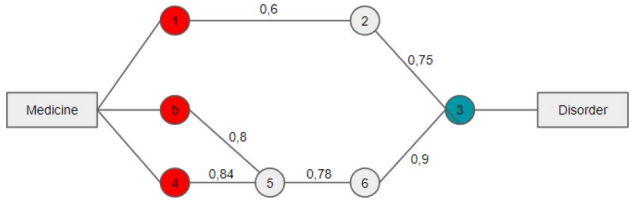


Figure 13. Modified connection: example of a connection between a medicine and a modified disease.

The final term is similar to the probabilistic union expressed below:

$$P(N_1,N_3)\cup P(N_4,N_3)\tag{5}$$

2.1.3. Medicine affecting disease in more than one way

A case can occur where more than one pathway connects two proteins. The example in Figure 12 is similar to the example in Figure 11 but with two ways of connecting proteins 1 and protein 3. proteins can connect through protein 2 or proteins 5 and 6. In cases like this, both paths will always be considered. The algorithm will interpret the graph as if it had a new vertex that unlinks the two paths of the graph, from which the calculations can be maintained as previously presented.

The example in Figure 13 presents the subdivision made by the algorithm of vertex 1 into two vertices 1 and b. The algorithm will act like this only when it finds a vertex that can give rise to two paths (in the example vertex 1) to the other vertices (vertex 2, 5, 4 . . .) the algorithm will act normally considering the original graph, in this example the graph in Figure 12.

2.1.4. Several proteins of interest

It may be relevant for the study to know the probability of a medicine acting on different proteins. Proteins can be, for example, in parallel, the cause of a single disease, as exemplified in Figure 14, or of several diseases. In this case, how do we calculate the reliability of using the medicine?

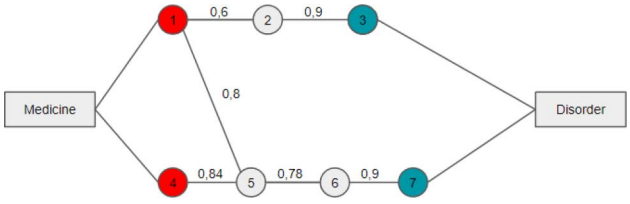


Figure 14. Complete connection: example of a connection between a medicine and a disease.

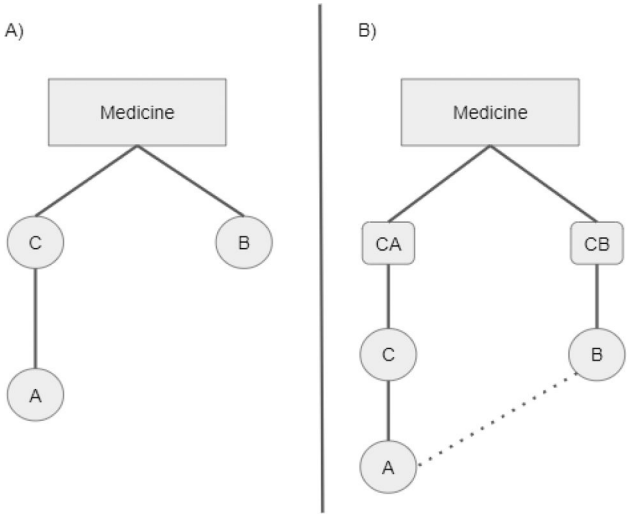


Figure 15. Union between two proteins: rectangles are medicines; waits are proteins; superellipses are subdivided components of the medicine.

Assuming transitive medicine-protein relationships, it can be implied that the relationship between a medicine and several proteins is dependent. A medicine can be divided into several components (as shown in Figure 15). For example, Benegrip, a medication commonly used Carlos Cristiano Hasenclever Borgesto combat flu symptoms, according to its package insert, is composed of the items: dipyrone (analgesic and antipyretic), chlorpheniramine maleate (antiallergic) and caffeine (stimulant). Therefore, when ingesting Benegrip, the three described components of the medicine will work together.

Thus, if the same medicine reaches two different proteins, the method will imply that the two pathways are dependent, it will perform the union of the pathways. Statistics already predict a union or sum of probabilities for dependent events. For better visualization of the explanation, this sum will be symbolized as: $\pm$. $\pm$ is defined as $A \pm B = A + B - (A \cdot B)$. The final subtraction prevents this sum from assuming values greater than 1 considering $0 \leq A \leq 1$ and $0 \leq B \leq 1$. In short $\overline{+}$ is a sum endowed with a subtraction of the dependent part of the sets.

See that what happens in Figure 15 is different from what happens in Figure 10, for example. In Figure 15, the medicine initiates two paths in parallel, the final proteins (A and B) are reached in parallel, considering the effectiveness of the two paths. Therefore, the calculation that will be used here must be different from the one used previously (at the beginning of the Section 2.1).

3. Tests and case study

The proposed program was compared with several others present and common in the literature. The objective is to verify if the proposed program presents similar results to the existing ones. Initially the proposed program is compared to two existing ones and finally a case study on a set of diseases is manufactured and presented.

The objective with the three experiments, mentioned above, is to verify whether the proposed method adequately outperforms other strategies in the literature. In the three experiments, a different graph was developed to evaluate the proposed method, all experimental specificities were presented.

3.1. Stitch in Duchenne muscular dystrophy

The program suggested throughout the work has great similarities with the program Stitch (Kuhn et al. 2007; Szklarczyk et al. 2016). The program suggested by this work may represent an evolution of Stitch. For this reason, initially, Stitch will be evaluated in relation to a specific disease. After applying Stitch, the improvements made by our program will be announced and validated.

For comparison between our method and the Stitch database, we used Duchenne muscular dystrophy, a genetic disorder caused by variation in the DMD gene, as a case study. This disease is characterized by a progressive and irreversible degeneration of the skeletal muscles (Aparecida 1999). It is commonly treated by the medicines Prednisone, Prednisolone and Deflazacort (Shieh et al. 2021).

The name of the protein altered by Duchenne muscular dystrophy (i.e. DMD) was used as input for Stitch. The output of the program, in summary form, can be seen in Figure 16. The input protein DMD can be seen in the figure, as well as the medicines initially sought: Prednisone, Prednisolone and Deflazacort.

The proposed program will answer questions that Stitch is unable to answer such as: which medicine,

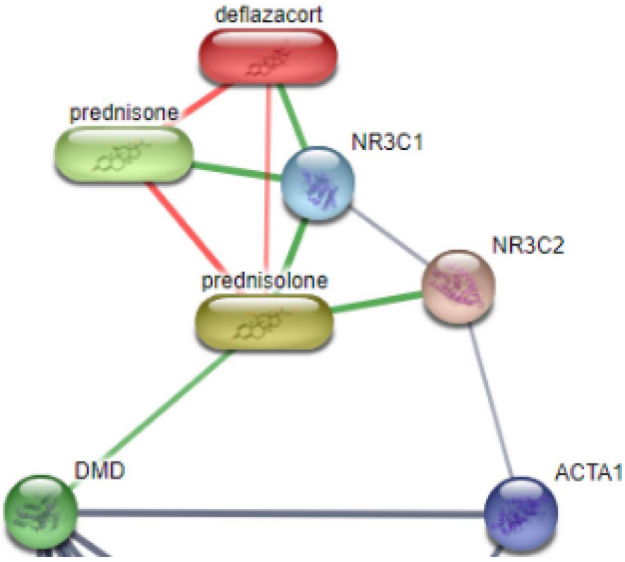


Figure 16. Stitch output: DMD protein input. Spheres represent proteins, rounded rectangles represent medicines.

Predinisolone or Predinisone, is more likely to act on the DMD protein. In this specific case, the Predinisolone score (i.e. 0.578) would exceed the Predinisone performance score (i.e. 0.126), by the calculation suggested here. Predinisolone has a direct path of action on the DMD protein, in addition to having another path similar to the medicine Predinisone.

3.2. SCMFDD (similarity constrained matrix factorization method for the drug-disease association prediction)

SCMFDD is a program that can be used to evaluate and score connections between proteins and medicines. The SCMFDD offers a value that relates a medicine to a disease, allowing the entry of diseases that are known to modify the action of some protein. Its result is also a positive value between 0 and 1.

The SCMFDD was also used in this work to validate the medicines found. A similar response from SCMFDD is desired with the suggested program. Specifically, you want medicines that score high on the suggested program to also score high on the SCMFDD.

Several monogenetic diseases (shown in Table 2) were used to compare the two models. It is also shown in Table 2 that different values of Limit 1 and Limit 2 were used, where Limit 1 is the minimum score allowed on the edges of the graph, and Limit 2 is the maximum number of vertices between two other vertices in the graph.

Table 2. Information about testing with SCMFDD.

Diseases	Entry	Limit 1	Limit 2
Marfan syndrome	FBN1	0.8	3
Cystic fibrosis	CFTR	0.6	3
Cystisinosis	CTNS	0.09	2
Duchenne muscular dystrophy	DMD	0.5	3
Oculopharyngeal muscular dystrophy	PABPN1	0.75	3

Note: Value of limit 1 and limit 2 in this experiment for different diseases.

Each disease used was scored against our program and the SCMFDD, according to the score offered by both. All scores are presented in visual format in graphs.

The SCMFDD (**s**imilarity **c**onstrained **m**atrix **f**actorization **m**ethod for the **d**rug-disease association prediction) was compared to this program. The result, in graphical format, is shown in Figure 17.

In Figure 17 the X axis represents the Score provided by our program, the Y axis the Score provided by the compared program. Each point in the figure represents 1 medicine. Proportional scores are expected in both programs.

The purple line in all the sub-figures of Figure 17 represents the line that best approximates the points. The line is calculated from the linear regression of the points. The straight line helps to visualize the growth of the values in the X and Y axis. A straight line between 0 and 90 degrees represents that the Scores grow proportionally in the two programs, that is, the program suggested by this study managed, for this scenario, a result similar to the compared program, the SCMFDD.

3.3. Case study: similarity of treatment between ataxias

Spinocerebellar ataxias or SCAs are a group of more than 37 known diseases, none of which currently have a medical cure (Botez et al. 1998; Ghanekar et al. 2022). Ataxias are genetic and clinically heterogeneous diseases, however, with symptomatic similarities, Ataxias commonly affect the nervous system, indirectly the entire muscular system (Ashizawa and Xia 2016). The most common type of SCAs occurs in one to five cases per 100,000 people (Paulson 2009).

Several studies such as (Lim et al. 2006; Carnivali C 2020; Carnivali GS 2021) have already demonstrated the genetic and protein similarity of Ataxias. Ataxias, despite being different diseases, have several symptomatic and functional similarities (Paulson 2009; Ashizawa and Xia 2016). Therefore, it is expected that Ataxias also have similar medications that are recommended and found by the tool, showing a possible use and validation of the tool.

For this test, 10 genes that cause or possibly cause the appearance of a disease were selected, of these 10,

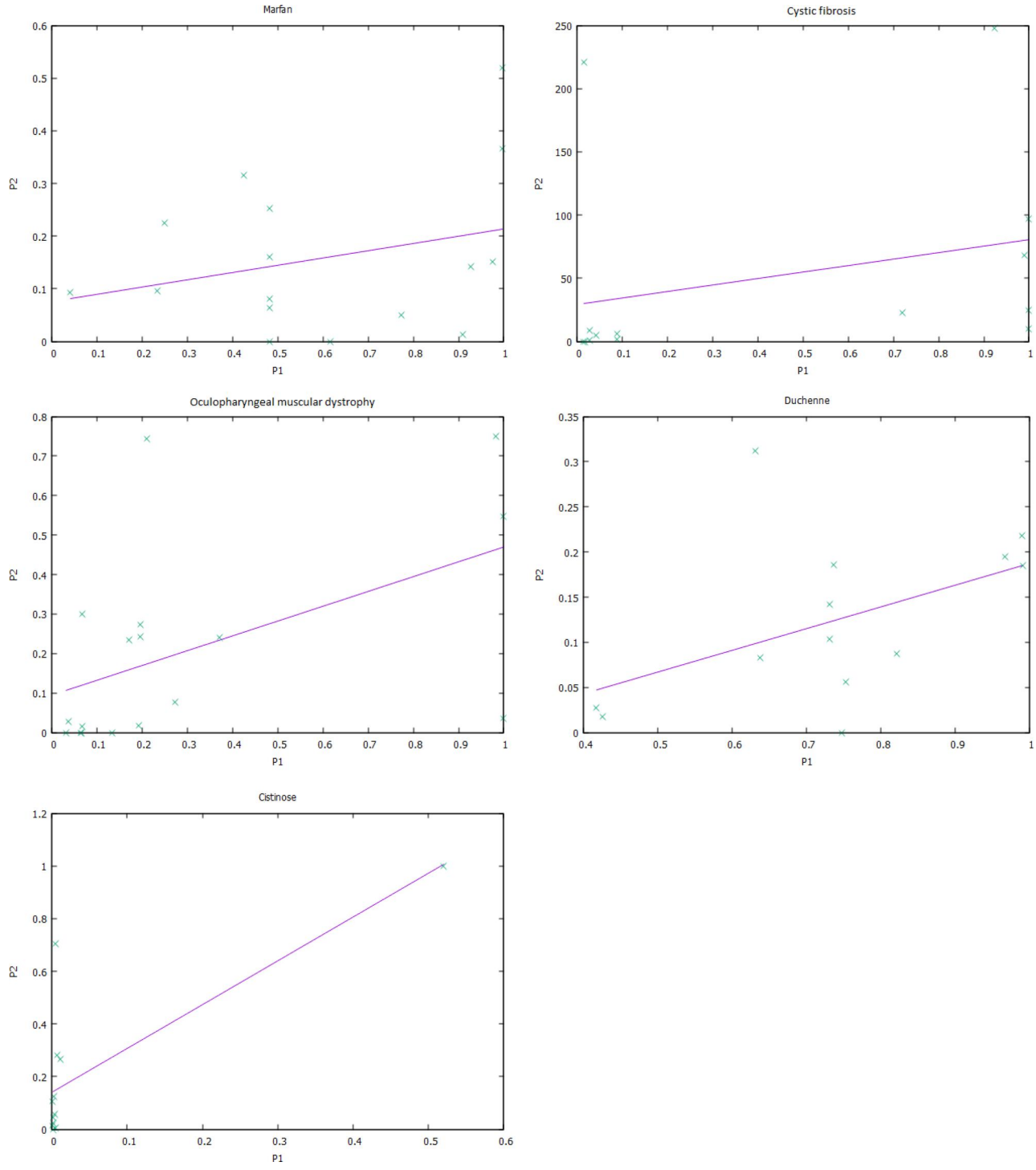


Figure 17. SCMFDD comparison: P1 represents the suggested program, P2 the compared program (SCMFDD), each point represents a medicine, purple line represents the linear regression of the points. Comparative result with the SCMFDD platform.

5 are responsible for some Ataxia. The test is expected to verify whether there are more similarities in drug indications in similar diseases. All 10 genes used are listed below:

- **Gene ATXN2**: one of the causes of the disease Spinocerebellar Ataxia type 2.
- **Gene ATXN2l**: one of the causes of the disease Spinocerebellar Ataxia type 2.
- **Gene ATXN3**: one of the causes of the disease Spinocerebellar Ataxia type 3.
- **Gene ATXN3l**: one of the causes of the disease Spinocerebellar Ataxia type 3.

Table 3. Comparison between genes.

x	atxn2	atxn2l	atxn3	atxn3l	ttbk2	brca1	brca2	cftr	hexa	atp7b
atxn2	x									
atxn2l	111	x								
atxn3	130	135	x							
atxn3l	127	128	127	x						
ttbk2	118	119	135	129	x					
brca1	137	136	132	133	134	x				
brca2	133	132	128	130	132	104	x			
cftr	152	152	153	143	153	156	154	x		
hexa	133	132	138	129	133	137	136	146	x	
atp7b	123	124	123	121	128	119	115	148	130	x

- **Gene TTBK2**: one of the causes of the disease Spinocerebellar Ataxia type 11.
- **Gene BRCA1**: one of the causes of breast cancer.
- **Gene BRCA2**: one of the causes of breast cancer.
- **Gene CFTR**: one of the causes of the disease Cystic Fibrosis.
- **Gene HEXA**: one of the causes of Tay–Sachs disease.
- **Gene ATP7B**: one of the causes of Wilson's disease.

The 10 genes were chosen due to their notorious association with a disease. All 10 genes were used as program input, and the 100 highest scoring drugs for each gene were collected. In this test, it was set that the minimum distance (parameter 1) defined was equal to 3 and the minimum score (parameter 2) was different for each gene, to obtain the 100 drugs initially set.

Table 3 shows the number of different medicines that were found for each comparison of two different genes. Tuples with a value less than 130 are in blue. One can see a greater similarity of medicines between the Ataxias as expected. The smallest value can be seen in the BRCA1 and BRCA2 tuple, which is also expected since both genes are associated with breast cancer. It is possible to verify small values with the ATP7B gene, this may represent a system error, however, the disease caused by the ATP7B gene has great symptomatic similarities with Ataxias (Prado and da Fonseca 2004).

4. Discussion of results and conclusion

The offered program, which implements the proposed method, is easy to use. No installation is required, just using the Linux platform command line. Allowing the variation of the precision of the results, generating, in parallel, a variation of the temporal cost of the same.

The program also allows easy parallelization, the parallel process is a more efficient way of working with information, it emphasizes the exploration of simultaneous events in the execution of a program. In the proposed method, proteins are searched through parallel paths, input proteins are analyzed independently. However, the final calculation, which evaluates the relationship between medicines and proteins, cannot be parallelized.

It was possible to perceive, due to the diversity of diseases used, that some protein relationships have more studies than others, generating an inevitable variation in the final score found. Therefore, it is necessary, for a future application of the presented tool, to better understand the results, the comparison with the scores of other medicines. In this work, a score close to 1 can be considered low compared to other scores.

A possible way to find more reliable protein connections is to analyze their topological characteristics in the available graph. For example, vertices or proteins with high degree (i.e. high number of connections), represent highly connected vertices in the graph (Gagliardi et al. 2003), this indirectly implies that several studies were used to study this protein.

Another metric that can also be used in other work is *betweenness*, a metric that evaluates the number of shortest paths that pass through a given vertex (Gagliardi et al. 2003). This metric can help find proteins that are relevant in other indirect protein connections.

This work presented the application of several diseases to the method. However, this study did not study all diseases, so it is proposed to apply this proposed methodology to other diseases.

It is also proposed to increase the number of databases used, is also suggested the insertion of new medicine-protein relationships with the use of other databases, new relationships between proteins, new co-expression relationships and, of course, the greater

study of these relationships in other possible tissues can also be done.

Acknowledgments

The authors thank Prof. Rafaela Ferreira (UFMG, Brazil) for discussions related to this study. The authors also thank FAPEMIG, Capes, UFMG and UFJF.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was supported by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior; Fundação de Amparo à Pesquisa do Estado de Minas Gerais.

References

Aparecida CF. 1999. Características do portador de distrofia muscular de duchenne (dmd) - revisão, Arquivos de Ciências da Saúde da UNIPAR. Tres Rios- MG.

Ashburn TT, Thor KB. 2004. Drug repositioning: identifying and developing new uses for existing drugs. Nat Rev Drug Discov. 3(8):673–683. doi: 10.1038/nrd1468.

Ashizawa T, Xia G. 2016. Ataxia, continuum: lifelong learning in neurology. 22(4):1208–1226. doi: 10.1212/CON.0000000000000362.

Asratian AS, Denley TM, Häggkvist R. 1998. Bipartite graphs and their applications. Vol. 131. EUA: Cambridge University Press.

Barabási A-L, et al. 2016. Network science. EUA: Cambridge University Press.

Bleakley K, Yamanishi Y. 2009. Supervised prediction of drug–target interactions using bipartite local models. Bioinformatics. 25(18):2397–2403. doi: 10.1093/bioinformatics/btp433.

Botez M, Botez-Marquard T, Mayer P, Marchand L, Lalonde R, Reader T. 1998. The treatment of spinocerebellar ataxias: facts and hypotheses. Med Hypotheses. 51(5):381–384. doi: 10.1016/s0306-9877(98)90032-9.

Calixto JB, Siqueira Junior JM. 2008. Desenvolvimento de medicamentos no brasil: desafios. Gazeta Médica da Bahia. Bahia - BR. 78(1).

Carnivali C. 2020. Does the ataxia group have genetic similarities? Encontro Acadêmico de Modelagem Computacional.

Carnivali GS. 2021. Machine learning method to diferentiete ataxias. Int J Appl Math Mach Learn. 15(1):53–67. doi: 10.18642/ijamml_710012230.

Carvalho JP. 2021. A introdução de novos medicamentos no mercado farmacêutico brasileiro: exame em um cenário de precificação regulada. Unesp - University of Bahia.

Cayir A. 2022. Rna modifications as emerging therapeutic targets. Wiley Interdiscip Rev RNA. 13(4):e1702.

Chen H, Zhang H, Zhang Z, Cao Y, Tang W. 2015. Network-based inference methods for drug repositioning. Comput Math Methods Med. 2015:130620–130627. doi: 10.1155/2015/130620.

Cheng F, Desai RJ, Handy DE, Wang R, Schneeweiss S, Barabási A-L, Loscalzo J. 2018. Network-based approach to prediction and population-based validation of in silico drug repurposing. Nat Commun. 9(1):2691. doi: 10.1038/s41467-018-05116-5.

Cheng F, Kovács IA, Barabási A-L. 2019. Network based prediction of drug combinations. Nat Commun. 10(1):1806. doi: 10.1038/s41467-019-09186-x.

da VM, Vieira M, Ohayon P. 2006. Inovação em fármacos e medicamentos: estado-da-arte no brasil e políticas de p&d. Revista Economia & Gestão. PUC - Minas - Brazil.6(13).

DePamphilis ML, et al. 1996. DNA replication in eukaryotic cells. Cold Spring Harbor Laboratory.

Dias TL, Schuch V, Beltrão-Braga PCB, Martins-de Souza D, Brentani HP, Franco GR, Nakaya HI. 2020. Drug repositioning for psychiatric and neurological disorders through a network medicine approach. Transl Psychiatry. 10(1):141. doi: 10.1038/s41398-020-0827-5.

Fakhraei S, Huang B, Raschid L, Getoor L. 2014. Network-based drug-target interaction prediction with probabilistic soft logic. IEEE/ACM Trans Comput Biol Bioinform. 11(5):775–787. doi: 10.1109/TCBB.2014.2325031.

Farha MA, Brown ED. 2019. Drug repurposing for antimicrobial discovery. Nat Microbiol. 4(4):565–577. doi: 10.1038/s41564-019-0357-1.

Feofiloff P, Kohayakawa Y, Wakabayashi Y. 2011. Uma introdução sucinta à teoria dos grafos. IME - USP - Brazil.

Gagliardi EO, Berón M, Peñalver GH. 2003. Evaluación de métricas en redes de computadoras. IX Congreso Argentino de Ciencias de la Computación.

Ghanekar SD, Kuo S-H, Staffetti JS, Zesiewicz TA. 2022. Current and emerging treatment modalities for spinocerebellar ataxias. Expert Rev Neurother. 22(2):101–114. doi: 10.1080/14737175.2022.2029703.

Gil C, Martinez A. 2021. Is drug repurposing really the future of drug discovery or is new innovation truly the way forward?

Gordon DE, Jang GM, Bouhaddou M, Xu J, Obernier K, White KM, O'Meara MJ, Rezelj VV, Guo JZ, Swaney DL, et al. 2020. A sars-cov-2 protein interaction map reveals targets for drug repurposing. Nature. 583(7816):459–468.

Gygi SP, Rochon Y, Franza BR, Aebersold R. 1999. Correlation between protein and mRNA abundance in yeast. Mol Cell Biol. 19(3):1720–1730. doi: 10.1128/MCB.19.3.1720.

Heard K, Green JL, Anderson V, Bucher-Bartelson B, Dart RC. 2014. A randomized, placebo-controlled trial to determine the course of aminotransferase elevation during prolonged acetaminophen administration. BMC Pharmacol Toxicol. 15(1):39. doi: 10.1186/2050-6511-15-39.

I. F. of Pharmaceutical Manufacturers & Associations. 2014. The pharmaceutical industry and global health–facts and figures. IFPMA - Genève, Suíça.

Kinsley AC, Rossi G, Silk MJ, VanderWaal K. 2020. Multilayer and multiplex networks: an introduction to

their use in veterinary epidemiology. Front Vet Sci. 7: 596. doi: 10.3389/fvets.2020.00596.
Koussounadis A, Langdon SP, Um IH, Harrison DJ, Smith VA. 2015. Relationship between differentially expressed mRNA and mRNA-protein correlations in a xenograft model system. Sci Rep. 5(1):10775. doi: 10.1038/srep10775.
Kuhn M, von Mering C, Campillos M, Jensen LJ, Bork P. 2007. Stitch: interaction networks of chemicals and proteins. Nucleic Acids Res. 36(Database issue):D684–D688. doi: 10.1093/nar/gkm795.
Lim J, Hao T, Shaw C, Patel AJ, Szabó G, Rual J-F, Fisk CJ, Li N, Smolyar A, Hill DE, et al. 2006. A protein–protein interaction network for human inherited ataxias and disorders of Purkinje cell degeneration. Cell. 125(4):801–814. doi: 10.1016/j.cell.2006.03.032.
McRae D, Allman M, James D. 2016. The redistribution of medicines: could it become a reality? Int J Pharm Pract. 24(6):411–418. doi: 10.1111/ijpp.12275.
Morgan BP, Harris AL. 1999. Complement regulatory proteins. Cambridge, MA: Academic Press.
Muthyala R. 2011. Orphan/rare drug discovery through drug repositioning. Drug Discov Today Ther Strateg. 8(3-4):71–76. doi: 10.1016/j.ddstr.2011.10.003.
Paulson HL. 2009. The spinocerebellar ataxias. J Neuro-Ophthalmol. 29(3):227–237. doi: 10.1097/WNO0b013e318 1b416de.
Pinto JPDM. 2017. Competências para inovar na indústria farmacêutica brasileira. Brazil: CGEE.
Pollard TD, Earnshaw WC, Lippincott-Schwartz J, Johnson G. 2016. Cell biology e-book. Amesterdã, Países Baixos: Elsevier Health Sciences.
Prado ALC, da Fonseca DC. 2004. Uma revisão sobre a doença de wilson relato de caso. Saúde. Blucher Medical Proceedings. Brazil. 69–75.
Re M, Valentini G. 2013. Network-based drug ranking and repositioning with respect to DrugBank therapeutic categories. IEEE/ACM Trans Comput Biol Bioinform. 10(6): 1359–1371. doi: 10.1109/TCBB.2013.62.
Rudrapal M, Khairnar SJ, Jadhav AG. 2020. Drug repurposing (DR): an emerging approach in drug discovery. Drug Repurposing Hypothesis Mol Asp Ther Appl. 1:1.
Shieh PB, Elfring G, Trifillis P, Santos C, Peltz SW, Parsons JA, Apkon S, Darras BT, Campbell C, McDonald CM. 2021. Meta-analyses of deflazacort versus prednisone/prednisolone in patients with nonsense mutation Duchenne muscular dystrophy. J Comp Eff Res. 10(18): 1337–1347. doi: 10.2217/cer-2021-0018.
Silva GCD. 2015. Estratégias em reposicionamento de fármacos. LUME. Universidade Federal do Rio Grande do Sul. Faculdade de Farmácia. Curso de Farmácia.
Szklarczyk D, Franceschini A, Wyder S, Forslund K, Heller D, Huerta-Cepas J, Simonovic M, Roth A, Santos A, Tsafou KP, et al. 2015. String v10: protein–protein interaction networks, integrated over the tree of life. Nucleic Acids Res. 43(Database issue):D447–D452. doi: 10.1093/nar/gku1003.
Szklarczyk D, Santos A, Von Mering C, Jensen LJ, Bork P, Kuhn M. 2016. Stitch 5: augmenting protein–chemical interaction networks with tissue and affinity data. Nucleic Acids Res. 44(D1):D380–D384. doi: 10.1093/nar/gkv1277.
Van Laarhoven T, Marchiori E. 2013. Predicting drug-target interactions for new drug compounds using a weighted nearest neighbor profile. PLoS One. 8(6):e66952. doi: 10.1371/journal.pone.0066952.
van Laarhoven T, Nabuurs SB, Marchiori E. 2011. Gaussian interaction profile kernels for predicting drug–target interaction. Bioinformatics. 27(21):3036–3043. doi: 10.1093/bioinformatics/btr500.
Wang R-S, Loscalzo J. 2016. Illuminating drug action by network integration of disease genes: a case study of myocardial infarction. Mol Biosyst. 12(5):1653–1666. doi: 10.1039/c6mb00052e.
Wishart DS, Feunang YD, Guo AC, Lo EJ, Marcu A, Grant JR, Sajed T, Johnson D, Li C, Sayeeda Z, et al. 2018. DrugBank 5.0: a major update to the DrugBank database for 2018. Nucleic Acids Res. 46(D1):D1074–D1082. doi: 10.1093/nar/gkx1037.
Yamanishi Y, Araki M, Gutteridge A, Honda W, Kanehisa M. 2008. Prediction of drug–target interaction networks from the integration of chemical and genomic spaces. Bioinformatics. 24(13):i232–i240. doi: 10.1093/bioinformatics/btn162.
Yan X-Y, Zhang S-W, Zhang S-Y. 2016. Prediction of drug–target interaction by label propagation with mutual interaction information derived from heterogeneous network. Mol Biosyst. 12(2):520–531. doi: 10.1039/c5mb00615e.
Zhang B, Horvath S. 2005. A general framework for weighted gene co-expression network analysis. Stat Appl Genet Mol Biol. 4(1):Article17. doi: 10.2202/1544-6115.1128.
Zheng X, Ding H, Mamitsuka H, Zhu S. 2013. Collaborative matrix factorization with multiple similarities for predicting drug-target interactions. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 1025–1033. doi: 10.1145/2487575.2487670.
Zhou Y, Hou Y, Shen J, Huang Y, Martin W, Cheng F. 2020. Network-based drug repurposing for novel coronavirus 2019-ncov/sars-cov-2. Cell Discov. 6(1):14. doi: 10.1038/s41421-020-0153-3.