UNIVERSIDADE FEDERAL DE MINAS GERAIS
Escola de Engenharia
Programa de Pós-Gradução em Engenharia Elétrica

Itallo Guilherme Machado

Aprendizagem Estrutural de Redes Bayesianas utilizando Algoritmo Genético Multi-Agente

Belo Horizonte

2021

Itallo Guilherme Machado

Aprendizagem Estrutural de Redes Bayesianas utilizando Algoritmo Genético Multi-Agente

Dissertação submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica.

Universidade Federal de Minas Gerais

Orientador Michel Bessani

Belo Horizonte – MG

Machado, Itallo Guilherme.
M149a Aprendizagem estrutural de redes bayesianas utilizando algoritmo genético multi-agente [recurso eletrônico] / Itallo Guilherme Machado. - 2021.
1 recurso online (70 f. : il., color.) : pdf.

Orientador Michel Bessani.

Dissertação (mestrado) - Universidade Federal de Minas Gerais, Escola de Engenharia.

Anexos: f. 68-70.

Bibliografia: f. 62-67.
Exigências do sistema: Adobe Acrobat Reader.

1. Engenharia elétrica - Teses. 2. Algoritmos genéticos - Teses. I. Bessani, Michel. II. Universidade Federal de Minas Gerais. Escola de Engenharia. III. Título.

CDU: 621.3(043)

Ficha catalográfica elaborada pelo bibliotecário Reginaldo Cesar Vital dos Santos CRB/6 2165
Biblioteca Prof. Mário Werneck, Escola de Engenharia da UFMG

"Aprendizagem Estrutural de Redes Bayesianas Utilizando Algoritmo Genético Multi-agente"

Itallo Guilherme Machado

Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Mestre em Engenharia Elétrica.

Aprovada em 22 de julho de 2021.

Por:

Prof. Dr. Michel Bessani
(UFMG) - Orientador

Prof. Dr. Carlos Dias Maciel
SEL (USP)

Prof. Dr. Eduardo Gontijo Carrano
DEE (UFMG)

AGRADECIMENTOS

Agradeço à minha família pelo apoio ao longo da minha vida e me deram toda a força para mais essa conquista. Graças a vocês pude concretizar mais um sonho.

Agradeço ao meu orientador, prof. Michel Bessani, pelo apoio e toda ajuda prestada ao longo de todo mestrado, excepcional orientação.

Por fim, agradeço à CAPES pelo apoio financeiro, sem o qual esse trabalho não teria sido possível.

RESUMO

MACHADO, I. G.. **Aprendizagem Estrutural de Redes Bayesianas utilizando Algoritmo Genético Multi-Agente**. 2021. 70 f. Dissertação (Mestrado Engenharia Elétrica) – Escola de Engenharia (UFMG), Belo Horizonte – MG.

Esse trabalho aborda o aprendizado estrutural de redes Bayesianas. Esse aprendizado da rede que melhor representa os dados disponíveis é um problema Np-difícil. O objetivo é a adaptação do Algoritmo Genético Multi-Agente (MAGA) para o aprendizado estrutural de redes Bayesianas. No algoritmo MAGA são utilizados os elementos de um sistema multi-agente com mecanismos dos algoritmos genéticos. Nesse trabalho, foram implementadas duas configurações do MAGA, as quais foram comparadas com outros algoritmos da literatura. Os resultados demonstraram que as duas configurações do MAGA apresentaram bons desempenhos no aprendizado estrutural de redes Bayesianas, principalmente em instâncias com muitas variáveis e grandes volumes de dados.

Palavras-chave: Redes Bayesianas, Aprendizado estrutural, Algoritmo genético, Algoritmo genético multiagente .

ABSTRACT

MACHADO, I. G.. **Aprendizagem Estrutural de Redes Bayesianas utilizando Algoritmo Genético Multi-Agente**. 2021. 70 f. Dissertação (Mestrado Engenharia Elétrica) – Escola de Engenharia (UFMG), Belo Horizonte – MG.

This work approaches the structural learning of Bayesian networks. The task of learning a Bayesian Network from data is an Np-hard problem. We present the Multi-Agent Genetic Algorithm (MAGA) to learn Bayesian network structures. The MAGA algorithm uses the elements of a multi-agent system, such as communication between agents, their interaction with the environment, and the mechanisms of genetic algorithms that subject agents to genetic operators. In this study, two configurations of MAGA were implemented, which were compared to other algorithms in the literature. The purpose of this research is to evaluate the performance of the MAGA, mainly in instances with many variables and large volumes of data. The experiment results show that the two configurations of MAGA are effective in learning the Bayesian Network structures, chiefly in instances with many variables and large volumes of data.

Keywords: Bayesian networks, Structural learning, Genetic Algorithm, Multi-Agent Genetic Algorithm.

LISTA DE ILUSTRAÇÕES

Figura 2.1 – Exemplo de uma rede Bayesiana. Adaptado de [Scanagatta, Salmerón e Stella 2019]. 20
Figura 2.2 – Modelo da grade de agentes. 31
Figura 3.1 – rede alvo do problema ASIA. Adaptado de [Scutari 2020]. 34
Figura 3.2 – rede alvo do Problema CHILD. Adaptado de [Scutari 2020]. 34
Figura 3.3 – rede alvo do Problema INSURANCE. Adaptado de [Scutari 2020]. 35
Figura 3.4 – Em (A) e (B) é mostrado duas redes,após o cruzamento dessas redes obtivemos a rede (C). 36
Figura 3.5 – A rede (A) foi utilizada para a mutação e podemos obter as redes (B) e (C) dessa mutação. 37
Figura 3.6 – Tempo em segundos do teste do *repair operator*. 37
Figura 4.1 – Em (A) é mostrado a rede alvo. Em (B) é mostrado uma rede igual à rede alvo mas com dois arcos invertidos. 43
Figura 4.2 – Redes utilizadas para o estudo do SLF, TLF e da acurácia. 43
Figura 4.3 – Comparação do tempo computacional dos algoritmos em todas as instâncias do problema ASIA. 44
Figura 4.4 – Comparação da taxa de convergência para a rede alvo dos algoritmos em todas as instâncias do problema ASIA. 45
Figura 4.5 – Comparação da métrica SLF dos algoritmos em todas as instâncias do problema ASIA. 46
Figura 4.6 – Comparação da métrica TLF dos algoritmos em todas as instâncias do problema ASIA. 47
Figura 4.7 – Comparação da acurácia dos algoritmos em todas as instâncias do problema ASIA. (A) é a comparação da acurácia da média das 20 execuções. (B) é comparação da acurácia das execuções que convergiram. 47
Figura 4.8 – Comparação do tempo computacional dos algoritmos em todas as instâncias do problema CHILD, em escala Log. 48
Figura 4.9 – Comparação da taxa de convergência para a rede alvo dos algoritmos em todas as instâncias do problema CHILD. 49
Figura 4.10–Comparação da métrica SLF dos algoritmos em todas as instâncias do problema CHILD. 50
Figura 4.11–Comparação da métrica TLF dos algoritmos em todas as instâncias do problema CHILD. 50

Figura 4.12–Comparação da acurácia dos algoritmos em todas as instâncias do problema CHILD. (A) é a comparação da acurácia da média das 20 execuções. (B) é comparação da acurácia das execuções que convergiram. 51
Figura 4.13–Comparação do tempo computacional dos algoritmos em todas as instâncias do problema INSURANCE. 52
Figura 4.14–Comparação da taxa de convergência para a rede alvo dos algoritmos em todas as instâncias do problema INSURANCE. 53
Figura 4.15–Comparação da métrica SLF dos algoritmos em todas as instâncias do problema INSURANCE. 54
Figura 4.16–Comparação da métrica TLF dos algoritmos em todas as instâncias do problema INSURANCE. 54
Figura 4.17–Comparação da acurácia dos algoritmos em todas as instâncias do problema INSURANCE. (A) é a comparação da acurácia da média das 20 execuções. (B) é comparação da acurácia das execuções que convergiram. 55
Figura 4.18–Tempo computacional de todas as instâncias para os algoritmos MAGA-Tabu, MAGA-Original e ABC. 56
Figura 4.19–Teste Tukey dos algoritmos MAGA-Tabu, MAGA-Original e ABC em relação ao tempo computacional. 57
Figura 4.20–taxa de convergência de todas as instâncias para os algoritmos MAGA-Tabu, MAGA-Original e ABC. 57
Figura 4.21–Teste Tukey dos algoritmos MAGA-Tabu, MAGA-Original e ABC em relação à taxa de convergência. 58

LISTA DE TABELAS

Tabela 3.1 – Os valores dos problemas *benchmark* 35
Tabela 3.2 – O número de avaliações da métrica BIC para várias instâncias do problema INSURANCE, CHILD e ASIA. 40
Tabela 4.1 – Valor da BIC dos algoritmos nas instâncias do problema ASIA. 45
Tabela 4.2 – Valor da BIC dos algoritmos nas instâncias do problema CHILD. 49
Tabela 4.3 – Valor da BIC dos algoritmos nas instâncias do problema INSURANCE. . . 53

SUMÁRIO

Lista de ilustrações vii

Lista de tabelas ix

Sumário x

1 INTRODUÇÃO 16
1.1 Objetivo 17
1.2 Organização 17
1.3 Produção Bibliográfica 18

2 REFERENCIAL TEÓRICO 19
2.1 Redes Bayesianas 19
2.2 Revisão Bibliográfica 22
2.3 MAGA 29

3 METODOLOGIA 33
3.1 Base de dados 33
3.2 MAGA 35
3.3 Outros Algoritmos 38
3.4 Experimentos 39

4 RESULTADOS 42
4.1 Observações das métricas utilizadas 42
4.2 Resultados dos algoritmos 43

5 CONCLUSÕES E TRABALHOS FUTUROS 59
5.1 Trabalhos Futuros 61

REFERÊNCIAS 62

APÊNDICE A DADOS 68

CAPÍTULO

1

INTRODUÇÃO

As Redes Bayesianas - RB (*bayesian networks*) [Koller e Friedman 2009] são um tipo de modelos gráficos probabilísticos que representam a relação entre variáveis aleatórias (v.a.), as quais podem ser discretas ou contínuas. Uma RB é composta por um conjunto de probabilidades condicionais e um grafo acíclico dirigido (*Directed Acyclic Graph*- DAG), onde as variáveis são representadas como nós e a dependência condicional entre elas são representadas com arcos. A direção do arco indica que um nó, chamado de pai, possui influência nos possíveis estados do nó apontado, sendo esse chamado de filho. O conjunto de probabilidades condicionais é representado por uma tabela chamada Tabela de Probabilidade Condicional (*Conditional Probability Table* - CPT), para os casos de variáveis discretas, ou por distribuição de probabilidade condicional no caso de variáveis contínuas [Koller e Friedman 2009]. Dentre o campo de pesquisa relacionado as RBs, temos o desafio de encontrar o DAG que melhor representa os dados amostrados. Esse tema é definido como aprendizagem estrutural das RBs e é um problema NP-difícil [Chickering, Geiger e Heckerman 1995] e de natureza combinatorial, resultando em intensa pesquisa nessa área [Scanagatta, Salmerón e Stella 2019].

Na literatura, podemos encontrar diversas aplicações das RBs em contextos onde a incerteza nas variáveis é relevante. Como, por exemplo, o uso de ferramentas auxiliar no entendimento e na tomada de decisão sobre problemas de saúde, como câncer de pulmão [Sesen *et al.* 2013] e Alzheimer [Gross *et al.* 2018]. Outra aplicação é em problemas de manutenção, como no diagnóstico de falhas em motores de aviões [Sahin *et al.* 2007]. Mais uma utilização é no contexto de predição, como da capacidade de manutenção dos sistemas de um *software* [Okutan e Yıldız 2014], do consumo de energia elétrica de múltiplos domicílios [Bessani *et al.* 2020], dos padrões de movimento de múltiplos veículos em redes tolerantes ao atraso [Wu *et al.* 2020]. Ainda na modelagem de RBs de relatórios de investigação de acidentes para avaliação de segurança da aviação [Zhang e Mahadevan 2021].

Recentemente com o avanço da tecnologia, a sociedade tem produzindo um volume

cada vez maior de informações. Tecnologias como a internet das coisas e armazenamento de informações em ambiente virtual deram o início a era do *Big Data*, que se caracteriza por gerar e gerenciar com uma grande quantidade de dados complexos e diversos [Oussous *et al.* 2018]. Portanto, há grande interesse no desenvolvimento de algoritmos para gerenciar esses grandes volumes de dados. O aprendizado das RBs de grande volume de dados está se tornando cada vez mais valioso para a modelagem e o raciocínio sobre incertezas em muitas áreas no *Big Data*. A maioria dos algoritmos de aprendizagem de estruturas RBs atuais têm deficiência em tratar problemas com grande volume de dados [Tang *et al.* 2019]. Dentre algumas dificuldades do aprendizado das RBs temos o alto tempo computacional que geralmente termina em falha do aprendizado devido a limitações de memória.

Nesse trabalho será implementado um adaptação do algoritmo híbrido chamado Algoritmo Genético Multi-Agente (MAGA) [Zhong *et al.* 2004] para o aprendizado estrutural de RBs. O MAGA foi proposto para otimização numérica global em problemas de larga escala. Esse algoritmo integrou um sistema multi-agente com operadores do Algoritmo GA. No trabalho em que o algoritmo foi proposto apresentou uma otimização de funções com 10.000 dimensões, na qual ele foi o primeiro GA que resolveu problemas com dimensões nessa escala. Além disso, o MAGA apresentou um bom desempenho em problemas de otimização combinatorial e problemas com restrições [Liu, Zhong e Jiao 2006, Liu, Zhong e Jiao 2009]. Os experimentos do trabalho [Zhang *et al.* 2015] mostra o desempenho do MAGA que foi projetado para seis problemas do *Optimization of Big Data 2015 Competition*. Os resultados mostraram, segundo o autor, que o MAGA foi superior aos algoritmos de linha de base em relação às soluções e ao tempo computacional relativamente mais baixo.

1.1 Objetivo

O objetivo desse trabalho é implementar e analisar uma adaptação do MAGA na aprendizagem estrutural de RBs, principalmente para problemas com maior número de variáveis e grande volume de dados. Serão implementadas duas configurações do algoritmo MAGA e outros algoritmos da literatura que serão comparados entre si.

1.2 Organização

Esse trabalho está organizado da seguinte forma. O Capítulo 2 apresenta a revisão bibliográfica, onde serão discutidos o problema de aprendizado de estrutural de RBs, os algoritmos estudados na literatura e o Algoritmo MAGA. O Capítulo 3 apresenta a metodologia, que mostra os bancos de dados utilizados, os métodos e algoritmos empregados e a organização dos experimentos. No Capítulo 4 serão apresentados os resultados dos experimentos. Por fim o Capítulo 5, apresentará as conclusões e os trabalhos futuros.

1.3 Produção Bibliográfica

Parte do trabalho científico desenvolvido nesse estudo foi publicado no XV Encontro Acadêmico de Modelagem Computacional (EAMC).

- MACHADO, I. G.;BESSANI, M. Aprendizagem Estrutural de Redes Bayesianas utilizando Algoritmo Genético Multi-Agente. XIV Encontro Acadêmico de Modelagem Computacional. Petrópolis - RJ. 2021

CAPÍTULO

2

REFERENCIAL TEÓRICO

Neste Capítulo, serão apresentados os estudos relacionados ao aprendizado de estruturas das Redes Bayesianas (RB) e o algoritmo MAGA. Primeiro serão apresentados os conceitos importantes das RBs. Depois terá a revisão bibliográfica dos algoritmos utilizados na aprendizagem estrutural de RBs. Por fim, será apresentado o algoritmo MAGA.

2.1 Redes Bayesianas

As Redes Bayesianas, também chamadas de Redes Casuais, Rede de Crença ou Gráficos de Dependência Probabilística, surgiram na década de 80 e têm sido aplicadas em uma grande variedade de atividades. Tais como: nas áreas de finanças, saúde, desenvolvimento de jogos, entre outros [Bobbio *et al.* 2001]. O teorema de Bayes descreve a probabilidade de dado um evento ocorrer, sendo que foi observado a ocorrência de outro determinado evento [Daly, Shen e Aitken 2011]. Esta probabilidade é calculada pela expressão:

$$P(E \mid F) = \frac{P(F \mid E)P(E)}{P(F)} \tag{2.1}$$

Em que $P(E)$ e $P(F)$ representam as probabilidades a priori dos eventos E e F; $P(E|F)$ é a probabilidade a posteriori (probabilidade condicionada) de E condicional a F e $P(F|E)$ é a probabilidade a posteriori de F condicional a E [Daly, Shen e Aitken 2011].

Na Figura 2.1 temos um exemplo de RBs em que podemos observar a relação entre as variáveis. Se os nós B e E apontam para o nó A, os estados da v.a. B e da v.a E afetam a probabilidade dos possíveis estados de A. Assim como a probabilidade do v.a. C ocorrer depende do que foi observado em A. Além disso, um grafo que represente uma rede Bayesiana deve ser um Grafo Acíclico Dirigido (*Directed Acyclic Graph* - DAG), de modo que um determinado nó não possua nenhuma ligação que comece e termine em si mesmo, não criando, portanto, circuitos no Grafo [Daly, Shen e Aitken 2011]. Dado o Grafo da RB, sendo um conjunto de

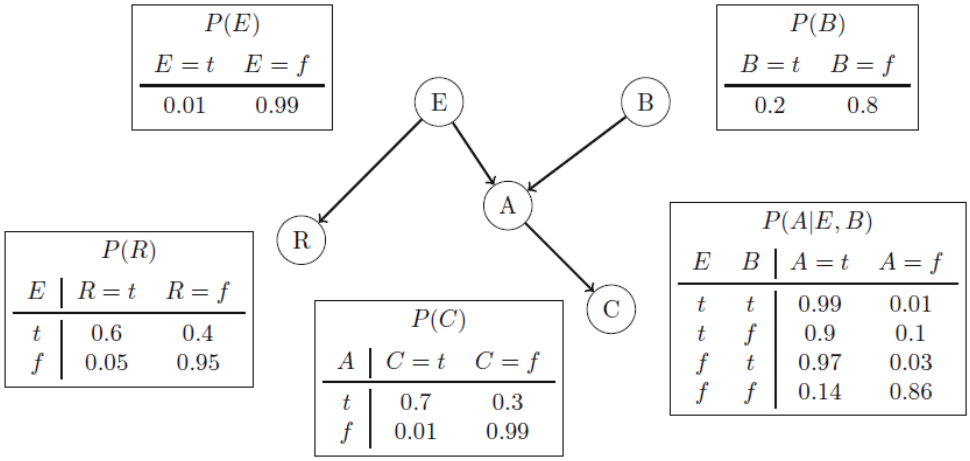

Figura 2.1 – Exemplo de uma rede Bayesiana. Adaptado de [Scanagatta, Salmerón e Stella 2019].

v.a. $X = X_1, X_2, ..., X_n$ temos a Eq.(2.2) que representa a distribuição de probabilidade conjunta dessas variáveis.

$$P(X) = P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i | P_{a_i}) \tag{2.2}$$

Onde P_{a_i} indica o conjunto de pais do nó X_i, e $P(X_i|P_{a_i})$ representa a distribuições de probabilidade condicional de cada nó. A distribuição de probabilidade conjunta pode ser representada pelo produto das probabilidades condicionais. Portanto, atribuindo determinados valores para parte das variáveis, podemos encontrar a probabilidade dos valores das variáveis não conhecidas ocorrerem.

Para variáveis discretas, as probabilidades condicionais de cada variável são representadas por uma tabela chamada Tabela de probabilidade condicional (*Conditional Probability Tables* - CPT) [Koller e Friedman 2009]. Como é mostrado na Figura 2.1, utilizando a variável "A"como exemplo. Ela possui dois pais, "E"e "B", portanto, ela detém uma relação de dependência dessas duas variáveis. Pela Eq.(2.2) obtemos $P(E)$, $P(B)$ e $P(A|E,B)$, esses valores são mostrados na Figura 2.1. Podemos observar que os estados de "E"e "B" influenciam os estados de "A". Para compreender a estrutura de RBs, é preciso obter a distribuição de probabilidade conjunta (*joint probability distribution*- JPD) de todas as variáveis e a estrutura a partir dos dados. O processo de estimar a distribuição condicional, incluindo parâmetros dos dados, é chamado de aprendizado de parâmetro. Na literatura, temos a abordagem de verossimilhança [Aldrich *et al.* 1997], algoritmo de maximização de expectativa [Dempster, Laird e Rubin 1977], entre outros. A aprendizagem de estrutura é o processo para obter a estrutura que melhor representa o conjunto de dados e que seja uma DAG. Na literatura, temos três principais abordagens para o problema de aprendizagem estrutural [Contaldi 2016]: *Constraint-based*, *Search and Score* (SS) e a híbrida.

Constraint-based foi o primeiro método estudado para a aprendizagem estrutural de redes Bayesiana. A principal ideia é encontrar a melhor estrutura possível, restringindo o espaço

de busca progressivamente de alguma maneira. Na literatura os algoritmos que utilizam esse método, seguem três passos que são originados do teste de independência condicional (IC) que foram estudados nos algoritmos *Causality Search* (PC) e *Grow-Shrink* (GS) [Contaldi 2016]. O primeiro passo, opcional, define o *Markov blanket* para cada nó com o objetivo de eliminar o maior número possível de candidatos na criação das estruturas. O *Markov blanket* obtém o conjunto de nós que possui alguma dependência do nó avaliado, que irá resultar em um conjunto de possíveis pais, filhos e os outros pais de seus filhos. Logo após é feito o teste de simetria, verificando se dado um nó X que está no conjunto dos nós possíveis de Y e o nó Y está também no conjunto dos nós possíveis de X. Caso se encontre uma assimetria, os nós serão removidos de cada conjunto. No segundo passo, o objetivo é a criação da estrutura não direcionada, que a literatura chama de esqueleto. Em que consistem em identificar se há dependência entre os nós, caso tenha ocorrido o primeiro passo há uma redução no número possível de combinações de nós. Por fim, o terceiro passo é a definição da direção dos arcos da estrutura criada anteriormente.

O *Search and Score* é definido por uma função de pontuação para estimar a qualidade de uma rede. O objetivo de utilizar essa métrica é encontrar no espaço de busca a rede que possui melhor qualidade possível. Dentre essas métricas de avaliação das redes, algumas analisam a RBs quanto à complexidade estrutural e à verossimilhança com os dados.

O *Log-likelihood* (LL) representa o logaritmo da verossimilhança, e para uma rede Bayesiana é calculada pela Eq. (2.3).

$$LL(R \mid D) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} log \left(\frac{N_{ijk}}{N_{ij}} \right) \tag{2.3}$$

A métrica BIC [Schwarz *et al.* 1978] é calculada pela verossimilhança menos a complexidade estrutural da rede com uma penalidade nessa complexidade. A BIC é apresentada pela Eq. (2.4) em que n é o número de nós; sendo r_i é o número de estados do i-ésimo nó; q_i é o número de pais do i-ésimo nó, sendo que o valor de q_i é igual a 1 se o i-ésimo nó não possuir pais; N é o número de observações na amostra D; N_{ijk} é o número total de observações na amostra em que o i-ésimo nó assume seu k-ésimo estado e o seus pais assumem o j-ésimo estado; N_{ij} é a quantidade de observações do i-ésimo nó quando seus pais assumem o j-ésimo estado.

$$BIC(R \mid D) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} log \left(\frac{N_{ijk}}{N_{ij}} \right) - \frac{1}{2} log(N) \sum_{i=1}^{n} (r_i - 1) q_i \tag{2.4}$$

Além da BIC, na literatura podemos encontrar outras métricas como a *Akaike information criterion* (AIC) [Akaike 1974], *Bayesian Dirichlet equivalent uniform* (BDeu) *score* [Heckerman, Geiger e Chickering 1995], *Bayesian Dirichlet with Jeffrey's prior (BDJ) score* [Scutari 2016] e *K2 score* [Cooper e Herskovits 1991]. As métricas buscam qualificar as Redes Bayesianas, proporcionando uma melhor representação das amostras aleatórias disponíveis e penalizando

as estruturas com maior número de arcos. As RBs com maiores valores de métricas tendem a possuir menores erros estruturais.

A outra abordagem é a hibrida. Ela é uma combinação das duas abordagens anteriores, das quais os testes de independência são realizados para restringir o espaço de busca que será explorado pelo SS. Primeiro é aplicado o teste de independência, para a criação da estrutura esqueleto, logo após é aplicado uma métrica de pontuação que irá direcionar os arcos dessa estrutura.

2.2 Revisão Bibliográfica

Os artigos citados têm como objetivo desenvolver algoritmos que sejam eficientes para aprendizagem estrutural de RBs buscando ter um baixo tempo computacional. Visando atingir esse fim, os autores utilizam métodos exatos e heurísticos. Além disso, em geral, os artigos citados a seguir baseiam-se nos problemas do repositório [Scutari 2020] para a obtenção dos resultados dos estudos propostos, sendo ASIA, CANCER, ALARM e INSURANCE, os mais utilizados nos trabalhos acadêmicos.

O aprendizado estrutural de RB pode ser apresentado como um problema restrito, tornando o problema NP-completo. A programação Linear é área de estudo com métodos de aproximação de problemas de otimização combinatória teoria de grafos [Bartlett e Cussens 2017]. A aplicação da programação linear no aprendizado estrutural de RB é estudo desde 2006 [Guo e Schuurmans 2006]. Dentre as restrições aplicadas nesse tipo de abordagem temos a restrição dos números de pais de cada nó e que a rede seja DAG. A função de pontuação da rede com essas restrições podem ser escritas como função linear de variáveis binárias, podendo então utilizar a programação linear Inteira para o aprendizado estrutural [Bartlett e Cussens 2017]. Abordagem como programação dinâmica [Silander e Myllymaki 2012], A * search [Yuan e Malone 2013] e Branch and-Bound [Campos e Ji 2011] são utilizadas para o aprendizado estrutural com restrições. O trabalho [Campos e Ji 2011] comparou o método de programação dinâmica e Branch and Bound. O resultado mostrou que para problemas com poucas variáveis é mais indicado a programação dinâmica, para problema tiver tamanho médio, o método branch and-bound é mais indicado e para problemas grandes o branch-and-bound irá dar uma aproximação da solução, mas a programação dinâmica irá falhar.

Na abordagem híbrida, os trabalhos na literatura focam nos métodos de restrição para criação da estrutura esqueleto. No trabalho [Tsamardinos, Brown e Aliferis 2006] apresentou o método *Max-Min Hill-Climbing* (MMHC). Nesse trabalho, o MMHC utiliza o Máximo-Mínimo PC (MMPC) para criação da estrutura esqueleto e a Escalada de Colina (*Greedy Hill-Climbing*) para direcionar os arcos. Os resultados dos testes constataram que o MMHC possui um desempenho superior aos demais algoritmos testados: PC, *Greedy Search* (GS), *Sparse Candidate* (SC), dentre outros. Ademais, o autor demonstra que a utilização do MMHC dispensa

restrições quanto ao número de pais para cada nó, uma vez que essa estratégia resulta em um aumento no tempo computacional. No trabalho [Contaldi, Vafaee e Nelson 2019], apresentou um algoritmo chamado elitGA. Esse algoritmo apresenta uma idéia de redução dos parâmetros do Algoritmo GA. Uma restrição é aplicada nesse algoritmo que limita o número de pais que cada nó pode ter, essa restrição é uma forma de redução do espaço de busca. Os resultados mostram que os métodos evolutivos têm um desempenho geralmente superior que os algoritmos não evolutivos em problemas com grandes números de variáveis. Além disso, os resultados mostraram que a parametrização adaptativa proposta melhora o valor da métrica de avaliação e obtêm um menor erro estrutural nos testes. Foi utilizado nesse algoritmo, para redução do espaço de busca, um algoritmo chamado Opt01SS [Villanueva e Maciel 2014] para encontrar a estrutura esqueleto da RBs. No artigo [Dai *et al.* 2020], o algoritmo proposto é um algoritmo genético (GA) aprimorado, onde se utiliza a métrica Informações Mútuas (MI) para modificar adaptativamente o espaço de busca, essa abordagem foi chamado MIIGA. Os resultados experimentais demonstraram que o algoritmo proposto tende a ter um melhor desempenho em termos de qualidade das soluções, velocidade de convergência e tempo de execução, em comparação com os outros três algoritmos baseados em população.

O trabalho [Cooper e Herskovits 1992] propôs um algoritmo denominado K2 para o aprendizado de estruturas de RBs, como uma Busca Gulosa (*greedy search*). Inicialmente, o algoritmo K2 assume que os nós não possuem pais. Então, para cada nó, adiciona de maneira incremental um pai, o que resulta no aumento da métrica K2 (verossimilhança da estrutura resultante). Quando a adição de pais aos nós deixa de aumentar probabilidade da estrutura resultante, o algoritmo K2 para. Apesar de não garantir a melhor estrutura global, o K2 tem sido amplamente utilizado, combinado com algumas métricas de avaliação, para aprendizado estrutural de Redes Bayesianas. Utilizando os problemas ASIA e ALARM, foi analisado o desempenho dos parâmetros de controle do algoritmo genético. Foi comparado o uso do elitismo no GA e sem o uso dele.

O artigo [Larrañaga *et al.* 1996] foi um dos primeiros a utilizar o algoritmo genético para o aprendizado de estruturas das RBs. Foram realizados análises dos parâmetros do algoritmo Genético e testes dos problemas apresentados pela literatura. A classe do algoritmo de aprendizagem estrutural utilizado foi a abordagem *Search and Score* e a métrica utilizada de pontuação foi a k2 *score*. A partir dos resultados, os autores recomendaram um algoritmo híbrido e que, além disso, utilizasse algum método de elitismo, junto a um tamanho de população não muito baixo e uma porcentagem de mutação baixa.

O trabalho [Chickering 2002] abordou um algoritmo de busca gulosa chamado *Greedy Equivalence Search* (GES). Esse algoritmo consiste em duas fases. Na primeira fase, o algoritmo adiciona arcos que possuam melhores avaliações dentre os arcos possíveis, até que não seja possível adicionar uma arco que melhore o valor da métrica ou que crie um grafo cíclico. Na segunda fase, o grafo gerado passa por um processo onde são removidos todos os arcos que

resultem na melhoria da avaliação do grafo. Os experimentos desse trabalho foram de comparar o GES com outras três variantes do GES, os dados utilizados são problemas reais. Os resultados mostraram que a qualidade da solução era praticante a mesma das outras abordagem de busca gulosa. Outro resultado mostrado pelo autor foi que limitar o número de vizinhos resulta em um aumento no tempo computacional.

Em 2007, foram apresentados os primeiros trabalhos relevantes utilizando o *Particle Swarm Optimization* (PSO) para problemas de aprendizado de estruturas de Redes Bayesianas. Os trabalhos [Xing-Chen *et al.* 2007], [Sahin e Devasia 2007] descrevem o PSO de forma discreta, nomeado de *Binary* PSO pelos autores. Um trabalho do mesmo ano [Sahin *et al.* 2007] demonstra uma aplicação do PSO discreto em um problema de diagnóstico em falha de motores de avião. Nesses três trabalhos, apesar de os testes terem sido realizados com problemas de até 16 variáveis, os autores apontam o PSO discreto como um algoritmo promissor para problemas com muitas variáveis.

Kabli [Kabli, Herrmann e McCall 2007] propôs um algoritmo chamado ChainGA, o qual consiste em uma nova métrica de avaliação para o algoritmo genético K2GA [Larranaga *et al.* 1996]. Para tanto, adicionou-se à métrica de avaliação K2 - procedente do K2GA - um método de estruturas em cadeia para a seleção e avaliação de ordenação de nós. Resultando assim, no algoritmo proposto, que funciona da seguinte forma: primeiro, o GA cria as ordens dos nós; em seguida, o método em cadeia seleciona as melhores ordens desses nós; e, por fim, o K2 avalia essa seleção. A abordagem de aprendizagem estrutural utilizada foi a *Search and Score*. A redução do número de avaliações da métrica K2 foi a grande vantagem da utilização do método em cadeia. Em função disso, os resultados do estudo apresentaram um menor tempo computacional do ChainGA em relação ao K2GA. Contudo, o erro estrutural revelou-se maior para o ChainGA.

Wang [Wang e Yang 2010] apresentou, uma nova abordagem do PSO discreto, na qual propôs duas modificações no PSO binário. A primeira trata-se da regra de atualização da velocidade baseada na representação binária e, a segunda consiste da regra de atualização da posição com base no que o autor denominou “operação de mutação estocástica”. Ao comparar os resultados do PSO proposto ao GA e ao PSO clássico, concluiu-se que o primeiro revelou uma convergência superior aos demais. Além disso, apesar do PSO proposto mostrar-se mais lento do que o GA, visto que o tempo computacional das gerações do primeiro aumentou mais rapidamente em relação ao segundo, o PSO proposto ainda possui uma demanda significativamente menor de gerações para convergir. Ademais, o autor descreve também que o GA possui escalabilidade inferior ao PSO proposto, uma vez que, à medida que se aumenta o número de v.a., a taxa de convergência é mais reduzida no primeiro do que no segundo.

O trabalho [Cao e Fang 2010] implementou um algoritmo híbrido baseado em GA e *Simulated Annealing* (SA). O autor utilizou o GA pela sua forte capacidade de fazer uma procura global. Para refinar os resultados obtidos de cada nova geração, foi empregado o SA. Nesse

trabalho foram feitas comparações do algoritmo proposto com outros quatro algoritmos, sendo o *Hill Climbing*, o algoritmo K2, o SA e o GA. Os resultados mostram que o algoritmo proposto teve um menor tempo computacional e uma menor taxa de erro. Nesse trabalho o autor testou os algoritmos apenas em três problemas com apenas cinco variáveis.

O artigo de Fournier [Fournier *et al.* 2010], apresentou um comparativo entre quatro métodos de aprendizagem estrutural de Redes Bayesianas, aos pares: a métrica de avaliação K2 comparada à métrica de avaliação Chain, e o algoritmo genético comparado ao algoritmo de colônia de formigas. Assim, os métodos testados foram os seguintes: K2GA, ChainGA, K2ACO e ChainACO. Os resultados revelaram as melhores estruturas para o algoritmo K2GA, porém, o tempo computacional é o que apresentou maior valor. A abordagem utilizada nesse trabalho foi a *Search and Score*. Esse trabalho de 2010, está contido em uma tese de doutorado defendida em 2012 [Wu 2012].

No trabalho [Adabor, Acquaah-Mensah e Oduro 2015] foi mostrado uma aplicação de um algoritmo híbrido chamado SAGA. Esse algoritmo utiliza o SA para a busca no espaço global e o *Greedy Search* para refinar a solução do SA. Nesse artigo foi utilizado a abordagem de algoritmos *Search and Score* e a métrica foi a BDe. O experimento nesse trabalho comparou o algoritmo proposto com outros, sendo eles o *Greedy Search* com reinício randômico (GR), O SA com *Reannealing* (SAR), o *Greedy Search* (GS) e o SA. Os resultados mostraram que como em outros trabalhos da literatura, todos os algoritmos tiveram bons resultados para problemas com poucas variáveis, mas quando o problema possui grandes números de variáveis os algoritmos encontram dificuldades. O SAGA obteve menos erro estrutural e maior pontuação na métrica BDe comparado com os demais algoritmos.

O artigo [Gheisari e Meybodi 2016] descreveu uma nova abordagem para o algoritmo PSO, o qual foi intitulado *Bayesian Network Construction Algorithm Using PSO* (BNC-PSO). Para operações de cálculos de velocidade e posição, o algoritmo proposto utilizou operadores de cruzamento e mutação, baseados em algoritmo evolutivo; para impedir soluções inválidas, empregou um procedimento de remoção de circuito; e para a convergência global, utilizou o teorema da cadeia de Markov. O algoritmo BNC-PSO é classificado como *Search and Score* e o autor utilizou o *Bayesian Dirchlet, Structure learning factor* (SLF) e *topology learning factor* (TLF) como métrica de avaliação. A partir dos problemas apresentados na literatura, o autor comparou o BNC-PSO a outros algoritmos da literatura, como o *Hill Climb*, *Genetic Algorithm*, ACO básico, MMHC e *Greedy Search*. Os resultados revelaram que o BNC-PSO apresentou o menor erro de estruturas, obtendo a melhor qualidade da estrutura da rede Bayesiana. Além disso, o autor apontou como vantagens do PSO, a fácil implementação e o fato das partículas possuírem memória, o que possibilita a retenção do conhecimento de boas soluções. Em contrapartida, as desvantagens do PSO é o elevado tempo computacional para cada geração do algoritmo, visto que, quanto mais gerações forem necessárias para obter um resultado, mais custoso será computacionalmente; e os problemas com grandes conjuntos de dados, assim como muitos

algoritmos da literatura, o PSO pode não encontrar a solução ideal, diferentemente do que ocorre para problemas com pequenos conjuntos de dados.

No trabalho [Ji *et al.* 2017], o autor realizou um estudo sobre métodos de enxame para o aprendizado de estruturas de Redes Bayesianas. No artigo, o autor utiliza três métodos de enxames, sendo eles, Colônia de Formiga (ACO), Colônia de Abelha (ABC) e *Bacterial Foraging Optimization* (BFO) que é um método baseado em bactérias. Esses métodos foram analisados, definidas as suas principais características e, por fim, comparados. O autor utilizou três comparações iniciais: métrica K2, diferenciação estrutural e tempo computacional. Primeiro, comparou-se em relação à métrica de avaliação K2, no qual foi considerado o valor máximo, mínimo e a média da métrica K2. Como segunda comparação, foram utilizadas a diferenciação estrutural entre as estruturas de redes obtidas e a original, comparando-se a menor, a maior e a média da diferença estrutural. Por fim, foi comparado o tempo computacional, obtendo o menor, o maior e a média dos tempos computacionais de cada execução. Além dessas, o autor comparou também a robustez dos algoritmos em relação a erros nos dados, denominado ruído nos dados, utilizando os seguintes valores: 5%, 10%, 15%, 20% e 25%, e adicionou dois novos algoritmos: subida de encosta (*hill climb*) e busca tabu.

Os resultados do estudo evidenciaram que o ACO não apresentou vantagens significativas em relação aos demais algoritmos. Já o ABC obteve maiores valores de avaliação K2 e, ainda, apresentou o menor tempo computacional e maior verossimilhança. O BFO apresentou-se superior quando se deseja uma menor diferença estrutural em relação à rede alvo. Para os testes de robustez, os resultados ressaltaram que, com exceção do tempo, os três algoritmos de enxames foram superiores aos outros dois algoritmos - *hill climb* e busca - tabu, principalmente quando os dados possuem ruídos e o espaço de busca é grande.

Em [Khanteymoori *et al.* 2018], o autor apresentou um algoritmo híbrido chamado *Breeding Swarm*. Esse algoritmo utiliza o GA com uma forma discreta do PSO. Cada indivíduo do GA é representado também com uma partícula, há cada geração o algoritmo é dividido em dois grupos onde uma passa pelos operadores do PSO para o cálculo de posição e velocidade e o outro grupo passa pelos operadores do GA. A definição de quantos indivíduos vão ter em cada grupo depende de uma variável chamada *breeding rate*. Os experimentos iniciais mostram o comportamento do algoritmo proposto com teste dos problemas conhecidos da literatura e experimentos que definem um valor apropriado para a variável *breeding rate*. As métricas utilizadas nesse trabalho foi a *Bayesian Dirchlet, Structure learning factor* (SLF) e *topology learning factor (TLF)*. Depois o autor comparou o algoritmo com treze algoritmos da literatura, sendo quatro deles baseados em GA e PSO. Os resultados demonstraram que o algoritmo proposto superou completamente os quatro algoritmos populacionais e também apresentou um desempenho promissor em relação aos outros nove métodos. Nesses testes foram avaliados o erro estrutural e a métrica *Bayesian Dirchlet*. Os problemas utilizados na literatura foram de 8 até 58 variáveis.

O artigo [Zhang *et al.* 2018] apresentou uma nova abordagem do algoritmo ACO. Intitulado coACO, esse algoritmo inicialmente divide uma colônia de formiga em grupos independentes de formigas. Em seguida, um algoritmo evolucionário denominado Evolução Diferencial (ED), é utilizado para orientar o processo de coevolução para todos os grupos, sendo que cada grupo de formigas possui uma matriz de feromônios. As operações de mutação e cruzamento do DE podem ajustar efetivamente as informações de cooperação e conduzir todos os grupos de formiga à evolução em direção ao ideal de maneira cooperativa. Os problemas ASIA e ALARM foram utilizados para o teste dos algoritmos, a abordagem utilizada foi a *Search and Score* e utilizou a métrica BIC. Os resultados desse trabalho, apresentam um aumento na velocidade e na qualidade de convergência em relação ao ACO básico e a outros algoritmos testados. Por fim, o autor não mencionou porque o tempo do CPU foi maior para o coACO.

O trabalho [Li, Wang e Li 2019] utilizou o método *Intuitionistic Fuzzy Set* (IFS) para integrar o conhecimento prévio de especialistas ao aprendizado do Algoritmo Genético. Nesse artigo, o autor criou uma estrutura inicial para uma população inicial baseada no conhecimento prévio de especialistas sobre o problema. Posteriormente, por meio da métrica de avaliação BIC, aplicou-se os conhecimentos de especialistas como um peso na equação da avaliação BIC, adaptando a métrica, a qual foi intitulada IBIC. Em seguida, o autor utiliza o Algoritmo Genético com a métrica IBIC. Os resultados do novo algoritmo revelam um erro de estrutura reduzido em comparação a vários algoritmos, dentre eles, chainGA, *Hill Climb* (HC), algoritmo K2 e algoritmo PC. Apesar disso, a eficiência do algoritmo é dependente do conhecimento de especialistas a priori.

Em 2019 foi proposto um algoritmo [Sun *et al.* 2019] de colônia de abelhas em que o autor dividiu a população em subgrupos baseados no algoritmo *cuckoo*, chamado *multi-group artificial bee colony algorithm based on cuckoo algorithm* (CMABC). Esse algoritmo híbrido foi testado com os problemas da literatura com 8, 27 e 37 variáveis. E depois foi comparado com algoritmos da literatura com o GA, ABC básico. Os resultados mostraram que o CMABC comparado com os demais algoritmos obteve valores de BIC superiores e com menor número de interações que os algoritmos GA e ABC básico.

Em 2019 também um outro trabalho relacionado a algoritmos híbridos foi publicado. O artigo [Lee e Kim 2019] mostra uma abordagem nova ao algoritmo SAGA de 2014. O algoritmo proposto pelo autor, chamado *Parallel Simulated Annealing with a Greedy Algorithm* (PSAGA), emprega uma abordagem paralela para execução simultânea em várias instâncias do SAGA. Baseado no *multi-thread* para acelerar as operações de pesquisa, cada *thread* executa independentemente uma instância SAGA baseada na cadeia de Markov (*markov chain*). O artigo compara o PSAGA com algoritmo da literatura como o SA, *simulated reannealing* (SR), *parallel* SR (PSR) e o SAGA. Os resultados mostraram que o PSAGA resulta em redes com maior verossimilhança e com menor erro estrutural comparado com os outros algoritmos.

O trabalho [Scutari, Graafland e Gutiérrez 2019] mostrou uma comparação entre os

tipos de classes de aprendizagem estrutural de RBs. Foram utilizados três algoritmos para a comparação entre as abordagens; para a abordagem *constraint-based* o trabalho comparou os algoritmos: Pc-stable, Grow-shrink e Inter-IAMB. Na abordagem *Search and Score* foram utilizados os algoritmos busca tabu, *simulating annealing* e *Greedy Equivalent Search* (GES). E na abordagem híbrida foram usados os algoritmos *Max-min Hill Climbing* (MMHC), RSMAX2 e H2PC. Nesse artigo foi observado que os algoritmos *constraint-based* obtiveram maiores acurácias que os algoritmos de abordagem *Search and Score*. Os autores não conseguiram perceber uma diferença na acurácia obtida pelos algoritmos de abordagem *constraint-based* e os de abordagem hibrida. E ainda perceberam que os algoritmos de abordagem hibrida são mais rápidos que os algoritmos de abordagem *constraint-based* e de abordagem *Search and Score*.E por fim Concluíram que a busca tabu é mais rápido do que a maioria das abordagens híbridas e *constraint-based*.

O trabalho [Behjati e Beigy 2020] apresentou uma nova proposta para o algoritmo K2. Nesse algoritmo possui uma ordem de variáveis de formas aleatórias, o algoritmo proposto pelo autor contribuiu na ordenação das variáveis. Inicialmente é contraído um grafo a partir dos dados e em seguida é extraído os componentes fortemente conectados a esse grafo. Em seguida, é criado a ordem das variáveis através desses componentes. O algoritmo proposto foi significativamente superior aos métodos utilizados para comparação, na maioria dos conjuntos de dados. Mas o desempenho do método proposto depende da qualidade do conjunto pai para cada variável. Para grandes redes, é um grande desafio fornecer um conjunto de pais candidatos para cada variável. Para superar esse desafio, o autor propõe que o conjunto de pais seja aproximado ao invés de ser exato. A grande vantagem do algoritmo proposto é o seu tempo computacional menor que os outros métodos testados, e a qualidade das redes aprendidas pelo algoritmo.

No trabalho [Martins *et al.* 2021], foi estudado a variação de um algoritmo chamado *Hybrid Multi-objective Bayesian Estimation of Distribution Algorithm* (HMOBEDA), aplicado no problema combinatório da paisagem MNK. O HMOBEDA é uma abordagem híbrida de EDA. O termo híbrido é referente a inclusão de uma busca local em sua estrutura baseada em PGM para melhorar o desempenho permitindo ao algoritmo um melhor processo de busca, a busca local é usado para amostrar novos indivíduos. Nos experimentos foram avaliados três aspectos: habilidade de otimização, robustez e eficiência de aprendizagem. Foram utilizados os indicadores de hipervolume (HV-) e a métrica de distância geracional invertida (IGD). Na comparação entre os algoritmos foram testados: HMOBEDAk2, HMOBEDAhck2, HMOBEDAsparse e HMOBEDAtabu que utiliza a abordagem *Search and Score*;HMOBEDAiamb e HMOBEDApc para a abordagem *constraint-based*; e o algoritmo HMOBEDAmmhc para a abordagem híbrida. No artigo, foi descrito que os algoritmos de aprendizagem estrutural baseada em *Search and Score* parecem ser a melhor escolha. O algoritmo HMOBEDAk2 mostrou ser compatível com as outras variações em tempo de execução de convergência e cobertura da fronteira de Pareto final para problemas com muitos objetivos. Além disso, o HMOBEDAk2 apresentou melhores resultados quando a comparação foi utilizando o *structural Hamming Distance* (SHD). Por

fim, na análise de robustez o autor concluiu que o HMOBEDAk2 e o HMOBEDAhc-k2 são menos sensíveis aos ruídos em comparação as outras variantes do HMOBEDA. E ainda, o HMOBEDAk2 possui menor tempo computacional, podendo ser considerado a melhor opção para os modelos RB para HMOBEDA.

O artigo [Constantinou *et al.* 2021] propõe uma metodologia que aplica aos algoritmos incorporá aos dados um ruído sintético, para que se entenda melhor o desempenho dos algoritmos de aprendizagem estrutural quando aplicado a dados reais. Foram testados algoritmos das classes *Search and Score*, *constraint-based* e híbrido. Os algoritmos usados foram: PC-Stable, FGES, FCI, GFCI, RFCI-BSC, Inter-IAMB, MMHC, GS, HC, TABU, H2PC, SaiyanH, ILP, WINASOBS e NOTEARS. Os casos estudados foram o ASIA, ALARM, pathfinder, sports, ForMed e Property. Os resultados dos experimentos foram obtidos através de 10.000 execuções individuais, e ainda se restringiu o tempo de aprendizado por cada execução em seis horas. Embora o objetivo do trabalho seja o estudo da aprendizagem estrutural com ruídos nos dados, devido ao grande número de experimentos, os resultados permitiram demostrar os pontos fracos e forte de cada algoritmo em problemas pequenos e grandes, para dados com e sem ruído. Os resultados apresentados no trabalho sugere que o ruído nos dados pode ter um impacto considerável na precisão dos grafos obtidos. Os experimentos têm implicações importantes, dado que os trabalhos na literatura utilizam dados sem ruídos, os resultados obtidos nesses trabalhos podem estar superestimando o desempenho dos algoritmos no mundo real em um grau mais alto do que o assumido. Mas ainda assim, os experimentos com dados sem ruído continuam importantes na avaliação de algoritmos de aprendizagem estrutural. Além disso, os resultados sugerem que a redução da dimensionalidade nem sempre reduz o tempo computacional ou a complexidade do aprendizado. Por fim, o artigo não explorou a implementação dos *knowledge-based constraints* no processo de aprendizagem estrutural. Pois, os autores tiveram dificuldade em configurar os experimentos de forma justa. A implementação do *knowledge-based constraints* nos algoritmos é uma área que os autores desejam investigar.

Os problemas utilizados nesse trabalho serão ASIA, CHILD e INSURANCE. Nos trabalhos [Larrañaga *et al.* 1996, Wang e Yang 2010, Sun *et al.* 2019] foram utilizados instâncias com amostras de dados iguais a 500, 1000 e 2000 para o problema ASIA. No artigo [Sun *et al.* 2019] temos os valores de amostras do INSURANCE iguais à 500, 1000, 2000 e 5000. No trabalho [Gheisari e Meybodi 2016] os valores são 500, 1500, 3000 e 5000 e por fim no trabalho [Lee e Kim 2019] temos valores de 50000 e 100000 para os problemas do CHILD e INSURANCE. Esses valores são bem próximos e alguns iguais ao utilizado nesse trabalho.

2.3 MAGA

Para a realização desse trabalho, será implementado um adaptação do algoritmo híbrido chamado MAGA para o aprendizado de estruturas de RBs. O MAGA é um sistema multi-agente

com operadores do algoritmo genético. Nesse algoritmo cada um dos agentes será representado por um agente físico, que irá interagir com o ambiente que estão. O agente possui as seguintes propriedades: pode viver e atuar no ambiente; é capaz de obter informações do ambiente e interagir com essas informações. Um sistema multi-agente é composto de vários agentes que interagem entre si e trabalham em conjunto para a resolução de um problema. No MAGA, cada agente representa uma solução para o problema. Os operadores de cruzamento, mutação e seleção são aplicados nos agentes em cada geração. Os trabalhos [Zhong *et al.* 2004, Zhang *et al.* 2015, Li e Liu 2016, Huang, Liu e Yao 2017, Wang e Liu 2017] mostraram algumas das características do MAGA. Dentre essas características, temos as que os agentes estão em formato de rede onde cada agente tem quatro outros agentes na sua vizinhança, podendo comunicar-se apenas com esses vizinhos. Outra característica é que o algoritmo possui 3 principais funções: competição entre os vizinhos para escolher qual o vizinho que será substituído, os operadores de mutação e cruzamento entre os vizinhos e a função de auto-aprendizagem do agente. Esses trabalhos citados indicam a utilização da auto-aprendizagem no melhor agente da geração ou em um pequeno grupo dos melhores agentes da geração, pois, é um operador caro computacionalmente.

O trabalho [Huang, Liu e Yao 2017] mostra uma aplicação do MAGA no problema de agrupamento de módulos de *software* (SMCP). Esse problema é para encontrar um conjunto específico de módulos, os quais são organizados em *clusters* de acordo com critérios predefinidos. O artigo [Peng *et al.* 2019] aplicou o MAGA no *job shop scheduling problem* - JSSP. Esse problema tem como objetivo a otimização do tempo na fabricação do produto ou no custo da fabricação, através da organização da ordem do processamento de cada trabalho a ser processado em cada máquina usando os recursos da máquina e das matérias-primas. Alguns artigos como os [Li e Liu 2016, Wang e Liu 2017] trabalham com a detecção de comunidades em redes complexas. Nesses trabalhos são apresentados a construção de estruturas de redes com mais de 20 nós, focados na otimização da partição entre os grupos de nós.

Os trabalhos já citados sobre o MAGA, mostram que ele se comporta bem em problemas com grande volume de dados. Nesse trabalho, o objetivo é a implementação de uma adaptação do MAGA para o aprendizado de estruturas de RBs, desejando observar o seu comportamento em diferentes volumes de dados, observando principalmente em instâncias com maior número de variáveis e grande volume de dados. Os trabalhos anteriormente citados na literatura para a aprendizagem estrutural de RBs, mostraram que os algoritmos já propostos possuem uma dificuldade quando se trata de problemas com muitas variáveis ou grande volume de dados.

No MAGA, os agentes representam os indivíduos que estão distribuídos em um ambiente em formato de grade, de tamanho L, como mostra a Figura 2.2. O tamanho da grade, L, é definido com $L_{size} \times L_{size}$, onde $L_{size} \in \mathbb{N}^*$. Supondo que um agente se localiza na posição (i, j) sendo $i, j \in \{1, 2, \ldots L_{size}\}$, o conjunto de vizinhos é definido como mostra a Eq. (2.5).

$$Vizinhos_{i,j} = \left\{L_{i',j}, L_{i,j'}, L_{i'',j}, L_{i,j''}\right\} \tag{2.5}$$

$$i' = \begin{cases} i-1 & i \neq 1 \\ L_{size} & i = 1 \end{cases}, \quad j' = \begin{cases} j-1 & j \neq 1 \\ L_{size} & j = 1 \end{cases},$$

$$i'' = \begin{cases} i+1 & i \neq L_{size} \\ 1 & i = L_{size} \end{cases}, \quad j'' = \begin{cases} j+1 & j \neq L_{size} \\ 1 & j = L_{size} \end{cases}$$

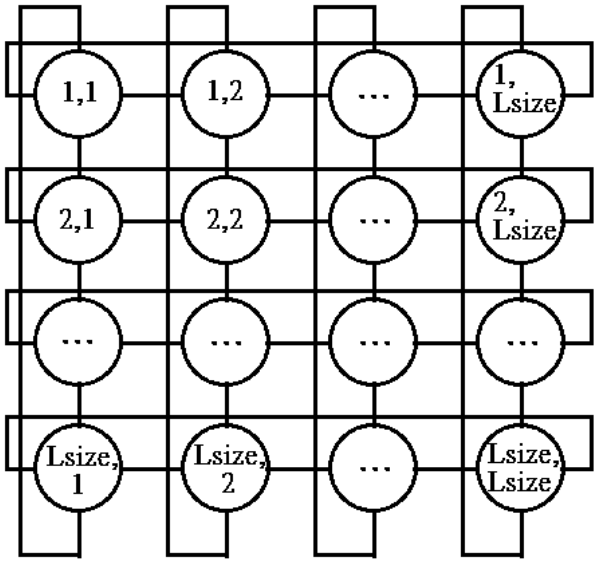

Figura 2.2 – Modelo da grade de agentes.

O MAGA possui os operadores convencionais do GA, cruzamento e mutação, além de uma competição entre os vizinhos e um operador específico chamado de auto-aprendizagem [Zhang *et al.* 2015] que utiliza alguma heurística para refinar a qualidade do agente que representa a melhor solução encontrada em cada iteração. O MAGA inicia gerando os agentes iniciais de forma aleatória. A interação entre os vizinhos consistem em: para cada agente, encontra-se o vizinho com melhor pontuação. Se essa pontuação do melhor vizinho for maior que o agente avaliado, os dois agentes passam por uma operação que gera um novo agente que irá substituir o agente avaliado. No trabalho citado, essa operação é definida em duas estratégias. O operador de cruzamento no MAGA segue a mesma interação entre os vizinhos. O cruzamento é definido por uma equação e esse novo agente tem uma probabilidade de cruzamento (Pc) de substituir o agente escolhido.

Em relação ao operador de mutação, para cada agente, dado a probabilidade de mutação (Pm), é escolhido o agente que será mutado. Caso ele não seja o agente que possui melhor pontuação na geração, ele sofre mutação. Se a pontuação do agente mutado for maior que quando ele não tinha sido mutado, então o agente mantém a mutação. Se a mutação gerar um agente com pontuação menor, então a uma probabilidade Po de manter a mutação. Por fim, é feita a operação de auto-aprendizado no agente com melhor pontuação dentre os agentes da geração corrente. Esse processo é repetido em cada geração até alcançar o critério de parada definido.

O operador de auto-aprendizado segue parte dos processos do MAGA, dado um agente são gerados seus vizinhos através do operador de mutação. Os vizinhos do agente são distribuídos em uma grade de tamanho igual a sL_{size} *x* sL_{size} onde sL_{size} é inteiro e normalmente menor que L_{size}. Após a formação da grade de agentes, para cada um deles é obtido o vizinho com melhor pontuação. Caso o agente verificado tiver uma pior pontuação é feito a operação de cruzamento. Por fim, é aplicado para cada agente da grade a operação de mutação. Se a pontuação do agente mutado for maior que quando ele não tinha sido mutado, então o agente mantém a mutação a uma probabilidade de *sPm*. Se a mutação gerar uma agente com pontuação menor, então a uma probabilidade *Po* de manter a mutação. Esse processo é repetido até o máximo de interações *sGen*.

Os passos do MAGA são mostrados no pseudocódigo abaixo.

Algorithm 1 MAGA

```
Result: Solução Ótima
Construção da grade multi-agente e inicialização da população  Calcular o valor da métrica BIC
  para cada agente
while critério de parada do
    for Agentes do
        Competição entre os agentes
    end
    for Agentes do
        Cruzamento com probabilidade Pc
    end
    for Agentes do
        Mutação
        if o agente gerado possuir pontuação pior que o agente original then
            if probabilidade Po then
                aceitar o agente gerado
            else
                rejeitar o agente gerado
            end
        else
    end
    Método de auto-aprendizado no melhor agente da geração
end
```

CAPÍTULO

3

METODOLOGIA

3.1 Base de dados

Os dados utilizados nesse trabalho foram obtidos na literatura, e são amplamente utilizados em experimentos de aprendizagem estrutural de redes bayesianas (RBs). São eles ASIA, CHILD e INSURANCE.

O ASIA, representa um problema de diagnóstico de câncer de pulmão [Lauritzen e Spiegelhalter 1988]. A dispneia (falta de ar) pode ser causado por tuberculose, câncer de pulmão ou bronquite, como também outros motivos. O tabagismo é conhecido por aumentar o risco para câncer de pulmão e bronquite, enquanto uma visita a Asia pode aumentar as chances de contrair a tuberculose. Um único exame de radiografia do tórax não é suficiente para diagnosticar o câncer de pulmão e nem a tuberculose, assim como a dispneia também não é um diagnóstico suficiente. O problema ASIA possui 8 variáveis com 16 estados nos totais possíveis. A rede alvo possui 8 arcos e é mostrada na Figura 3.1.

O CHILD é referente a um problema de diagnóstico de doenças cardíacas em recém-nascidos [Spiegelhalter *et al.* 1993]. Esse problema faz parte de um estudo de crianças doentes em Londres da *Great Ormond Street Hospital*. O objetivo da rede CHILD é fornecer um mecanismo que, através da experiência clínica oriunda dos dados disponíveis, possa ajudar no auxílio do diagnóstico das doenças cardíacas congênitas que afetam os bebês e levam à doença *blue baby*. Essa doença é causada por condições que afetam o trasporte de sangue, deixando a pele do bebê azul. O CHILD possui 20 variáveis, com 230 possíveis estados. Na Figura 3.2 temos a rede alvo do CHILD. Podemos observar que o nó *Disease* possui seis possíveis valores, portanto, o CHILD estuda seis possíveis doenças.

O problema INSURANCE e relacionado com a estimativa dos riscos de um seguro de automóveis. A rede possui 27 variáveis, sendo apenas 15 variáveis observáveis e mais de 1400 estados. Três das variáveis são observações de "saída". A Figura 3.3, mostra todos os arcos e nós

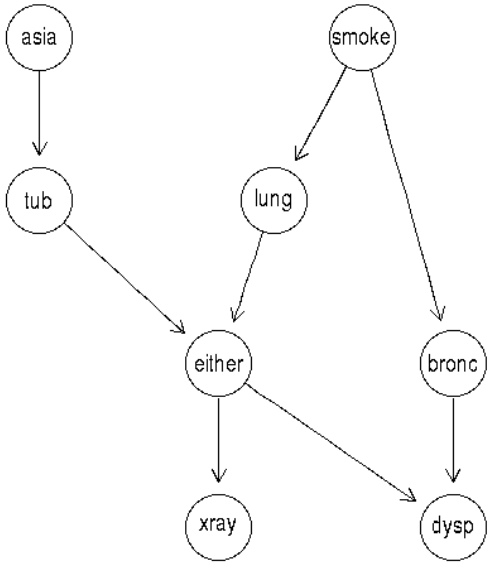

Figura 3.1 – rede alvo do problema ASIA. Adaptado de [Scutari 2020].

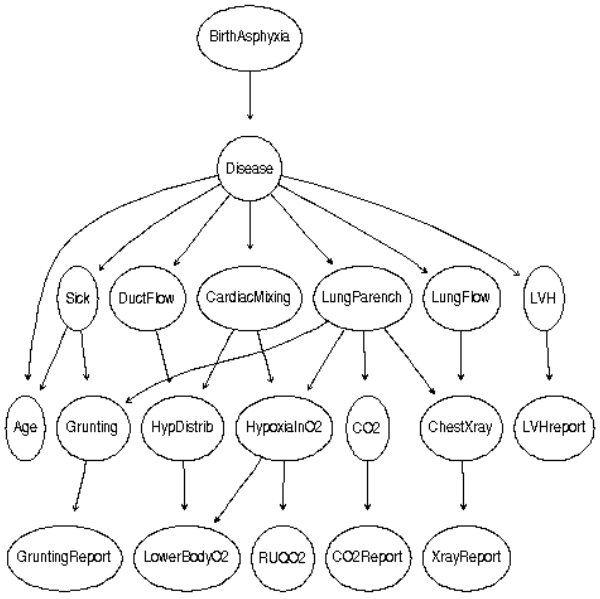

Figura 3.2 – rede alvo do Problema CHILD. Adaptado de [Scutari 2020].

da rede alvo. Dentre as variáveis desse problema observamos que o valor do seguro depende de questões sociais, como o nó SocioEcon que depende diretamente se a pessoa foi uma boa aluna (nó GoodStudent) ou se ela possui outro carro (nó OtherCar). O ano do carro (nó VehicleYear) tem relação direta com as variáveis como *Airbag* (nó Airbag) e valor do carro (nó CarValue), dentre outros fatores mostrados na Rede.

Através do trabalho [Scutari, Graafland e Gutiérrez 2019], foram escolhidos os valores para os tamanhos das amostras nos experimentos. Como apresentado nessa abordagem, o tamanho das amostras são escolhidos através da razão entre n sobre o número de parâmetros do problema, $n/|\Theta|$, sendo n tamanho dos dados. Essa abordagem permite uma comparação

Figura 3.3 – rede alvo do Problema INSURANCE. Adaptado de [Scutari 2020].

significativa, pois, o tamanho das RBs influência na complexidade. Por exemplo, $n = 1000$ isso é 1000 observações, esse valor pode ser grande para um problema com $\Theta = 10$, mas pequeno para $\Theta = 10000$. Usando uma constante para $n/|\Theta|$, podemos garantir que o tamanho escolhido para que a amostra cresça de forma consistente com a complexidade do problema. Os valores de $n/|\Theta|$ escolhidos foram 1, 5, 20, 50 e 100. Como desejamos valores grandes, para o problema ASIA utilizamos apenas $n/|\Theta|$ igual a 20, 50 e 100. Portanto, para o problema ASIA temos os tamanhos de amostras iguais a 400, 1000 e 2000. Para o problema CHILD foram 500, 2500, 10000, 25000 e 50000. E o problema INSURANCE foram 1000, 5000, 20000, 50000 e 100000. Todos os valores foram aproximados. Após testes, observamos que é melhor aproximar o número de parâmetro do CHILD para 500. Como apresentado na Revisão bibliográfica, os valores utilizados na literatura são compatíveis com os utilizados nesse trabalho. A Tabela 3.1 apresenta a quantidade de nós, arcos e de parâmetros para cada um dos problemas.

Tabela 3.1 – Os valores dos problemas *benchmark*

Problema	N° de Nós	N° de arcos	N° de Parâmetros
ASIA	8	8	18
CHILD	20	25	230
INSURANCE	27	52	984

3.2 MAGA

Essa Seção irá discutir a adaptação do MAGA descrito no capítulo anterior para o aprendizado estrutural de RB. O algoritmo inicia construindo a vizinhança entre os agentes.

Cada Agente é representado por uma rede, os agentes iniciais são formados por redes geradas aleatoriamente. Em seguida é iniciado a primeira geração.

Como nos trabalhos [Wang e Liu 2017, Peng *et al.* 2019], a adaptação do MAGA desse trabalho não implementou a competição entre os agentes, pois, esse operador aumentou o tempo computacional sem trazer um melhor desempenho do algoritmo. Primeiro todos os agentes são submetidos ao operador de cruzamento, os agentes selecionados para esse operador são escolhidos como no operador de cruzamento descrito na seção 2.3. O cruzamento utilizado no algoritmo funciona da seguinte forma: dados dois agentes selecionados, são mantidos os arcos que são comuns e que possuam a mesma direção nos dois agentes para o agente filho, e para os arcos diferentes são escolhidos ou do primeiro, ou do segundo agente com uma probabilidade de 50% para o novo agente gerado. Por exemplo, na Figura 3.4 (A) e (B) são dois grafos que foram submetidos ao cruzamento e resultou no grafo (C). Observamos que o arco entre os nós A e B, e o arco entre os nós B e D são comuns nos grafos (A) e (B). E pelo cruzamento foi escolhido o arco dos nós B e C do grafo (A) e o arco entre os nós A e D do grafo (B).

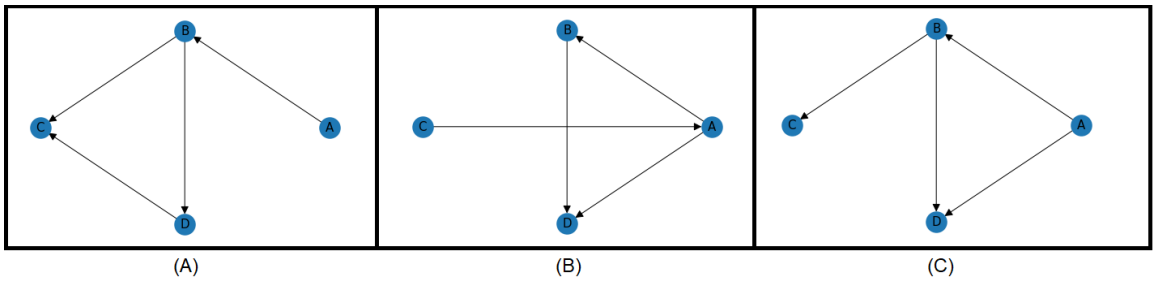

Figura 3.4 – Em (A) e (B) é mostrado duas redes,após o cruzamento dessas redes obtivemos a rede (C).

Logo após o operador de cruzamento, os agentes passam pela fase da mutação. O operador de mutação implementado nesse trabalho possui algumas diferenças do descrito na seção 2.3. A escolha do agente que será mutado é definida primeiro pelo cálculo do número total de mutações na geração utilizando o Pm. Então é selecionado um agente aleatório e aplicado a mutação. A mutação seleciona dois nós aleatórios então é verificado se há um arco entre os nós. Caso não exista um arco entre os nós, será adicionado um arco e sua direção é escolhida a uma probabilidade de 50%. Caso exista um arco entre os nós, é escolhido a uma probabilidade de 50% se essa será removida ou se será invertida a sua direção. Depois é selecionado outro agente aleatório e executado novamente a mutação até que o número de agentes selecionados seja igual ao número total de mutações na geração. Por exemplo, na Figura 3.5 é mostrado às duas possibilidades de uma mutação de dois nós. Os nós selecionados foram A e C, em (B) o arco foi invertido e em (C) o arco foi removido.

Tanto no cruzamento quanto na mutação se a adição ou inversão de um arco tornar a rede com circuito, busca-se de forma aleatória um dos arcos desse circuito, que não seja aquele alterado pelos operadores, e é invertido sua direção ou removida. Essa operação se repete até que não haja mais circuitos. Essa operação, chamada *repair operator*, foi descrita no

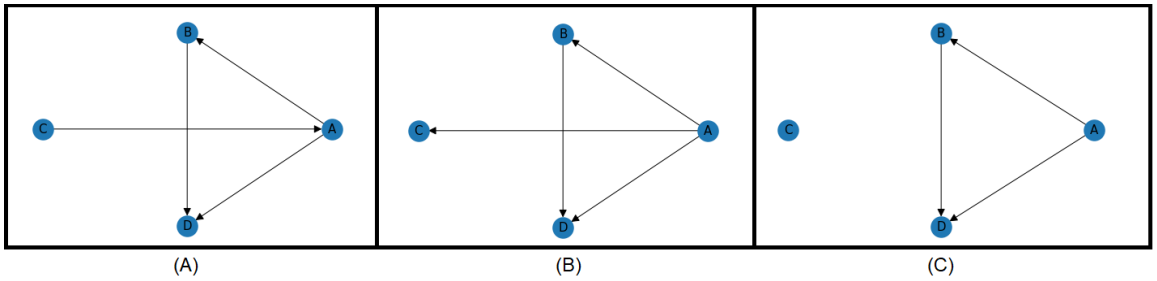

Figura 3.5 – A rede (A) foi utilizada para a mutação e podemos obter as redes (B) e (C) dessa mutação.

trabalho [Larrañaga *et al.* 1996] que mostra um melhor desempenho em relação ao algoritmo que não usou essa operação. Foi feito um teste para verificar a complexidade desse *repair operator*. Para todas as instâncias escolhidas nesse trabalho, foi testado 1000 redes aleatórias que violam a restrição DAG. A Figura 3.6 mostra que o tempo de cada um das instâncias. Observamos que conforme aumenta a complexidade das redes o tempo do operador possui um aumento, mas conforme o aumento do volume de dados não há uma influência direta no tempo. O maior tempo foi 1.2 segundos, Para esses problemas utilizados nesse trabalho, a complexidade desse operador é baixa.

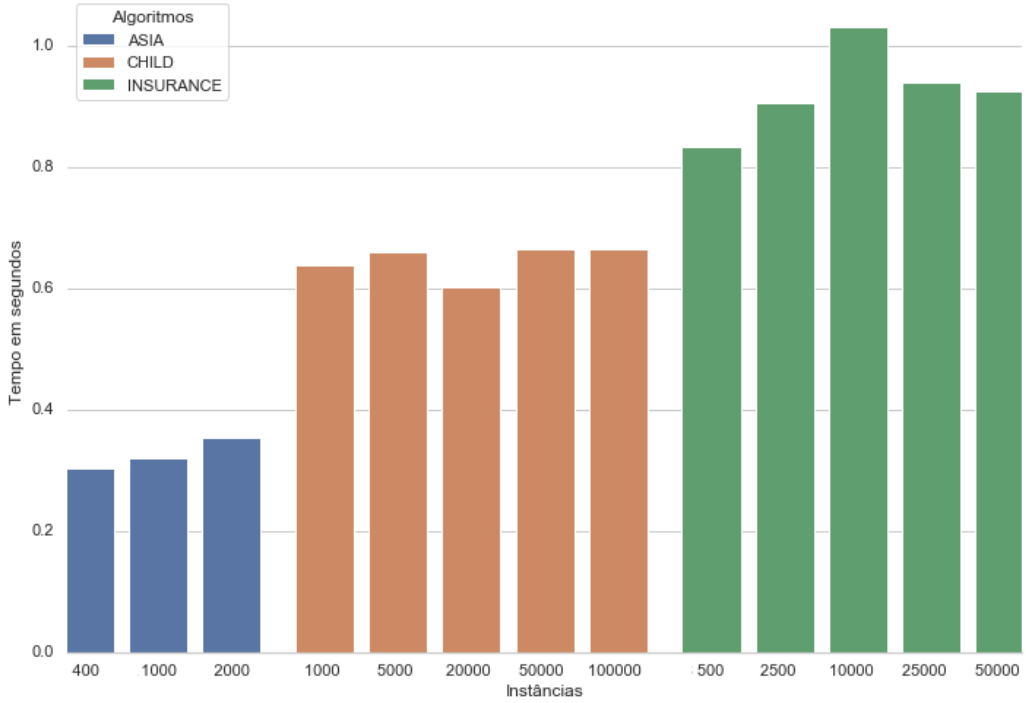

Figura 3.6 – Tempo em segundos do teste do *repair operator*.

Por fim, é aplicado o auto-aprendizado no melhor agente da geração. Nesse trabalho foi implementado dois algoritmos, que são adaptações do MAGA, a diferença entre eles e o operador de auto-aprendizado. O primeiro algoritmo chamado de MAGA-Tabu utilizou a heurística de busca tabu como auto-aprendizado, apresentou bons resultados com o MAGA [Peng *et al.* 2019] e também se mostrou uma heurística com bons resultados no aprendizado estrutural da

RBs [Scutari, Graafland e Gutiérrez 2019]. O segundo algoritmo chamado MAGA-Original, utilizou o auto-aprendizado original do MAGA.

Na busca tabu, a principal ideia é interativamente avaliar a vizinhança da solução. Para não ficar preso em mínimos locais, a utilização da lista tabu permite movimentações quando se explora soluções vizinhas com a pior pontuação no espaço de busca. Inicialmente é definido o melhor vizinho do agente atual que não esteja na lista tabu. Então é verificado se o melhor vizinho é melhor que o agente atual, caso positivo o agente atual é substituído pelo melhor vizinho, e Em seguida se atualiza a lista tabu de tamanho definido igual a 100. Por fim, se verifica se o agente atual é o melhor agente visto, caso positivo se atualiza o melhor agente visto. Esse processo segue até que não tenha um vizinho melhor que o melhor agente em t_{max} gerações e então o melhor agente visto é saída do algoritmo. O auto-aprendizado original foi o mesmo descrito no Capítulo da revisão bibliográfica, com os operadores genéticos descritos nessa Seção.

Após o operador de auto-aprendizado, termina a geração e começa uma nova caso não satisfeito o critério de parada. O critério de parada utilizado foi o número máximo de BIC ou se obter um agente com pontuação igual, ou menor que a rede alvo.

O MAGA foi executado com probabilidade de cruzamento (Pc) de $0,9$, probabilidade de mutação (Pm) de $0,01$, probabilidade Po de $0,05$, número de vizinhos da busca tabu em 25, o número máximo de gerações para a busca tabu em 10 e o L_{size} em 10, dessa forma o MAGA possui 100 agentes. As variáveis *sL*, *sPm* e *sGen* do auto-aprendizado original foram definidas iguais a 3, 0.01 e 10 respectivamente. Esses parâmetros foram ajustados através de testes preliminares dos algoritmos e foram aplicados nas duas versões do MAGA com a busca tabu e com o auto-aprendizado original.

3.3 Outros Algoritmos

Os algoritmos GA [Larranaga *et al.* 1996] , PSO-BNC [Gheisari e Meybodi 2016] ,SAGA [Adabor, Acquaah-Mensah e Oduro 2015], ABC [Ji, Wei e Liu 2013] e GES [Chickering 2002] foram escolhidos para a comparação com o MAGA nesse trabalho. A implementação dos algoritmos seguiu os trabalhos citados.

Os algoritmos PSO-BNC e GA foram testados com o cruzamento e mutação iguais ao MAGA descrito na Seção anterior, esses operadores obtiveram melhores resultados que os originalmente propostos nos artigos. Nos algoritmos PSO-BNC, GA e ABC foi implementado o operador *repair operator*. Os parâmetros utilizados foram conformes as referências para cada algoritmo, o tamanho da população no GA foi definida como 100 indivíduos. Foram utilizados tanto os mesmos parâmetros da referência, quanto a mesma forma de pesquisa de vizinhança da referência do algoritmo SAGA.

3.4 Experimentos

Nessa Seção serão discutidas as métricas utilizadas no trabalho, o critério de parada dos algoritmos e o ambiente computacional utilizado. Para esse trabalho, cada algoritmo foi executado 20 vezes para cada uma das instâncias.

As métricas utilizadas foram a BIC, a *Structure learning factor* (SLF), a *topology learning factor* (TLF), acurácia e a convergência para a rede alvo. A métrica BIC, descrita na Revisão bibliográfica, foi utilizada como função objetivo durante o aprendizado estrutural com os diferentes algoritmos.

O problema de otimização apresentado na abordagem *Search and Score* é maximizar o valor da métrica. Nesse trabalho será utilizado a métrica BIC que possui um valor negativo e diferente de zero. Portanto, utilizamos o módulo da métrica BIC e tornamos em um problema de minimização. O modelo utilizado é mostrado na Eq.(3.1).

$$\begin{aligned} Minimizar: &\quad |BIC(R \mid D)| \\ Sujeito\ a: &\quad R \in DAG \end{aligned} \tag{3.1}$$

A única restrição para as redes testadas (R), é que elas devem ser DAG.

As métricas SLF e TLF são definidas pelas Equações 3.2 e 3.3 respectivamente.

$$SLF = \frac{TC}{TE} \tag{3.2}$$

$$TLF = \frac{TC + IE}{TE} \tag{3.3}$$

Onde TC é o número de arcos que estão corretos, TE é o número de arcos na rede correta e IE é o número de arcos corretos com direção incorreta.

A acurácia representa a proximidade entre duas redes. Para isso ela é definida sendo a soma do verdadeiro positivo (isso é se há arco onde deveria ter) e o verdadeiro negativo (não possui arco onde não deveria ter) sobre todas as outras possibilidades, falso negativo, falso positivo, verdadeiro positivo e verdadeiro negativo. A Eq. (3.4) demostra a acurácia.

$$acurácia = \frac{verdadeiro\ positivo + verdadeiro\ negativo}{verdadeiro\ positivo + verdadeiro\ negativo + falso\ positivo + falso\ negativo} \tag{3.4}$$

Por fim, a taxa de convergência é quantas execuções resultaram em redes com um módulo da BIC igual ou menor que o módulo da BIC da rede alvo definida pela literatura, antes de alcançar o critério de parada.

O critério de parada para os algoritmos foram o número de avaliações da função BIC ou pela convergência para a rede alvo conhecido para cada instância. Os valores do número de avaliações em cada instância foram definidos através de testes. Para o problema ASIA foram definidos os valores 12000, 25000 e 50000 para as instâncias 400, 1000 e 2000, respectivamente. O problema CHILD foram 20000, 60000, 100000, 120000 e 140000 para as instâncias 500, 2500, 10000, 25000 e 50000 respectivamente. Para o INSURANCE os valores foram 60000, 100000, 140000, 160000 e 180000 para as instâncias 1000, 5000, 20000, 50000 e 100000.

Consultamos a literatura para obter um valor máximo de avaliações da métrica BIC como critério de parada. Encontramos, como exemplo, dois valores distintos nos trabalhos de [Larrañaga *et al.* 1996] e [Martins *et al.* 2021]. No primeiro, o resultado foi de 10000; e no segundo foi de 200000. Além de haver uma divergência grande entre os resultados, não encontramos nos textos a explicação de como esses resultados foram obtidos. Por essa razão, optamos por efetuar os testes para obter esses marcos. Os testes foram efetuados obedecendo aos seguintes passos: utilizamos os algoritmos MAGA-Original e GA; avaliamos os valores de convergência obtidos não limitando o valor máximo de avaliações e empregamos o arredondamento. Inicialmente, foram feitos os testes com o problema CHILD. Utilizando a instância 2500, o maior valor de convergência do MAGA-Original foi 56643 e do GA foi 58886. Portanto, o valor máximo escolhido para essa instância foi 60000. Na instância 10000, o MAGA-Original obteve 64742, o GA obteve 73251 e o valor máximo de avaliações escolhido foi 100000. Na instância 25000, o MAGA-Original obteve 111711 e o GA obteve 113694. Por isso, o valor de 120000 foi escolhido. Foram feitos testes similares com os problemas INSURANCE e ASIA. Depois desses testes, foi observado um padrão: quando o volume de dados dobrava era adicionado cerca de 20000 ao valor de número máximo de avaliações da BIC. A Tabela 3.2 foi construída a partir desses resultados. Os valores utilizados nesse trabalho foram obtidos dessa Tabela.

Tabela 3.2 – O número de avaliações da métrica BIC para várias instâncias do problema INSURANCE, CHILD e ASIA.

INSURANCE		CHILD		ASIA	
Dados	Número máx de BIC	Dados	Número máx de BIC	Dados	Número máx de BIC
100000	180000	50000	140000	2000	50000
50000	160000	25000	120000	1000	25000
20000	140000	10000	100000	400	12000
10000	120000	5000	80000	-	-
5000	100000	2500	60000	-	-
2000	80000	1000	40000	-	-
1000	60000	500	20000	-	-

O ambiente computacional utilizado para o desenvolvimento e a execução dos testes foi uma máquina com SO *Windows* 10, uma *CPU* i5-3570k 3.4GHz e com uma memória *RAM* de 8GB. Os algoritmos foram implementados na linguagem de programação *python*, e utilizaram as bibliotecas *networkx* [Hagberg, Schult e Swart 2008] para manipulação dos grafos, *numpy* [Harris, Millman e al. 2020] para manipulação de vetores, a *rpy2* para utilizar a função

de cálculo do BIC da biblioteca *bnlearn* [Scutari 2010] do R. Os dados utilizados foram gerados utilizando o R e a biblioteca *bnlearn*.

CAPÍTULO

4

RESULTADOS

Nesse Capítulo serão apresentados os resultados da comparação dos algoritmos. O Capítulo será dividido em: observações dos métodos utilizados que irá mostrar os resultados da análise das métricas utilizadas e a seção dos resultados dos algoritmos no aprendizado estrutural de Redes Bayesianas (RB). Os dados da média de todas as execuções dos algoritmos para todas as instâncias testadas estão no Apêndice A.

4.1 Observações das métricas utilizadas

O método de avaliação utilizado nesse trabalho foi a métrica BIC da biblioteca bnlearn. Durante os testes, foi observado que a pontuação dada para as redes merece algumas observações importantes. Na Figura 4.1 temos duas redes diferentes do problema ASIA. A rede (A) representa a rede alvo do problema e a rede (B) representa uma rede parecida com a rede (A) com apenas dois arcos invertidos. O nó tub está apontando para o nó asia e o nó lung está apontando para o nó smoke. Na avaliação BIC dessas duas redes que não são equivalentes, seus valores foram iguais a 2273.24. Portanto qualquer uma das duas poderiam ser consideradas uma rede alvo. Além desse exemplo, temos outros casos em que encontramos redes com pontuações menores que a pontuação dada pela rede alvo, tanto para o problema ASIA quanto nos demais utilizados nesse trabalho. Portanto foi considerado convergência, no critério de parada, as redes que obtiveram módulos de BIC iguais ou inferiores a pontuação da rede alvo da literatura. Esse comportamento foi observado em outras métricas de pontuação durante os testes.

No trabalho [Ji *et al.* 2017] observamos esse comportamento. O autor apresenta os valores da métrica K2 das redes objetivo e nos resultados do aprendizado dos algoritmos testados, os valores obtidos foram menores que os valores de K2 das redes ótimas. Observamos também que dentre as métricas de avaliação, eles utilizam a métrica que verifica a diferença entre a rede alvo e a obtida pelo algoritmo. As redes que obtiveram valores de K2 menores que os valores das redes ótimas, apresentaram erros estruturais. Nesse trabalho, não é apresentado o motivo

para essa diferença.

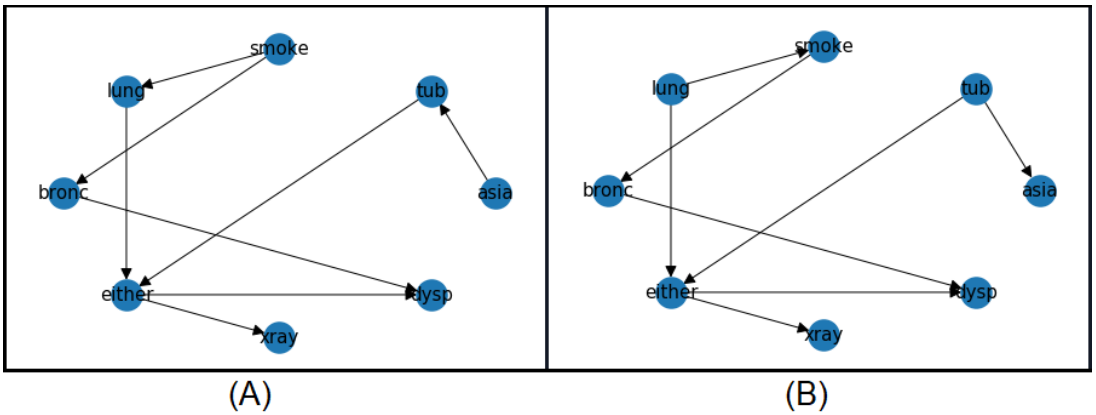

Figura 4.1 – Em (A) é mostrado a rede alvo. Em (B) é mostrado uma rede igual à rede alvo mas com dois arcos invertidos.

Alguns resultados apresentaram taxas de convergência de 100% sendo que o valor de SLF e TLF foram baixos. Para entender esses resultados, temos a Figura 4.2. Considerando o grafo (A) sendo a rede alvo, calculando o valor SLF, TLF e acurácia dos grafos (B) e (C). Obtivemos para a rede (B) o SLF igual a 0, o TLF igual a 1 e a acurácia igual a 0.8333 enquanto para a rede (C) o SLF foi 1, o TLF igual a 1 e a acurácia igual a 0.583. O TFL dos dois grafos deram iguais pois os dois apresentam o único arco da rede (A). Mas o SLF foi igual a 0 para (B) pois a direção do arco está invertido, e como o grafo (C) obteve a mesma direção do grafo (A) então o seu valor de SLF foi 1. Através das imagens das redes, podemos observar que a rede (B) aparenta ter maior semelhança com a rede (A) do que a rede (C). Portanto a acurácia dos grafos é importante para a análise dessas redes. Nos resultados a acurácia dos experimentos em alguns casos de taxas de convergência igual a 100% não apresentou acurácia igual a 1. Isso ocorreu provavelmente devido a observação dos valores da métrica BIC apresentada anteriormente.

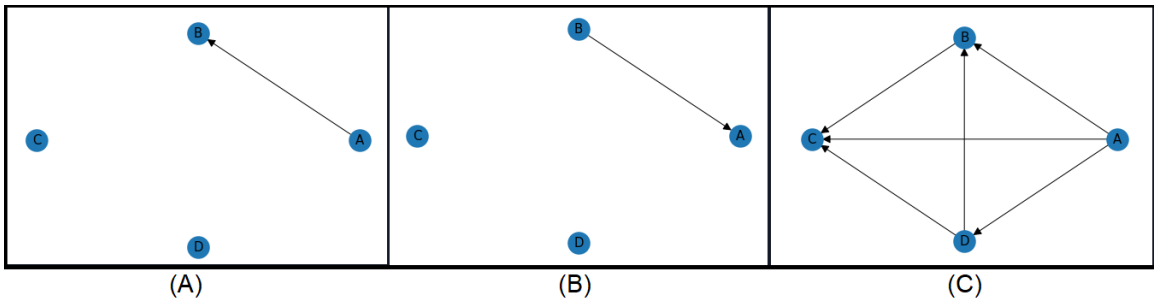

Figura 4.2 – Redes utilizadas para o estudo do SLF, TLF e da acurácia.

4.2 Resultados dos algoritmos

Nessa seção, primeiro vamos discutir como foi feito o teste estatístico. Depois foi debatido os resultados dos algoritmos no aprendizado estrutural de RB. Por fim, é mostrado os resultados dos testes estatísticos.

Para definir se o MAGA apresentou um resultado superior ou se há alguma diferença entre os MAGAs, foram feitos testes estatísticos. Os algoritmos comparados, foram os que apresentaram melhores resultados MAGA-Original, MAGA-Tabu e ABC. Foi utilizado um modelo linear com blocos completamente randomizado para estudar se há diferenças entre os algoritmos. O teste verifica se há alguma média que difere das demais média dos algoritmos. Para o problema desse trabalho, a hipótese nula é definida como todas as médias dos algoritmos são iguais, portanto a hipótese nula indica que não há evidências suficientes para provar que as médias são diferentes. Para o teste estatístico foi utilizado a linguagem R.

Como temos instâncias de problemas diferentes nesse trabalho, focamos na comparação dos algoritmos e não no fator das instâncias. Portanto, foi usado a blocagem para que as instâncias não influenciassem na comparação dos algoritmos.

Na Figura 4.3 é mostrado o tempo computacional dos algoritmos nas três instâncias do problema ASIA. Podemos observar que o PSO-BNC possui elevado tempo computacional nas instâncias 1000 e 2000. O GES possui o menor tempo em todas as instâncias, isso ocorre pois o GES é um algoritmo de busca local. Os MAGAs possuem um menor tempo computacional comparado com os outros algoritmos, exceto o GES. O MAGA-Original possui um tempo maior que o MAGA-Tabu nas duas primeiras instâncias, mas na terceira instância ele possui um tempo menor que o MAGA-Tabu. Podemos observar que quanto maior é o volume de dados, maior é o tempo da maioria dos algoritmos, apenas os MAGAs não obtiveram um aumento considerável no tempo computacional.

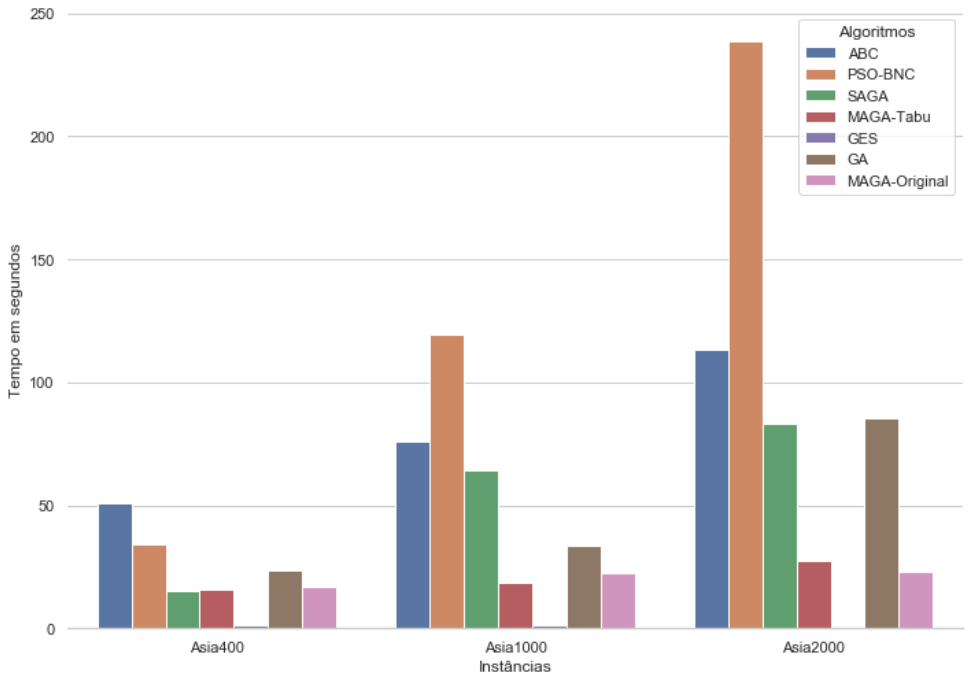

Figura 4.3 – Comparação do tempo computacional dos algoritmos em todas as instâncias do problema ASIA.

Na Figura 4.4 observamos que o GES convergiu para todas as execuções em todas as instâncias. Os MAGAs obtiveram melhores resultados comparados com os outros algoritmos,

	ABC	PSO-BNC	SAGA	MAGA-Tabu	GES	GA	MAGA-Original
ASIA 400	921.0302	919.0187	918.9213	918.9648	918.8693	920.4419	919.5954
ASIA 1000	2275.3022	2274.4914	2277.2980	2273.5730	2273.2422	2274.8368	2273.6019
ASIA 2000	4485.2962	4483.6578	4483.5827	4482.4134	4481.7679	4486.0891	4481.7679

Tabela 4.1 – Valor da BIC dos algoritmos nas instâncias do problema ASIA.

exceto o GES. O MAGA-Original convergiu melhor em instâncias com maior volume de dados que o MAGA-Tabu. Os algoritmos ABC e GA são os que possuíram menores taxas de convergência. O SAGA possui melhores taxas de convergência na primeira e ultima instância. Por fim, o PSO-BNC apresentou uma maior taxa de convergência na primeira instância e manteve aproximadamente constantes nas demais instâncias.

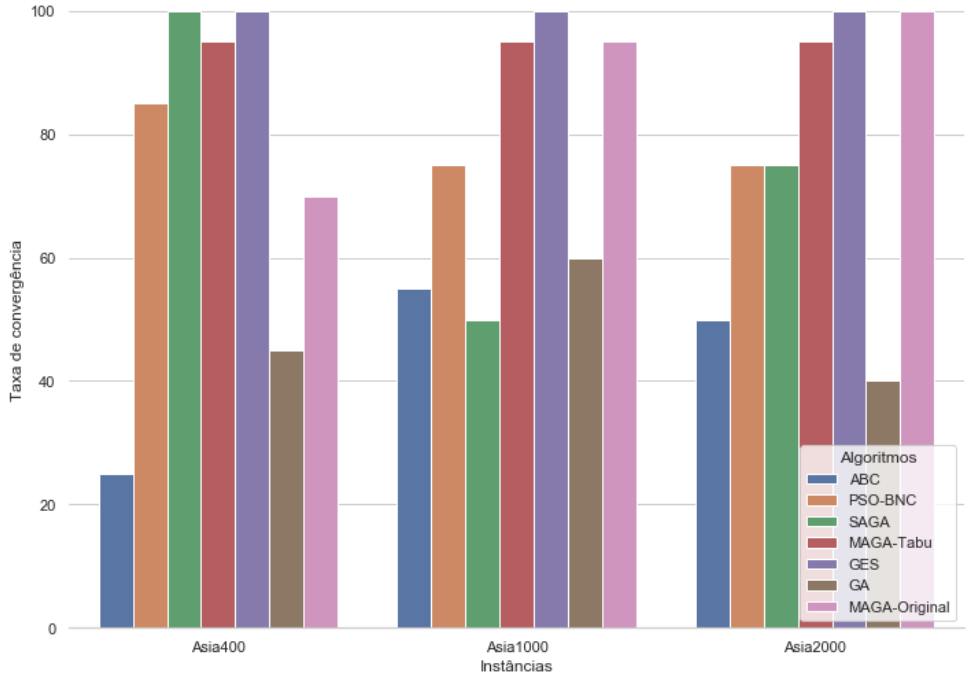

Figura 4.4 – Comparação da taxa de convergência para a rede alvo dos algoritmos em todas as instâncias do problema ASIA.

Na Tabela 4.1 são mostrados os valores de BIC dos algoritmos no problema ASIA. Na instância ASIA 400, observamos que o GES,o SAGA e o MAGA-Tabu obtiveram um valor médio de BIC menor que o valor de referência (918.9849). Na instância 1000 o GES obteve em todas as execuções o valor exato da referência, igual a 2273.242. Nessa instância e na última, observamos que os MAGAs obtiveram os menores valores de BIC juntamento com o GES. Na última instância, apenas os algoritmos ABC e GA obtiveram valores maiores que o da referência que é igual a 4484.5930. NA Tabela 4.1, temos que na instância 400 a maior diferença entre a pontuação obtida e a desejada foi aproximadamente de 3, na instância 1000 a maior diferença foi de aproximadamente 2 e na instância 2000 foi entorno de 2. Esses valores foram obtidos pelo GA ou pelo ABC, observamos que apesar da baixa convergência desses algoritmos, eles obtiveram valores próximo do valor de referência.

Nas Figuras 4.5 es 4.6 temos as métricas SLF e TLF para as instâncias do ASIA. Podemos observar que o algoritmo que possui maior proximidade das arestas corretas da rede ótima da literatura foi o GES, possuindo os maiores valores de SLF. Assim como na convergência os MAGAs obtiveram melhores resultados que os demais algoritmos, principalmente nas instâncias com maior volume de dados. Os algoritmos que apresentaram melhores valores de SLF foram os algoritmos que possuíram maior taxa de convergência, porém, um valor alto de TLF não necessariamente reflete uma qualidade do resultado do aprendizado estrutural. Por exemplo, o GA alcançou altos valores de TLF, principalmente na instância 1000, e mesmo assim não obteve taxas altas de convergência. A acurácia assim como a métrica SLF possui esse comportamento em relação à taxa de convergência. O MAGA-Original possui o TLF próximo ao MAGA-Tabu, nas duas ultimas instâncias. O PSO-BNC possui os menores valores de TLF. A maioria dos algoritmos obteve melhores valores de SLF e TLF na instância 1000. Nos testes observamos que nessa instância os algoritmos não obtiveram variações dos valores de BIC. Nas outras duas instâncias, obtivemos essa variação e observamos que os valores ligeiramente menores que na instância 1000.

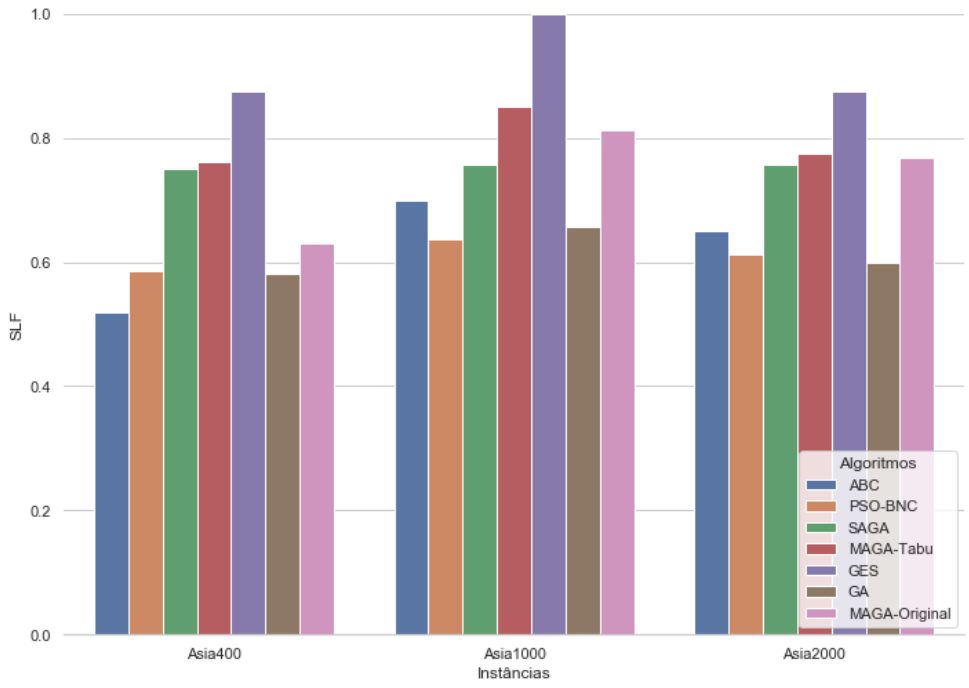

Figura 4.5 – Comparação da métrica SLF dos algoritmos em todas as instâncias do problema ASIA.

NA Figura 4.7 é apresentado a acurácia dos algoritmos no problema ASIA. Assim como nas outras métricas, o GES e os MAGAs obtiveram os melhores valores de acurácia comparado com os demais algoritmos. Os algoritmos GA e PSO-BNC obtiveram os menores resultados. Na instâncias 1000 do ASIA foi o menor valor de acurácia obtido em todos os problemas, o valor igual a 0.88 alcançado pelo PSO-BNC. O MAGA-original obteve um valor menor que o MAGA-tabu na primeira e na segunda instância e um valor maior na última instância.

No problema ASIA, o algoritmo que apresentou melhor desempenho foi o GES. Esse

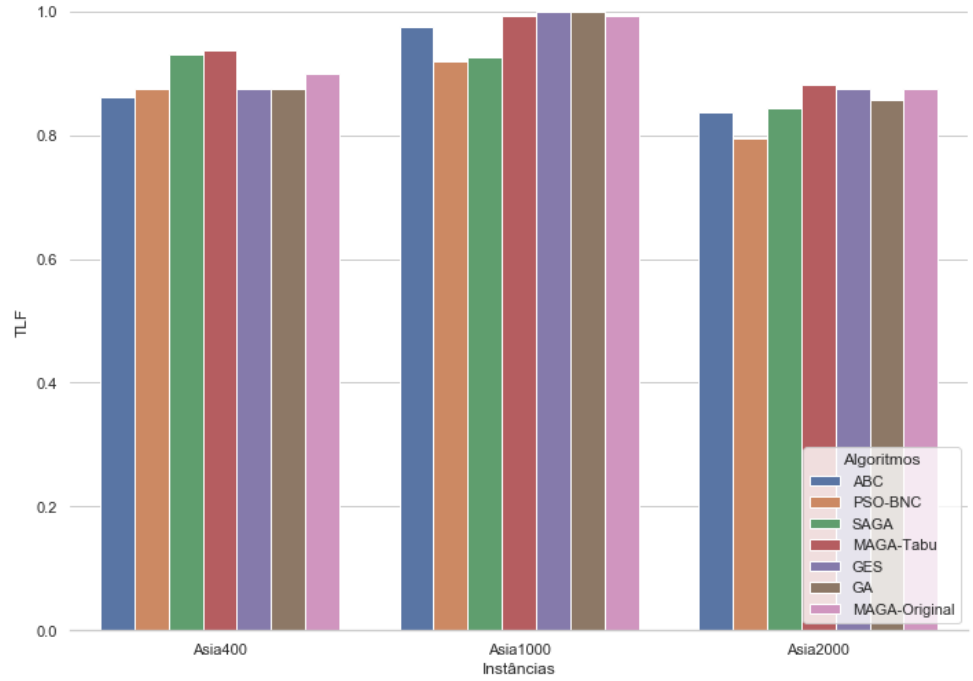

Figura 4.6 – Comparação da métrica TLF dos algoritmos em todas as instâncias do problema ASIA.

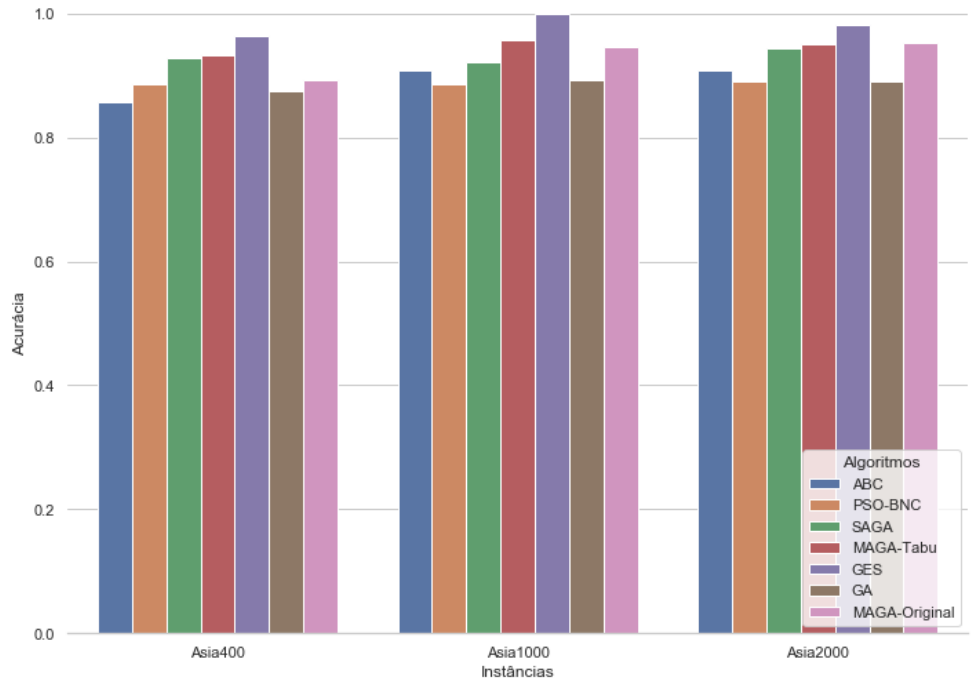

Figura 4.7 – Comparação da acurácia dos algoritmos em todas as instâncias do problema ASIA. (A) é a comparação da acurácia da média das 20 execuções. (B) é comparação da acurácia das execuções que convergiram.

resultado mostra que algoritmos de busca local apresentam melhores resultados em problemas com baixa complexidade. Os MAGAs apresentaram o segundo melhor desempenho, pois, possuíram menores tempos computacionais com melhores taxas de convergência, SLF e TLF nas instâncias 1000 e 2000. Com o maior tempo computacional, o PSO-BNC apresentou um resultado mediano comparado com os demais algoritmos, com a menor acurácia.

Na Figura 4.8 obtemos o tempo computacional dos algoritmos no problema CHILD, o tempo está em escala Log devido a diferença de tempo na ultima instância. O PSO-BNC possui os maiores tempos computacionais, obtendo uma alta diferença dos outros algoritmos conforme o aumento do volume de dados e se mostrando custoso computacionalmente em instâncias com grande volume de dados. Apenas na instância 500 o SAGA apresentou o maior tempo. O menor tempo foi o GES seguido dos MAGAs, em todas as instâncias. O tempo dos MAGAs foram próximos nas instâncias 2500, 10000 e 50000. Nas demais instâncias o MAGA-Tabu obteve custos computacionais menores que o MAGA-Original. Conforme o aumento do volume de dados maior é o tempo computacional dos algoritmos.

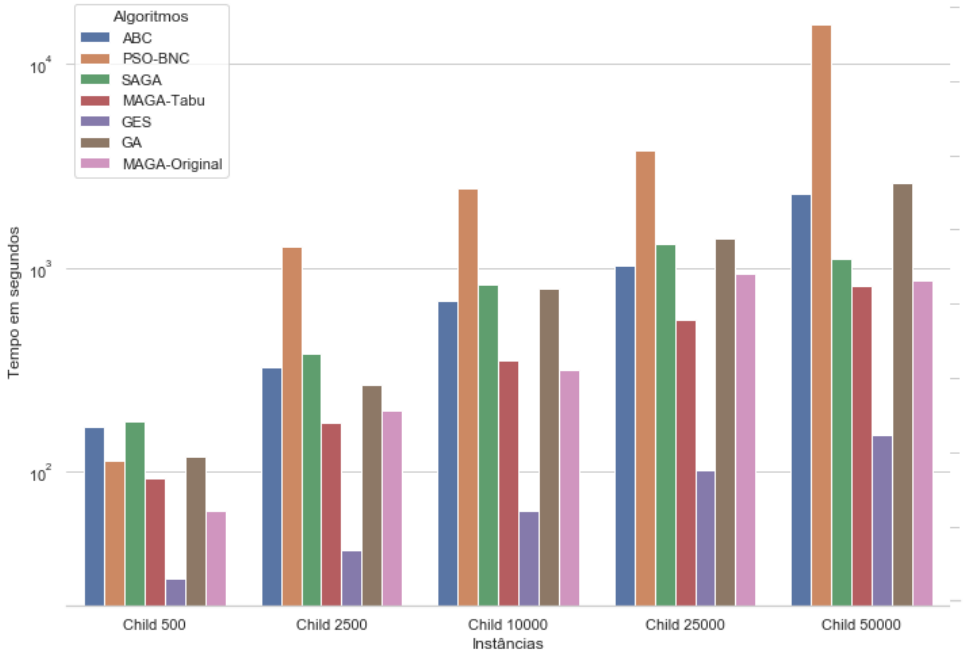

Figura 4.8 – Comparação do tempo computacional dos algoritmos em todas as instâncias do problema CHILD, em escala Log.

A Figura 4.9 apresenta a taxa de convergência dos algoritmos. Os melhores resultados foram os MAGAs. Na primeira instância, GES e MAGA-Original convergiram em todas as execuções. Os MAGAs obtiveram valores de convergência maiores que 70% em todas as instâncias enquanto os outros algoritmos apresentaram taxas de convergência menores que 70% em todas as instâncias exceto o GES na primeira instância. Dentre esses algoritmos o SAGA foi que obteve melhores valores, sendo o maior deles em 70% na última instância. O GES conseguiu convergência apenas na primeira instância. O PSO-BNC não conseguiu convergência apenas na instância ultima instância. O MAGA-Original obteve uma taxa de convergência maior que o MAGA-Tabu em três das cinco instâncias testadas. Sendo na última instância, o MAGA-Original obteve uma taxa de convergência de 100%.

Observamos pela Tabela 4.2 os valores BIC dos algoritmos no problema CHILD. Na instância 500 os MAGA-Original e o GES obtiveram valores de BIC menores que a referência (6674.0388). O MAGA-tabu, que possui uma convergência de 90%, possuiu uma diferença de

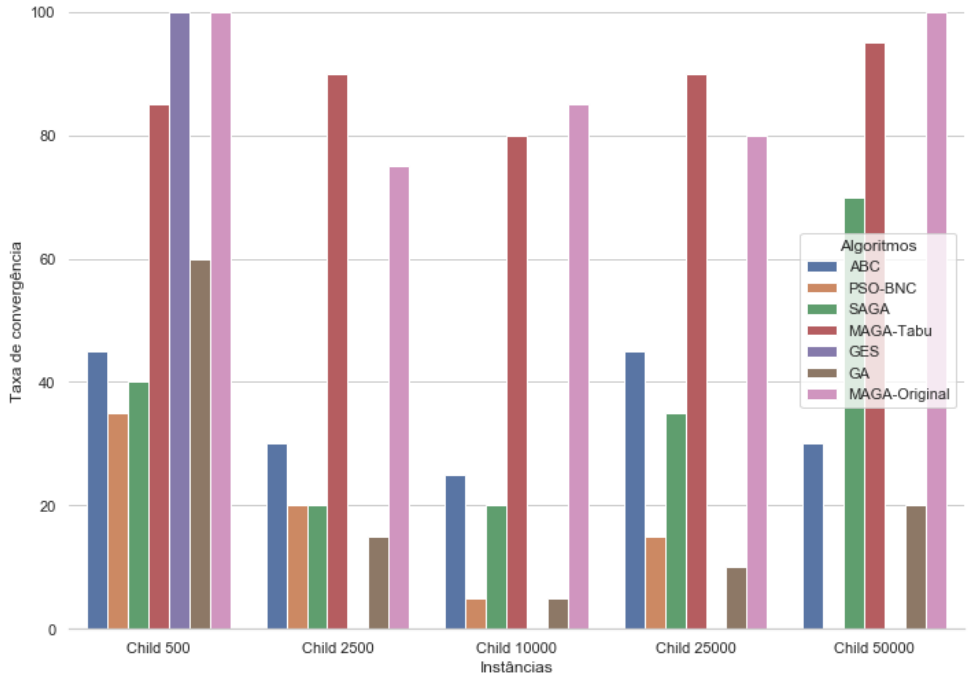

Figura 4.9 – Comparação da taxa de convergência para a rede alvo dos algoritmos em todas as instâncias do problema CHILD.

	ABC	PSO-BNC	SAGA	MAGA-Tabu	GES	GA	MAGA-Original
Child 500	6711.2138	6750.8955	6762.3931	6674.4059	6673.9954	6703.4779	6656.8844
Child 2500	31339.4596	31386.6060	31797.6114	31233.7360	31326.9937	31385.7381	31290.0374
Child 10000	122985.6388	123215.1416	123953.1543	122808.7055	123086.0946	123169.0290	122813.6559
Child 25000	306774.0982	307439.7849	308226.7282	306662.8740	307010.0789	307502.6594	306697.6253
Child 50000	610401.5970	611042.1904	611049.6629	610168.8625	610729.4425	610642.7034	610109.8003

Tabela 4.2 – Valor da BIC dos algoritmos nas instâncias do problema CHILD.

0.4 da referência. O SAGA possui a maior diferença com valor de 88 da referência. Na instância 2500, a diferença entre os algoritmos e o valor da referência ficou entre 8.2 a 572. Na instância 10000 essa variação foi de 29.4444 a 1173.9444. Já nas instâncias 25000 e 50000, foi 27.5062 a 1591.3062 e 0 a 939.8997, respectivamente. Por essas diferenças observamos que a última instância obteve menor diferença com a referência, dentre as 4 ultimas instâncias. O valor de 1591.3062 é 5% do valor de referência e o 1173.9444 é aproximadamente 3.7%, esses valores são dos algoritmos que obtiveram menos de 20% de taxa de convergência. Mostrando que os algoritmos, apesar da baixa convergência, obtiveram resultados próximos ao desejado.

As métricas SLF e TLF são mostradas nas Figura 4.10 e 4.11. Na primeira instância, o melhor algoritmo foi o SAGA seguido do MAGA-Tabu. Nas outras instâncias os MAGAs foram os algoritmos com melhores valores de SLF. Os algoritmos que apresentaram menores SLF foram o GES e o PSO-BNC. Observamos que os algoritmos nesse problema tiveram maiores valores de TLF, que conforme o aumento do volume de dados maior ficou a média dos valores de TLF entre os algoritmos. Isso mostra que a maior dificuldade encontrada pelos algoritmos foi no direcionamento dos arcos. Em relação ao TLF, o PSO-BNC e o SAGA tiveram um crescimento conforme o aumento dos dados. Observamos que os algoritmos que obtiveram maiores valores

de SLF alcançaram também os maiores valores de taxas de convergência, esse fenômeno não ocorreu apenas na primeira instância. Os algoritmos obtiveram altos valores de TLF, mas isso não influenciou na taxa de convergência.

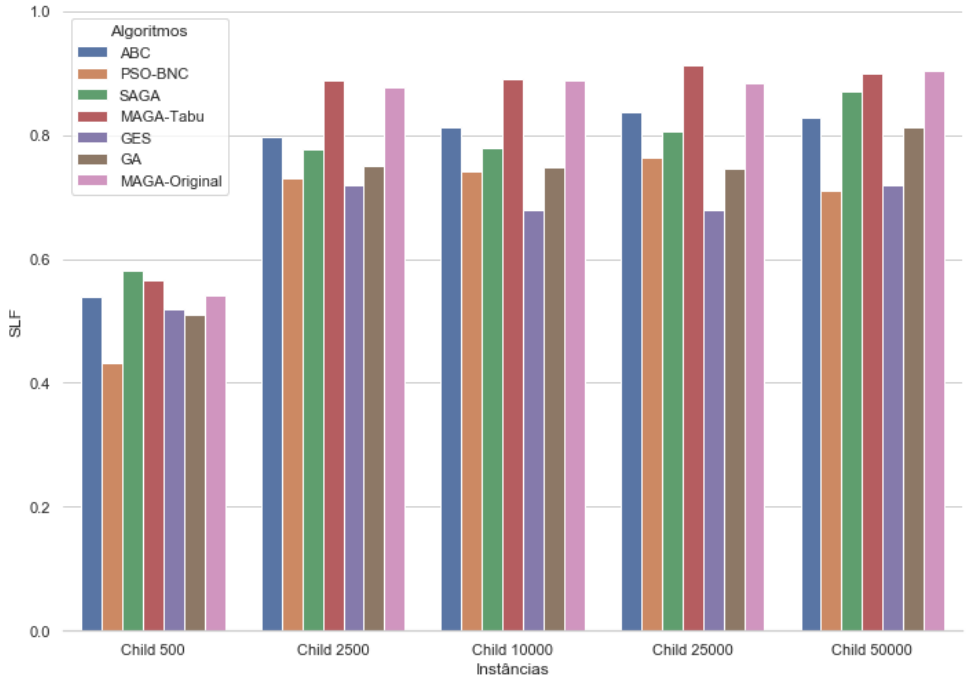

Figura 4.10 – Comparação da métrica SLF dos algoritmos em todas as instâncias do problema CHILD.

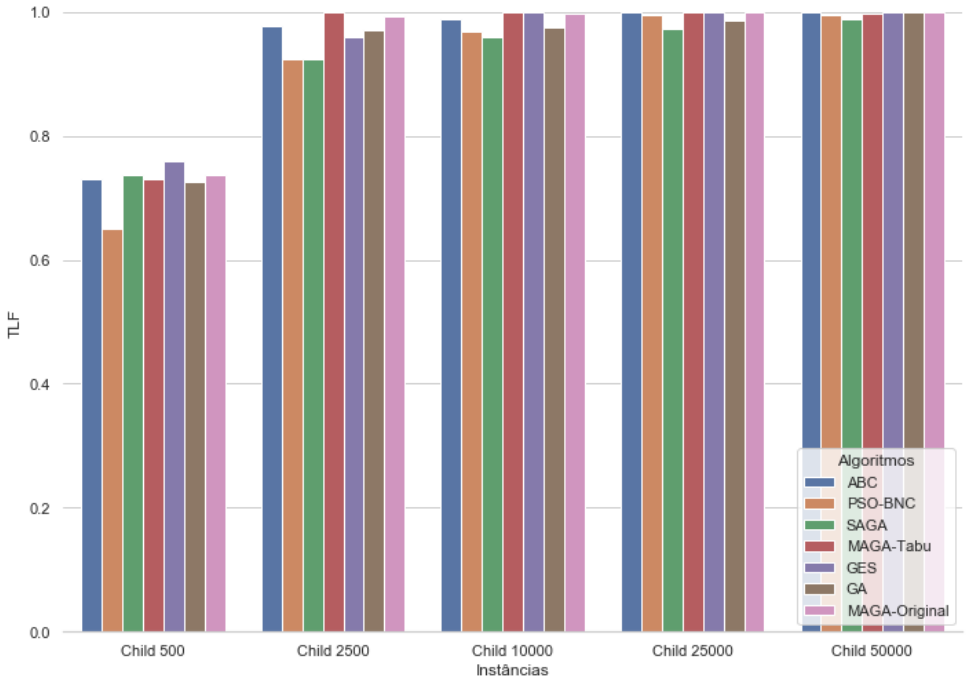

Figura 4.11 – Comparação da métrica TLF dos algoritmos em todas as instâncias do problema CHILD.

Figura 4.12 apresenta a acurácia do problema CHILD. Podemos observar que o valor de acurácia, em geral, dos algoritmos foram altos, sendo quanto maior o volume de dados, maior a

acurácia obtida. O GES teve valores altos de TLF nas instâncias 10000,20000 e 50000, mas não convergiu em nenhuma dessas instâncias, nessas instâncias os valores de SLF e acurácia média das execuções do GES foram baixos. Outro exemplo foi o PSO-BNC que obteve um alto valor de TLF e menores valores em SLF e acurácia e não convergiu na instância 50000. Isso mostra que a não convergência está relacionada, dentre outros motivos, na dificuldade de direcionamento dos arcos.

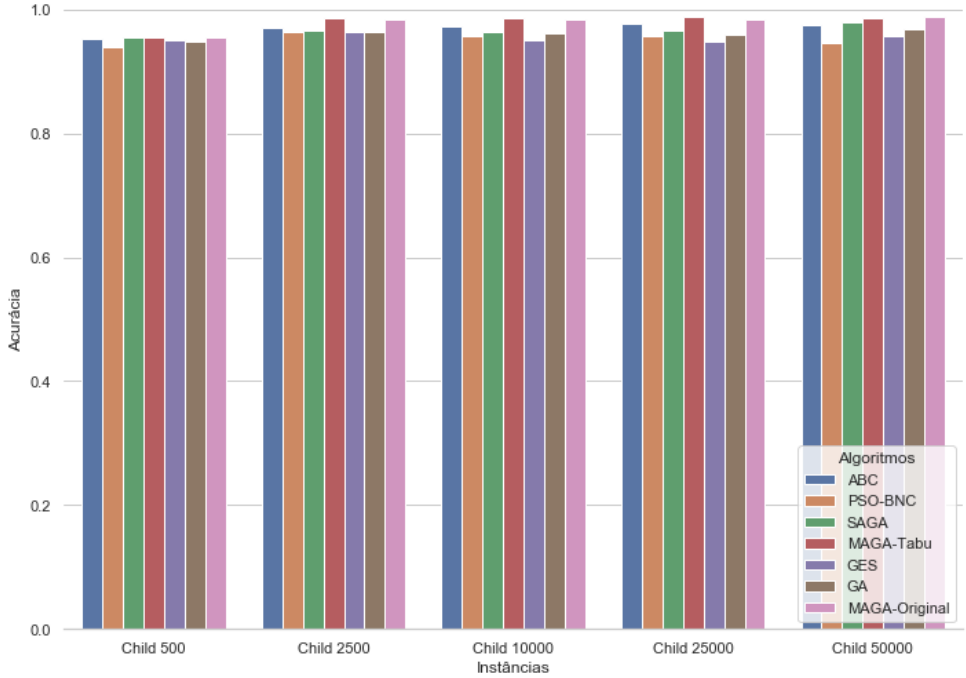

Figura 4.12 – Comparação da acurácia dos algoritmos em todas as instâncias do problema CHILD. (A) é a comparação da acurácia da média das 20 execuções. (B) é comparação da acurácia das execuções que convergiram.

Os resultados do problema CHILD. Os MAGAs apresentaram melhores resultados em todas as métricas utilizadas. Sendo pouca a diferença entre os MAGAs. O GES obteve os piores resultados em convergências e nas métricas STF, TLF e acurácia. O PSO-BNC obteve as menores taxas de convergência e os maiores custos computacionais. Nos gráficos SLF, TLF e a acurácia, observamos que os valores na primeira instância foram os menores e quanto maior o volume de dados maior foram os valores obtidos.

No problema INSURANCE, devido ao alto tempo computacional e as baixas taxas de convergência dos algoritmos. Foram escolhidos os 3 melhores algoritmos das instâncias 1000, 5000, 20000 e 50000 para a execução da instância 100000, os outros algoritmos não foram executados.

A Figura 4.13 mostra os tempos computacionais dos algoritmos. O algoritmo que possui menor tempo foi o GES e o que apresenta o maior tempo computacional foi o PSO-BNC. Os MAGAs possuem os menores tempos computacionais depois do GES. Na instância 100000, o MAGA-Original possui o menor tempo dentre os três algoritmos, sendo o MAGA-Tabu o maior

tempo. Observamos que o tempo computacional dos algoritmos cresce conforme aumenta o volume de dados.

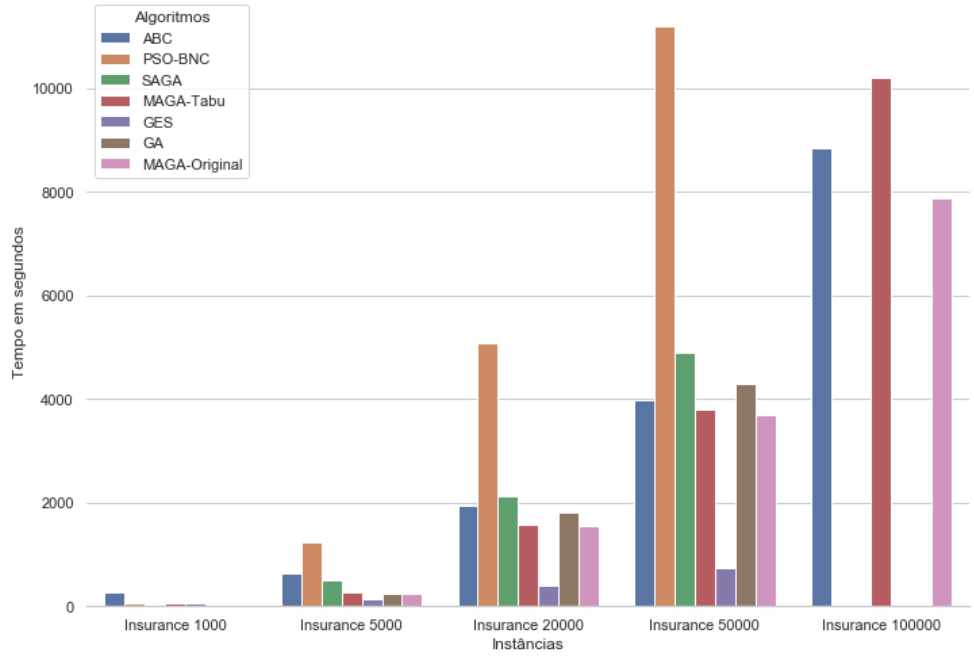

Figura 4.13 – Comparação do tempo computacional dos algoritmos em todas as instâncias do problema INSURANCE.

Na Figura 4.14 foi apresentado a taxa de convergência dos algoritmos. Na primeira instância observamos que todos os algoritmos convergiram em todos as execuções. Na instância5000 o ABC, MAGA-Tabu, MAGA-Original e GES mantiveram a convergência em todos os casos, o PSO-BNC obteve uma redução para 60% de taxa de convergência e o SAGA para 55%. Na instância 20000, os MAGAs apresentaram melhores taxas de convergência comparado com os demais algoritmos. Os algoritmos PSO-BNC e GES não conseguiram convergência, e o SAGA e GA obtiveram uma taxa de convergência igual e menor que 30% respectivamente. Na instância 50000, o MAGA-Tabu e o ABC possuem a maior taxa de convergência. O SAGA,GES, PSO-BNC não convergiram. Na instância 100000, os MAGAs possuíram o maior valor da taxa de convergência sendo igual a 25% e o ABC obteve uma taxa de convergência equivalente a 20%.

Na Tabela 4.3 temos os valores da BIC dos algoritmos para o problema INSURANCE. Na primeira instância observamos que todos os algoritmos obtiveram valores menores de BIC que o valor da referência. Na instância 5000, apenas os algoritmos PSO-BNC e o SAGA, sendo o SAGA a maior diferença em relação a referência, aproximadamente 533. Na instância 200000, a diferença entre o BIC dos algoritmos e o valor referencia foi entre 207 à 3308. Apesar de possui convergência o SAGA obteve o pior diferença em relação à referência, nessa instância. Na instância 50000, essa diferença foi entre 792 até 9015, o valor de 792 obtido pelo ABC é aproximadamente 0.30% do valor de referência, apesar dos 40% de taxa de convergência. O valor de 9015 é aproximadamente 3.3% do valor de convergência, mostrando que apesar da

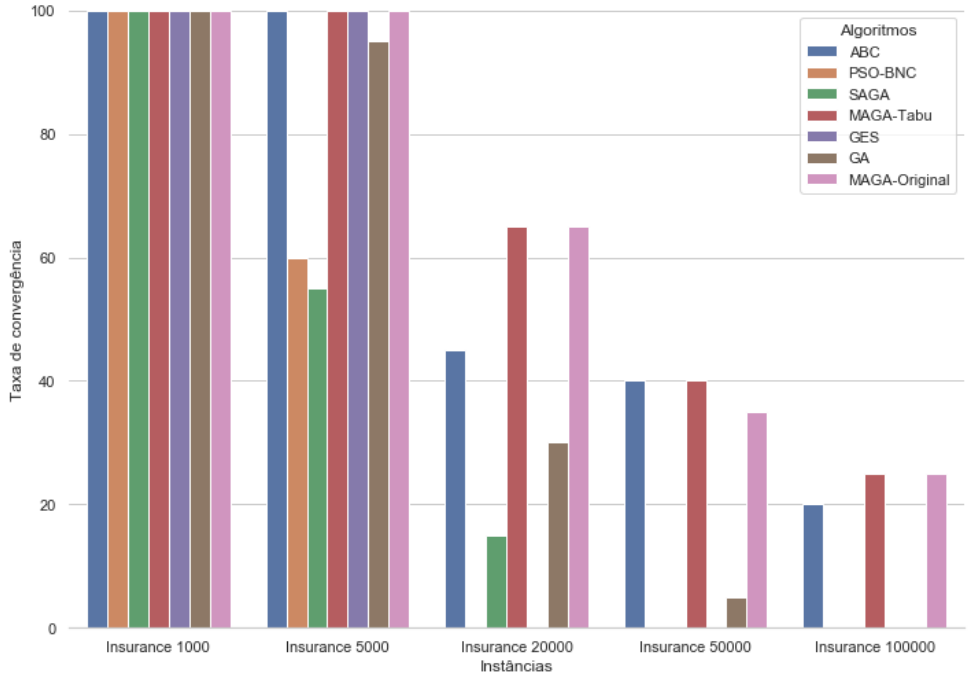

Figura 4.14 – Comparação da taxa de convergência para a rede alvo dos algoritmos em todas as instâncias do problema INSURANCE.

	ABC	PSO-BNC	SAGA	MAGA-Tabu	GES	GA	MAGA-Original
INSURANCE 1000	16154.5426	16161.2555	16172.1291	16071.1732	16152.3175	16157.8713	16101.8141
INSURANCE 5000	69078.0864	69331.6396	69654.6095	69042.9879	68999.9943	69067.8519	69053.5384
INSURANCE 20000	266282.0342	268752.2937	269268.3410	266496.5387	266226.8635	266963.0642	266168.4228
INSURANCE 50000	659541.7774	664809.0019	667764.2516	659717.1180	661017.3948	660989.9571	659739.7541
INSURANCE 100000	1313387.8649	-	-	1312754.8067	-	-	1312740.5569

Tabela 4.3 – Valor da BIC dos algoritmos nas instâncias do problema INSURANCE.

não convergência dos algoritmos, eles conseguiram valores próximos da referência. Em todas as instâncias que foi testado, o SAGA obteve os maiores valores de BIC. Na última instância, a diferença dos algoritmos e o valor da referência foi entre 1452 à 2099. O valor de 2099 é aproximadamente 0.16% do valor da referência. Apesar dos três algoritmos possuírem a maior taxa de convergência igual a 25%, os algoritmos obtiveram valor muito próximos do valor de referência.

As Figuras 4.15 e 4.16 representam o SLF e TLF, respectivamente, do problema INSURANCE. Na instância 1000, o MAGA-Tabu e o SAGA obtiveram o maior SLF e o menor valor foi obtido pelo PSO-BNC. Nas instâncias 5000, 20000 e 50000 os menores valores foram obtidos pelo PSO-BNC, GES e SAGA. Nessas instâncias os melhores SLF foram o MAGA-Tabu, MAGA-Original e ABC. Nas instâncias 1000, 5000 e 20000 os valores de SLF dos algoritmos não foram superiores à 0.6. Nas instâncias 50000 e 100000,o MAGA-Tabu, MAGA-Original e ABC obtiveram valores maiores. Os valores de TLF obtido pelos algoritmos na instância 1000 foram abaixo de 0.4 sendo o MAGA-Tabu o melhor e o GES o menor valor. Na instância 5000 continuou o MAGA-Tabu sendo o melhor valor e o GES o menor. Nas instâncias 20000 e 50000 o MAGA-Tabu, MAGA-Original e ABC se mostraram os melhores TLF e o PSO-BNC

com os menores valores. Na instância 100000, os três algoritmos obtiveram valores próximos a 0.84. Observamos que tanto no SLF quanto no TLF quanto maior o volume de dados maior é o valor médio obtido pelos algoritmos. No INSURANCE, os algoritmos apresentaram maiores dificuldades para encontrar os arcos do problema, diferente do ocorrido no CHILD.

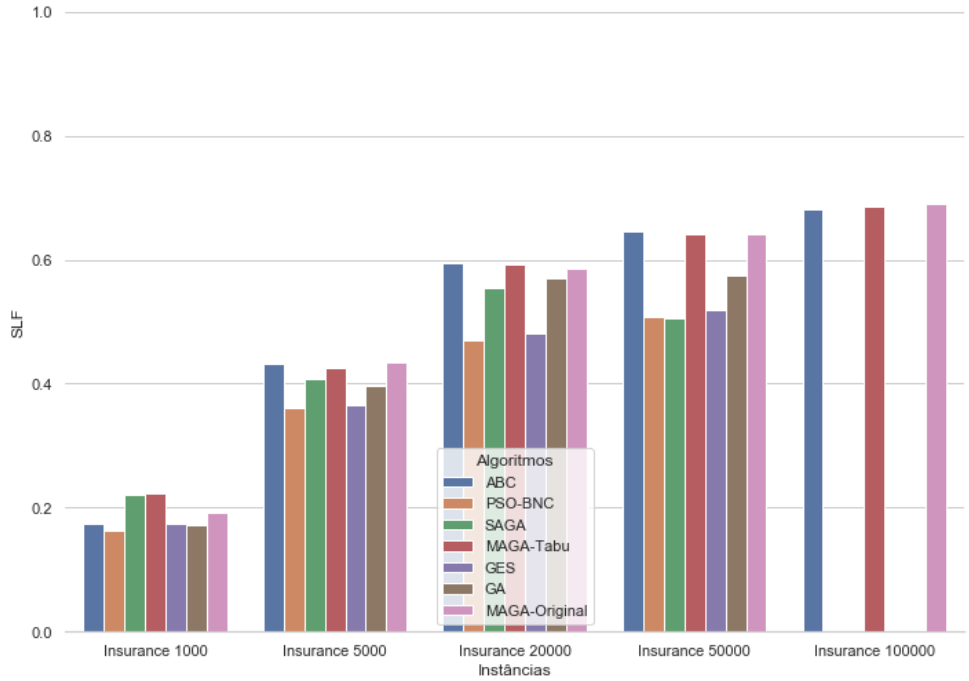

Figura 4.15 – Comparação da métrica SLF dos algoritmos em todas as instâncias do problema INSURANCE.

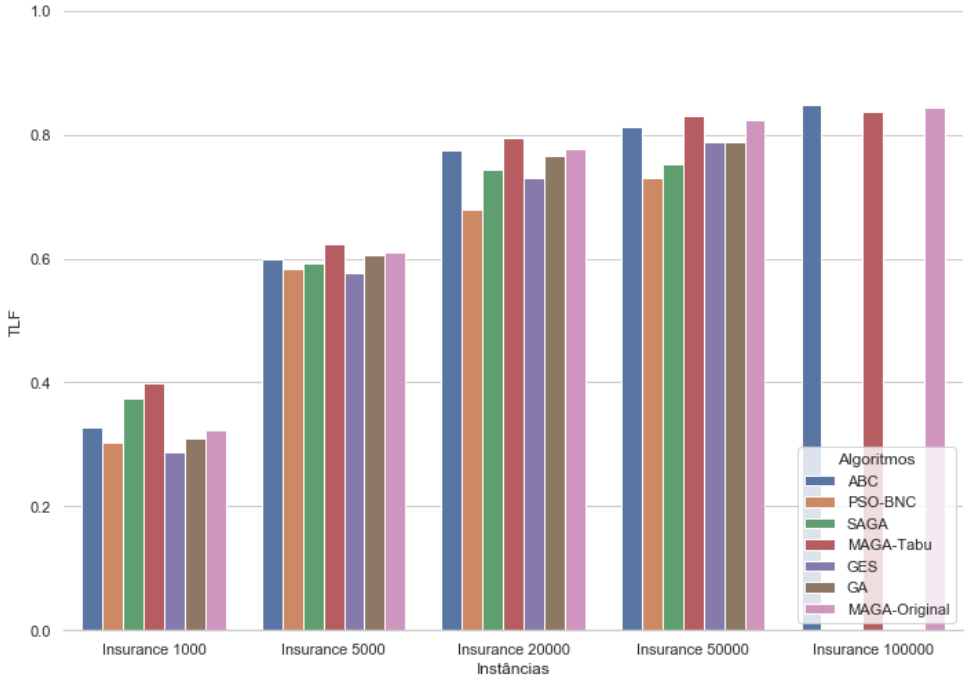

Figura 4.16 – Comparação da métrica TLF dos algoritmos em todas as instâncias do problema INSURANCE.

Na Figura 4.17 é apresentado a acurácia dos algoritmos no problema INSURANCE. Os MAGAs e o ABC são os algoritmos que obtiveram melhores valores de acurácia. Podemos

observar que para a instância 1000, apesar de que todos os algoritmos convergiram, os algoritmos obtiveram os menores valores de acurácia. Na instância 5000,o maior valor foi o GES e o menor foi o PSO-BNC. Na instância 20000, os algoritmos ABC, MAGA-tabu e MAGA-Original obtiveram os maiores valores. Nessa instância os algoritmos PSO-BNC e GES tiveram os menores valores e também são os algoritmos que não convergiram nessa instância. Na instância 50000, o SAGA, PSO-BNC e GES obtiveram os menores valores e não convergiram nessa instância. Observamos que os valores de acurácia dos algoritmos foram aumentando conforme o aumento do volume de dados. À medida que há o aumento dos dados, os algoritmos apresentaram dificuldade em encontrar o grafo ótimo. Pois, pelos gráficos de acurácia, TLF e SLF, podemos observar que os valores médios dessas métricas dos algoritmos crescem conforme o aumento dos volumes de dados. Esse fenômeno causa uma menor taxa de convergência na maioria dos algoritmos. Na instância 50000, o MAGA-Tabu, MAGA-Original e ABC apresentaram superioridade em relação aos demais algoritmos. O GA conseguiu convergência nessa instância, mas com uma taxa baixa e teve maior tempo computacional comparado com os MAGAs e o ABC.

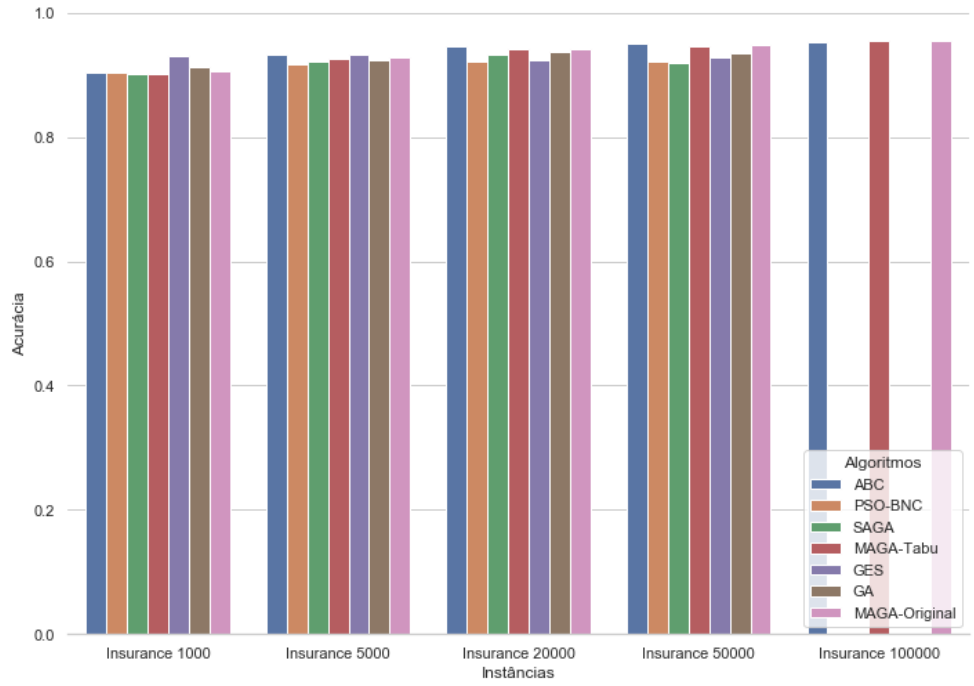

Figura 4.17 – Comparação da acurácia dos algoritmos em todas as instâncias do problema INSURANCE. (A) é a comparação da acurácia da média das 20 execuções. (B) é comparação da acurácia das execuções que convergiram.

No problema INSURANCE, os algoritmos MAGA-Tabu, MAGA-Original e ABC apresentaram os resultados superiores em todas as métricas. Em relação à taxa de convergência e tempo computacional os MAGAs apresentaram melhores resultados, o menor tempo foi o GES, mas ele não teve convergência em duas das quatro instâncias. O PSO-BNC apresentou os piores resultados de tempo computacional e taxa de convergência. Os MAGAs obtiveram uma taxa de convergência até a última instância e apresentaram os melhores resultados comparado com o

ABC. Sendo o MAGA-Original o menor tempo computacional na última instância.

O primeiro teste estatístico foi feito na variável tempo computacional. Na Figura 4.18 são apresentados os três algoritmos comparados em cada uma das instâncias.

Aplicando um teste com blocos, obtivemos o valor de p-valor igual a 2.76e-06 portanto rejeitamos a hipótese nula e há pelo menos uma média diferente das médias dos algoritmos.

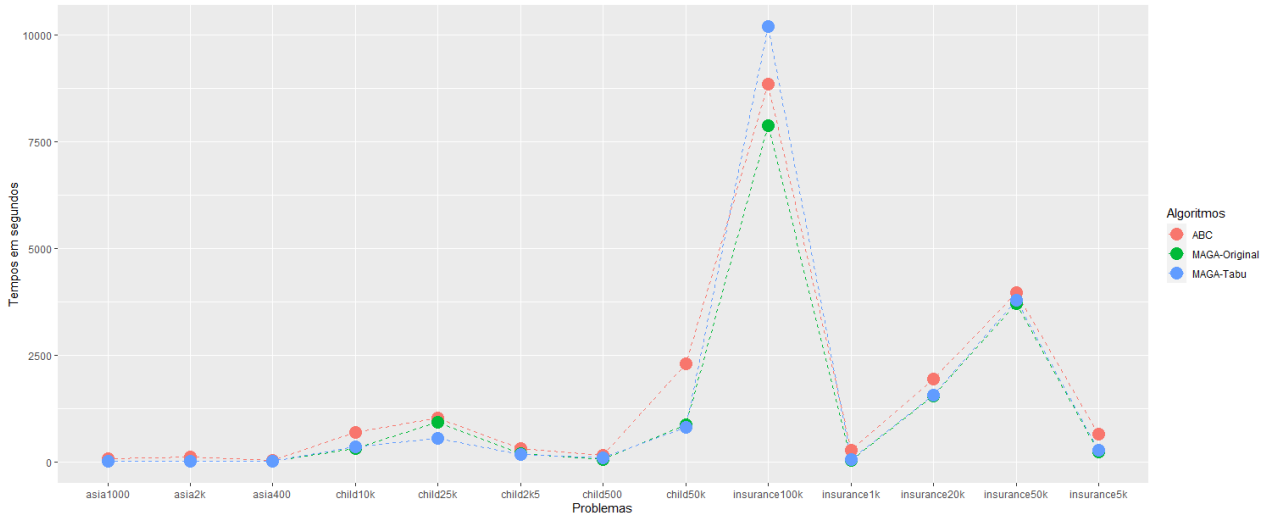

Figura 4.18 – Tempo computacional de todas as instâncias para os algoritmos MAGA-Tabu, MAGA-Original e ABC.

Então foi feito o teste de *Shapiro-Wilk* para verificar a normalidade dos resíduos. O p-valor obtido foi 0.8997. Portanto, não foi rejeitado a hipótese nula, o resíduo é normal. Verificaram-se os resíduos e eles mostraram não ser influenciados pelos fatores, indicando que existe homocedasticidade dos mesmos.

Atendendo as premissas do teste. Para verificar qual ou quais algoritmos divergem da média. Foi usado o teste comparativo chamado teste Tukey.

Na Figura 4.19 obtivemos o resultado do teste Tukey. O resultado mostra que comparando o MAGA-Original com o ABC e o MAGA-Tabu com o ABC, temos que os MAGAs apresentam menores tempos computacionais que o ABC. Enquanto que a diferença do tempo computacional entre os MAGAS não é significativa. Portanto, não há diferença entre o MAGA-Original e o MAGA-Tabu em relação ao tempo computacional.

O segundo teste estatístico foi comparando a taxa de convergência entre os algoritmos. Na Figura 4.20 é mostrado a taxa de convergência dos algoritmos em cada uma das instâncias.

Aplicando a blocagem nas instâncias obtivemos o p-valor igual a 4.74e-06 para os algoritmos. Portanto, se rejeita a hipótese nula e há pelo menos um algoritmo com média de taxa de convergência diferente.

Aplicando o teste de *Shapiro-Wilk* no resíduo. Obtivemos o p-valor igual a 0.5489. O resíduo é normal. Verificando os resíduos foi observado que não houve influência dos fatores,

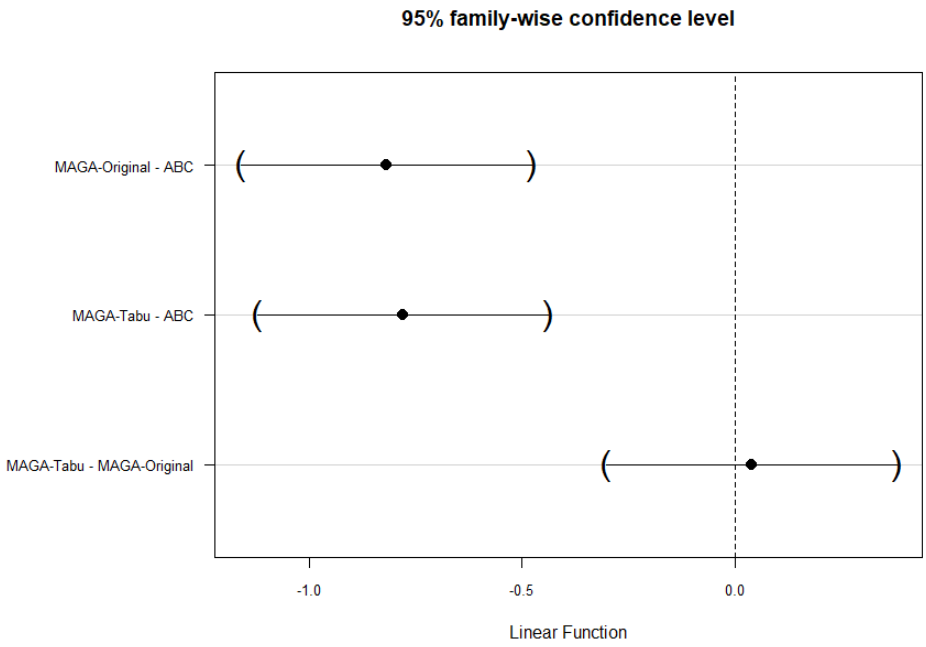

Figura 4.19 – Teste Tukey dos algoritmos MAGA-Tabu, MAGA-Original e ABC em relação ao tempo computacional.

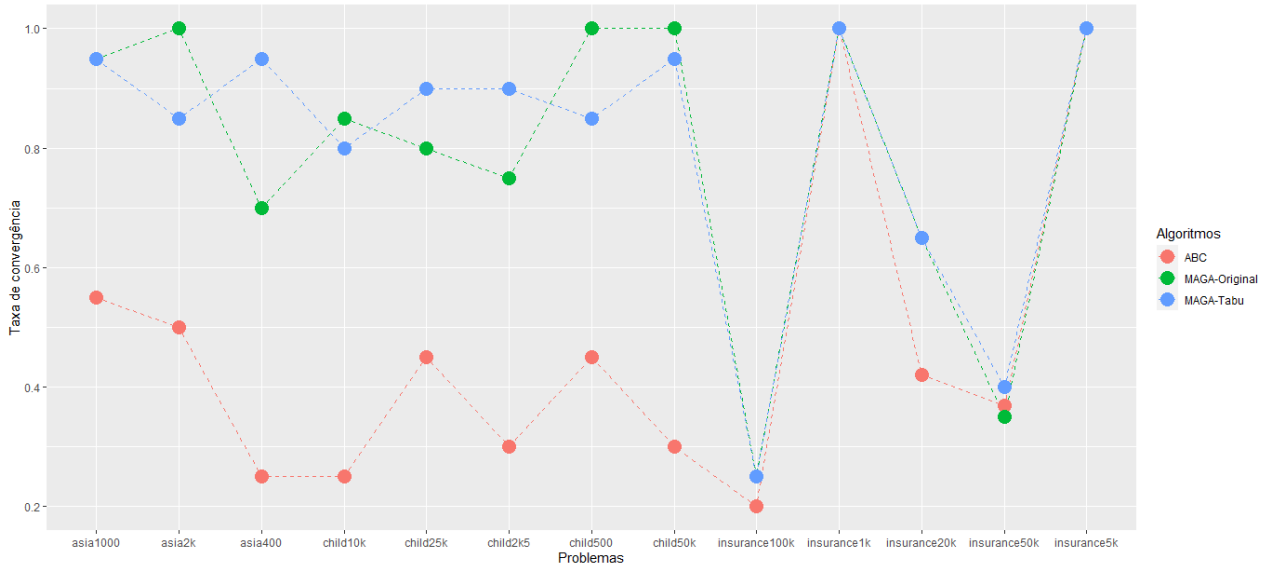

Figura 4.20 – taxa de convergência de todas as instâncias para os algoritmos MAGA-Tabu, MAGA-Original e ABC.

indicando a homocedasticidade.

Depois de satisfazer as premissas, o teste de Tukey foi utilizado para a comparação entre os algoritmos. Na Figura 4.21, observamos que na comparação entre MAGA-Original com o ABC e o MAGA-Tabu com o ABC, os MAGAs obtiveram maiores taxas de convergência. E não existe diferença significativa entre a taxa de convergência dos MAGAs. Portanto, não há diferença entre os MAGAS quando comparado com a taxa de convergência, com os dados disponíveis.

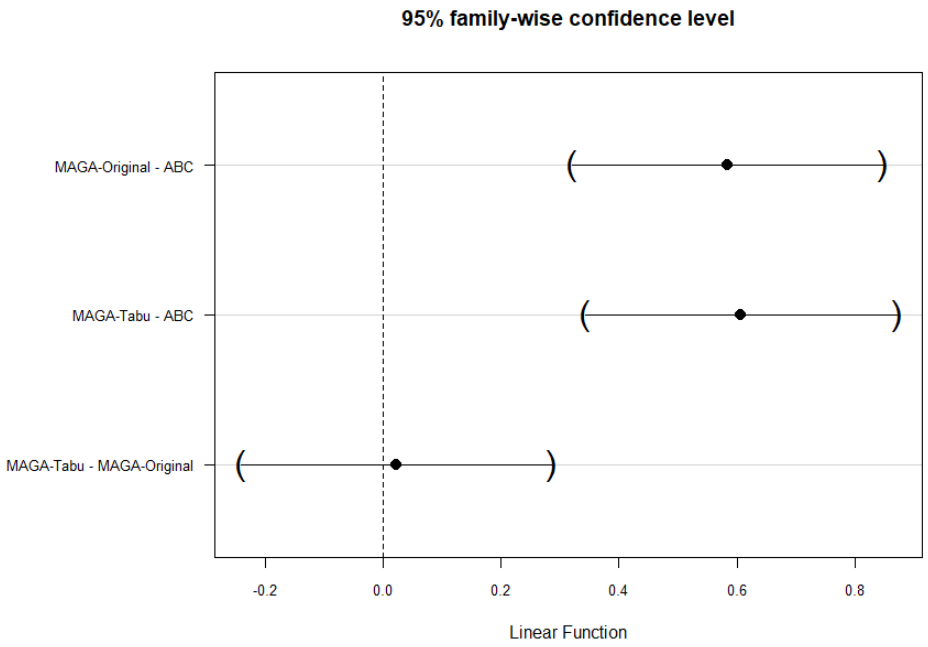

Figura 4.21 – Teste Tukey dos algoritmos MAGA-Tabu, MAGA-Original e ABC em relação à taxa de convergência.

Nessa seção foi feito o teste estatístico para verificar se há diferença entre os algoritmos MAGA-Tabu, MAGA-Original e ABC. Não há evidências estatísticas suficientes para dizer qual dos MAGAs possui melhor tempo computacional ou taxa de convergência. Mas foi observado que os MAGAs possuem melhor tempo computacional e taxas de convergência que o ABC. Concluindo o Capítulo de resultados, observamos que os MAGAs tiveram melhores resultados em todas as métricas utilizadas, comparados com todos os algoritmos testados. O GA, assim como o MAGA, é um algoritmo populacional. Nesse trabalho, o GA foi utilizado com os mesmos operadores genéticos e de reparo que o MAGA. Mas o GA não conseguiu o mesmo desempenho que os MAGAs. Acreditamos que os melhores resultados obtidos pelo MAGA possam ser um reflexo de como os agentes interagem, pois, o cruzamento acontece apenas com os vizinhos e o processo de auto-aprendizagem ocorre com o melhor agente em cada geração.

CAPÍTULO

5

CONCLUSÕES E TRABALHOS FUTUROS

Nesse trabalho foi abordado a aprendizagem estrutural de Redes Bayesianas (RB) utilizando o MAGA. Foram comparados os algoritmos ABC, PSO-BNC, SAGA, GES e GA, algoritmos utilizados na literatura para esse tipo de problema. Foram estudadas duas configurações do MAGA, uma abordagem que utiliza o auto-aprendizado original implementada pelo autor do MAGA e outra abordagem implementando a Busca Tabu como auto-aprendizado. O ASIA, INSURANCE e CHILD foram os problemas escolhidos para esse estudo, em diferentes instâncias para cada problema. O estudo desse trabalho teve como objetivo verificar o comportamento do MAGA na aprendizagem estrutura de RBs com ênfase em seu comportamento quanto as instâncias com maior número de variáveis e grande volume de dado.

A escolha do número de avaliação do BIC como critério de parada dos algoritmos foi feito com o objetivo de compensar os tipos de algoritmos testados. Observamos pelos resultados que o PSO-BNC não apresentou bons resultados, O tempo computacional foi muito elevado em todos os problemas, principalmente com grande volume de dados, esse resultado foi coerente com o reportado no trabalho [Gheisari e Meybodi 2016]. Seus resultados foram melhores nas primeiras instâncias do CHILD e do INSURANCE e um resultado mediano no problema do ASIA. O GES é outro algoritmo que apresentou melhores resultados em instâncias com baixo volume de dados. Porém, diferente do PSO-BNC, apresentou um tempo computacional baixo, sendo o menor em todos os problemas. O SAGA e o ABC, junto aos MAGAs foram os algoritmos que apresentaram melhores resultados com grandes volumes de dados, tanto em relação ao tempo computacional quanto em relação a convergência. Os MAGAs mostraram graficamente melhores resultados que os algoritmos comparados, principalmente em relação as instâncias com mais variáveis e grandes volumes de dados, mantendo uma taxa de convergências alta com um baixo tempo computacional como mostrada nas duas ultimas instâncias dos problemas CHILD e INSURANCE. Comparando os dois MAGAs, observamos que seus resultados são parecidos. Graficamente no problema ASIA o MAGA-Original obteve maior taxa de convergência na última instância apresentando um tempo parecido com o MAGA-Tabu, nas demais instâncias o

MAGA-Original obteve um menor tempo que o MAGA-Tabu e o MAGA-Tabu teve uma maior convergência na primeira instância. Mas nos problemas INSURANCE e CHILD os resultados ficaram próximos. Na última instância do INSURANCE as convergências dos dois MAGAs foram iguais, mas o MAGA-Original teve um menor tempo computacional e na última instância do CHILD o tempo computacional foi praticamente igual enquanto o MAGA-original obteve uma maior taxa de convergência. Estatisticamente foi testado os MAGAs e o ABC que apresentou melhor convergência no INSURANCE. Dado os resultados dos experimentos não há evidências suficientes para observar diferença entre os algoritmos MAGA-original e MAGA-Tabu, mas os dois obtiveram melhores resultados que o ABC.

Em relação aos erros estruturais, foram utilizados as métricas SLF, TLF e acurácia. Os resultados mostraram que os algoritmos obtiveram valores maiores de TLF do que de SLF, isso mostra que os algoritmos tiveram maior dificuldade em encontra a direção do arco do que se há ou não arco. Portanto, definir a direção da relação de dependência se mostra mais difícil que encontra a dependência entre as variáveis. Comparando os algoritmos, observamos que todos os algoritmos apresentaram um valor alto de acurácia, a menor acurácia foi o PSO na instância 1000 do ASIA com valor igual a 0.885. Nos problemas CHILD e INSURANCE, os MAGAs e o ABC obtiveram os melhores resultados nas métricas SLF, TLF e acurácia. Nos problemas estudados, podemos observar que quanto maior o volume de dados, maior é o valor obtido da acurácia, as métricas SLF e TLF acompanham essa relação. Esses resultados mostram que há vantagem no estudo do aprendizagem estrutural de RBs com grandes volumes de dados. Pois, quanto maior o volume de dado maior é qualidade do resultado, no INSURANCE nas instâncias 1000 os algoritmos convergiram com acurácias iguais a 0.90, na instância 5000 convergiram com 0.917, nas instâncias 20000 e 50000 os algoritmos que convergiram obtiveram acurácias maiores que 0.93, os algoritmos que não convergiram obtiveram valores entorno de 0.92, e na instância 100000 a acurácia foi entorno de 0.95 para os três algoritmos.

Os resultados mostram que o MAGA é recomendado para lidar com o aprendizado estrutural de RBs com maior número de variáveis e grande volume de dados. Os métodos de auto-aprendizado utilizados não mostram grande influência no resultado. O mecanismo de cruzamento somente com os vizinhos e o auto-aprendizado, acreditamos que seja o motivo do melhor resultado dos MAGAs no aprendizado estrutural de RBs.

Nos testes desse trabalho, observamos que as métricas de pontuação, principalmente a BIC que foi utilizada em todos os testes, apresentam uma imprecisão na pontuação da rede. Duas redes diferentes obtiveram valores iguais e obtivemos redes diferentes da rede alvo do problema com menores valores de BIC. Observamos que quanto maior é o volume de dados, maiores são os valores das métricas SLF, TLF e acurácias. Isso mostra que um grande volume de dados reduz essa imprecisão apresentada pelas métricas de pontuação.

5.1 Trabalhos Futuros

Como observado graficamente, nas últimas instâncias dos problemas CHILD e INSURANCE o MAGA-original obteve um desempenho um pouco melhor no tempo computacional ou na convergência em relação ao MAGA-Tabu. Para trabalhos futuros, podemos testar essas configurações do MAGA, em outros problemas e outros volumes de dados, e verificar se alguma dessas adaptações do MAGA apresenta um melhor resultado.

Para trabalhos futuros, a pesquisa de novas heurísticas para o auto-aprendizado poderá melhorar o tempo computacional ou a convergência do MAGA. O GES foi uma heurística que apresentou um baixo tempo computacional em todas as instâncias e manteve um valor alto nas métricas SLF, TLF e acurácias. Utilizar o GES como auto-aprendizado do MAGA seria uma implementação interessante. O estudo de outras heurísticas que apresentam bons resultados em lidar com grandes volumes de dados, poderia melhorar os resultados do MAGA.

Outro trabalho futuro seria estudar com profundidade o motivo de obter valores de BIC menores ou iguais à rede referência como descrita na seção 4.1. Esse estudo poderia melhorar os resultados de convergência e precisão dos algoritmos.

REFERÊNCIAS

ADABOR, E. S.; ACQUAAH-MENSAH, G. K.; ODURO, F. T. Saga: a hybrid search algorithm for bayesian network structure learning of transcriptional regulatory networks. **Journal of biomedical informatics**, Elsevier, v. 53, p. 27–35, 2015. Citado 2 vezes nas páginas: 25 e 38.

AKAIKE, H. A new look at the statistical model identification. **IEEE transactions on automatic control**, Ieee, v. 19, n. 6, p. 716–723, 1974. Citado na página 21.

ALDRICH, J. *et al.* Ra fisher and the making of maximum likelihood 1912-1922. **Statistical science**, Institute of Mathematical Statistics, v. 12, n. 3, p. 162–176, 1997. Citado na página 20.

BARTLETT, M.; CUSSENS, J. Integer linear programming for the bayesian network structure learning problem. **Artificial Intelligence**, Elsevier, v. 244, p. 258–271, 2017. Citado na página 22.

BEHJATI, S.; BEIGY, H. Improved k2 algorithm for bayesian network structure learning. **Engineering Applications of Artificial Intelligence**, Elsevier, v. 91, p. 103617, 2020. Citado na página 28.

BESSANI, M.; MASSIGNAN, J. A.; SANTOS, T. M.; JR, J. B. L.; MACIEL, C. D. Multiple households very short-term load forecasting using bayesian networks. **Electric Power Systems Research**, Elsevier, v. 189, p. 106733, 2020. Citado na página 16.

BOBBIO, A.; PORTINALE, L.; MINICHINO, M.; CIANCAMERLA, E. Improving the analysis of dependable systems by mapping fault trees into bayesian networks. **Reliability Engineering & System Safety**, Elsevier, v. 71, n. 3, p. 249–260, 2001. Citado na página 19.

CAMPOS, C. P. d.; JI, Q. Efficient structure learning of bayesian networks using constraints. **Journal of Machine Learning Research**, v. 12, n. Mar, p. 663–689, 2011. Citado na página 22.

CAO, W.; FANG, X. An improved method for bayesian network structure learning. In: IEEE. **2010 Sixth International Conference on Natural Computation**. [S.l.], 2010. v. 6, p. 3133–3137. Citado na página 24.

CHICKERING, D.; GEIGER, D.; HECKERMAN, D. Learning bayesian networks: Search methods and experimental results. In: **proceedings of fifth conference on artificial intelligence and statistics**. [S.l.: s.n.], 1995. p. 112–128. Citado na página 16.

CHICKERING, D. M. Optimal structure identification with greedy search. **Journal of machine learning research**, v. 3, n. Nov, p. 507–554, 2002. Citado 2 vezes nas páginas: 23 e 38.

CONSTANTINOU, A. C.; LIU, Y.; CHOBTHAM, K.; GUO, Z.; KITSON, N. K. Large-scale empirical validation of bayesian network structure learning algorithms with noisy data. **International Journal of Approximate Reasoning**, Elsevier, v. 131, p. 151–188, 2021. Citado na página 29.

CONTALDI, C. **Bayesian Network Hybrid Learning Using a Parent Reducing Site-specific Mutation Rate Genetic Algorithm**. Tese (Doutorado) — University of Illinois at Chicago, 2016. Citado 2 vezes nas páginas: 20 e 21.

CONTALDI, C.; VAFAEE, F.; NELSON, P. C. Bayesian network hybrid learning using an elite-guided genetic algorithm. **Artificial Intelligence Review**, Springer, v. 52, n. 1, p. 245–272, 2019. Citado na página 23.

COOPER, G. F.; HERSKOVITS, E. A bayesian method for constructing bayesian belief networks from databases. In: **Uncertainty Proceedings 1991**. [S.l.]: Elsevier, 1991. p. 86–94. Citado na página 21.

____. A bayesian method for the induction of probabilistic networks from data. **Machine learning**, Springer, v. 9, n. 4, p. 309–347, 1992. Citado na página 23.

DAI, J.; REN, J.; DU, W.; SHIKHIN, V.; MA, J. An improved evolutionary approach-based hybrid algorithm for bayesian network structure learning in dynamic constrained search space. **Neural Computing and Applications**, Springer, v. 32, n. 5, p. 1413–1434, 2020. Citado na página 23.

DALY, R.; SHEN, Q.; AITKEN, S. Learning bayesian networks: approaches and issues. **The knowledge engineering review**, Cambridge University Press, v. 26, n. 2, p. 99–157, 2011. Citado na página 19.

DEMPSTER, A. P.; LAIRD, N. M.; RUBIN, D. B. Maximum likelihood from incomplete data via the em algorithm. **Journal of the Royal Statistical Society: Series B (Methodological)**, Wiley Online Library, v. 39, n. 1, p. 1–22, 1977. Citado na página 20.

FOURNIER, F. A.; WU, Y.; MCCALL, J.; PETROVSKI, A.; BARCLAY, P. J. Application of evolutionary algorithms to learning evolved bayesian network models of rig operations in the gulf of mexico. In: IEEE. **2010 UK Workshop on Computational Intelligence (UKCI)**. [S.l.], 2010. p. 1–6. Citado na página 25.

GHEISARI, S.; MEYBODI, M. R. Bnc-pso: structure learning of bayesian networks by particle swarm optimization. **Information Sciences**, Elsevier, v. 348, p. 272–289, 2016. Citado 4 vezes nas páginas: 25, 29, 38 e 59.

GROSS, T. J.; ARAUJO, R. B.; VALE, F. A. C.; BESSANI, M.; MACIEL, C. D. Dependence between cognitive impairment and metabolic syndrome applied to a brazilian elderly dataset. **Artificial intelligence in medicine**, Elsevier, v. 90, p. 53–60, 2018. Citado na página 16.

GUO, Y.; SCHUURMANS, D. Convex structure learning for bayesian networks: Polynomial feature selection and approximate ordering. **Pages 208–216 of: Proc. of the 22nd Conference on Uncertainty in Artificial Intelligence. Arlington, Virginia: AUAI Press**, 2006. Citado na página 22.

HAGBERG, A. A.; SCHULT, D. A.; SWART, P. J. Exploring network structure, dynamics, and function using networkx. In: VAROQUAUX, G.; VAUGHT, T.; MILLMAN, J. (Ed.). **Proceedings of the 7th Python in Science Conference**. Pasadena, CA USA: [s.n.], 2008. p. 11 – 15. Citado na página 40.

HARRIS, C. R.; MILLMAN, K. J.; AL., S. J. van der Walt et. Array programming with NumPy. **Nature**, Springer Science and Business Media LLC, v. 585, n. 7825, p. 357–362, set. 2020. Disponível em: <https://doi.org/10.1038/s41586-020-2649-2>. Citado na página 40.

HECKERMAN, D.; GEIGER, D.; CHICKERING, D. M. Learning bayesian networks: The combination of knowledge and statistical data. **Machine learning**, Springer, v. 20, n. 3, p. 197–243, 1995. Citado na página 21.

HUANG, J.; LIU, J.; YAO, X. A multi-agent evolutionary algorithm for software module clustering problems. **Soft Computing**, Springer, v. 21, n. 12, p. 3415–3428, 2017. Citado na página 30.

JI, J.; WEI, H.; LIU, C. An artificial bee colony algorithm for learning bayesian networks. **Soft Computing**, Springer, v. 17, n. 6, p. 983–994, 2013. Citado na página 38.

JI, J.; YANG, C.; LIU, J.; LIU, J.; YIN, B. A comparative study on swarm intelligence for structure learning of bayesian networks. **Soft Computing**, Springer, v. 21, n. 22, p. 6713–6738, 2017. Citado 2 vezes nas páginas: 26 e 42.

KABLI, R.; HERRMANN, F.; MCCALL, J. A chain-model genetic algorithm for bayesian network structure learning. In: **Proceedings of the 9th annual conference on Genetic and evolutionary computation**. [S.l.: s.n.], 2007. p. 1264–1271. Citado na página 24.

KHANTEYMOORI, A. R.; OLYAEE, M.-H.; ABBASZADEH, O.; VALIAN, M. A novel method for bayesian networks structure learning based on breeding swarm algorithm. **Soft Computing**, Springer, v. 22, n. 9, p. 3049–3060, 2018. Citado na página 26.

KOLLER, D.; FRIEDMAN, N. **Probabilistic graphical models: principles and techniques**. [S.l.]: MIT press, 2009. Citado 2 vezes nas páginas: 16 e 20.

LARRANAGA, P.; KUIJPERS, C. M.; MURGA, R. H.; YURRAMENDI, Y. Learning bayesian network structures by searching for the best ordering with genetic algorithms. **IEEE transactions on systems, man, and cybernetics-part A: systems and humans**, IEEE, v. 26, n. 4, p. 487–493, 1996. Citado 2 vezes nas páginas: 24 e 38.

LARRAÑAGA, P.; POZA, M.; YURRAMENDI, Y.; MURGA, R. H.; KUIJPERS, C. M. H. Structure learning of bayesian networks by genetic algorithms: A performance analysis of control parameters. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 18, n. 9, p. 912–926, 1996. Citado 4 vezes nas páginas: 23, 29, 37 e 40.

LAURITZEN, S. L.; SPIEGELHALTER, D. J. Local computations with probabilities on graphical structures and their application to expert systems. **Journal of the Royal Statistical Society: Series B (Methodological)**, Wiley Online Library, v. 50, n. 2, p. 157–194, 1988. Citado na página 33.

LEE, S.; KIM, S. B. Parallel simulated annealing with a greedy algorithm for bayesian network structure learning. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, 2019. Citado 2 vezes nas páginas: 27 e 29.

LI, H.; WANG, F.; LI, H. Integrating expert knowledge for bayesian network structure learning based on intuitionistic fuzzy set and genetic algorithm. **Intelligent Data Analysis**, IOS Press, v. 23, n. 1, p. 41–56, 2019. Citado na página 27.

LI, Z.; LIU, J. A multi-agent genetic algorithm for community detection in complex networks. **Physica A: Statistical Mechanics and its Applications**, Elsevier, v. 449, p. 336–347, 2016. Citado na página 30.

LIU, J.; ZHONG, W.; JIAO, L. A multiagent evolutionary algorithm for constraint satisfaction problems. **IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)**, IEEE, v. 36, n. 1, p. 54–73, 2006. Citado na página 17.

____. A multiagent evolutionary algorithm for combinatorial optimization problems. **IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)**, IEEE, v. 40, n. 1, p. 229–240, 2009. Citado na página 17.

MARTINS, M. S.; YAFRANI, M. E.; DELGADO, M.; LÜDERS, R.; SANTANA, R.; SIQUEIRA, H. V.; AKCAY, H. G.; AHIOD, B. Analysis of bayesian network learning techniques for a hybrid multi-objective bayesian estimation of distribution algorithm: a case study on mnk landscape. **Journal of Heuristics**, Springer, p. 1–25, 2021. Citado 2 vezes nas páginas: 28 e 40.

OKUTAN, A.; YILDIZ, O. T. Software defect prediction using bayesian networks. **Empirical Software Engineering**, Springer, v. 19, n. 1, p. 154–181, 2014. Citado na página 16.

OUSSOUS, A.; BENJELLOUN, F.-Z.; LAHCEN, A. A.; BELFKIH, S. Big data technologies: A survey. **Journal of King Saud University-Computer and Information Sciences**, Elsevier, v. 30, n. 4, p. 431–448, 2018. Citado na página 17.

PENG, C.; WU, G.; LIAO, T. W.; WANG, H. Research on multi-agent genetic algorithm based on tabu search for the job shop scheduling problem. **PloS one**, Public Library of Science, v. 14, n. 9, 2019. Citado 3 vezes nas páginas: 30, 36 e 37.

SAHIN, F.; DEVASIA, A. **Distributed particle swarm optimization for structural Bayesian network learning**. [S.l.]: Citeseer, 2007. Citado na página 24.

SAHIN, F.; YAVUZ, M. Ç.; ARNAVUT, Z.; ULUYOL, Ö. Fault diagnosis for airplane engines using bayesian networks and distributed particle swarm optimization. **Parallel Computing**, Elsevier, v. 33, n. 2, p. 124–143, 2007. Citado 2 vezes nas páginas: 16 e 24.

SCANAGATTA, M.; SALMERÓN, A.; STELLA, F. A survey on bayesian network structure learning from data. **Progress in Artificial Intelligence**, Springer, p. 1–15, 2019. Citado 3 vezes nas páginas: vii, 16 e 20.

SCHWARZ, G. *et al.* Estimating the dimension of a model. **The annals of statistics**, Institute of Mathematical Statistics, v. 6, n. 2, p. 461–464, 1978. Citado na página 21.

SCUTARI, M. Learning bayesian networks with the bnlearn R package. **Journal of Statistical Software**, v. 35, n. 3, p. 1–22, 2010. Citado na página 41.

____. An empirical-bayes score for discrete bayesian networks. In: **Conference on probabilistic graphical models**. [S.l.: s.n.], 2016. p. 438–448. Citado na página 21.

SCUTARI, M.; GRAAFLAND, C. E.; GUTIÉRREZ, J. M. Who learns better bayesian network structures: Accuracy and speed of structure learning algorithms. **International Journal of Approximate Reasoning**, Elsevier, v. 115, p. 235–253, 2019. Citado 3 vezes nas páginas: 27, 34 e 38.

SCUTARI, P. M. **Bayesian Network Repository**. 2020. <https://www.bnlearn.com/bnrepository/>. [Online; acesso 02-Abril-2020]. Citado 4 vezes nas páginas: vii, 22, 34 e 35.

SESEN, M. B.; NICHOLSON, A. E.; BANARES-ALCANTARA, R.; KADIR, T.; BRADY, M. Bayesian networks for clinical decision support in lung cancer care. **PloS one**, Public Library of Science, v. 8, n. 12, 2013. Citado na página 16.

SILANDER, T.; MYLLYMAKI, P. A simple approach for finding the globally optimal bayesian network structure. **arXiv preprint arXiv:1206.6875**, 2012. Citado na página 22.

SPIEGELHALTER, D. J.; DAWID, A. P.; LAURITZEN, S. L.; COWELL, R. G. Bayesian analysis in expert systems. **Statistical science**, JSTOR, p. 219–247, 1993. Citado na página 33.

SUN, X.; CHEN, C.; WANG, L.; KANG, H.; SHEN, Y.; CHEN, Q. Hybrid optimization algorithm for bayesian network structure learning. **Information**, Multidisciplinary Digital Publishing Institute, v. 10, n. 10, p. 294, 2019. Citado 2 vezes nas páginas: 27 e 29.

TANG, Y.; WANG, J.; NGUYEN, M.; ALTINTAS, I. Penbayes: A multi-layered ensemble approach for learning bayesian network structure from big data. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 19, n. 20, p. 4400, 2019. Citado na página 17.

TSAMARDINOS, I.; BROWN, L. E.; ALIFERIS, C. F. The max-min hill-climbing bayesian network structure learning algorithm. **Machine learning**, Springer, v. 65, n. 1, p. 31–78, 2006. Citado na página 22.

VILLANUEVA, E.; MACIEL, C. D. Efficient methods for learning bayesian network superstructures. **Neurocomputing**, Elsevier, v. 123, p. 3–12, 2014. Citado na página 23.

WANG, S.; LIU, J. A multi-agent genetic algorithm for improving the robustness of communities in complex networks against attacks. In: IEEE. **2017 IEEE Congress on Evolutionary Computation (CEC)**. [S.l.], 2017. p. 17–22. Citado 2 vezes nas páginas: 30 e 36.

WANG, T.; YANG, J. A heuristic method for learning bayesian networks using discrete particle swarm optimization. **Knowledge and information systems**, Springer, v. 24, n. 2, p. 269–281, 2010. Citado 2 vezes nas páginas: 24 e 29.

WU, J.; GUO, Y.; ZHOU, H.; SHEN, L.; LIU, L. Vehicular delay tolerant network routing algorithm based on bayesian network. **IEEE Access**, IEEE, v. 8, p. 18727–18740, 2020. Citado na página 16.

WU, Y. Problem dependent metaheuristic performance in bayesian network structure learning. 2012. Citado na página 25.

XING-CHEN, H.; ZHENG, Q.; LEI, T.; LI-PING, S. Learning bayesian network structures with discrete particle swarm optimization algorithm. In: IEEE. **2007 IEEE Symposium on Foundations of Computational Intelligence**. [S.l.], 2007. p. 47–52. Citado na página 24.

YUAN, C.; MALONE, B. Learning optimal bayesian networks: A shortest path perspective. **Journal of Artificial Intelligence Research**, v. 48, p. 23–65, 2013. Citado na página 22.

ZHANG, X.; MAHADEVAN, S. Bayesian network modeling of accident investigation reports for aviation safety assessment. **Reliability Engineering & System Safety**, Elsevier, v. 209, p. 107371, 2021. Citado na página 16.

ZHANG, X.; XUE, Y.; LU, X.; JIA, S. Differential-evolution-based coevolution ant colony optimization algorithm for bayesian network structure learning. **Algorithms**, Multidisciplinary Digital Publishing Institute, v. 11, n. 11, p. 188, 2018. Citado na página 27.

ZHANG, Y.; ZHOU, M.; JIANG, Z.; LIU, J. A multi-agent genetic algorithm for big optimization problems. In: IEEE. **2015 IEEE Congress on Evolutionary Computation (CEC)**. [S.l.], 2015. p. 703–707. Citado 3 vezes nas páginas: 17, 30 e 31.

ZHONG, W.; LIU, J.; XUE, M.; JIAO, L. A multiagent genetic algorithm for global numerical optimization. **IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)**, IEEE, v. 34, n. 2, p. 1128–1141, 2004. Citado 2 vezes nas páginas: 17 e 30.

APÊNDICE

A

DADOS

Asia400							
Algoritmos	tempo	BestScore	SLF	TLF	f1score	accuracy	Covergência
ABC	51.0562	921.0302	0.5188	0.8625	0.5121	0.8563	25
PSO-BNC	34.1615	919.0187	0.5855	0.8750	0.5953	0.8853	85
SAGA	15.4736	918.9213	0.75	0.9313	0.75	0.9286	100
MAGA-Tabu	15.5038	918.9648	0.7625	0.9375	0.7625	0.9321	95
GES	1.1648	918.8693	0.875	0.875	0.875	0.9648	100
GA	23.7539	920.4419	0.5813	0.875	0.5757	0.8759	45
MAGA-Original	17.1572	919.5954	0.6313	0.9	0.6279	0.8919	70
Asia1k							
Algoritmos	tempo	BestScore	SLF	TLF	f1score	accuracy	Covergência
ABC	76.0267	2275.3022	0.7	0.975	0.6868	0.9071	55
PSO-BNC	119.4724	2274.4914	0.6375	0.9188	0.6224	0.8866	75
SAGA	64.1574	2277.2980	0.7563	0.925	0.7360	0.9205	50
MAGA-Tabu	18.5275	2273.5731	0.85	0.9938	0.8478	0.9563	95
GES	1.0481	2273.2422	1	1	1	1	100
GA	33.3844	2274.8358	0.6563	1	0.6453	0.8938	60
MAGA-Original	22.6162	2273.6019	0.8125	0.9938	0.8107	0.9455	95
Asia2k							
Algoritmos	tempo	BestScore	SLF	TLF	f1score	accuracy	Covergência
ABC	113.3104	4485.2962	0.65	0.8375	0.6766	0.9080	50
PSO-BNC	238.6005	4483.6577	0.6125	0.7938	0.6212	0.8902	75
SAGA	83.3069	4483.5827	0.7563	0.8438	0.7954	0.9438	75
MAGA-Tabu	27.5335	4482.4134	0.775	0.8813	0.8188	0.9509	95
GES	0.9936	4481.7679	0.875	0.875	0.9333	0.9821	100
GA	85.6390	4486.0891	0.6	0.8563	0.6164	0.8902	40
MAGA-Original	22.7800	4481.7679	0.7688	0.875	0.82	0.9518	100

Insurance 1k							
Algoritmos	tempo	BestScore	SLF	TLF	f1score	accuracy	Covergência
ABC	274.8502	16154.5426	0.1740	0.3279	0.2135	0.9046	100
PSO-BNC	70.9909	16161.2555	0.1625	0.3029	0.2032	0.9048	100
SAGA	9.2702	16172.1291	0.2212	0.375	0.2513	0.9025	100
MAGA-Tabu	54.9768	16071.1731	0.2221	0.3981	0.2502	0.9014	100
GES	47.2991	16152.3175	0.1731	0.2885	0.26866	0.9302	100
GA	37.8619	16157.8713	0.1711	0.3106	0.2228	0.9118	100
MAGA-Original	35.9742	16101.8141	0.1923	0.3231	0.2334	0.9063	100
Insurance 5k							
Algoritmos	tempo	BestScore	SLF	TLF	f1score	accuracy	Covergência
ABC	647.7005	69078.0864	0.4317	0.5981	0.4852	0.9321	100
PSO-BNC	1245.8259	69331.6395	0.3615	0.5827	0.3944	0.9175	60
SAGA	511.7043	69654.6095	0.4067	0.5923	0.4376	0.9225	55
MAGA-Tabu	274.8973	69042.9879	0.4259	0.6231	0.4576	0.9251	100
GES	139.6482	68999.9943	0.3653	0.5769	0.4471	0.9330	100
GA	244.5982	69067.8519	0.3962	0.6067	0.4338	0.9234	95
MAGA-Original	239.4597	69053.5384	0.4337	0.6106	0.4711	0.9278	100
Insurance 20k							
Algoritmos	tempo	BestScore	SLF	TLF	f1score	accuracy	Covergência
ABC	1945.0023	266282.0342	0.5951	0.7753	0.6180	0.9453	45
PSO-BNC	5075.6249	268752.2937	0.4692	0.6798	0.4725	0.9221	0
SAGA	2137.9321	269268.3410	0.5548	0.7442	0.5557	0.9337	15
MAGA-Tabu	1571.0058	266496.5387	0.5933	0.7942	0.6017	0.9416	65
GES	397.5513	266226.8635	0.4808	0.7308	0.4854	0.9245	0
GA	1809.2957	266963.0642	0.5692	0.7663	0.5781	0.9380	30
MAGA-Original	1540.1878	266168.4228	0.5865	0.7779	0.5996	0.9419	65
Insurance 50k							
Algoritmos	tempo	BestScore	SLF	TLF	f1score	accuracy	Covergência
ABC	3971.4246	659541.7774	0.6468	0.8117	0.6585	0.9501	40
PSO-BNC	11195.5600	664809.0019	0.5087	0.7298	0.4925	0.9217	0
SAGA	4890.5295	667764.2516	0.5048	0.7529	0.4824	0.9185	0
MAGA-Tabu	3799.5476	659717.1180	0.6404	0.8308	0.6412	0.9467	40
GES	740.6849	661017.3948	0.5192	0.7885	0.5192	0.9288	0
GA	4304.3217	660989.9571	0.5740	0.7885	0.5656	0.9344	5
MAGA-Original	3706.8881	659739.7541	0.6404	0.8240	0.6459	0.9479	35
Insurance 100k							
Algoritmos	tempo	BestScore	SLF	TLF	f1score	accuracy	Covergência
MAGA-Tabu	10193.2538	1312754.8067	0.6856	0.8375	0.6893	0.9539	25
MAGA-Original	7876.5623	1312740.5569	0.6913	0.8442	0.6978	0.9555	25
ABC	8858.0585	1313387.8649	0.6817	0.8471	0.6857	0.9533	20
Child 500							
Algoritmos	tempo	BestScore	SLF	TLF	f1score	accuracy	Covergência
ABC	164.5085	6711.2138	0.538	0.73	0.5983	0.9522	45
PSO-BNC	113.6823	6750.8955	0.432	0.65	0.4837	0.9391	100
SAGA	175.8472	6762.3931	0.582	0.738	0.6335	0.9555	40
MAGA-Tabu	91.9970	6674.4059	0.566	0.73	0.6271	0.9558	85
GES	29.8410	6673.9954	0.52	0.76	0.5778	0.95	100
GA	117.6246	6703.4779	0.51	0.726	0.5684	0.9488	60
MAGA-Original	64.0449	6656.8843	0.542	0.736	0.6094	0.9542	100

Child 2500							
Algoritmos	tempo	BestScore	SLF	TLF	f1score	accuracy	Covergência
ABC	323.4377	31339.4596	0.796	0.978	0.7820	0.9705	30
PSO-BNC	1267.8471	31386.6060	0.73	0.924	0.7256	0.9636	20
SAGA	379.3379	31797.6115	0.776	0.924	0.7512	0.9657	20
MAGA-Tabu	174.5485	31233.7361	0.888	1	0.8866	0.985	90
GES	41.2924	31326.9937	0.72	0.96	0.72	0.9632	0
GA	266.5349	31385.7382	0.75	0.97	0.7321	0.9634	15
MAGA-Original	198.9824	31290.0374	0.876	0.992	0.8708	0.9828	75
Child 10k							
Algoritmos	tempo	BestScore	SLF	TLF	f1score	accuracy	Covergência
ABC	688.7543	122985.6388	0.812	0.988	0.7969	0.9726	25
PSO-BNC	2462.1495	123215.1416	0.742	0.968	0.7016	0.9582	5
SAGA	826.7675	123953.1543	0.78	0.96	0.7419	0.9636	20
MAGA-Tabu	350.2287	122808.7055	0.89	1	0.8867	0.985	80
GES	63.7999	123086.0946	0.68	1	0.6415	0.95	0
GA	790.8635	123169.0290	0.748	0.974	0.7210	0.9616	5
MAGA-Original	315.1229	122813.6559	0.888	0.998	0.8850	0.9847	85
Child 20k							
Algoritmos	tempo	BestScore	SLF	TLF	f1score	accuracy	Covergência
ABC	1030.02447	306774.0982	0.838	1	0.8256	0.9764	45
PSO-BNC	3789.9494	307439.7849	0.764	0.996	0.7065	0.9574	15
SAGA	1317.2678	308226.7282	0.806	0.972	0.7675	0.9664	35
MAGA-Tabu	552.8166	306662.8740	0.912	1	0.9103	0.9882	90
GES	101.7146	307010.0789	0.68	1	0.6296	0.9474	0
GA	1383.1694	307502.6594	0.746	0.986	0.7060	0.9587	10
MAGA-Original	930.8182	306697.6253	0.884	1	0.8802	0.9841	80
Child 50k							
Algoritmos	tempo	BestScore	SLF	TLF	f1score	accuracy	Covergência
ABC	2297.9026	610401.5970	0.828	1	0.8097	0.9742	30
PSO-BNC	15633.8080	611042.1904	0.71	0.994	0.6425	0.9471	0
SAGA	1114.3928	611049.6629	0.87	0.988	0.8509	0.9793	70
MAGA-Tabu	811.7232	610168.8625	0.9	0.998	0.8983	0.9866	95
GES	150.8661	610729.4425	0.72	1	0.6923	0.9579	0
GA	2625.0735	610642.7034	0.812	1	0.7796	0.9692	20
MAGA-Original	871.3974	610109.8003	0.904	1	0.904	0.9874	100