SANDRO ROBERTO LOIOLA DE MENEZES

ANÁLISE DO PERCURSO DA QUALIDADE DE DIFERENTES MÉTRICAS PARA AGRUPAMENTO DE DADOS UTILIZANDO ALGORITMO PARALELO INSPIRADO EM ORGANISMOS SIMBIÓTICOS

Dissertação apresentada ao Programa de Pós-Graduação em Computação Aplicada da Universidade do Estado de Santa Catarina, como requisito parcial para obtenção do grau de Mestre em Computação Aplicada.

Orientador: Rafael Stubs Parpinelli
Coorientador: Rebeca Schroeder Freitas

JOINVILLE, SC
2016

M534a Menezes, Sandro Roberto Loiola
Análise do percurso da qualidade de diferentes métricas para agrupamento de dados utilizando algoritmo paralelo inspirado em organismos simbióticos /Sandro Roberto Loiola Menezes. – 2016.
89 p. : il. ; 21 cm

Orientador: Rafael Stubs Parpinelli
Bibliografia: 85-89 p.
Dissertação (mestrado) – Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Computação Aplicada, Joinville, 2016.

1. Banco de dados. 2. Exploração de dados (técnica). 3.Algoritmos de computador.
I. Parpinelli, Rafael Stubs. II. Universidade do Estado de Santa Catarina. Programa de Pós-Graduação em Computação Aplicada. III. Título.

CDD: 005.74 - 23. ed.

"Análise do Percurso da Qualidade de Diferentes Métricas para Agrupamento de Dados Utilizando Algoritmo Paralelo Inspirado em Organismos Simbióticos"

Por

Sandro Roberto Loiola de Menezes

Esta dissertação foi julgada adequada para a obtenção do título de

Mestre em Computação Aplicada

área de concentração em "Ciência da Computação",
e aprovada em sua forma final pelo

Mestrado Acadêmico em Computação Aplicada

CENTRO DE CIÊNCIAS TECNOLÓGICAS DA

UNIVERSIDADE DO ESTADO DE SANTA CATARINA.

Banca Examinadora:

Orientador:

Dr. Rafael Stubs Parpinelli
CCT/UDESC

Membros:

Dr. Daniel Gomes Ferrari
TalkDesk - Portugal

Dr. Guilherme Piêgas Koslovski
CCT/UDESC

Dra. Rebeca Schroeder Freitas
Co-orientadora

Joinville, 03 de novembro de 2016.

Dedico a Deus a realização deste trabalho por conduzir meus pensamentos nos momentos mais difíceis. Dedico a minha querida esposa Vladia Medrado Mendes de Brito de Menezes e meu filho José Lobo de Menezes Neto. Dedico aos meus pais José Lobo de Menezes e Judite Loiola de Menezes. Dedico também a minha afilhada Carla Lorena da Costa Lobo além de meus irmãos, tios e primos. Dedico a memória de meus avós maternos Rita Dantas Loiola e Antônio Loiola Neto como também aos meus avós paternos Areovaldo da Silva Menezes e Azinda Lobo de Menezes.

Agradecimentos

Primeiramente agradeço a Deus por conduzir meus pensamentos e ações na direção correta para desenvolver esse trabalho. Nos momentos mais difíceis Ele sempre se mostrou presente nesse desafio. Não sei como traduzir em palavras a gratidão que tenho por minha esposa, Vladia Medrado Mendes de Brito de Menezes, por me apoiar, compreender e incentivar ao longo de todo o curso. Agradeço ao meu filho, José Lobo de Menezes Neto, por proporcionar momentos de felicidades que me ajudaram a superar barreiras. Não poderia de deixar de agradecer aos meus pais José Lobo de Menezes e Judite Loiola de Menezes por todas as palavras de superação e incentivo nos momentos que pensei em desistir.

Gostaria de agradecer de forma especial ao meu orientador Prof. Dr. Rafael Stubs Parpinelli por acreditar em mim e aceitar ser meu orientador. Tenho muita gratidão, admiração e respeito pelo professor. Além da orientação acadêmica sou grato pelas orientações que levarei para outras áreas da vida.

Sou grato também a minha coorientadora Profa. Dra. Rebeca Schroeder Freitas pela orientações que foram muito importantes para o desenvolvimento desse trabalho. Agradeço também a todos os professores do PPGCA que tive a oportunidade de conhecer, em especial ao Prof. Dr. Guilherme Piegas Koslovski, Prof. Dr. Charles Christian Miers e a Profa. Dra. Avanilde Kemczinski. Agradeço também aos integrantes do grupo CoCa e Colmeia.

Gostaria de Agradecer a Ana Cristina Calegari Correa e Liriane Zimmermann Bittencourt pelo incentivo e apoio para conciliar o curso com o trabalho.

Agradeço a todas as pessoas que de qualquer forma influenciaram positivamente no desenvolvimento desse trabalho.

"O sucesso nasce do querer, da determinação e persistência em se chegar a um objetivo. Mesmo não atingindo o alvo, quem busca e vence obstáculos, no mínimo fará coisas admiráveis."(José de Alencar)

RESUMO

MENEZES, Sandro. **Análise do Percurso da Qualidade de Diferentes Métricas para Agrupamento de Dados Utilizando Algoritmo Paralelo Inspirado em Organismos Simbióticos**. 2016. 89 p. Dissertação (Mestrado em Computação Aplicada - Área: Sistemas de Computação). Universidade do Estado de Santa Catarina. Programa de Pós-Graduação em Computação Aplicada. Joinville, 2016.

Algoritmos para mineração de dados tem se destacado em várias áreas de aplicação devido ao auxílio na tomada de decisão embasada no conhecimento extraído de um conjunto de dados. Entretanto, realizar tarefas de mineração de dados, como agrupamento, no cenário de massas de dados é dificultoso devido às suas características complexas, como o grande volume e alta dimensionalidade. Este trabalho propõe uma abordagem, *MRCSOS*, para realizar agrupamento de dados utilizando o algoritmo Inspirado em Organismos Simbióticos (*SOS*) projetado na infraestrutura fornecida pelo *Apache Hadoop MapReduce*. Essa combinação agrega trazidos pela técnica, como exploração eficiente na busca pela melhor solução, conciliada a escalabilidade e paralelismo fornecidas pelo *Hadoop*. A principal contribuição deste trabalho é a análise de correlação dos resultados de pureza, entropia e diversidade genotípica usando diferentes métricas de agrupamento de dados no decorrer do processo de otimização. Com a análise dos resultados dos experimentos com três bases de dados foi evidenciado que algumas métricas de agrupamento não conseguem conservar a qualidade do agrupamento no decorrer do processo. Nesses casos, o agrupamento de dados final possui qualidade inferior a qualidade obtida em iterações anteriores. Dentre as métricas analisadas a função *f5*, ou função Silhueta, se sobressai em relação as demais. A mesma consegue conservar o agrupamento com melhor qualidade do início ao fim do processo de otimização. Além disso, *MRCSOS* utilizando a função *f5* obteve resultados com qualidades superiores ou competitivas aos resultados de outras abordagens já existentes, como *MRCPSO* e *MRCGSO*, e melhores que o *K-Means*.

Palavras-chaves: Agrupamento de Dados; *Hadoop MapReduce*; Algoritmos Bio-Inspirados; Algoritmo Inspirado em Organismos Simbióticos.

ABSTRACT

Algorithms for data mining have being highlighted in several application areas, helping the decision making process based on knowledge extracted from datasets. However, data mining tasks in the scenario of massive datasets, such as clustering, are difficult because of its complexity like the large volume of data and high dimensionality. This dissertation proposes an approach to perform data clustering using the Symbiotic Organisms Search (SOS) algorithm using the infrastructure provided by Apache the Hadoop MapReduce, named MRCSOS. This combination provides an efficient exploration of the search space and the scalability with parallelism provided by the Hadoop. The main contribution of this work is the correlation analysis of purity, entropy and genotypic diversity using different metrics of data clustering during the optimization process. The results obtained with three different datasets showed that some clustering metrics did not maintain the clustering quality during the optimization process. In these cases the final data clustering got worst quality than the quality obtained in previous iterations. Among the analyzed metrics, the Silhouette function (F5) was better than others. This function can maintain the best cluster through the whole optimization process. Furthermore, MRCSOS using the function F5 got better results, or at least competitive, when compared with other existing approaches in literature.

Key-words: Data Clustering, Hadoop MapReduce, Bio-Inspired Algorithms, Symbiotic Organisms Search.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ACD	Arquivo de *Cache* Distribuído
ACO	Otimização por Colônia de Formigas
AD	Arquivo de Dados
DE	Evolução Diferencial
GA	Algoritmos Genéticos
GP	Programação Genética
GSO	Otimização por Enxame de Vermes Luminescentes
HDFS	*Hadoop Distributed File System*
LSI	*Latent Semantic Indexing*
MDG	Medida da Diversidade Genotípica
MOA	*Massive Online Analysis*
MRCGSO	*Map Reduce Clustering Glowworm Swarm Optimization*
MRCSOS	*Map Reduce Clustering Symbiotic Organisms Search*
MRCPSO	*Map Reduce Clustering Particle Swarm Optimization*
NMD	Normalizador da Métrica da Diversidade
NC	Número de Centróides
PSO	Otimização por Enxame de Partículas
QI	Quantidade de Iterações
SOS	Otimização por Organismos Simbióticos
SSE	*Sum Squared Errors*
TCP	*Transmission Control Protocol*
TP	Tamanho da População
UCI	*Machine Learning Repository*
VM	Vetor Mutualismo

Sumário

1 INTRODUÇÃO

Técnicas de mineração de dados têm recebido grande atenção por parte da comunidade científica devido aos benefícios que podem produzir em diversas áreas como Educação, Bioinformática, Economia, Negócios, entre outras. Tais benefícios podem ser obtidos através da descoberta de padrões, correlações e tendências de informações sendo que esses padrões podem ser úteis na tomada de decisão em um determinado processo (SACHIN; VIJAY, 2012). Na área de Negócios, por exemplo, dados de compras de clientes podem ser utilizados como fonte de mineração para obter um padrão de compras e assim embasar o planejamento estratégico de mercado (CHUNG; CHUNG, 2013). Na área de segurança de redes, estas técnicas podem ser utilizadas para auxiliar na detecção de invasão de intrusos (ALJARAH; LUDWIG, 2013). No ramo das Ciências Biológicas, a utilização do processo de mineração contribui na descoberta de novos tipos de câncer (BHAVANI; SADASIVAM; KUMARAN, 2011). Na Bioinformática, bases de dados com informações de DNA podem ser mineradas obtendo assim identificação de espécies (DAOUDI et al., 2014). Na Educação, novas técnicas de aprendizado e de avaliação de estudantes tem sido desenvolvidas utilizando técnicas de mineração (PATHAN et al., 2014). Além dessas áreas, técnicas de mineração de dados vem sendo introduzidas em aplicações de segurança nacional e segurança cibernética (THURAISINGHAM, 2009).

Nas últimas décadas o volume de dados gerado e armazenado vem aumentando rapidamente. A tendência é que o volume de bancos de dados corporativos aumente em 40% a cada ano (TALLON, 2013). Entre outros fatores, os avanços e o barateamento das tecnologias de coleta e armazenamento possibilitaram as aplicações computacionais fornecerem funcionalidades como operações de gravação e recuperação de dados com uma capacidade superior ao fornecido anteriormente. Além disso, o rápido desenvolvimento de redes e o aumento da capacidade de coleta de dados são outros fatores que influenciaram na tendência de aumento do volume em várias fontes de dados como mídias sociais, sensores, arquivo de *logs*, dentre outras (JAFAR, 2010) (TALLON, 2013). Portanto, existe uma vasta área de massa de dados que algoritmos de mineração de dados podem explorar e, entre estes, existe a tarefa de agrupamento de dados. Agrupamento de dados é uma tarefa de mineração que busca obter padrões para inferir conhecimento (FAHAD et al., 2014). O agrupamento consiste em dividir um conjunto de objetos de dados em diferentes grupos de acordo com a similaridade, onde cada grupo possui dados com características bem próximas.

Entretanto, existem alguns contextos de aplicação em que o processo de mineração de dados fica impossibilitado ou se torna inviável devido à existência de algum requisito que não é atendido de forma eficiente pelas tecnologias convencionais. Por exemplo, no contexto de grandes volumes de dados, um processo de mineração pode ocasionar em

alto consumo de tempo de processamento. Outro fator limitante é a dimensionalidade dos dados que influencia diretamente no desempenho do algoritmo: quanto maior a dimensionalidade de dados, maior será o tempo de processamento (KATAL; WAZID; GOUDAR, 2013). Além disso, as técnicas convencionais de mineração de dados precisam realizar a carga dos dados que serão analisados na memória antes de realizar a mineração (LESKOVEC; RAJARAMAN; ULLMAN, 2014). No contexto de grandes volumes de dados, realizar esse carregamento na memória é mais um ponto crítico, pois delimita a capacidade máxima de dados que serão minerados em um determinado momento em uma estação de processamento.

Para realizar mineração em grandes massas de dados, é necessário que o processamento seja realizado em uma infraestrutura de processamento escalável, para que seja possível realizar a alocação de recursos de acordo com a demanda requerida pelo volume de dados a ser processado. A escalabilidade pode ser vertical ou horizontal. A escalabilidade vertical é obtida através do incremento de mais recursos computacionais (núcleos, memória, processamento) em um servidor original, incrementando assim o poder de processamento (YU et al., 2007)(MICHAEL et al., 2007). Já a escalabilidade horizontal é obtida através do incremento de um grande número de servidores menores e mais baratos, fortemente interconectados em formato de *cluster* (YU et al., 2007)(MICHAEL et al., 2007). A escolha desse tipo de escalabilidade vem crescendo pois é mais acessível e possui melhor desempenho de processamento em relação à escalabilidade vertical (YU et al., 2007). Assim, a linha de pesquisa deste trabalho considera o contexto de escalabilidade horizontal. Utilizando escalabilidade horizontal é possível realizar processamento distribuído e paralelo. Diante disso, é necessário utilizar técnicas com características mais adequadas ao processamento em paralelo, como é o caso de algoritmos bio-inspirados.

Algoritmos bio-inspirados pertencem a uma vertente da Computação conhecida como Computação Natural, que utiliza a natureza como fonte de inspiração para desenvolver técnicas computacionais não-determinísticas para resolver problemas complexos de otimização (JR et al., 2013). Inteligência de Enxame e Computação Evolucionária são exemplos de paradigmas que propõem algoritmos bio-inspirados. Inteligência de Enxame é um conjunto de técnicas de pesquisa e otimização que são inspirados no comportamento social de alguns organismos vivos como formigas, abelhas, pássaros, baratas, dentre outros, que vivem em sociedade (PARPINELLI; LOPES, 2011). Já a Computação Evolucionária é inspirada na teoria da evolução de Darwin que explora conceitos de seleção natural e sobrevivência dos mais bem adaptados ao ambiente (HOLZINGER et al., 2014). Como exemplo desse paradigma pode-se citar o algoritmo Inspirado em Organismos Simbióticos (*Symbiotic Organisms Search - SOS*) que é baseado em relações simbióticas entre organismos distintos dentro de um ecossistema (CHENG; PRAYOGO, 2014). Inteligência de

Enxame e Computação Evolucionária englobam abordagens populacionais que processam um conjunto de indivíduos no qual cada indivíduo representa uma possível solução para o problema. Essa característica populacional tem contribuído na resolução de problemas de grande escala, alta dimensionalidade e multiobjetivos (CHENG et al., 2013). Outra característica vantajosa desses paradigmas bio-inspirados é o não determinismo aliado com a intensificação e diversificação. Isso implica em uma exploração mais eficiente do espaço de busca e em diferentes regiões (RAJARAMAN; VAIDYANATHAN; CHOKKALINGAM, 2015). Mineração em massas de dados envolve espaço de busca de solução bastante extenso e altamente não-linear. Nesse caso, a diversificação pode beneficiar a busca da melhor solução. Soluções determinísticas tendem a falhar quando aplicadas em problemas em que o espaço de busca é muito grande (BINITHA; SATHYA, 2012) (YANG et al., 2013). A mineração de dados pode ser vista como um problema de otimização onde busca-se encontrar a melhor solução, como os valores que representam os grupos de dados resultantes de uma tarefa de agrupamento, em um determinado tempo limite. Algoritmos Bio-inspirados são técnicas de pesquisa e otimização que possuem características adequadas para análise de dados (CHENG et al., 2013).

No contexto de mineração de massas de dados, é necessário embasar a técnica escolhida para realizar a mineração em uma infraestrutura que dê suporte de armazenamento e processamento paralelo. Ferramentas convencionais como Sistemas de Gerenciamento de Banco de Dados Relacional não possuem disponibilidade e desempenho eficientes nesse contexto (SIVARAMAN; MANICKACHEZIAN, 2014) (DOSHI et al., 2013). Tolerância a falhas é outra questão crucial para técnicas que manipulam dados intensivos executados em paralelo (KUTLU; AGRAWAL; KURT, 2012). Assim, é imprescindível que esse requisito esteja presente na infraestrutura de processamento.

Entre as ferramentas existentes, *Hadoop* é um projeto *open source* criado pela *Apache Software Fundation* que vem sendo utilizado como ferramenta no contexto de mineração de dados. A utilização dessa ferramenta é justificada pelo fornecimento de um *framework* de processamento paralelo, fácil escalabilidade e utilização, *Hadoop MapReduce*, além de usar um sistema de arquivos distribuído e com alta taxa de transferência (*Hadoop Distributed File System (HDFS))* (MOHAMMED; FAR; NAUGLER, 2014).

Dessa forma, a principal discussão apresentada neste trabalho foca em soluções de agrupamento de dados utilizando algoritmos bio-inspirados projetados na infraestrutura *Hadoop MapReduce*. Também serão discutidos aspectos relacionados com a análise da qualidade de agrupamento de dados relacionados com a técnica e também relacionados com a análise de escalabilidade fornecida pela ferramenta.

1.1 MOTIVAÇÃO

O aumento de volume e dimensionalidade de dados dificultam a realização de mineração de dados devido a necessidade de escalabilidade, tratamento de falhas e técnicas de mineração mais adequadas a esse cenário. Diante da tendência de expansão deste cenário, a principal motivação deste trabalho é contribuir com uma abordagem de agrupamento de dados que atendam requisitos de infraestrutura escalável conciliado com técnica eficiente de agrupamento. Para isso, este trabalho revisa abordagens de mineração de dados que utilizam algoritmos bio-inspirados embasados na infraestrutura *Hadoop* para desenvolver uma abordagem nessa mesma linha de pesquisa além de analisar os resultados obtidos. A escolha do algoritmo *SOS* é justificada pelo bom desempenho obtido no domínio de otimização de funções contínuas (CHENG; PRAYOGO, 2014). A hipótese a ser verificada é se, no domínio de mineração de dados a abordagem proposta embasada no *SOS* é capaz de obter resultados competitivos aos das abordagens existentes. A melhoria da qualidade do agrupamento de dados também motiva este trabalho a entender e discutir o curso do nível de qualidade do agrupamento através da análise de diferentes funções à medida que o processo de agrupamento evolui.

1.2 OBJETIVO

O objetivo geral deste trabalho é desenvolver uma abordagem de agrupamento de dados embasada numa infraestrutura que forneça paralelismo, e que utilize uma técnica de agrupamento eficiente. Também objetiva analisar o curso da qualidade dos agrupamentos obtidos aplicando diferentes métricas de agrupamento. A abordagem desenvolvida é composta pelo algoritmo *(SOS)* projetado na infraestrutura fornecida pelo *Hadoop* chamado de Agrupamento por Organismos Simbióticos *MapReduce*, *MapReduce Clustering Symbiotic Organisms Search (MRCSOS)* (MENEZES; PARPINELLI, 2016b).

Para atingir o objetivo geral, alguns objetivos específicos são traçados:

- Verificar a qualidade dos resultados de agrupamento obtidos por *MRCSOS* em relação a outros algoritmos bio-inspirados. Dessa forma, será possível verificar se a qualidade dos resultados obtidos são melhores que os existentes.

- Avaliar a qualidade das diferentes funções de agrupamento aplicadas no *MRCSOS* durante o processo de otimização. Com isso pretende-se verificar se o curso da qualidade do agrupamento é estável, ou seja, sempre melhora com a evolução do sistema ou se ocorrem instabilidades.

- Analisar o desempenho do algoritmo *SOS* dentro da arquitetura *MapReduce*. Esse objetivo visa verificar o nível de escalabilidade do proposto.

1.3 ORGANIZAÇÃO DO TRABALHO

Este trabalho está estruturado da seguinte maneira. O Capítulo 2 apresenta os fundamentos para o melhor entendimento deste trabalho e a revisão da literatura. O Capítulo 3 descreve o método utilizado para desenvolver a abordagem *MRCSOS* além das funções e métricas empregadas na mesma. O Capítulo 4 apresenta configurações de ambiente, parâmetros e bases de dados utilizadas nos experimentos além da análise dos resultados obtidos. O Capítulo 5 apresenta as considerações finais e trabalhos futuros.

2 REVISÃO DA LITERATURA

Este capítulo apresenta alguns conhecimentos básicos sobre mineração de dados, algoritmos bio-inspirados e sobre o *framework Hadoop* para melhor compreensão sobre o tema deste trabalho. Em seguida, são descritos alguns trabalhos encontrados na literatura que utilizam algoritmos bio-inspirados projetados na arquitetura *Hadoop MapReduce* para realizar mineração de dados. Além disso, diante da análise desses trabalhos, três modelos de projetos *MapReduce* foram identificados e são exibidos nesse capítulo.

Na primeira seção a seguir serão apresentados alguns conhecimentos sobre mineração de dados, visto que é a classe de problemas que a abordagem *MRCSOS* é aplicada.

2.1 MINERAÇÃO DE DADOS

Mineração de Dados é um processo de análise de conjuntos de dados a fim de encontrar relacionamentos inesperados e de resumir os dados de uma forma que eles sejam tanto úteis quanto compreensíveis (HAND; MANNILA; SMYTH, 2001). O processo de análise envolve técnicas matemáticas, estatísticas e computacionais. Processos ou tarefas no domínio de mineração de dados visam construir modelos eficientes capazes de analisar e extrair conhecimento além de predizer tendências futuras no comportamento dos dados. Tarefas de mineração de dados podem ser classificadas em duas categorias: preditiva e descritiva. Modelos preditivos aprendem através da análise de um conjunto de dados, ou dados de treinamento, e fazem predições no comportamento de novos dados. Já técnicas descritivas proveem sumarização de dados. Dentre as tarefas mais comuns de mineração de dados é possível citar: Classificação, Agrupamento, Associação e Regreção (GHOSH; NATH, 2004). As tarefas de Classificação e Agrupamento são descritas na sequência por serem abordadas nos trabalhos revisados. A tarefa de Agrupamento será mais detalhada por se tratar do problema de mineração tratado nesse trabalho.

A classificação é uma tarefa preditiva que infere a determinação de qual classe um novo dado está classificado. Basicamente, essa tarefa divide-se em duas fases. Na primeira fase ocorre o processo de aprendizagem. Nesta fase, o modelo preditivo que define qual classe um determinado dado pertence é construído a partir da análise de um conjunto de dados, ou dados de treinamento. Na maioria das vezes os dados de treinamento são rotulados, ou seja, cada dado já possui a informação correta de qual classe ele pertence (atributo meta ou descritor alvo). Nesses casos a classificação é dita como supervisionada. Na segunda fase ocorre a avaliação do modelo em dados de teste, disjunto dos dados de treinamento, para encontrar a precisão do padrão de classificação (GHOSH; NATH, 2004) (KOTURWAR; GIRASE; MUKHOPADHYAY, 2015). Aplicar essa tarefa em grandes massas de dados com número muito grande de atributos e com características redundantes

pode trazer prejuízos à precisão de classificação (LIU; MOTODA, 1998). Desta maneira, é comum antes de realizar a análise para gerar o modelo de classificação executar uma etapa de seleção de atributos que serão considerados pelo algoritmo de aprendizagem (GUYON; ELISSEEFF, 2003).

2.1.1 Agrupamento

A tarefa de agrupamento tem como principal objetivo dividir um conjunto de objetos de dados em diferentes grupos (ALJARAH; LUDWIG, 2012). Normalmente os dados utilizados nessa tarefa são armazenados em tabelas, nas quais os objetos são representados pelas linhas e os atributos pelas colunas. Dessa forma, o conjunto de tabelas forma uma base de dados (FERRARI et al., 2014). Seja D uma base de dados, o agrupamento consiste em encontrar partições de D, denotadas por $\tau(D)$, de tal forma que $\tau(D) = \{c_1, c_2, ..., c_k\}$, onde cada c_i representa um centroide e k representa o número de centroides. Não deve existir grupo vazio e os mesmos devem ser disjuntos. Além disso, a união dos grupos deve formar D (DAOUDI et al., 2014).

Cada grupo é formado por objetos que possuem uma maior similaridade. Dentre as medidas de similaridade, ou medida de distância, pode-se citar a distância *Euclidiana* e a distância de *Manhattan* (ALJARAH; LUDWIG, 2013). A escolha da distância utilizada na abordagem de agrupamento pode depender da base de dados minerada. Para determinar se a técnica utilizada no processo de agrupamento é eficiente deve-se mensurar a qualidade dos grupos formados. Essa medida de qualidade pode ser avaliada conforme dois critérios: compactação e separação. A compactação representa quanto os objetos de um mesmo grupo estão próximos entre sí. Quanto menor a distância, melhor a compactação. A distância entre objetos de um mesmo grupo é chamada de *intragrupo*. O critério separação representa quanto os diferentes grupos estão distantes entre sí (*intergrupo*). Quanto maior a distância melhor (FERRARI et al., 2014). Dessa forma, a similaridade interna do grupo deve ser maximizada. Quanto menor for a similaridade de objetos de grupos diferentes, maior será a qualidade do agrupamento, logo essa similaridade deve ser minimizada (ALJARAH; LUDWIG, 2012).

Os dados agrupados podem estar rotulados ou não. Caso estejam rotulados o agrupamento da base de dados é *a priori*, pois o rótulo informa qual grupo aquele objeto pertence antes da mineração. Agrupamento de dados é uma tarefa não supervisionada. Assim, mesmo se o objeto de dado for rotulado, na grande maioria dos casos o rótulo não é utilizado no processo de agrupamento. O rótulo normalmente é utilizado para comparar o desempenho entre diferentes algoritmos (FERRARI et al., 2014). O desempenho dos algoritmos estão diretamente relacionados com as funções de avaliação, ou função *fitness*, utilizadas pelos mesmos. A próxima subseção cita algumas funções de avaliação

de agrupamento utilizadas tanto por alguns trabalhos revisados quanto por este trabalho.

2.1.1.1 Funções *fitness*

Uma das análises deste trabalho é verificar a influência de cinco diferentes funções *fitness* na qualidade do agrupamento formado. Todas as cinco funções utilizam índices internos, ou seja, utilizam os próprios objetos (seus atributos) e similaridade em suas equações (FERRARI et al., 2014). Quatro dessas funções referenciadas por *f1*,*f2*,*f3*,*f4* estão disponíveis em (ALJARAH; LUDWIG, 2013). A função *f5* é a versão simplificada da função de Silhueta disponível em (HRUSCHKA; CAMPELLO; CASTRO, 2006 apud FERRARI et al., 2014). Os cálculos comuns para as funções *f1*,*f2*,*f3*, como o cálculo da distância intragrupo definida na Equação (2.1), são expostos a seguir.

$$IntraD_j = \sum_{i=1}^{|cr_j|} Distance(cr_{ji}, c_j) \tag{2.1}$$

Onde cr_{ji} representa os itens de dados pertencente ao centroide c_j e $|cr_j|$ representa a quantidade de itens que pertencem ao centroide c_j. A distância utilizada nas Equações (2.1), (2.2) e (2.3) é a Euclidiana. A distância intergrupo é calculada conforme Equação (2.2).

$$InterDist = \sum_{i=1}^{k} \sum_{j=i}^{k} (Distance(c_i, c_j))^2 \tag{2.2}$$

Onde k é o número de centroides e c_i, c_j representam os centroides do organismo. O cálculo do erro, *Sum Squared Errors (SSE)*, também é realizado e considerado em algumas funções. A equação (2.3) define *SSE*.

$$SSE = \sum_{j=1}^{k} \sum_{i=1}^{|C_j|} (Distance(c_i, c_j))^2 \tag{2.3}$$

Onde $|C_j|$ representa a quantidade de itens cobertos pelo centroide c_j, e c_i representam itens cobertos por c_j. Enfim, a função *f1* é definida pela Equação (2.4), *f2* conforme Equação (2.5) e *f3* conforme Equação (2.6).

$$f1 = \frac{InterDist}{SSE} \times \frac{intraD_j}{max(intraD_j)} \tag{2.4}$$

$$f2 = \frac{InterDist}{\frac{intraD_j}{max(intraD_j)}} \tag{2.5}$$

$$f3 = \frac{1}{SSE \times \frac{intraD_j}{max(intraD_j)}} \tag{2.6}$$

As funções *f1* e *f2* consideram a distância intergrupo em suas equações, sendo que a função *f2* dá um peso maior à distância intergrupo. As três funções consideram a maior

distância intragrupo em suas equações. A distância intragrupo apresentada nas funções acima representa a média dessa distância de todos os grupos da solução. Além disso, as três funções são de maximização, ou seja, quanto maior seu valor melhor é a solução representada pela solução candidata.

A função *f4* é uma função de minimização e não utiliza nenhuma das equações apresentadas anteriormente sendo definida conforme Equação (2.7).

$$f4 = \frac{\sum_{j=1}^{k} \frac{\sum_{i=1}^{n_j} Distance(R_i, C_j)}{n_j}}{k} \tag{2.7}$$

Onde k é o número de centroides, n_j é o número de itens cobertos pelo centroide C_j e R_i representa os itens cobertos por C_j. A distância utilizada na Equação (2.7) é a distância de *Manhattan*. Essa função é de minimização, ou seja, quanto menor seu valor melhor é a solução.

A função *f5* é uma função de maximização e pode ser definida conforme Equação (2.8) (HRUSCHKA; CAMPELLO; CASTRO, 2006 apud FERRARI et al., 2014).

$$f5 = \frac{1}{n} \sum_{i=1}^{n} \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \tag{2.8}$$

Onde n é o número de itens de dados, *a(i)* é a distância do item i até o centroide do grupo que esse item pertence e *b(i)* é a distância do item i até os centroides dos outros grupos. A distância utilizada é a distância Euclidiana. Os resultados dessa função variam no intervalo [-1, 1]. Por ser uma função de maximização, 1 é o melhor resultado que pode ser obtido.

Para que seja possível avaliar qual das funções possui melhor qualidade no agrupamento é necessário definir as métricas de qualidade. Na próxima seção são descritas as métricas de qualidade e de desempenho utilizadas para avaliar os resultados obtidos.

2.1.1.2 Métricas de Qualidade e de Desempenho

Para o melhor entendimento da definição das métricas de qualidade é necessário saber que um grupo de dados possui um conjunto de itens de dados. Cada item pertence a uma classe de dados pré-estabelecida. Dessa forma, um grupo de dados pode possuir mais de uma classe de dados. Entretanto, um agrupamento de boa qualidade é aquele no qual cada grupo de dados possui muitos itens de uma mesma classe, com o menor número de classes. As métricas de qualidade descritas a seguir utilizam índices externos, ou seja, utiliza informações das classes de dados pré-estabelecidas em suas equações (FERRARI et al., 2014).

A métrica de pureza *FScore* (ZHAO; KARYPIS, 2002) é utilizada para avaliar a qualidade dos grupos formados. A mesma utiliza a informação do rótulo do dado, ou classe,

na realização do cálculo da pureza. Dessa forma, a base de dados precisa ser rotulada para utilizar essa métrica. Alguns trabalhos relacionados, *MRCGSO* (AL-MADI; ALJARAH; LUDWIG, 2014) e *MRCPSO* (ALJARAH; LUDWIG, 2012), utilizaram essa métrica de pureza para realizar a análise comparativa de resultados.

Dada uma classe de dados representada por L_r, o seu tamanho é referenciado por n_r. Para um grupo identificado por S_i e tamanho n_i, suponha n_i^r registros no grupo S_i pertencente a L_r, então o valor F dessa classe nesse grupo é definida pela Equação (2.9) (ZHAO; KARYPIS, 2002).

$$F(L_r, S_i) = \frac{2 * R(L_r, S_i) * P(L_r, S_i)}{(R(L_r, S_i) + P(L_r, S_i))} \tag{2.9}$$

Onde $R(L_r, S_i) = n_i^r/n_r$ e $P(L_r, S_i) = n_i^r/n_i$. O $FScore$ da classe L_r é o máximo valor F encontrado em qualquer grupo, conforme Equação (2.10).

$$FScore(L_r) = max_{S_i \in T} F(L_r, S_i) \tag{2.10}$$

Enfim, a pureza de todo agrupamento é calculada considerando a média de todas as classes, conforme a Equação (2.11), onde c é o número total de classes. Uma solução de agrupamento perfeita é aquela cujo resultado da Equação (2.11) é igual a 1 (ZHAO; KARYPIS, 2002). Isso indica que existe pelo menos um grupo de cada classe totalmente homogêneo.

$$FScore = \sum_{r=1}^{c} (\frac{n_r}{n}) FScore(L_r) \tag{2.11}$$

A segunda métrica de qualidade, a entropia, pode ser definida na Equação (2.12) (ALJARAH, 2013).

$$Entropia = \sum_{j=1}^{k} \frac{|C_j|}{n} E(C_j) \tag{2.12}$$

Onde C_j contém todos os itens de dados que pertencem ao grupo j do agrupamento, n representa o total de itens de toda a base de dados, k é o número de grupos e $E(C_j)$ representa a entropia individual do grupo j definida na Equação (2.13). Onde L_i representa dados de uma determinada classe i, q é o número de classes.

$$E(C_j) = -\frac{1}{\log q} \sum_{i=1}^{q} \frac{|C_j \bigcap L_i|}{|C_j|} \log(\frac{|C_j \bigcap L_i|}{|C_j|}) \tag{2.13}$$

Definidas as métricas de qualidade dos grupos, é necessário avaliar o desempenho do *MRCSOS* com relação a escalabilidade no uso dos recursos computacionais. Para isso, as métricas *Speedup*, definida na Equação (2.14) (ALJARAH; LUDWIG, 2012 apud HUANG

et al., 2006), e de eficiência definida na Equação (2.15) (GRAMA; GUPTA; KUMAR, 1993) são utilizadas.

$$Speedup = \frac{T_2}{T_n} \tag{2.14}$$

Na equação 2.14, T_2 representa o tempo de execução considerando dois computadores e T_n é o tempo de execução considerando n computadores. Além disso, n é múltiplo de 2. Nesse trabalho foi utilizado uma variação do *speedup*. O *speedup* inicial é com 2 computadores e o final com 8 computadores.

$$Eficiência = \frac{Speedup}{p} \tag{2.15}$$

Onde p é o número de computadores do *cluster* - 1. Esse decremento representa o servidor *Hadoop* que não processa dados na mineração.

A técnica utilizada para realizar agrupamento de dados utilizando a arquitetura *Hadoop MapReduce,MRCSOS* , é embasada no algoritmo inspirado em organismo simbiótico. A secção a seguir abordará algoritmos bio-inspirados além de detalhar o algoritmo inspirado em organismos simbióticos.

2.2 ALGORITMOS BIO-INSPIRADOS

Algoritmos bio-inspirados tem sido utilizados como métodos de otimização para problemas complexos normalmente não estacionários e multidimensionais (SMUTNICKI, 2010). A natureza tem sido utilizada como fonte de inspiração para o desenvolvimento de algoritmos de otimização sendo assim denominados de Algoritmos Bio-Inspirados. Segundo (JR et al., 2013), essa categoria de algoritmos pode ser subdividida em quatro grupos. São eles: Computação Evolucionária, Inteligência de Enxame, Redes Neurais Artificiais e Sistemas Imunológicos Artificiais. Inteligência de Enxame e Computação Evolucionária possuem a semelhança de representarem meta-heurísticas estocásticas e serem baseadas em população de possíveis soluções, chamadas de indivíduos.

Em uma população de indivíduos, cada indivíduo representa uma potencial solução do problema que está sendo otimizado. A população de indivíduos tende a mover-se para áreas onde estão localizadas as melhores soluções considerando todo o espaço de solução de um determinado problema. Além disso, Inteligência de Enxame e Computação Evolucionária mantém e sucessivamente melhoram uma população de potenciais soluções até que alguma condição de parada seja satisfeita. A métrica utilizada para determinar o quanto uma solução é boa para um determinado problema é uma função de eficiência ou função *fitness*. Através dessa função é possível determinar qual é o melhor indivíduo da população em um determinado momento (CHENG et al., 2013).

Inteligência de Enxame é um conjunto de técnicas de pesquisa e otimização que são inspiradas no comportamento social de alguns organismos vivos como formigas, abelhas, pássaros, baratas, dentre outros. Auto-organização e controle descentralizado são características essenciais da Inteligência de Enxame. Essas características conduzem ao comportamento emergente que surge através de interações locais entre componentes do sistema e não pode ser obtido por um componente atuando sozinho (PARPINELLI; LOPES, 2011). Inteligência de Enxame possui algumas meta-heurísticas inspiradas na natureza. Otimização por Enxame de Partícula, Otimização por Colônia de Formigas e Otimização por Enxame de Vermes Luminescentes são exemplos de algoritmos pertencentes a Inteligência de Enxame que foram abordados nos artigos revisados (Seção 2.4).

A Computação Evolucionária é outro paradigma inserido na Computação Natural e baseia-se na teoria de Charles Darwin para simular o processo evolutivo. A teoria de Darwin apresenta o processo de seleção natural que ocorre na natureza evidenciando a sobrevivência dos mais bem adaptados (ZELINKA et al., 2014). No contexto computacional, cada indivíduo representa uma solução candidata do espaço de solução do problema. Computação Evolucionária é composta por vários subprocessos inspirados na evolução das espécies. São eles: seleção, reprodução, combinação ou *crossover* e mutação. Algoritmos Genéticos, Evolução Diferencial e Programação Genética são exemplos de algoritmos evolucionários utilizados nos artigos revisados (Seção 2.4). A abordagem de agrupamento deste trabalho utiliza o algoritmo evolucionário inspirado no comportamento de Organismos Simbióticos (*SOS*) e é detalhado na Seção 2.2.1.

A seguir são detalhados alguns algoritmos pertencentes a Inteligência de Enxame e Computação Evolucionária. O detalhamento destes algoritmos, em específico, se dá pelo fato destes possuírem trabalhos correlatos ao presente desenvolvimento. Ao final desta seção é apresentado a importância do cálculo da diversidade populacional.

Em 1995, Otimização por Enxame de Partículas (*Particle Swarm Optimization - PSO*) foi proposta por Kennedy (KRISHNANAND; GHOSE, 2005). Esse algoritmo é inspirado no comportamento coordenado dos pássaros ao voarem e do movimento de cardume de peixes. O peixe ou o pássaro é considerado como uma partícula. No contexto computacional, cada partícula representa uma solução do problema. Esse algoritmo utiliza os conceitos de velocidade e posição para varrer o espa ço de busca. Os parâmetros do *PSO* são: o tamanho da população, o peso da inércia e os coeficientes de aceleração.

Proposto por Dorigo e Blum em 1999, a Otimização por Colônia de Formigas (*Ant Colony Optimization - ACO*) é inspirada no comportamento das formigas pela busca de alimentos (DORIGO; BLUM, 2005). Foi observado que, no decorrer do tempo, as formigas são capazes de descobrir o caminho mais curto entre a fonte de alimento e sua colônia utilizando feromônio, ou seja, o melhor caminho. Os parâmetros definidos no *ACO* são:

peso relativo da trilha, a atratividade da trilha, o tamanho da população e a taxa de evaporação do feromônio.

Otimização por Enxame de Vermes Luminescentes (*Glowworm Swarm Optimization - GSO*) foi proposto por Krishnanand e Ghose em 2005. Foi inspirado na capacidade do controle de emissão de luminosidade própria desses vermes em determinadas situações. O *GSO* possui os seguintes parâmetros: número de indivíduos, taxa de decaimento da luciferina, raio de abrangência da vizinhança e ponderador da função objetivo considerada no cálculo do nível de luciferina que é a substância que influência o nível de luminosidade.

Propostos por John Holland em 1975, os Algoritmos Genéticos (*Genetic Algorithms - GA*) são os mais conhecidos e utilizados dentre os Algoritmos Evolucionários (DAVIS et al., 1991). São inspirados no processo de Seleção Natural, segundo Darwin, no qual os indivíduos mais bem adaptados ao meio possuem maior chance de sobreviver e assim produzir mais dependentes. Além da seleção natural, a hereditariedade genética também é inspiração desses algoritmos. Parâmetros do *GA* são: o tamanho da população, a probabilidade de *crossover*, probabilidade de mutação e número de gerações.

Evolução Diferencial (*Differential Evolution - DE*) foi proposta por Storn e Price (STORN; PRICE, 1997). Também possui uma população de indivíduos onde cada indivíduo possui um tamanho fixo e representa uma solução. O indivíduo da população é representado por um vetor e cada indivíduo possui um custo calculado com base no próprio vetor. Esse algoritmo consiste em um esquema de gerar vetores de parâmetros experimentais na busca das melhores soluções no espaço de busca. Os parâmetros utilizados no *DE* são: o tamanho da população, a probabilidade de *crossover*, o fator de mutação e número de gerações.

Em 1992 John Koza utilizou *GAs* para desenvolver programas computacionais aplicados em certas tarefas computacionais, surgindo assim a Programação Genética (*Genetic Programming - GP*) (KOZA, 1992). O indivíduo da população é um programa composto por funções e terminais. Os terminais representam as variáveis e constantes de um programa. Essa algoritmo utiliza a reprodução dos indivíduos combinando variaáveis e constantes na busca das melhores soluções. São parâmetros da *GP*: tamanho da população, probabilidade de *crossover*, probabilidade de mutação e número de gerações.

2.2.1 Otimização por Organismos Simbióticos

O algoritmo inspirado em Organismos Simbióticos (*Symbiotic Organisms Search*) foi criado por Min-Yuan Cheng e Doddy Prayogo em 2014. Esse algoritmo é inspirado na relação simbiótica entre organismos distintos dentro de um ecossistema e é o algoritmo utilizado no desenvolvimento deste trabalho (CHENG; PRAYOGO, 2014).

Pode-se entender uma relação simbiótica como sendo uma relação estabelecida

entre dois organismos buscando a sobrevivência ou a adaptação de um ou de ambos no ecossistema. As relações simbióticas mais comuns encontradas na natureza são mutualismo, comensalismo e parasitismo. Mutualismo denota uma relação simbiótica entre duas espécies diferentes em que as duas se beneficiam. Comensalismo é uma relação simbiótica em que um dos indivíduos é beneficiado e o outro não é afetado. O parasitismo é uma relação simbiótica que um dos indivíduos é beneficiado enquanto o outro é prejudicado. O algoritmo *SOS* modela computacionalmente estas três relações. Cada organismo representa uma solução e é um indivíduo da população. Cada relação simbiótica possui uma característica específica para reprodução de novos organismos.

No mutualismo, para cada organismo da população O_i é selecionado aleatoriamente outro organismo O_j para então gerar dois novos organismos O_{inew} e O_{jnew}, conforme equações (2.16) e (2.17), respectivamente. O vetor de mutualismo, *MutualVetor*, representa as características do relacionamento de O_i e O_j e é obtido com o cálculo da média aritmética das dimensões dos organismos, calculado na equação (2.18).

$$O_{inew} = O_i + rand(0,1) * (O_{best} - MutualVetor * BF_1) \tag{2.16}$$

$$O_{jnew} = O_j + rand(0,1) * (O_{best} - MutualVetor * BF_2) \tag{2.17}$$

$$MutualVetor = (O_i + O_j)/2 \tag{2.18}$$

Onde as variáveis BF_1 e BF_2 podem assumir o valor 1 ou 2. O valor dessas variáveis é escolhido de forma aleatória caracterizando o não-determinismo, juntamente com os ponderadores aleatórios *rand*. A variável O_{best} representa o melhor organismo da população atual. A operação $(O_{best} - MutualVector * BF)$ nas equações (2.16) e (2.17) representa o mutualismo pois seu objetivo é aumentar a vantagem mutua de sobrevivência dos dois novos organismos no ecossistema utilizando o melhor indivíduo sem prejuízo a nenhum dos envolvidos. Após a reprodução, na fase de seleção é verificado qual é o melhor indivíduo. Se o novo organismo for melhor que o seu ancestral o novo organismo substitui seu antecessor no ecossistema. Caso contrário, o novo organismo é descartado.

No comensalismo, para cada organismo da população O_i, é gerado um novo organismo, O_{inew}, utilizando o melhor organismo da população, O_{best}, e outro organismo aleatório, O_j, conforme a equação (2.19).

$$O_{inew} = O_i + rand(-1,1) * (O_{best} - O_j) \tag{2.19}$$

A operação $(O_{best} - O_j)$ caracteriza o comensalismo, pois o novo organismo é beneficiado com a influência do melhor organismo do ecossistema sem prejudicar o melhor organismo.

Se o novo organismo for melhor que o seu ancestral o novo organismo substitui seu antecessor no ecossistema. Caso contrário, o novo organismo é descartado.

No parasitismo, para cada organismo do ecossistema, é feita uma cópia do mesmo e após isso é alterada uma única dimensão aleatória de seu vetor. Dessa forma, um novo organismo, O_{inew}, chamado de vetor parasita é gerado. Em seguida é selecionado um outro organismo de forma aleatória, O_j, que assume o papel de hospedeiro ao participar da seleção de organismos. Se o *fitness* do vetor parasita for melhor que o *fitness* do hospedeiro o parasita substitui o hospedeiro no ecossistema.

Os organismos mais adaptados ao ecossistema são mantidos na evolução. Esse algoritmo possui os parâmetros: tamanho da população, número máximo de iterações e máximo de avaliações. Esses dois últimos definem o critério de parada da execução.

2.2.2 Diversidade Populacional

A diversidade genotípica é uma métrica que identifica o grau de heterogeneidade ou homogeneidade entre os indivíduos de uma população. Essa métrica serve para monitorar o balanceamento entre a intensificação e a diversificação. O balanceamento da intensificação e diversificação é fundamental para o bom desempenho de um algoritmo populacional. A intensificação está relacionada com a concentração das buscas numa determinada região do espaço de solução que é considerada promissora. A diversificação está relacionada com a busca global. A mesma amplia as buscas por soluções em diferentes regiões visando identificar uma possível região promissora ainda não explorada. Quando o algoritmo apresenta elevada intensificação e baixa diversificação, caracterizado pela pouca diversidade genotípica na população de indivíduos, o processo de otimização se torna ineficiente. Isso porque a busca do algoritmo não irá contemplar diferentes regiões no espaço de solução. Nesse caso, a busca tende a estagnar em uma determinada região local. Quando o algoritmo apresenta alta diversificação e baixa intensificação, caracterizado pela manutenção da alta diversidade genotípica em todo o processo de otimização, a busca tende a ser aleatória perdendo assim a inteligência na busca da solução. Para um algoritmo populacional realizar a busca no espaço de solução de forma eficiente a diversidade genotípica inicialmente deve ser elevada e à medida que o processo de otimização evolui a diversidade genotípica deve ir sendo decrementada. Isso caracteriza a presença maior da diversificação em relação a intensificação no início do processo de otimização e com o decorrer do tempo a intensificação ganha mais presença no processo. A medida de diversidade genotípica (MDG) pode ser calculada conforme Equação (2.20) (CORRIVEAU et al., 2013).

$$MDG = \frac{\sum_{i=1}^{TP-1} ln\left(1 + \min_{j\in[i+1,TP]} \frac{1}{dm}\sqrt{\sum_{k=1}^{dm}\left(O_{i,k} - O_{j,k}\right)^2}\right)}{NMD} \tag{2.20}$$

Onde *TP* é o tamanho da população, *NMD* é o normalizador da métrica da diversidade genotípica. Essa variável mantém a maior diversidade genotípica da geração atual, *dm* representa o número total de dimensões de um indivíduo, O_j e O_i são indivíduos da população e k representa uma dimensão do indivíduo.

Assim, a diversidade genotípica possibilita analisar até quando um algoritmo populacional é capaz de buscar melhores soluções, enquanto possui diversidade genotípica, e definir quando não é mais útil a continuidade do processamento. Além disso, pode-se analisar se existe o balanceamento adequando entre a intensificação e diversificação no processo de otimização.

Algoritmos bio-inspirados tem sido aplicados em problemas de grande escala e multiobjetivos, como em tarefas de mineração de dados, com eficácia na resolução desses problemas. A combinação de algoritmos bio-inspirados e técnicas de mineração de dados é adequada para análise de dados principalmente devido a característica populacional desses algoritmos (CHENG et al., 2013).

A ferramenta *Hadoop*, detalhada na seção seguinte, fornece uma infraestrutura com recursos adequados à técnica de mineração de dados no cenário de agrupamento de dados.

2.3 HADOOP

Hadoop é uma ferramenta de código aberto utilizada para armazenar e manipular grandes massas de dados (NURAIN et al., 2012). Com essa ferramenta, o grande problema existente relacionado com tempo no processo de leitura de uma grande massa de dados é amenizado com a leitura de múltiplos discos de forma simultânea. Nesse tipo de leitura podem ocorrer algumas falhas que também são contornadas através da replicação de dados. Esse armazenamento compartilhado é provido pelo componente *(HDFS)* (KATAL; WAZID; GOUDAR, 2013). Outro componente provido pelo *Hadoop* é o sistema de análise chamado de *framework MapReduce*. *MapReduce* explora a arquitetura de armazenamento distribuída do *HDFS* provendo assim escalabilidade, confiabilidade e processamento paralelo (NARAYAN; BAILEY; DAGA, 2012). Nas subseções a seguir, os componentes *HDFS* e *Hadoop MapReduce* serão detalhados, respectivamente.

2.3.1 HDFS

HDFS é um sistema de arquivos projetado para armazenar arquivos de dados volumoso com um padrão contínuo de acesso a dados e em *cluster*. *HDFS* possui arquitetura cliente-servidor e utiliza o protocolo de comunicação *Transmission Control Protocol (TCP)*. Existem dois tipos de nodo. Um deles é o servidor, chamado *namenode* ou referenciado também por *Master*. O *Master* é responsável por gerenciar a localização de blocos

de arquivos fragmentados e replicados do sistema de arquivos, além de manutenir os metadados e a árvore do sistema de arquivos. O outro tipo de nodo é o cliente, chamado de *datanode* ou também referenciado por *Worker*. O *Worker* possui como função armazenar e recuperar blocos de dados solicitados por aplicações de *software* ou pelo *Master*. Em alguns casos a utilização de *HDFS* pode não ser adequada, como nos casos que envolvem o contexto de baixa latência de acesso a dados porque o tempo requerido para inicialização do serviço, divisão e junção das tarefas, além da comunicação entre os nós certamente inviabiliza a utilização do mesmo (KATAL; WAZID; GOUDAR, 2013).

2.3.2 MapReduce

MapReduce é um modelo de programação que possui principalmente duas funções, chamadas de *map* e *reduce*. Cada uma dessas funções recebe como entrada de dados um conjunto de pares chave-valor e após o processamento dessas informações, de acordo com a função implementada pelo programador, tem como saída um conjunto de pares chave-valor. Basicamente existem duas fases de execução no processamento *MapReduce*, cada fase executando uma função. Na primeira fase é executado o processo de mapeamento através da função *map*, na segunda fase é executado o processo de redução através da função *reduce*. Essas fases podem ser executadas em paralelo através dos *datanodes* disponíveis em um determinado *cluster*. O tempo de processamento depende do volume de dados processados, e do método de distribuição e replicação de dados no *cluster*. Quanto maior o volume de dados, maior será o tempo de processamento e quanto maior o *cluster* menor será o tempo de execução (NARAYAN; BAILEY; DAGA, 2012). Um *job MapReduce* representa um processo completo de execução do *framework MapReduce* num determinado arquivo armazenado no *HDFS*.

A Figura 1 mostra uma visão geral do fluxo de execução de um programa utilizando o *framework MapReduce*. No fluxo com rótulo (1) o *framework MapReduce* faz algumas cópias do programa do usuário para as estações *Master* e *Workers* do *cluster*. No fluxo com rótulo (2) o nodo *Master* faz a atribuição das tarefas *map* e *reduce* para os *workers*. No fluxo cujo rótulo é (3) os *workers* que irão realizar a execução da função *map* lêem o conteúdo correspondente ao dado de entrada e transformam esses dados em pares no formato chave-valor. Os dados nesse formato são passados como entrada para a função *map*. A saída de dados da função *map* também está em formato de pares chave-valor e fica armazenada na memória. No fluxo correspondente ao rótulo (4) os dados de saída da função *map*, que estão em memória, são gravados em disco local, chamados de arquivos intermediários, e posteriormente é enviada a localização desses arquivos para o *Master*. No fluxo (5) os *workers* responsáveis por fazer a redução são notificados pelo *Master* com a informação de localização dos arquivos intermediários. Assim, os *workers* fazem a leitura

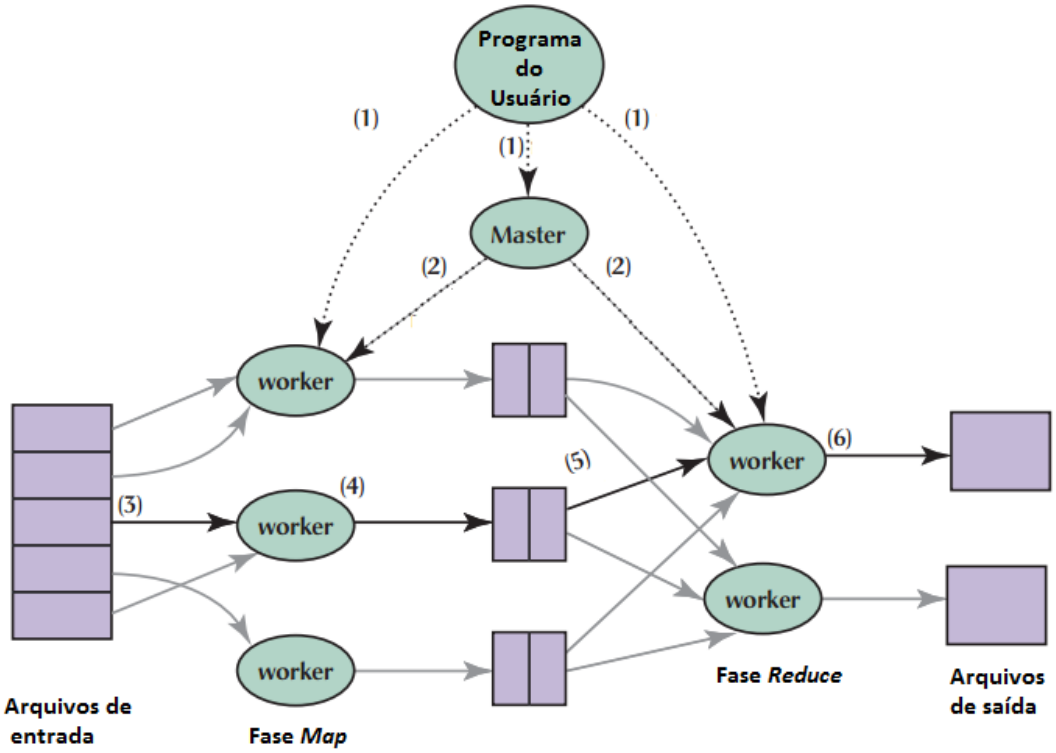

Figura 1 – Execução de um *job MapReduce*. Adaptado de (DEAN; GHEMAWAT, 2008).

remota dos arquivos intermediários. Os *workers* de redução ordenam os dados no formato de pares chave-valor de tal forma que todos os valores de uma mesma chave são agrupados e relacionados àquela chave. Assim, todos os pares terão chave diferentes e cada chave terá o conjunto de valores relacionado a ela. Em seguida, ainda no fluxo de número (5), cada chave com seus valores são enviados para a função *reduce*. Após o processamento, representado pelo fluxo de número (6), a saída de dados é persistida em um arquivo final. Ao finalizar todas as tarefas de mapeamento e redução, o *Master* retorna para o programa do usuário (DEAN; GHEMAWAT, 2008).

Visto que os conceitos que embasam a linha de pesquisa deste trabalho foram descritos, na próxima seção será apresentada de forma resumida algumas abordagens de mineração de dados que utilizam algoritmos bio-inspirados projetados que usam *Hadoop MapReduce*.

2.4 TRABALHOS RELACIONADOS

O objetivo dessa seção é identificar as principais abordagens que realizam mineração de massas de dados, tornando possível obter uma visão de quais algoritmos bio-inspirados são empregados, além de identificar as tarefas de mineração utilizadas.

Alguns dos trabalhos relacionados utilizaram o algoritmo *K-Means*. Esse algoritmo

foi proposto por J. MacQueen em 1967 e é um dos algoritmos tradicional de agrupamento que é mais utilizado além de possuir mais variações (FABER, 1994).

Como todos os trabalhos utilizam o *framework Hadoop*, a paralelização com tolerância a falhas e balanceamento de carga é uma característica implícita e comum em todos os trabalhos comentados a seguir.

Aljarah (2012) (ALJARAH; LUDWIG, 2012) propôs um algoritmo que realiza a tarefa de agrupamento e tem como base o algoritmo *PSO* adaptado ao paradigma *MapReduce*. Os resultados mostraram uma melhor qualidade comparados com o algoritmo *K-Means* além de possuir uma boa relação de desempenho mantendo a qualidade do agrupamento. Esse mesmo algoritmo foi utilizado em Aljarah (2013) (ALJARAH; LUDWIG, 2013) como um componente que realiza agrupamento em volume de dados de tráfico de redes de grande escala. Com essa arquitetura os resultados obtiveram uma boa taxa de detecção de invasão verdadeira e baixa taxa de detecção falsa. Além disso, foi mantido um *speedup* não linear. O ideal é que o *speedup* fique próximo de linear.

Judith (2015) (JUDITH; JAYAKUMARI, 2015) apresenta um sistema que realiza a tarefa de agrupamento em documentos utilizando o algoritmo *PSO*. Além de *PSO*, o sistema proposto também utiliza o algoritmo de agrupamento de dados *K-Means* e a técnica de redução de dimensionalidade *Latent Semantic Indexing (LSI)*. O sistema também demonstrou redução do tempo de execução com a alocação de mais recursos.

Al-Mad (2014) (AL-MADI; ALJARAH; LUDWIG, 2014) apresenta um algoritmo que realiza a tarefa de agrupamento e tem como base o algoritmo *GSO*. A cada iteração do algoritmo *GSO* ocorre a execução de um *job MapReduce* no arquivo de dados. Os resultados mostram que esse algoritmo reduz o tempo de execução com a alocação de mais recursos e mantem melhor qualidade nos grupos formados do que o *K-Means* e pelo algoritmo proposto por Aljarah (2012) (ALJARAH; LUDWIG, 2012).

O algoritmo *ACO* foi proposto por Bhavani (2014) (BHAVANI; SADASIVAM, 2014) para realizar a tarefa de agrupamento. Nessa abordagem o *ACO* é utilizado na poda de uma árvore de cobertura mínima até a formação de grupos. Esse algoritmo conseguiu alcançar um *speedup* abaixo de linear. O mecanismo distribui o algoritmo *ACO* de tal forma que no mapeamento a quantidade de feromônios influencia no cálculo da árvore de cobertura mínima. Já na redução, o feromônio é atualizado.

Outro trabalho que utiliza *ACO* para realizar a tarefa de agrupamento foi apresentado por Yang (2012) (YANG et al., 2012). Nos resultados desse trabalho é possível perceber que o tempo de execução do sistema em pequenas bases de dados aumenta a medida que é incrementado o número de nodos que compõe o *cluster* devido ao tempo consumido na comunicação entre os nodos na rede. Em relação a qualidade do agrupamento, foi comparado os resultados da versão do sistema com e sem a arquitetura *MapReduce* e

não foi possível concluir melhoria na qualidade dos grupos.

Hans (2015) (HANS; MAHAJAN; OMKAR, 2015) apresenta um aplicativo que realiza agrupamento de dados utilizando *GA*. O autor comparou a versão do aplicativo executado de forma sequencial com a versão projetada em *Hadoop MapReduce*. Foi demonstrado que os resultados de qualidade dos *clusters* foram semelhantes. Em relação ao tempo de execução, a versão do aplicativo projetado em *Hadoop MapReduce* consumiu menos tempo.

O algoritmo *GA* também foi utilizado para realizar a etapa de seleção de atributos antes de realizar a geração de regras de classificação em Ferrucci (2013) (FERRUCCI et al., 2013). Nesse trabalho foi construído um *framework* de desenvolvimento que paraleliza o algoritmo *GA* utilizando *Hadoop MapReduce*. O *framework* adapta o paradigma *MapReduce* de tal forma que é possível executar em uma única iteração vários processos de mapeamento antes e depois da redução. Os resultados demonstram que a precisão e tempo de execução foram melhores para a aplicação desenvolvida como base no *framework Hadoop MapReduce* em comparação com o *Weka*[1] e com uma versão pseudo-distribuída com apenas uma ilha. Al-Madi (2013) (AL-MADI; LUDWIG, 2013) apresenta um algoritmo que realiza classificação e tem como base o algoritmo *GP*. Resultados mostram que uma alta precisão de classificação além de manter escalabilidade e diminuir o tempo de execução por mapeamento.

Daoudi (2014) (DAOUDI et al., 2014) apresenta a abordagem de agrupamento que utiliza o algoritmo *DE*. Os resultados comprovam que esse modelo é mais preciso que *K-Means* e é mais estável e robusto que Aljarah (2012) (ALJARAH; LUDWIG, 2012) pois possui melhor desempenho a medida que o volume da base de dados aumenta.

Bhavani (2011) (BHAVANI; SADASIVAM; KUMARAN, 2011) apresenta duas estratégias para realizar agrupamento. Uma envolve *DE* e *K-Means* e a outra envolve o uso de *DE*, *ACO* e *K-Means*. Em ambas estratégias o algoritmo *DE* é utilizado para realizar a atualização global dos centroides enquanto que *K-Means* é utilizado localmente. O segundo mecanismo obteve resultados com melhor qualidade e desempenho. Ambos mecanismos aplicam o algoritmo DE apenas na redução, mas no segundo mecanismo o algoritmo o *ACO* é distribuído de tal forma que o feromônio seja analisado no mapeamento e é atualizado na redução.

[1] *Weka* é uma coleção de algoritmos de aprendizado de máquina utilizados em tarefas de mineração de dados. Os algoritmos podem ser aplicados diretamente em um conjunto de dados ou serem chamados de um código *Java*. *Weka* contém ferramentas para pré-processamento, classificação, regressão, agrupamento, associação (FERRUCCI et al., 2013).

2.4.1 MODELOS DE PROJETOS HADOOP MAPREDUCE

Três modelos de paralelismo foram identificados nos trabalhos revisados e são descritos a seguir (MENEZES; FREITAS; PARPINELLI, 2016).

A principal característica do Modelo 1 é a execução de apenas um *job MapReduce* por ciclo de iteração do algoritmo (conforme Figura 2, na qual o ícone com o desenho do planeta Terra representa o algoritmo bio-inspirado). Além disso, cada execução do *job MapReduce* processa toda a população de indivíduos no mapeamento do volume de dados. Como apresentado na Figura 2, o algoritmo bio-inspirado está projetado no *job Hadoop MapReduce*, portanto será paralelizado. Sete trabalhos usam esse modelo.

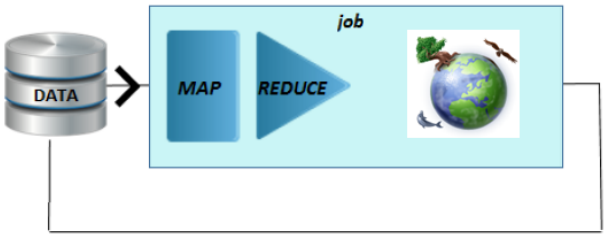

Figura 2 – Modelo 1:Um *job MapReduce*

O Modelo 2 tem como principal característica a execução de dois *jobs MapReduce* executados um após o outro (Rótulos 1 e 2 da Figura 3). Logo em seguida, é executada uma rotina que finaliza o ciclo de iteração do algoritmo (Rótulo 3 da Figura 3).

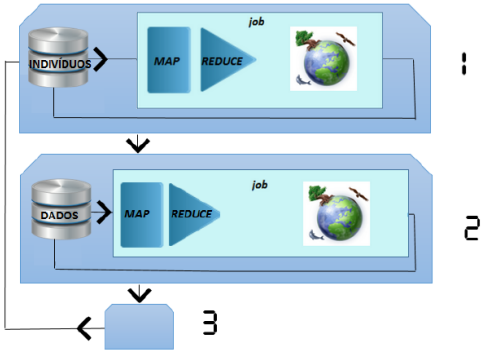

Figura 3 – Modelo 2: Dois *jobs MapReduce*

Conforme é apresentado, o primeiro *job MapReduce* processa dados que representam a movimentação dos indivíduos da população no espaço de solução. Já o segundo *job MapReduce* processa o volume de dados que está sendo minerado considerando a população de indivíduos. Na sequência, uma rotina é executada para realizar as últimas operações de um ciclo do algoritmo bio-inspirado. Esse modelo processa a população de indivíduos de forma paralela através do *job MapReduce* representado pelo Rótulo 1. Quatro trabalhos utilizam esse modelo. Apesar de (JUDITH; JAYAKUMARI, 2015) não realizar a execução de um *job MapReduce* no arquivo com a população de indivíduos, o

mesmo foi classificado no Modelo 2 por executar dois *jobs MapReduce*, um após o outro, que corresponde a principal característica desse modelo.

O Modelo 3 tem como principal característica a execução de um *job MapReduce* onde múltiplas funções de mapeamento são processadas antes e depois da função de redução (Figura 4).

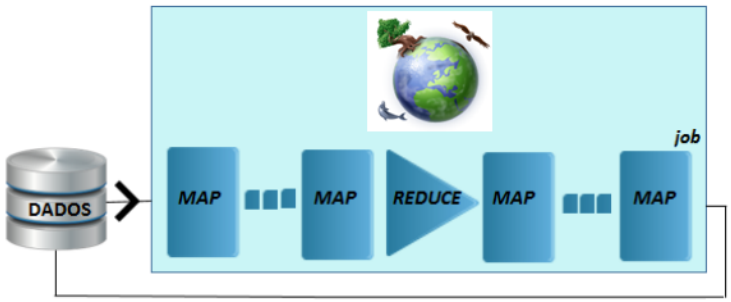

Figura 4 – Modelo 3: *Job com múltiplos Maps e Reduce*

Esse modelo consegue executar em um único *job MapReduce* várias fases do algoritmo bio-inspirado. Cada fase do algoritmo pode ser executada na função de mapeamento ou de redução. A desvantagem é a complexidade necessária para alterar seu *framework Hadoop MapReduce* e assim alterar o comportamento padrão do *job MapReduce* que é executar apenas um mapeamento e uma redução. Apenas um trabalho utilizou esse modelo.

A Figura 5 exibe a sumarização dos trabalhos revisados agrupados pelo modelo que cada trabalho melhor se assemelha.

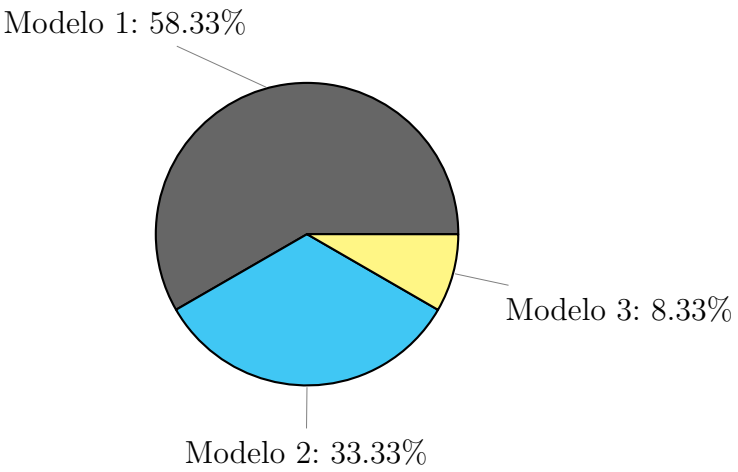

Figura 5 – Modelos de projeto *Hadoop MapReduce*

O Modelo 1 tende a ser mais vantajoso por ser mais simples de ser desenvolvido e por conseguir utilizar os benefícios do *framework MapReduce* no processamento dos dados que estão sendo minerados. Esse modelo é o mais utilizado nos trabalhos revisados. Como não é vantajoso executar o *job MapReduce* em arquivos pequenos (YANG et al., 2012)

(KATAL; WAZID; GOUDAR, 2013), como ocorre no arquivo que armazena a população de indivíduos no Modelo 2, o Modelo 2 torna o desenvolvimento mais trabalhoso sem ganho de desempenho com a paralelização devido ao tempo consumido na comunicação entre os nodos do *cluster*. Dessa forma, a ocorrência do Modelo 2 nos trabalhos revisados foi menor que a ocorrência do Modelo 1. Apenas um trabalho foi classificado no Modelo 3 devido a particularidade não muito comum de envolver alteração no comportamento padrão do *framework Hadoop MapReduce*.

Diante disso, o Modelo 1 foi empregado na abordagem deste trabalho. Assim, o algoritmo *SOS* está projetado no *Hadoop MapReduce* conforme o Modelo 1. No capítulo a seguir o método utilizado nesse trabalho será apresentado.

3 *MRCSOS*

Este capítulo detalha como o algoritmo *SOS* é projetado na arquitetura *Hadoop MapReduce* para agrupamento de dados. Além disso, é apresentada uma função *fitness* que terá sua qualidade avaliada juntamente com as outras funções apresentadas na revisão da literatura (Seção 2.1.1.1).

Para um melhor entendimento do funcionamento do algoritmo *MRCSOS*(*Map Reduce Clustering Symbiotic Organisms Search*), o Algoritmo 1 detalha seu pseudocódigo.

Algorithm 1 MRCSOS

Entrada: NC, TP, ACD, AD, QI
Saída: Percentual de pureza

1. Inicializar TP organismos no ecosistema com itens aleatórios de AD;
2. Armazernar TP organismos em ACD;
3. Executar JOB MAPREDUCE **início**
   4. Recuperar organismos de ACD;
   5. Map: calcular distâncias utilizadas na função *fitness*;
   6. Reduce: calcular função *fitness* dos organismos;

fim

7. Ler arquivo de saída e atualizar *fitness* dos organismos;
8. Armazenar melhor organismo em O_{best};
9. Inicializar i = 0;
10. **while** $i < QI$ **do**
    11. Limpar ACD;
    12. O_j com j=1 **repita**
        //`Relação Simbiótica de Mutualismo`
        13. Selecionar um organismo aleatório O_k != O_j;
        14. Criar VM com O_j e O_k;
        15. Criar novo organismo $Onew_j$ via perturbação de O_j utilizando VM, O_{best} e BF1;
        16. Criar novo organismo $Onew_k$ via perturbação de O_k utilizando VM, O_{best} e BF2;
        17. Guardar a referência de O_j em $Onew_j$;
        18. Guardar a referência de O_k em $Onew_k$;
        19. Armazenar $Onew_j$ e $Onew_k$ em ACD;
        20. j = j + 1;
    até $j = TP$;
    21. Executar JOB MAPREDUCE ;
    22. Ler arquivo de saída e atualizar *fitness* dos organismos $Onew_j$ e $Onew_k$;
    23. O_{jk} com jk=1 **repita**
        24. **se** *fitness de* $Onew_{jk}$ *for melhor que o de* O_{jk} **então**
            25. Substituir O_{jk} por $Onew_{jk}$
        26. jk = jk + 1;
    até $j = TP * 2$;
    27. Limpar ACD;
    28. O_j com j=1 **repita**
        //`Relação Simbiótica de Comensalismo`
        29. Selecionar um organismo aleatório M != O_j;
        30. Criar novo organismo $Onew_j$ via perturbação de O_j utilizando M e O_{best};
        31. Guardar a referência de O_j em $Onew_j$;
        32. Armazenar $Onew_j$ em ACD;
        33. j = j + 1;
    até $j = TP$;
    34. Executar JOB MAPREDUCE ;
    35. Ler arquivo de saída e atualizar *fitness* dos organismos $Onew_j$;
    36. O_j com j=1 **repita**
        37. **se** *fitness de* $Onew_j$ *for melhor que o de* O_j **então**
            38. Substituir O_j por $Onew_j$
        39. j = j + 1;
    até $j = TP$;
    40. Limpar ACD;
    41. O_j com j=1 **repita**
        //`Relação Simbiótica de Parasitismo`
        42. Selecionar um organismo aleatório M != O_j;
        43. Criar novo organismo $Onew_j$ via perturbação de O_j;
        44. Guardar a referência de M em $Onew_j$;
        45. Armazenar $Onew_j$ em ACD;
        46. j = j +1;
    até $j = TP$;
    47. Executar JOB MAPREDUCE ;
    48. Ler arquivo de saída e atualizar fitness dos organismos $Onew_j$;
    49. O_j com j=1 **repita**
        50. **se** *fitness de* $Onew_j$ *for melhor que o de M* **então**
            51. Substituir M por $Onew_j$
    até $j = TP$;
    52. Selecionar o melhor organismo e armazena-lo na variável O_{best}.
    53. i = i + 1;

//`Cálculo da pureza de` O_{best}

54. Limpar ACD;
55. Armazenar O_{best} em ACD;
56. Executar JOB MAPREDUCE **início**
    57. Recuperar O_{best} de ACD;
    58. Map : calcular quantos itens de cada classe está coberto por cada centroíde;
    59. Reduce : calcular pureza e entropia do agrupamento;

fim

60. Ler valor da pureza final do arquivo de saída;

O funcionamento do algoritmo *MRCSOS* pode ser dividido em três fases principais de execução: inicialização (linhas 1 a 9), processamento (linhas 10 a 53) e finalização (linhas 54 a 60).

Na fase de inicialização (linha 1) são recebidos os parâmetros: número de centroides (NC), tamanho da população (TP), arquivo de cache distribuído (ACD), arquivo de dados (AD) e quantidade de iterações (QI). Com essas informações o *MRCSOS* inicializa os organismos de acordo com a quantidade parametrizada. Cada organismo possui um identificador único e representa um conjunto de centroides. Cada centroide possui um identificador e é representado por um vetor de u dimensões, onde u é igual a quantidade de atributos da base de dados. Os valores dos centroides são inicializados com itens selecionados aleatoriamente da base de dados. Dessa forma é possível garantir que cada centroide possua pelo menos um item agrupado. Isso garante que todas as soluções iniciais são válidas, pois cada centroide deve agrupar pelo menos um item da base. Ainda nessa fase é calculada a função *fitness* de cada organismo da população. Para isso, os organismos são armazenados no ACD (linha 2) para serem recuperados de forma eficiente na execução do *job Hadoop MapReduce* no cálculo do *fitness*. No processo de mapeamento (linha 5), função *map*, é calculada a distância entre cada item de dados e os centroides de cada organismo simbiótico e emitido para a fase de redução. Na execução da função *reduce* (linha 6), as distâncias recebidas são utilizadas no cálculo da função *fitness* e os resultados são armazenados no arquivo de saída do *job Hadoop MapReduce*. Todos os cálculos de *fitness* neste algoritmo são realizados dessa forma (linhas 3 a 6). Em seguida, é feita a leitura do arquivo de saída do *job Hadoop MapReduce* contendo o valor do *fitness* de cada organismo e assim a atualização do organismo com seu respectivo *fitness* (linha 7). Finalizando a fase de inicialização, é realizada a seleção do melhor organismo, através da comparação do *fitness* e o mesmo é armazenado na variável O_{best} (linha 8).

Na fase de processamento ocorre a evolução dos organismos através das relações simbióticas de mutualismo (linhas 11 a 26), comensalismo (linhas 27 a 39) e parasitismo (linhas 40 a 53) até que a condição de parada, número máximo de iterações, seja alcançada (linha 10). Inicialmente ocorre o mutualismo onde para cada organismo O_j é selecionado um organismo diferente O_k (linha 13) para então realizar o cálculo (linha 14) do vetor de mutualismo (VM), conforme Equação (2.18). Em seguida ocorre a pertubação do organismo O_j, definida na Equação (2.17), gerando o novo organismo $Onew_j$ (linha 15). Da mesma forma ocorre a pertubação do organismo O_k, definida na Equação (2.16), gerando o novo organismo $Onew_k$. Posteriormente, a referência dos organismos já existentes O_j e O_k são armazenadas nos organismos recém-gerados $Onew_j$ e $Onew_k$ (linhas 17 e 18). No final dessa relação simbiótica os novos organismos são armazenados no ACD. Com isso realiza-se o cálculo do *fitness* para cada novo organismo (linha 24) e se o *fitness* do novo

organismo for melhor que o organismo antigo o mesmo substitui seu respectivo organismo mais antigo (linha 25). Na sequência ocorre o comensalismo, para cada organismo O_j é realizada a perturbação do mesmo utilizando a Equação (2.19) originando $Onew_j$ (linha 30). A referência de O_j é guardada em $Onew_j$ para futura seleção evolutiva (linha 31). Após isso, cada $Onew_j$ é armazenado em ACD (linha 32). Dessa forma, é realizado o cálculo da função *fitness* e a atualização dos organismos recém gerados (linhas 34 e 35). Finalizando o comensalismo, ocorre a seleção entre cada organismo O_j e seu respectivo descendente $Onew_j$ onde quem possuir o melhor *fitness* será mantido no ecossistema (linhas 37 e 38). Por último ocorre o parasitismo, onde cada organismo O_j é perturbado, conforme perturbação de parasitismo (Seção 2.2.1), originando o organismo $Onew_j$ (linha 43). Como característica do parasitismo, o organismo $Onew_j$ guarda a referência (linha 44) de um terceiro organismo, M, para ser utilizado no processo de seleção. Em seguida, cada organismo $Onew_j$ é armazenado em ACD para realizar o cálculo do *fitness* (linha 45). Após isso é calculado o fitness dos organismos e atualizado nos mesmos (linhas 47 e 48). Finalizando o parasitismo, cada organismo $Onew_j$ é comparado com o seu respectivo organismo M onde é mantido o organismo com melhor *fitness* (linhas 50 e 51). Finalizando a fase de processamento, é selecionado o melhor organismo do ecossistema e armazenado em O_{best} (linha 52), além de incrementar a variável de controle de iterações (linha 53).

Na fase de finalização é verificada a qualidade do agrupamento através do cálculo da pureza utilizando o melhor organismo do ecossistema (O_{best}) visto que o mesmo representa a melhor solução encontrada pelo *MRCSOS*. Para isso, ocorre a limpeza de ACD (linha 54) e armazenamento de O_{best} no ACD (linha 55) para calcular a pureza utilizando um *job Hadoop MapReduce*. No mapeamento (linha 58), O_{best} é recuperado de ACD e é verificado para cada item de dados qual é o centroide de O_{best} que cobre aquele item, através da menor distância, e assim envia-se a classe daquele item com o respectivo centroide para a redução. Na redução ocorre o cálculo da pureza utilizando a função *fScore* e a entropia com as informações recebidas do mapeamento (linha 59). No final da redução é emitido o valor do percentual de pureza e a entropia de O_{best} no arquivo de saída. Finalizando execução do algoritmo ocorre a leitura do arquivo com o percentual de pureza e a entropia do agrupamento (linha 60).

Para uma macrovisão do fluxo de processos existentes na execução do *MRCSOS* é apresentado o fluxograma na Figura 6. Nele também é possível identificar quais processos são executados em paralelo e quais são executados de maneira sequencial.

No fluxograma é possível identificar a fase de inicialização, grupo de processos tracejado de azul, contendo os processos de recebimento de parâmetros, leitura da base, inicialização dos organismos, cálculos do *fitness* dos organismos e seleção do melhor organismo no ecossistema. Percebe-se que o processo que calcula o *fitness* dos organismos está

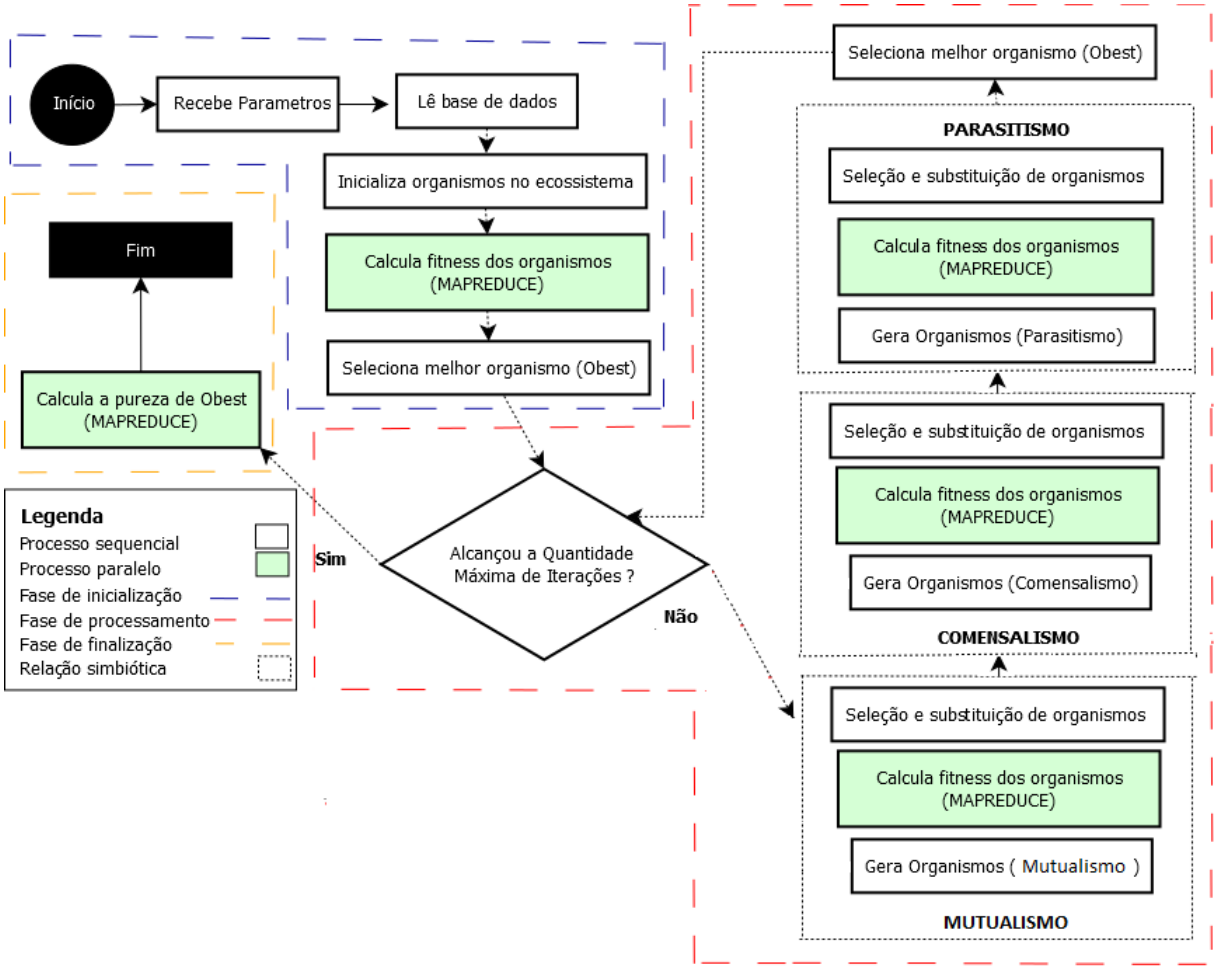

Figura 6 – Fluxograma do *MRCSOS*

destacado em verde informando que esse processo é executado em paralelo. Na sequência, o fluxo de execução entra na fase de processamento, grupo de processos tracejado de vermelho, possuindo os processos que abrangem as relações simbióticas de mutualismo, comensalismo e parasitismo, além do processo de seleção do melhor organismo. Assim como na fase de inicialização, os processos de cálculo do *fitness* presentes no mutualismo, parasitismo e comensalismo estão destacados em verde indicando sua execução em paralelo. Percebe-se também nessa fase a existência de um processo decisório indicando um ciclo de iterações. O ciclo é executado até que a quantidade máxima de iterações seja alcançada. Finalizando, o fluxo de execução chega na fase de finalização, grupo de processos tracejado de amarelo, onde está presente o processo de cálculo da pureza que é executado em paralelo. Ao final desse processo é obtida a pureza indicando o nível de qualidade do agrupamento formado.

No próximo capítulo serão descritos os experimentos e apresentados os resultados obtidos além da análise comparativa com resultados de outras abordagens.

4 EXPERIMENTOS, RESULTADOS E ANÁLISES

Os experimentos foram realizados utilizando três bases de dados, as mesmas que foram utilizadas por alguns trabalhos relacionados, amplamente utilizadas na literatura: *MAGIC Gamma Telescope*, *Poker Hand* e *Electricity*.

A base *MAGIC Gamma Telescope* pertence a área da Física. Os dados dessa base foram gerados em 2004 para simular o registro de partículas de raios gama de alta energia num telescópio *Cherenkov* atmosférico. Essa base possui 19.020 instâncias de dados, 10 atributos previsores, atributo-meta com 2 classes, 3 *Megabytes* e não existe ausência de valores nos atributos. A mesma está disponível no *UCI Machine Learning Repository* (https://archive.ics.uci.edu/ml/index.html) (LICHMAN, 2013).

A base *Poker Hand* também está no repositório *UCI* e pertence a área de jogos. Cada registro dessa base de dados representa uma mão de 5 cartas. Um par de atributos representa uma carta onde um deles representa o naipe da carta e o outro representa o valor da carta. Assim, 10 atributos representam 5 cartas. Essa base possui 1.025.010 instâncias de dados, 10 atributos previsores, atributo-meta com 10 classes, 25,2 *Megabytes* e não possui valores faltantes (LICHMAN, 2013).

A base de dados *Electricity* pertence a área de comércio. Os dados dessa base foram coletados no mercado *Australian New South Wales Electricity Market*. Neste comércio, os preços não são fixos e são afetados pela oferta e procura do mercado. Os preços são fixados a cada cinco minutos. A etiqueta de classe identifica a alteração do preço relativo a uma média móvel das últimas 24 horas. Essa base possui 45.312 instâncias de dados, 8 atributos previsores, atributo-meta com 2 classes, 6 *Megabytes* e está disponível no *Massive Online Analysis (MOA)* (http://moa.cms.waikato.ac.nz/datasets/).

Os experimentos foram realizados em *clusters* configurados nos laboratórios do departamento do curso de Ciência da Computação da Universidade do Estado de Santa Catarina em Joinville-SC. Em cada *cluster*, um computador assume a função de servidor, como visto na Figura 1 (Seção 2.3 do Capítulo 2), e não realiza processamento de dados. Dessa forma, o número de computadores de um *cluster* que executam o processamento de dados em paralelo é o total de computadores do *cluster* menos 1. Todos os resultados analisados foram coletados executando os experimentos num *cluster Hadoop* com 9 computadores. Cada computador possui a seguinte configuração: 8 *Gigabytes* de RAM, processador *Intel(R) Core(TM)* i7-4770 CPU 3.40 GHz, Sistema Operacional *Ubuntu* 14.04 LTS 64-bit, Hadoop versão 2.7.1, *Java* 1.7 e configuração de rede com 100Mbps . O algoritmo *MRCSOS* foi desenvolvido na linguagem de programação Java. Os parâmetros utilizados foram: tamanho da população igual a 100, número de execuções

igual a 10, número máximo de iterações igual a 100. Com esta configuração tem-se um total de 40.000 avaliações de função *fitness* por execução. A parametrização foi definida dessa forma para que a quantidade total de avaliações por execução não fosse superior a quantidade de avaliações utilizadas em (AL-MADI; ALJARAH; LUDWIG, 2014) que é de 100.000 avaliações por execução. Dessa forma, a quantidade de avaliações aplicada no *MRCSOS* é inferior devido ao tempo necessário para realizar os experimentos. Apesar disso, essa parametrização permite comparar a competitividade entre as abordagens avaliadas.

Dois tipos de análises são realizadas considerando os resultados obtidos. A primeira análise refere-se a qualidade dos agrupamentos encontrados que está exibida na Seção 4.1. A segunda análise está relacionada com o desempenho do *MRCSOS* descrito na Seção 4.2.

4.1 ANÁLISE DE QUALIDADE

A análise de qualidade dos resultados obtidos contempla as métricas de pureza e entropia do agrupamento de dados, Equação (2.11) e Equação (2.13) descritas na Seção 2.1.1.2 do Capítulo 2, respectivamente. Além disso, é analisada a correlação entre diferentes funções *fitness* do agrupamento, diversidade genotípica, pureza e entropia no decorrer do processo de agrupamento de dados. A análise da correlação é realizada com base no comportamento dos respectivos gráficos. A primeira métrica analisada é a pureza, posteriormente a entropia para então finalizar essa subseção com as análises de correlação.

4.1.1 Pureza

Antes de analisar os resultados é importante relembrar que quanto maior o valor da pureza melhor é a qualidade do agrupamento e que o valor máximo que a pureza pode assumir é 1. O intervalo de valores que a pureza pode assumir é [0,1]. Os resultados das médias de pureza dos agrupamentos estão exibidos na Tabela 1 onde a primeira coluna informa o algoritmo empregado. Os resultados obtidos pelos algoritmos *K-Means*, *MRCPSO* e *MRCGSO* foram coletados dos trabalhos de (ALJARAH; LUDWIG, 2012) e (AL-MADI; ALJARAH; LUDWIG, 2014) e são mostrados nas três primeiras linhas. Vale ressaltar que não foi informado o desvio padrão dos resultados destes trabalhos, sendo estes os melhores valores encontrados dentre todas as execuções. O desvio padrão é uma medida de dispersão usada com a média. Mede a variabilidade dos valores à volta da média. O valor mínimo do desvio padrão é 0 indicando que não há variabilidade, ou seja, que todos os valores são iguais à média. Nas demais linhas é mostrado o resultado do *MRCSOS* com diferentes funções de *fitness*, descritos na Seção 2.1.1.1 do Capítulo 2. Na segunda

coluna são exibidos os resultados de pureza na base *Magic*, onde os mesmos correspondem a média e desvio padrão e melhor pureza encontrada dentre todas as execuções destacada entre parênteses. Na terceira coluna são exibidos os resultados na mesma estrutura da segunda coluna, entretanto, para a base *Electricity*. Da mesma forma estão descritos na quarta coluna os resultados utilizando a base *Poker Hand*. Os resultados que estão com destaque em negrito representam o melhor resultado de pureza considerando a média e desvio padrão de cada base de dados.

Tabela 1 – Resultados obtidos para a pureza dos agrupamentos

Algoritmo	*Magic*	*Electricity*	*Poker Hand*
K-Means	0.60	0.51	0.11
MRCPSO	0.65	0.58	0.51
MRCGSO	0.66	0.58	**0.53**
MRCSOS(f1)	0.54±0.305(0.55)	0.54±1.249(0.56)	0.19±1.823(0.22)
MRCSOS(f2)	**0.69±0.001 (0.69)**	0.54±1.678(0.56)	0.34±3.218(0.39)
MRCSOS(f3)	**0.69±0.008 (0.69)**	0.55±0.803(0.56)	0.28±2.236(0.32)
MRCSOS(f4)	0.56±0.189(0.56)	0.61±1.435(0.64)	0.18±0.896(0.19)
MRCSOS(f5)	**0.69±0.000 (0.69)**	**0.67±0.000 (0.67)**	0.35±4.256(0.39)

Em relação aos resultados na base *Magic*, as médias de pureza obtidas pelo *K-Means*, *MRCPSO* e *MRCGSO* são superiores à média de pureza das versões de *MRCSOS(f1)* e *MRCSOS(f4)*. Contudo, as médias dos resultados obtidos nessa base utilizando as versões *MRCSOS(f2)*, *MRCSOS(f3)* e *MRCSOS(f5)* são as maiores considerando todos os algoritmos da Tabela 1.

Nessa base verifica-se também que os resultados obtidos pelas versões *MRCSOS(f2)*, *MRCSOS(f3)* e *MRCSOS(f5)* são equivalentes, possuindo sobreposição dos intervalos de desvio padrão, conforme Tabela 1. Dentre essas abordagens a *MRCSOS(f5)* foi a única a possuir o desvio padrão igual a 0 considerando até a terceira casa decimal. A Figura 7 exibe a distribuição dos resultados cujas médias estão exportas na Tabela 1. Nessa figura pode-se verificar não ocorre sobreposição de resultados obtidos por *MRCSOS(f1)* e *MRCSOS(f4)*.

Na base *Electricity*, as médias da pureza obtida pelas versões *MRCSOS(f1)*, *MRCSOS(f2)* e *MRCSOS(f3)* são inferiores às médias obtidas pelas versões *MRCPSO* e *MRCGSO*. As versões *MRCSOS(f4)* e *MRCSOS(f5)* apresentam média de pureza superior a dessas abordagens. A melhor média de pureza na base *Electricity* foi obtida pela versão *MRCSOS(f5)* não ocorrendo sobreposição de resultados considerando o desvio padrão. Na base *Electricity* todas as versões de *MRCSOS* obteveram melhores resultados que o *K-Means*.

O gráfico *Boxplot* com os resultados de pureza obtidas na base *Electricity* está visível na Figura 8. Nessa figura pode-se perceber que a distribuição de todos resultado de

pureza obtido pela versão *MRCSOS(f5)* não se sobrepõem com nenhum resultado obtido por todas as outras versões. Da mesma forma, a versão *MRCSOS(f4)* apresenta a distribuição de resultados melhores que as versões *MRCSOS(f1)*, *MRCSOS(f2)* e *MRCSOS(f3)*. As versões *MRCSOS(f1)*, *MRCSOS(f2)* e *MRCSOS(f3)* obtiveram sobreposição de resultados de pureza indicando que os mesmos são equivalentes estatisticamente.

Os resultados obtidos utilizando a base *Poker Hand* evidenciam que a abordagem *MRCGSO* encontrou a melhor resultado de pureza. O segundo melhor resultado nessa base foi obtido pela abordagem *MRCPSO*. Todas as versões de *MRCSOS* obtiveram resultados melhores que o *K-Means*. Dentre as versões de *MRCSOS*, a versão *MRCSOS(f5)* obteve o melhor resultado da média de pureza. Entretanto, ocorre sobreposição do intervalo de desvio padrão da versão *MRCSOS(f5)* com a versão *MRCSOS(f2)* , conforme Tabela 1. Na Figura 9 é possível confirmar que ocorre sobreposição na distribuição dos resultados de *MRCSOS(f5)* e *MRCSOS(f2)*. Após essas duas versões, a melhores médias de pureza são obtidas respectivamente pelas versões *MRCSOS(f3)*, *MRCSOS(f1)* e *MRCSOS(f4)*. As versões *MRCSOS(f2)*, *MRCSOS(f3)* e *MRCSOS(f5)* apresentam resultados estatisticamente equivalentes. Pode-se verificar a sobreposição na distribuição entre os resultados de *MRCSOS(f2)*, *MRCSOS(f3)* e *MRCSOS(f5)*. Essas três versões obtiveram o melhor resultado de pureza. Na Seção 4.1.3, experimentos com *MRCSOS(f5)* são estendidos para fins de análise de convergência.

Visto os resultados de pureza nas três bases conforme Tabela 1, percebe-se que a abordagem *MRCSOS(f5)* se sobressai diante das demais na busca por soluções de agrupamento de dados com alto grau de pureza. Apesar de obter o terceiro melhor resultado de pureza na base *Poker Hand* a mesma obteve os melhores resultados de pureza juntamente com *MRCSOS(f2)* e *MRCSOS(f3)* na base *Magic*. *MRCSOS(f5)* também obteve o melhor resultado da média de pureza na base *Electricity*. Nota-se, dentre os algoritmos analisados, que *MRCSOS(f5)* tende a encontrar os melhores resultados de pureza nas bases experimentadas.

Como as três bases foram analisadas considerando a pureza como critério de qualidade de agrupamento, a seguir é apresentada a análise de qualidade considerando a métrica de entropia. A análise de entropia considera apenas as versões da abordagem *MRCSOS* tendo em vista que as outras abordagens não apresentaram resultados de entropia para as bases analisadas.

4.1.2 Entropia

Ao contrário dos resultados de pureza, é importante recordar que quanto menor o valor da entropia melhor é a qualidade do agrupamento. O resultado da entropia de um agrupamento pode assumir qualquer valor dentro do intervalo [0,1].

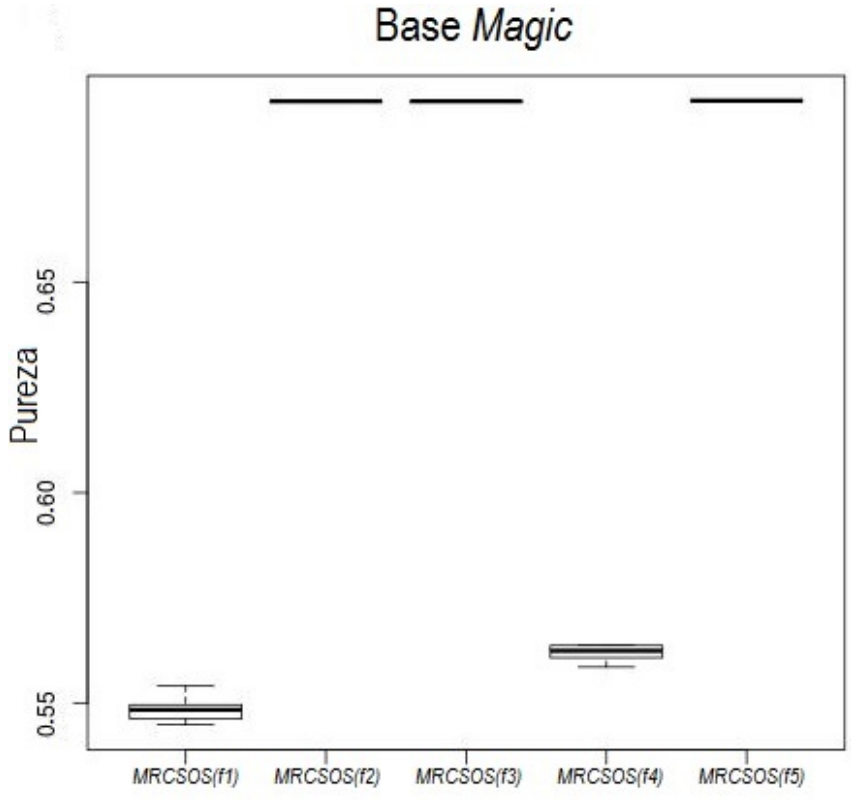

Figura 7 – Gráfico *Boxplot* com resultados de pureza obtidos na base *Magic*. O eixo-x representa o percentual de pureza e o eixo-y representa a versão de *MRCSOS*.

Os resultados dos experimentos considerando a métrica de entropia estão dispostos na Tabela 2. A primeira coluna informa o algoritmo empregado. Na segunda coluna são exibidos os resultados da entropia na base *Magic*, onde os mesmos correspondem a média e desvio padrão e melhor entropia dentre todas as execuções, mostrada entre parênteses. Na terceira coluna são exibidos os resultados na mesma estrutura da segunda coluna, entretanto, para a base *Electricity*. Da mesma forma, estão descritos na quarta coluna os resultados utilizando a base *Poker Hand*. Os resultados que estão com destaque em negrito representam o melhor resultado de entropia considerando a média e desvio padrão da base de dados representada na coluna da Tabela 2.

Tabela 2 – Resultados obtidos para a entropia dos agrupamentos

Algoritmo	*Magic*	*Electricity*	*Poker Hand*
MRCSOS(f1)	0.956±0.000(0.956)	0.943±0.004(0.936)	0.429±0.007(0.429)
MRCSOS(f2)	**0.530±0.000 (0.530)**	0.939±0.009(0.925)	0.442±0.011(0.430)
MRCSOS(f3)	**0.538±0.011 (0.530)**	0.942±0.004(0.935)	0.435±0.016(0.410)
MRCSOS(f4)	0.958±0.000(0.958)	0.943±0.008(0.925)	0.428±0.008(0.420)
MRCSOS(f5)	**0.530±0.000 (0.530)**	**0.524±0.000 (0.534)**	**0.421±0.005 (0.414)**

A média dos resultados de entropia das versões *MRCSOS(f2)* e *MRCSOS(f5)* foram os melhores na base *Magic*. A versão *MRCSOS(f3)* obteve média de entropia um

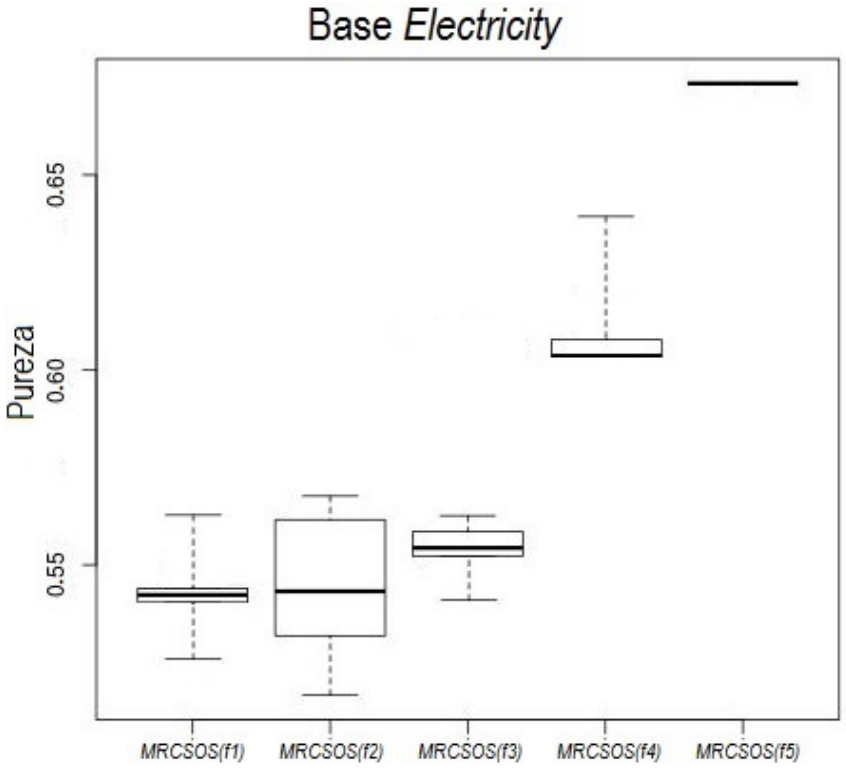

Figura 8 – Gráfico *Boxplot* com resultados de pureza obtidos na base *Electricity*. O eixo-x representa o percentual de pureza e o eixo-y representa a versão de *MRCSOS*.

pouco inferior aos resultados de *MRCSOS(f2)* e *MRCSOS(f5)* apesar de ocorrer sobreposição do intervalo de desvio padrão entre essas três versões. Estatisticamente, essas três versões obtiveram resultados de entropia equivalentes. A distribuição dos resultados pode ser observada no gráfico *Boxplot* na Figura 10. As versões *MRCSOS(f1)* e *MRCSOS(f4)* também obtiveram resultados de entropia estatísticamente semelhantes além de serem os piores resultados de entropia nessa base.

Na base *Electricity* a versão *MRCSOS(f5)* obteve a melhor média de entropia e não ocorre sobreposição na distribuição de resultados com nenhuma outra versão conforme gráfico *Boxplot* da Figura 11. Estatisticamente, essa versão é a que apresenta resultados de entropia significativamente melhores que as outras versões. As outras quatros versões de *MRCSOS* apresentam sobreposição de valores de entropia. Dessa forma, as versões *MRCSOS(f1)*, *MRCSOS(f2)*, *MRCSOS(f3)* e *MRCSOS(f4)* apresentam resultados de entropia estatisticamente correspondentes.

Na base *Poker Hand*, conforme Tabela 2, a versão *MRCSOS(f5)* apresenta melhor resultado de entropia. Entretanto, conforme Figura 12, os resultados de entropia obtidos pela versão *MRCSOS(f5)* apresenta sobreposição de resultado com as versões de *MRCSOS(f1)*, *MRCSOS(f3)* e *MRCSOS(f4)*. Isso caracteriza que a versão *MRCSOS(f5)* possui equivalência estatística de entropia com as versões *MRCSOS(f1)*, *MRCSOS(f3)* e *MRCSOS(f4)*. A versão *MRCSOS(f5)* possui melhores resultados significativos de entropia que

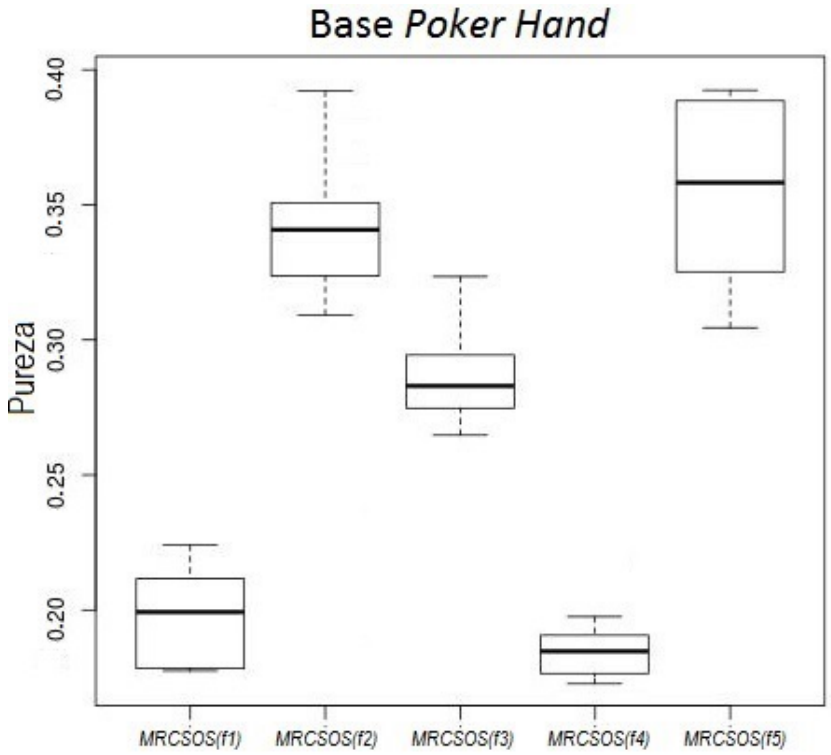

Figura 9 – Gráfico *Boxplot* com resultados de pureza obtidos na base *Poker Hand*. O eixo-x representa o percentual de pureza e o eixo-y representa a versão de *MRCSOS*.

os resultados obtidos pela versão *MRCSOS(f2)* nessa base de dados.

Conforme visto na Tabela 2, a versão *MRCSOS(f5)* obteve melhores resultados de entropia que as outras versões. Na base *Magic* a versão *MRCSOS(f5)* obteve a melhor média de entropia juntamente com as versões *MRCSOS(f2)* e *MRCSOS(f3)*. Além disso, essa versão da abordagem obteve melhor média de entropia tanto na base *Electricity* quanto na base *Poker Hand*. Dessa forma, pode-se entender que dentre as 5 versões da abordagem *MRCSOS* a que utiliza a função *f5* tende a encontrar melhores resultados de entropia dos agrupamentos formados.

Após realizar a análise da qualidade do agrupamento de dados considerando as métricas de pureza e entropia, a seguir será descrita a análise de correlação do comportamento entre as funções *fitness*, diversidade genotípica, pureza e entropia.

4.1.3 Correlações

A análise feita nessa seção correlaciona o gráfico de convergência (iterações vs *fitness*) das diferentes funções de agrupamento com seus respectivos gráficos de diversidade genotípica, pureza e entropia. O objetivo é observar o comportamento destas funções com relação a estes indicadores. Ou seja, avaliar a qualidade das funções de agrupamento segundo o grau de pureza e entropia durante a evolução do processo de otimização consi-

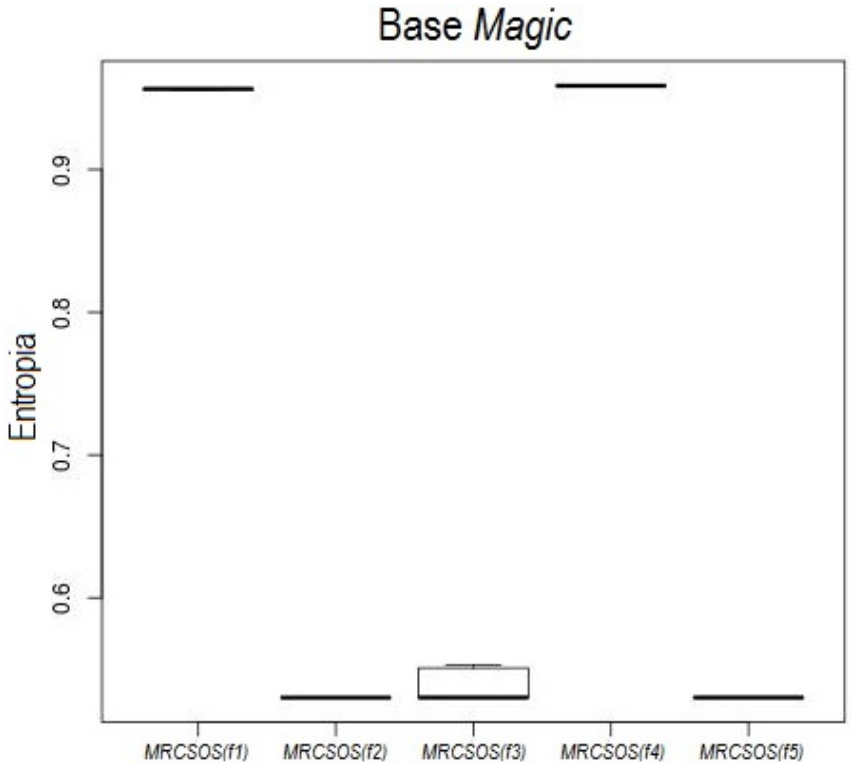

Figura 10 – Gráfico *Boxplot* com resultados de entropia obtidos na base *Magic*. O eixo-x representa o valor de entropia e o eixo-y representa a versão de *MRCSOS*.

derando também a diversidade das soluções envolvidas nesse processo.

A análise de correlação considera 15 cenários resultantes da combinação das execuções das versões *MRCSOS(f1)*, *MRCSOS(f2)*, *MRCSOS(f3)*, *MRCSOS(f4)* e *MRCSOS(f5)* utilizando as bases *Magic*, *Electricity* e *Poker Hand*. Cada base de dados pode ser considerada como um problema diferente com complexidade diferente. Cada versão de *MRCSOS* é uma diferente solução para agrupamento que é definida de acordo com as características da sua respectiva função de agrupamento. Assim, cada versão tende a possuir um comportamento diferente quando aplicado em bases de dados diferentes. Cada cenário de análise possui uma figura com a representação da função *fitness*, diversidade genotípica, pureza e entropia conforme disposto no modelo apresentado na Figura 13.

O eixo-x nos gráficos representa a quantidade de iterações. Nos eixos-y estão os valores da função *fitness*, os valores da diversidade genotípica, os percentuais das purezas do agrupamento (%) e os valores da entropia do agrupamento, respectivamente. Vale ressaltar que os gráficos foram gerados utilizando a média de 10 execuções.

O gráfico de *fitness* serve para evidenciar a evolução da função *fitness* à medida que vão sendo executadas as iterações. Uma atenção deve ser dada aos gráficos de pureza e de entropia na análise de instabilidades. A instabilidade mencionada refere-se ao comportamento oscilante do valor da métrica de qualidade. Ou seja, quando o comportamento do resultado da métrica analisada não é constante crescente ou decrescente no decorrer das

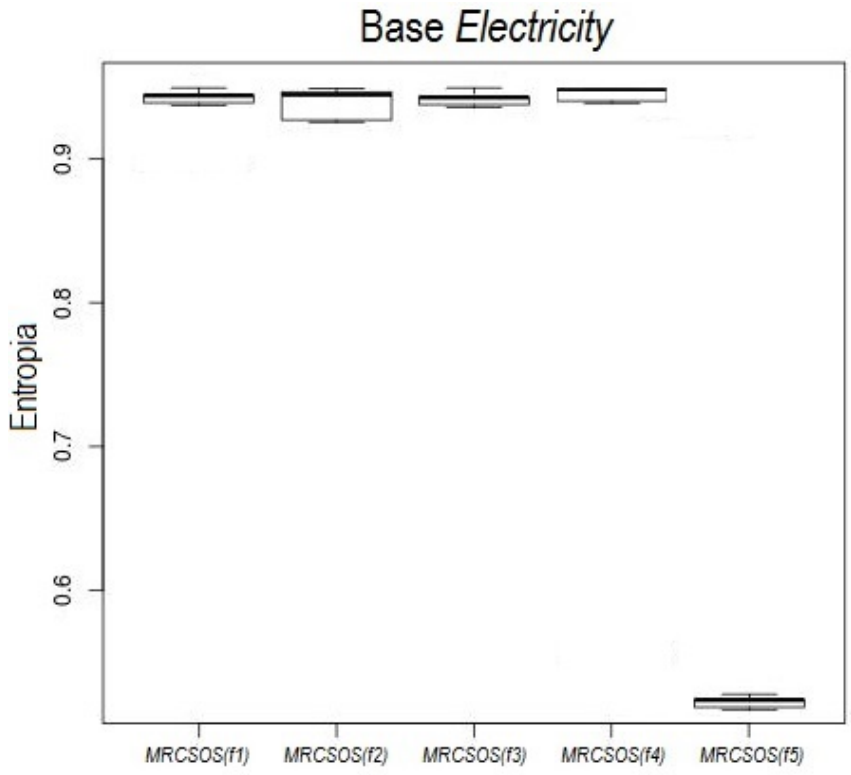

Figura 11 – Gráfico *Boxplot* com resultados de entropia obtidos na base *Electricity*. O eixo-x representa o valor de entropia e o eixo-y representa a versão de *MRCSOS*.

iterações (comportamento não-monotônico). Quando isso ocorre, indica que a melhoria da função *fitness* nem sempre reflete na melhoria da qualidade do agrupamento (MENEZES; PARPINELLI, 2016a). Por vezes, em alguns gráficos de pureza e entropia haverá um marcador indicando tais instabilidades. Além disso, deve-se atentar para o resultado final da métrica de qualidade na última iteração do processo verificando se existe algum resultado nas iterações anteriores melhor que o resultado final. O gráfico da diversidade genotípica auxilia a análise e informa quanto o processo de otimização ainda possui capacidade para evoluir. Desta maneira, se a diversidade genotípica não estiver próxima de 0 e a função *fitness* não estiver estagnada, o processo de otimização pode evoluir em mais iterações. Os próximos 5 cenários apresentam o comportamento das funções *f1, f2, f3, f4* e *f5* na base *Magic*.

Nos resultados da função *f1* na base *Magic*, conforme Figura 14, percebe-se que mesmo com melhora constante da função *fitness* (comportamento assintótico) existe instabilidade acentuada na pureza entre as iterações 0 e 18. Próximo da iteração 8 é o ponto no processo de otimização onde se apresenta o melhor resultado de pureza. Entretanto, nas iterações seguintes ocorre piora sistemática da pureza. De forma semelhante, o gráfico da entropia também apresenta instabilidade entre as iterações 0 e 18 além de apresentar na iteração 8 o melhor resultado de entropia com perda sistemática posterior. Percebe-se

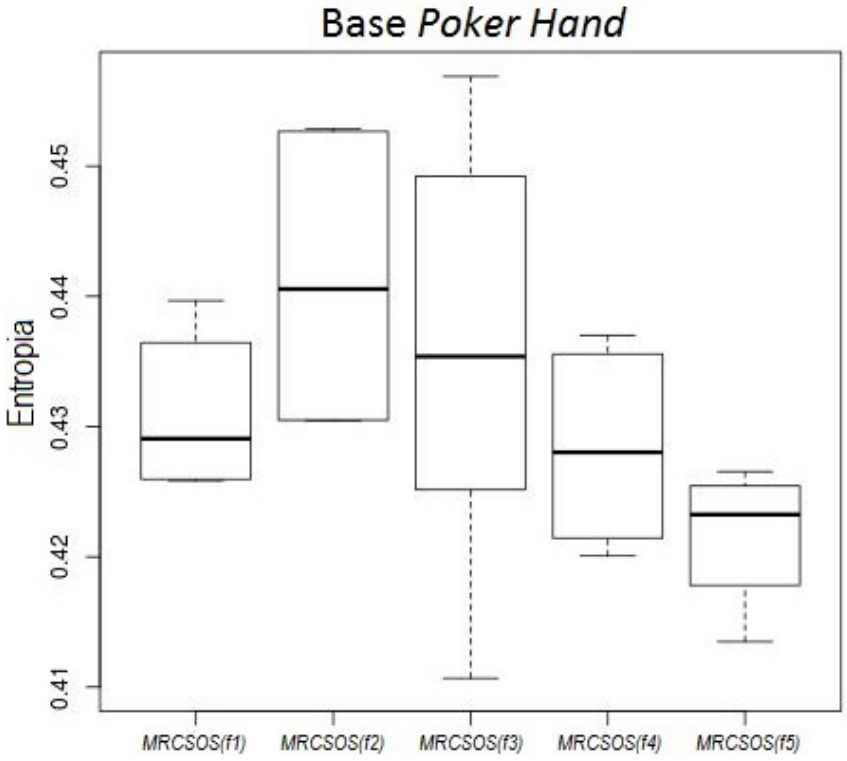

Figura 12 – Gráfico *Boxplot* com resultados de entropia obtidos na base *Poker Hand*. O eixo-x representa o valor de entropia e o eixo-y representa a versão de *MRCSOS*.

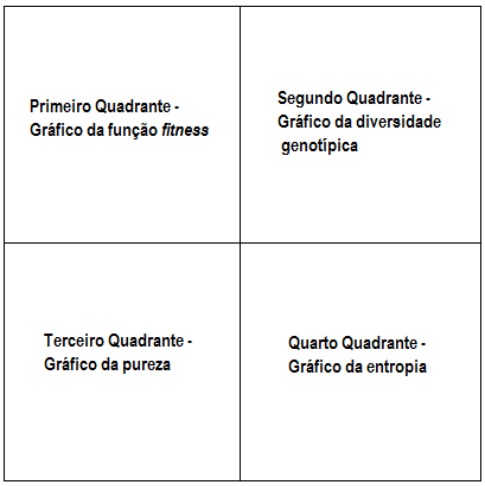

Figura 13 – Disposição dos gráficos da função *fitness*, diversidade genotípica, pureza e entropia para análise de correlação.

na sequência das iterações que a qualidade final do agrupamento foi pior que o melhor resultado encontrado no decorrer do processo considerando as duas métricas. No gráfico da diversidade genotípica pode-se perceber que, no decorrer do processo, a diversidade é decrementada constantemente e próximo da iteração 87 não existe mais diversidade genotípica. Visto que, ao final das 100 iterações não existe mais diversidade genotípica e

a função *fitness* demonstra-se estar estagnada, pode-se entender que o sistema convergiu para um atrator local.

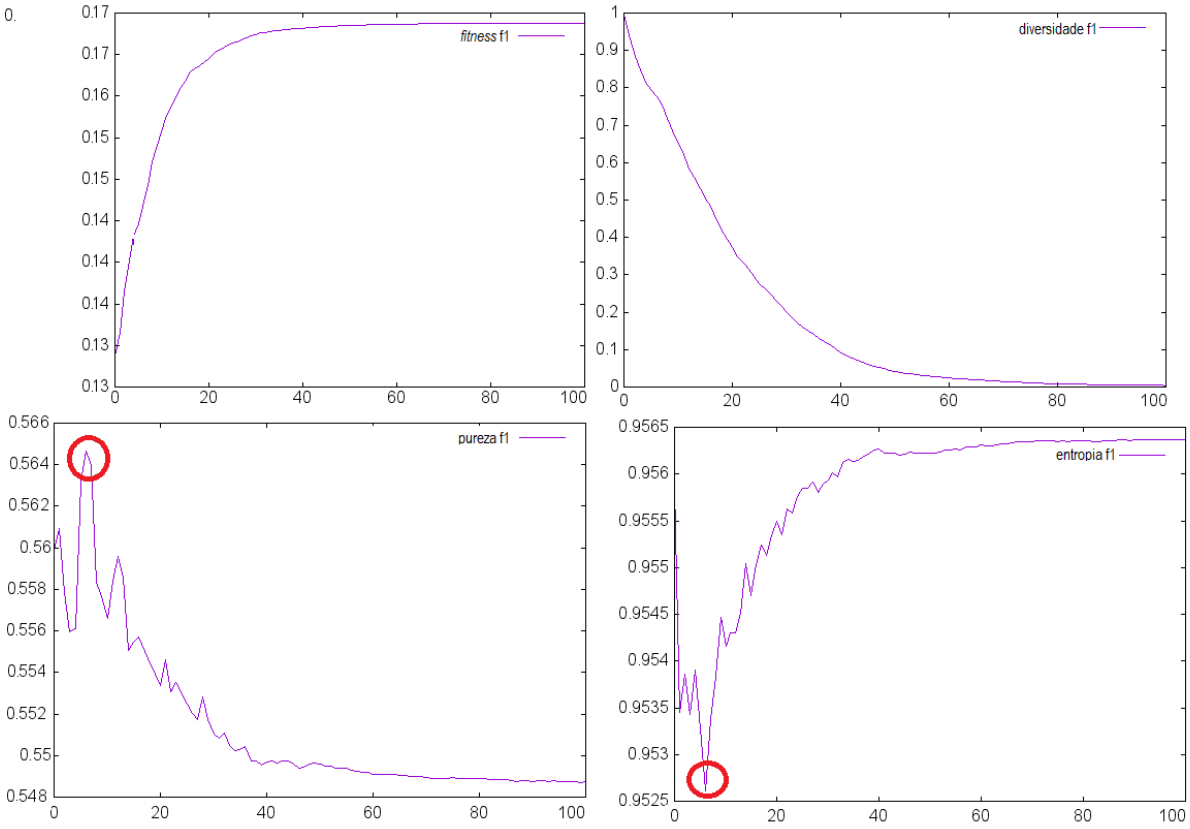

Figura 14 – Gráficos da evolução do *fitness*, diversidade genotípica, pureza e entropia para a função *f1* na base *Magic*.

Na Figura 15, os gráficos da função *f2* demonstram que não ocorre instabilidade na pureza. A pureza sempre melhora à medida que o *fitness* melhora. A entropia apresenta uma pequena instabilidade entre as iterações 0 e 6. Posterior a iteração 6, a entropia melhora constantemente assim como o *fitness* e a pureza. A diversidade genotípica é decrementada constantemente e desde a iteração 60 até o final do processo de otimização se mantém sem diversidade genotípica. Observa-se também que os resultados das duas métricas de qualidade obtidas ao final do processo são as melhores obtidas considerando todo o processo de otimização.

Percebe-se que os cursos da pureza e do *fitness* da função *f3*, Figura 16, possuem evolução semelhantes aos da função *f2*. Da mesma forma ocorre com a entropia. A pureza e entropia dessa função não apresentam instabilidades e além disso, sempre que ocorre a evolução da função *fitness* no decorrer das iterações tanto a pureza quanto a entropia melhoram. A qualidade dos agrupamentos, entropia e pureza, formados no final do processo utilizando a função *f3* é a melhor encontrada considerando todo o processo de otimização. A diversidade genotípica não existe desde a iteração 60 indicando a baixa capacidade do

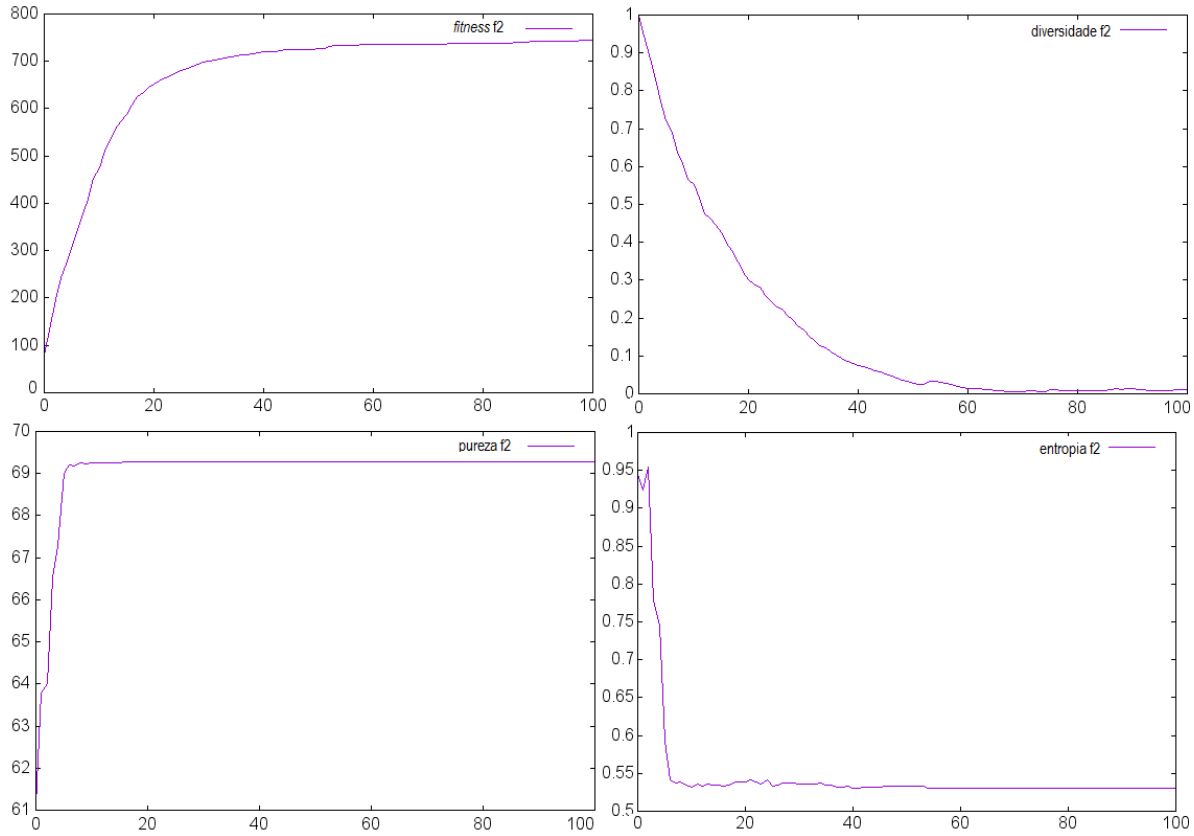

Figura 15 – Gráficos da evolução do *fitness*, diversidade genotípica, pureza e entropia para a função *f2* na base *Magic*.

sistema em evoluir e assim encontrar melhores soluções de agrupamento.

Analisando os gráficos da função *f4*, que é de minimização, na Figura 17, é possível verificar que até próximo da iteração 40 existe uma instabilidade acentuada tanto na pureza quanto na entropia mesmo com a evolução constante do *fitness*. Nos dois gráficos de qualidade, o melhor resultado é obtido na iteração 4. Assim como a função *f1*, na sequência das iterações, os melhores resultados de pureza e entropia obtidos anteriormente não são conservados até o final das 100 iterações. Em relação a diversidade genotípica, a inexistência da mesma ocorre a partir da iteração 45. Nessa iteração a função *fitness* também já convergiu.

A Figura 18 mostra os gráficos para a função *f5*. No gráfico de pureza da função *f5* ocorre um incremento abrupto até a iteração 4. A média da pureza encontrada por essa função na iteração 4 é a melhor pureza encontrada e a mesma é conservada até finalizar todas as iterações. A entropia dessa função apresenta uma variação entre as iterações 5 e 15. Contudo, após essa instabilidade a entropia final é a melhor apresentada considerando todas as iterações. No gráfico da diversidade percebe-se que o decremento da diversidade é mais lento que em todas as outras funções anteriores nessa mesma base. Ao final de 100 iterações diversidade final é de aproximadamente 0.35. Visto que após a iteração 80 ainda ocorre melhora na função *fitness* e existe diversidade genotípica o sistema ainda

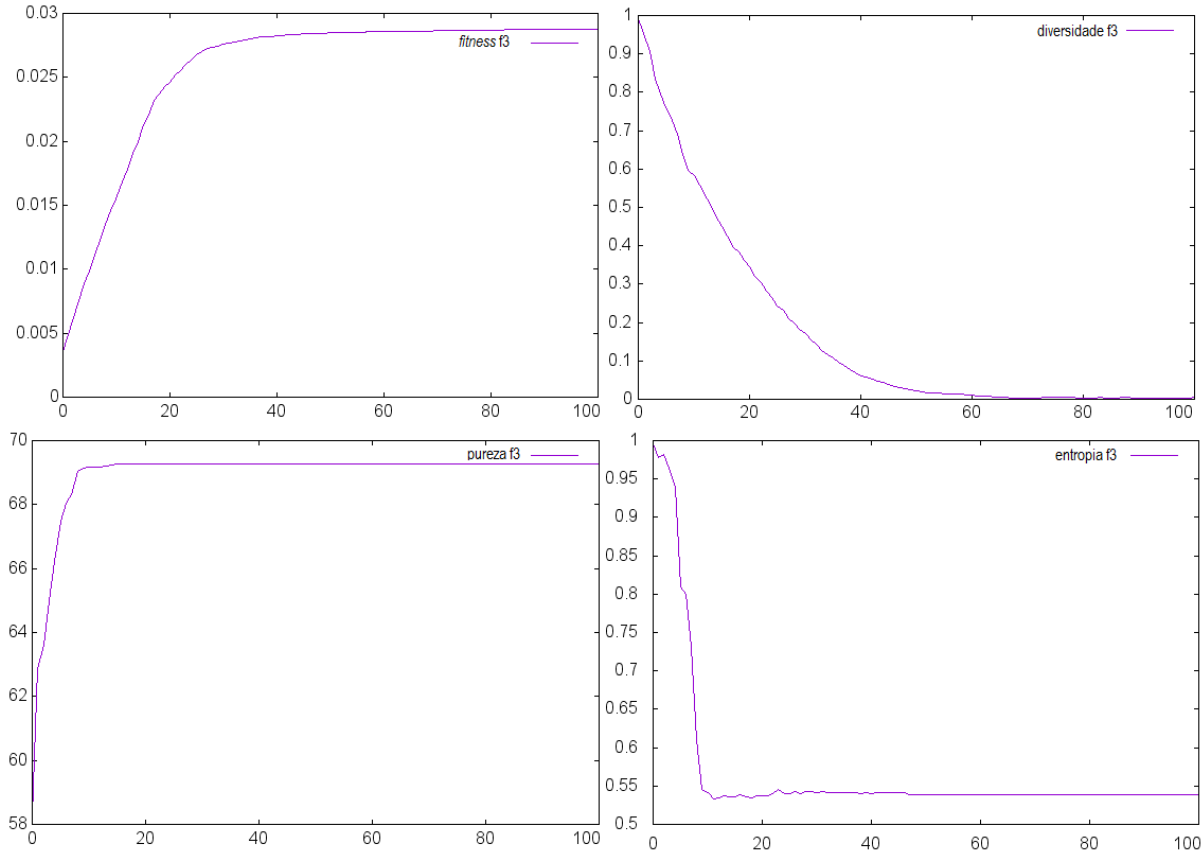

Figura 16 – Gráficos da evolução do *fitness*, diversidade genotípica, pureza e entropia para a função *f3* na base *Magic*.

não utilizou todos os recursos da diversidade para garantir que esses resultados são os melhores possíveis.

Os próximos 5 cenários avaliam o comportamento dos resultados obtidos pelas funções *f1, f2, f3, f4* e *f5* na base *Electricity*.

Na Figura 19, tanto a pureza quanto a entropia dos resultados utilizando a função *f1* na base *Electricity* apresentam o pico de melhor resultado entre as iterações 0 e 3. Após isso ocorre perda de qualidade em ambas as métricas até a iteração 10 seguida de pequenas instabilidades nas próximas iterações. Dessa forma, ambas as métricas de qualidade não conservam a qualidade no decorrer do processo. A diversidade genotípica fica próxima de 0 após a iteração 45. A convergência total do *fitness* pode ser verificada após iteração 80.

O comportamento da pureza e entropia nos resultados da função *f2* na base *Electricity*, conforme Figura 20, apresentam maior instabilidade até a iteração 40. Essas duas métricas de qualidade possuem um comportamento semelhante nessa base com a função *f2* em relação a melhoria da qualidade por iteração. Nas duas métricas ocorre perda de qualidade obtida entre as iterações 2 e 5. Assim, a qualidade final do agrupamento formado não é a melhor encontrada no decorrer do processo. A diversidade genotípica fica

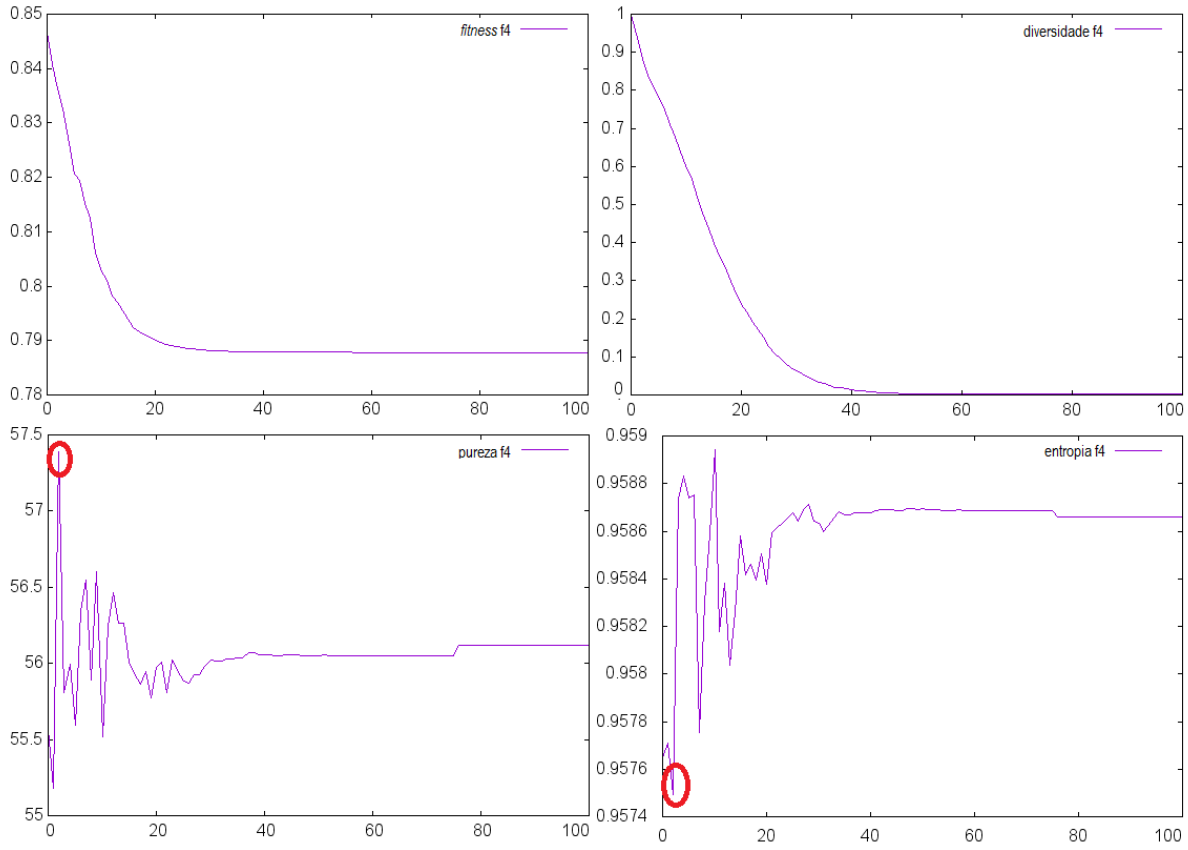

Figura 17 – Gráficos da evolução do *fitness*, diversidade genotípica, pureza e entropia para a função *f4* na base *Magic*.

próxima de 0 na iteração 80 quando também não ocorre mais evolução do *fitness*.

Semelhante aos resultados das funções *f1* e *f2*, a função *f3* também obteve, para as duas métricas, a perda de qualidade no decorrer do processo que foi obtida nas primeiras iterações, conforme Figura 21. No caso da função *f3*, o ganho da qualidade ocorre entre as iterações 1 e 4. Posteriormente ocorre uma perda intensa de qualidade seguida de instabilidades. A diversidade genotípica fica inexistente próximo da iteração 80.

O gráfico da função *f4* na base *Electricity* (Figura 22) demonstra que o comportamento da qualidade das duas métricas é mais estável do que para as funções *f1*, *f2* e *f3*. Mesmo assim também ocorre perda de qualidade adquirida anteriormente considerando as duas métricas assim como ocorreu nas funções *f1*, *f2* e *f3*. A pureza perde um pequeno percentual no resultado final em relação a pureza próxima da iteração 60. Pode-se dizer que esse percentual da perda de pureza não é muito significativo, contudo a melhor pureza não é conservada até o final do processo de agrupamento. A entropia tem uma perda mais significativa desde a iteração inicial. Como a diversidade genotípica é inexistente próximo da interação 40 e nessa iteração a função *fitness* já convergiu entende-se que não é necessário processar mais iterações para obter melhores resultados.

A função *f5* é a única função na base *Electricity* cujos melhores resultados de

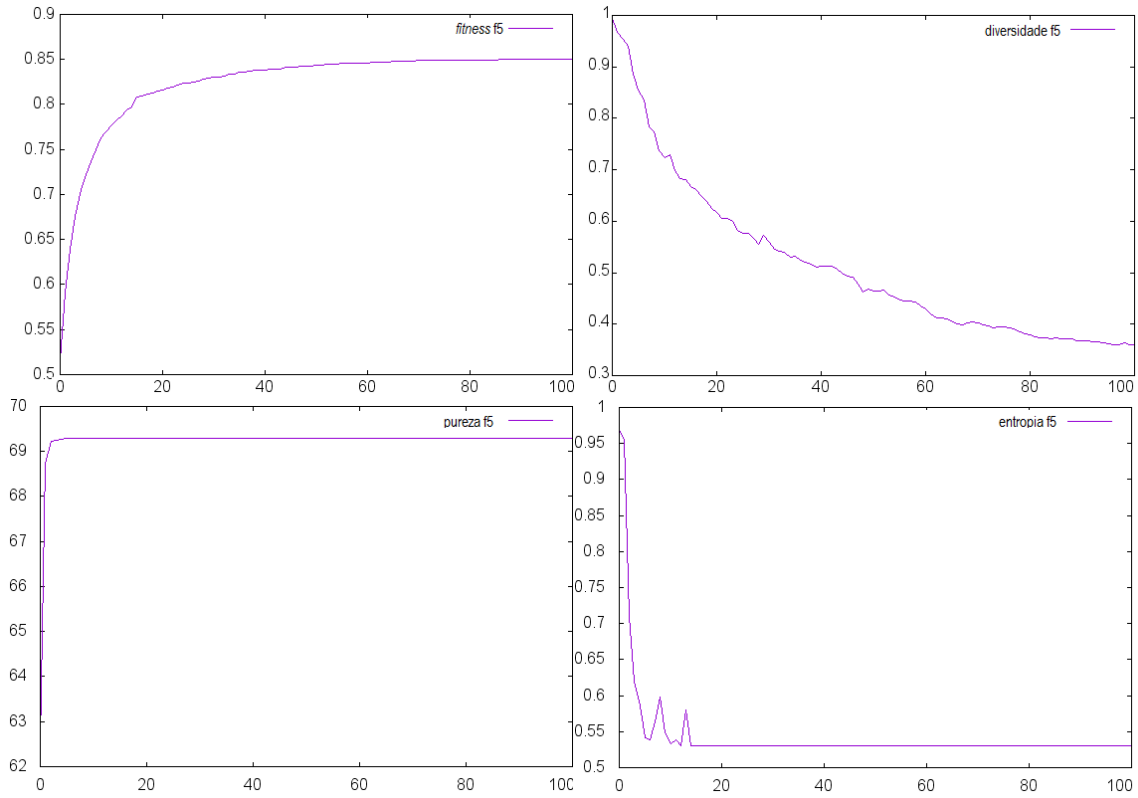

Figura 18 – Gráficos da evolução do *fitness*, diversidade genotípica, pureza e entropia para a função *f5* na base *Magic*.

pureza e entropia são mantidos até o final do processo, ou seja, não ocorre perda de qualidade. Para ambas as métricas de qualidade os melhores resultados são encontrados com essa função até a iteração 20, conforme Figura 23. A pureza dessa função não apresenta nenhuma instabilidade. Ao contrário da pureza, a entropia apresenta instabilidade nas iterações iniciais, até a iteração 17. Apesar disso, a entropia obtida no final do processo é a melhor entropia considerando todas as iterações. A diversidade genotípica na iteração 100 ainda existe e a função *fitness* ainda não parou de convergir. Isso indica que o sistema ainda possui a capacidade de evoluir se for aumentado o número de iterações.

Os próximos 5 cenários analisam o comportamento das funções *f1*, *f2*, *f3*, *f4* e *f5* na base *Poker Hand*.

A Figura 24 exibe os gráficos com os resultados da função *f1* na base *Poker Hand*. Tanto a entropia quanto a pureza apresentam o comportamento bastante instável. A pureza no decorrer do processo não foi superior a pureza inicial. O melhor resultado de entropia ocorre entre as iterações 15 e 19 mas nas iterações posteriores ocorre a perda desse resultado. Percebe-se no comportamento do *fitness* que na iteração 100 a função ainda não parou de convergir. A diversidade genotípica varia muito pouco após a iteração 15 mantendo um valor próximo do valor 0,5 na iteração 100. Isso indica que o sistema tem capacidade para evoluir em mais iterações no processo de otimização. Esse comportamento

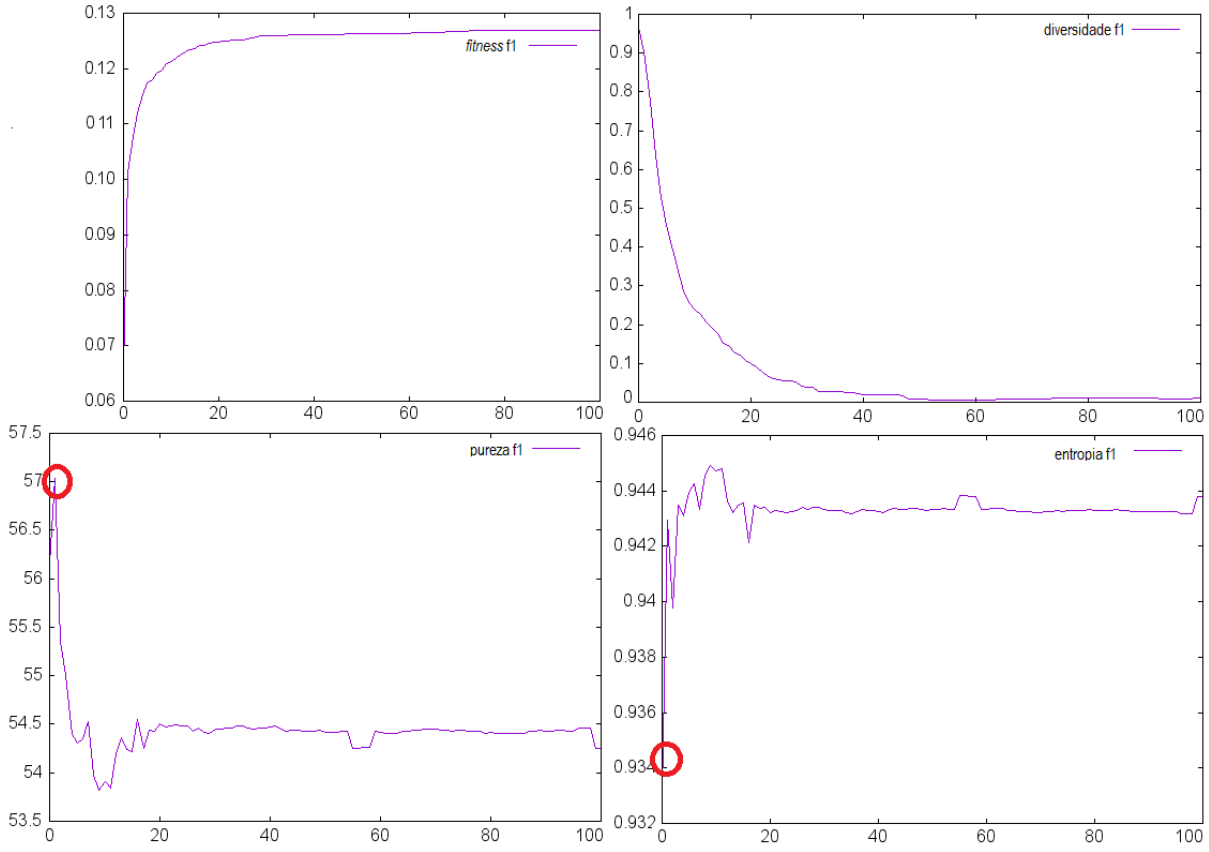

Figura 19 – Gráficos da evolução do *fitness*, diversidade genotípica, pureza e entropia para a função *f1* na base *Electricity*.

é semelhante em todas as funções da base *Poker Hand*.

A pureza da função *f2* obteve o comportamento crescente na base *Poker Hand* mesmo apresentando instabilidade, conforme Figura 25. Apesar desse comportamento, a pureza obtida ao final do processo não é a melhor encontrada visto que próximo da iteração 97 ocorre o ápice com a melhor pureza e posteriormente, na iteração 99, ocorre a perda da pureza. A entropia é bastante instável alcançando o melhor resultado próximo da iteração 7 . Contudo, no final do processo a entropia é pior que a entropia obtida na iteração 7. A diversidade genotípica é de 0,3 na iteração 100 indicando que ainda existe um pouco de diversidade.

Analisando a função *f3*, o gráfico de pureza exibido na Figura 26 demonstra que na iteração 23 ocorre o melhor resultado. Após isso ocorrem algumas instabilidades mas é perceptível que a tendência do comportamento é ascendente. Apesar disso, a pureza na iteração 100 é inferior que a pureza obtida próximo da iteração 23. A entropia alcança o melhor resultado próximo da iteração 3. Nas iterações seguintes ocorre a perda de qualidade da entropia mantendo a instabilidade. A entropia final é inferior a entropia próxima da iteração 3. A diversidade genotípica ainda existe na iteração 100 e é 0,3. Nessa iteração a função *fitness* ainda não finalizou a convergência. Com isso, o sistema

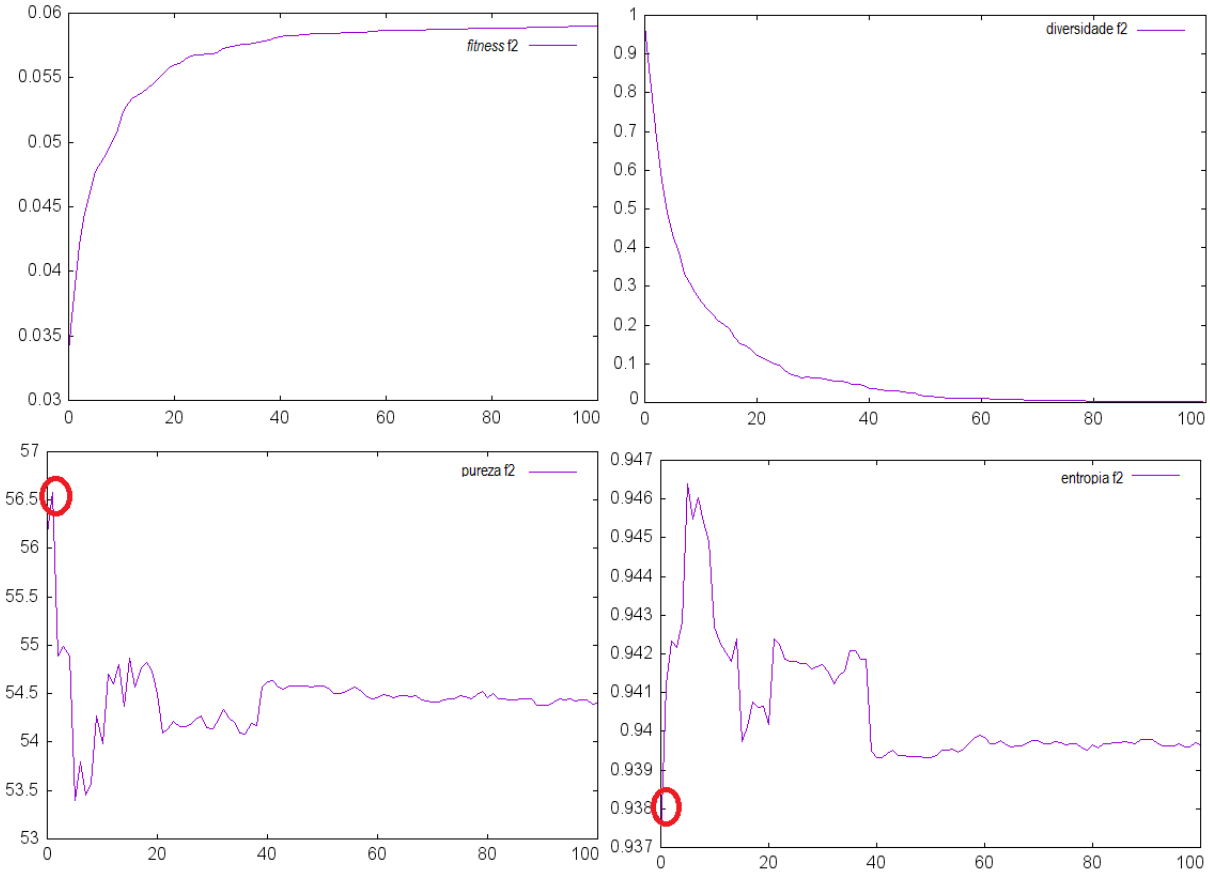

Figura 20 – Gráficos da evolução do *fitness*, diversidade genotípica, pureza e entropia para a função *f2* na base *Electricity*.

ainda possui capacidade de continuar a evoluir no processo de agrupamento caso seja aumentado o número de iterações.

O comportamento da função *f4* pode ser visto nos gráficos da Figura 27, onde os valores de qualidade inicial em ambas as métricas é melhor que as encontradas no final do processo de otimização. A duas métricas de qualidade apresentam o comportamento bastante instável do início ao fim do processo de otimização. Além disso, a diversidade genotípica permanece elevada, próximo de 0,45, na iteração 100.

Dentre as funções aplicadas, a *f5* é a função que obteve maior média de pureza utilizando a base *Poker Hand* (Figura 28). Além disso, essa função foi a única cuja pureza encontrada no final do processo de otimização é a melhor pureza considerando todo o processo de otimização. Entretanto, a entropia final não foi a melhor entropia obtida em todo o processo. A melhor entropia foi encontrada próxima da iteração 2. A diversidade genotípica final ficou elevada, em torno de 0,77. Visto que a função *fitness* ainda não parou de convergir pode-se perceber que a quantidade de iterações deve ser aumentada para o sistema continuar o processo de otimização.

Conforme visto nos cenários apresentados, o gráfico de *fitness* sempre possui comportamento assintótico crescente nas funções de maximização (*f1*, *f2*, *f3* e *f5*) e decrescente

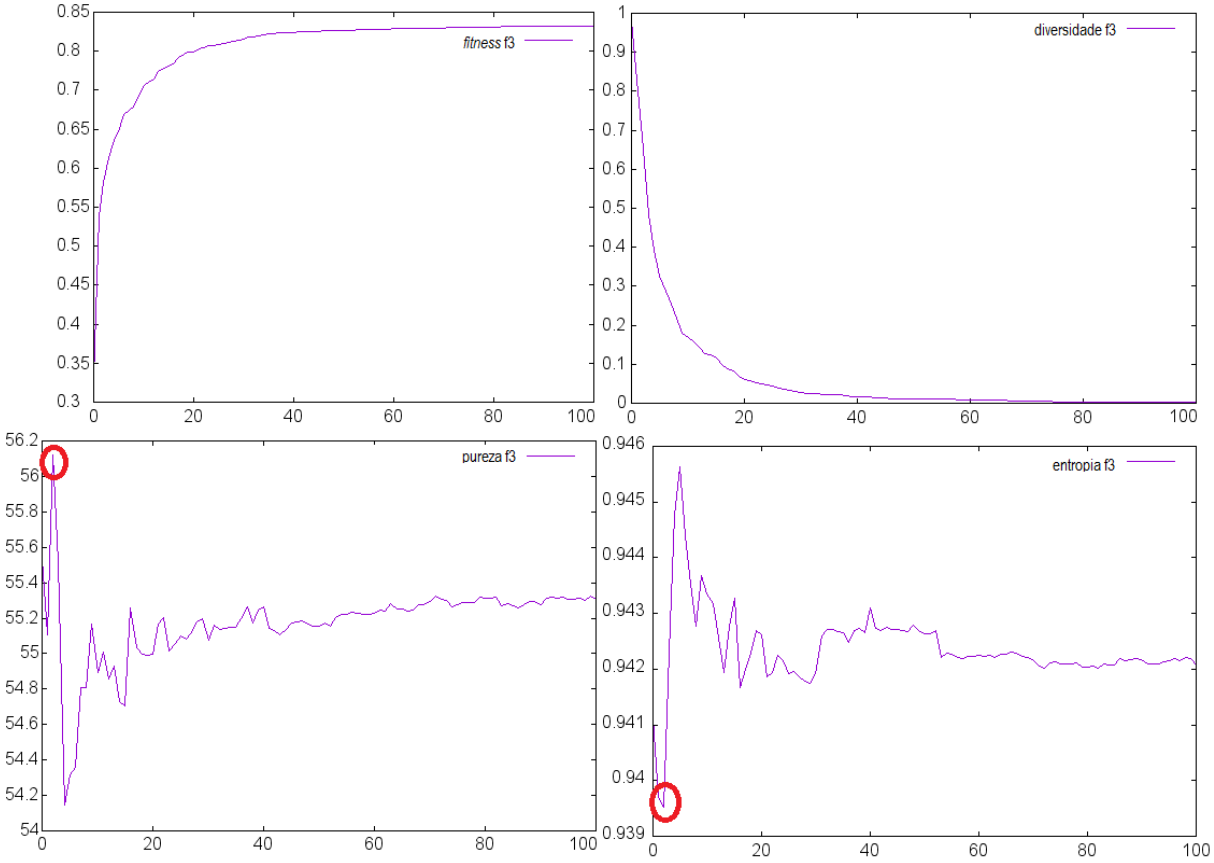

Figura 21 – Gráficos da evolução do *fitness*, diversidade genotípica, pureza e entropia para a função *f3* na base *Electricity*.

na função de minimização (*f4*). O gráfico da função *fitness* exibe a evolução dos organismos no decorrer das iterações. O gráfico da função *fitness* das 5 versões de *MRCSOS* demonstram a não convergência na base *Poker Hand* em 100 iterações. Nas bases *Magic* e *Electricity* o comportamento da função *fitness* em 100 iterações indica que as 5 funções convergiram. A diversidade genotípica é praticamente inexistente ao final das 100 iterações nas bases *Magic* e *Electricity* em todas as versões de *MRCSOS*. Contudo, na base *Poker Hand* a diversidade genotípica se mantem alta nas últimas iterações, especialmente a versão *MRCSOS(f5)* que manteve a maior diversidade genotípica dentre as 5 versões. Dessa forma, entende-se que a quantidade de iterações igual a 100 é suficiente nas bases *Magic* e *Electricity* pois o *fitness* consegue convergir e não existe mais diversidade genotípica. Ao contrário do que ocorre nessas duas bases, a não convergência da função *fitness* e a existência de diversidade genotípica na base *Poker Hand* indicam que 100 iterações não são suficientes para o algoritmo evoluir e explorar a diversidade de soluções ainda existentes. Dessa forma, foi realizado um experimento com 200 iterações, totalizando 80.000 avaliações de *fitness*, em cada uma das 10 execuções exclusivamente para a versão *MRCSOS(f5)* na base *Poker Hand*. A média da pureza obtida foi de 50% com desvio padrão de ± 0.70. Mesmo com 200 iterações a diversidade genotípica ainda se manteve elevada, em

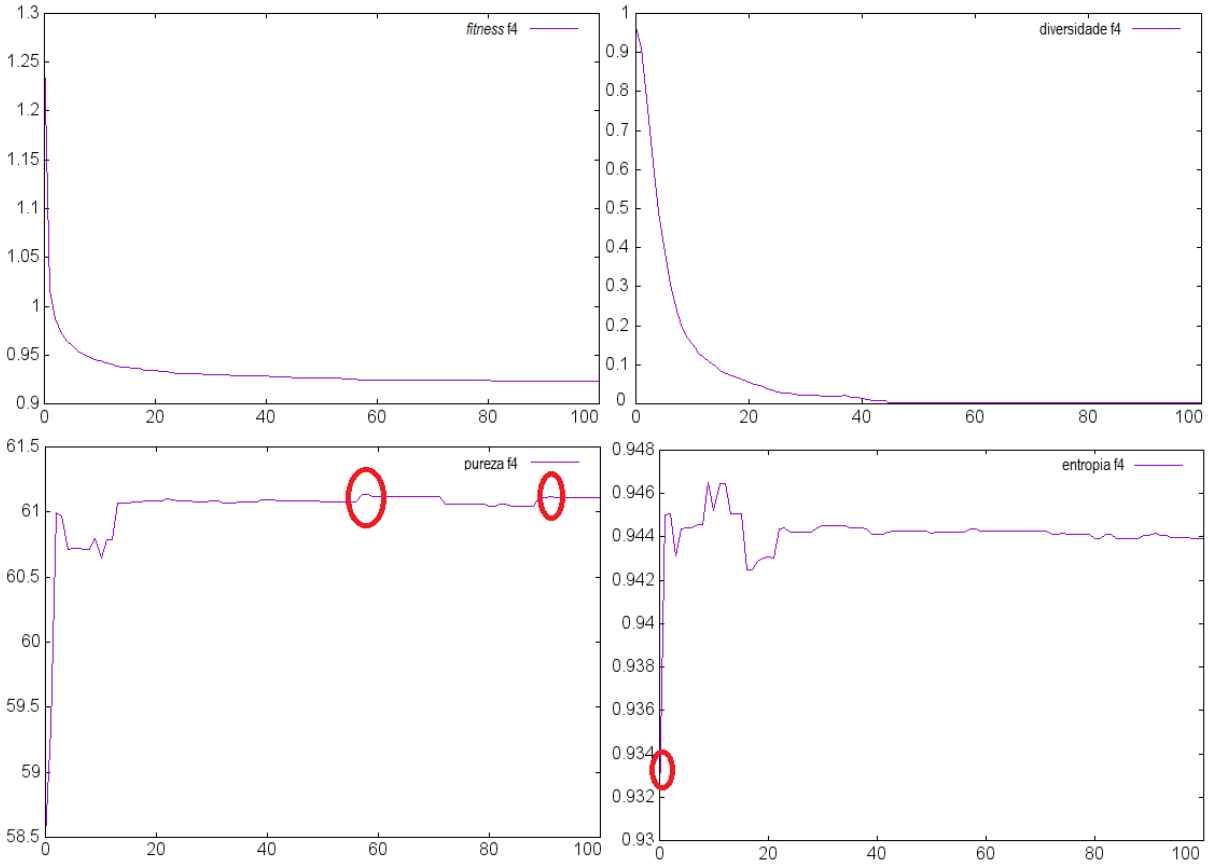

Figura 22 – Gráficos da evolução do *fitness*, diversidade genotípica, pureza e entropia para a função *f4* na base *Electricity*.

torno de 0.71. Isso indica que o processo de otimização ainda poderia alcançar melhores resultados aumentando a quantidade de iterações. Desta forma, os resultados obtidos com *MRCSOS(f5)* com 200 iterações se mostra competitivo com os obtidos pelos algoritmos da literatura (Tabela1).

Em relação ao comportamento da pureza e entropia, percebe-se que, mesmo com a constante evolução da função *fitness*, as versões *MRCSOS(f1)*, *MRCSOS(f2)*, *MRCSOS(f3)* e *MRCSOS(f4)* apresentaram resultados com grau de pureza e entropia pior ao final do processo que obtidas em iterações anteriores nas bases *Electricity* e *Poker Hand*. O mesmo comportamento ocorre com os resultados de pureza e entropia obtidos pelas versões *MRCSOS(f1)* e *MRCSOS(f4)* na base *Magic*. Apenas nessa base as versões *MRCSOS(f2)* e *MRCSOS(f3)* conservaram o melhor resultado de pureza e entropia até o final do processo de agrupamento. Foi percebido que pelo fato das funções de agrupamento *f2* e *f3* utilizarem a razão entre a maior distância intra-grupo e a distância intra-grupo, $\frac{intraD}{max(intraD)}$, no denominador de suas equações, conforme Equação 2.5 e Equação 2.6 respectivamente, as mesmas alcançaram o comportamento de pureza crescente e entropia decrescente assintoticamente na base *Magic*. Exclusivamente nessa base, quando o resultado de $\frac{intraD}{max(intraD)}$ diminui o resultado das funções *f2* e *f3* aumenta indicando a melhoria

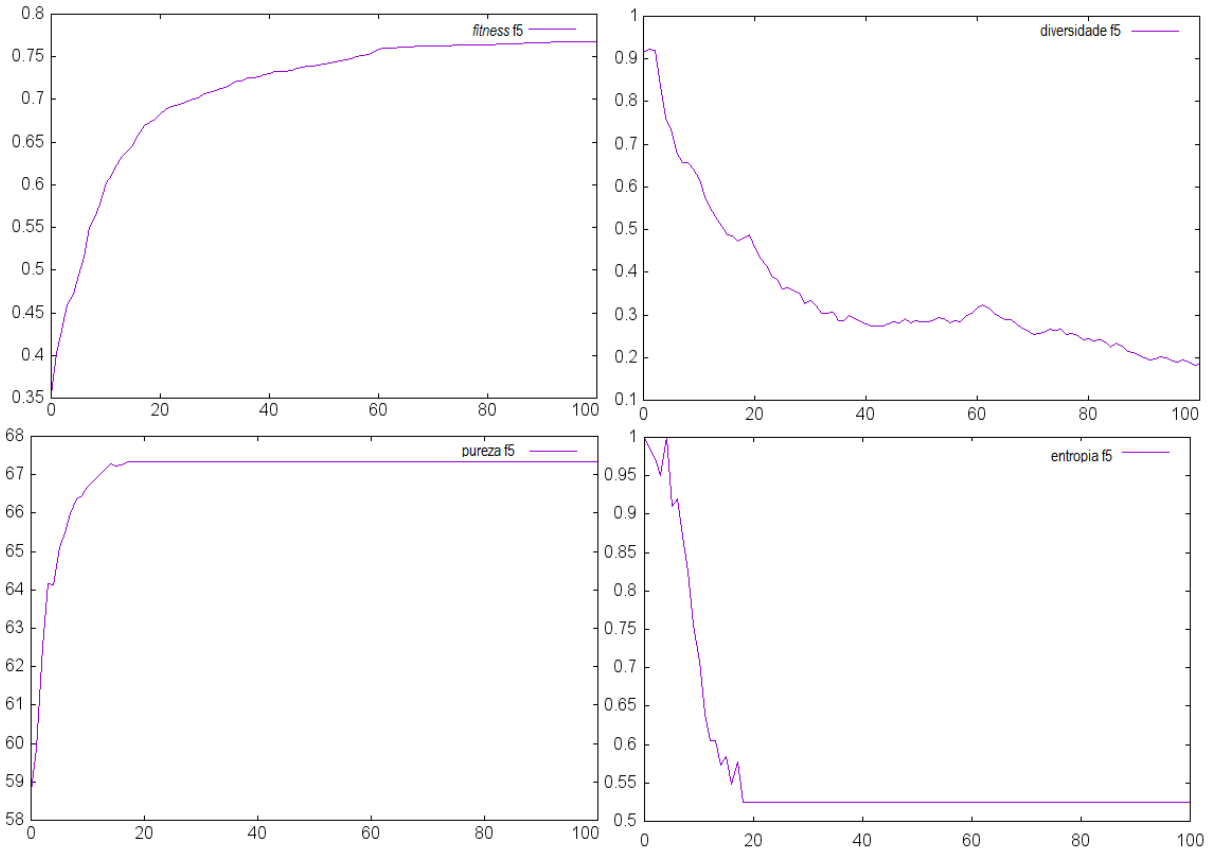

Figura 23 – Gráficos da evolução do *fitness*, diversidade genotípica, pureza e entropia para a função *f5* na base *Electricity*.

da função. Quando isso ocorre, a pureza e entropia do agrupamento também melhoram. Percebe-se que em todos os casos citados onde ocorre a perda de qualidade, a instabilidade está presente no comportamento dos gráficos de pureza e gráficos de entropia. Assim, entende-se que a presença da instabilidade no comportamento dos gráficos de entropia e pureza é um indício que pode ocorrer perda de pureza no decorrer do processo de agrupamento de dados.

A versão *MRCSOS(f5)* é a única a conseguir obter média de pureza final superior do que em qualquer iteração anterior nas bases *Magic*, *Electricity* e *Poker Hand*. A entropia final obtida nas bases *Magic* e *Electricity* pela versão *MRCSOS(f5)* é a melhor entropia encontrada no decorrer do processo. Contudo, na base *Poker Hand*, essa versão apresentou perda de entropia. A instabilidade nos gráficos de entropia e pureza para a versão *MRCSOS(f5)* ocorrem apenas na base *Poker Hand*. Apesar disso, mesmo com instabilidade, a pureza final obtida na base *Poker Hand* é a melhor pureza obtida em todo o processo.

A função *f5* valoriza os agrupamentos que possuem itens de dados cobertos por um centroide que são mais afastados do segundo centroide mais próximo, conforme Equação 2.8. Assim essa função contempla no seu resultado a proximidade com itens similares,

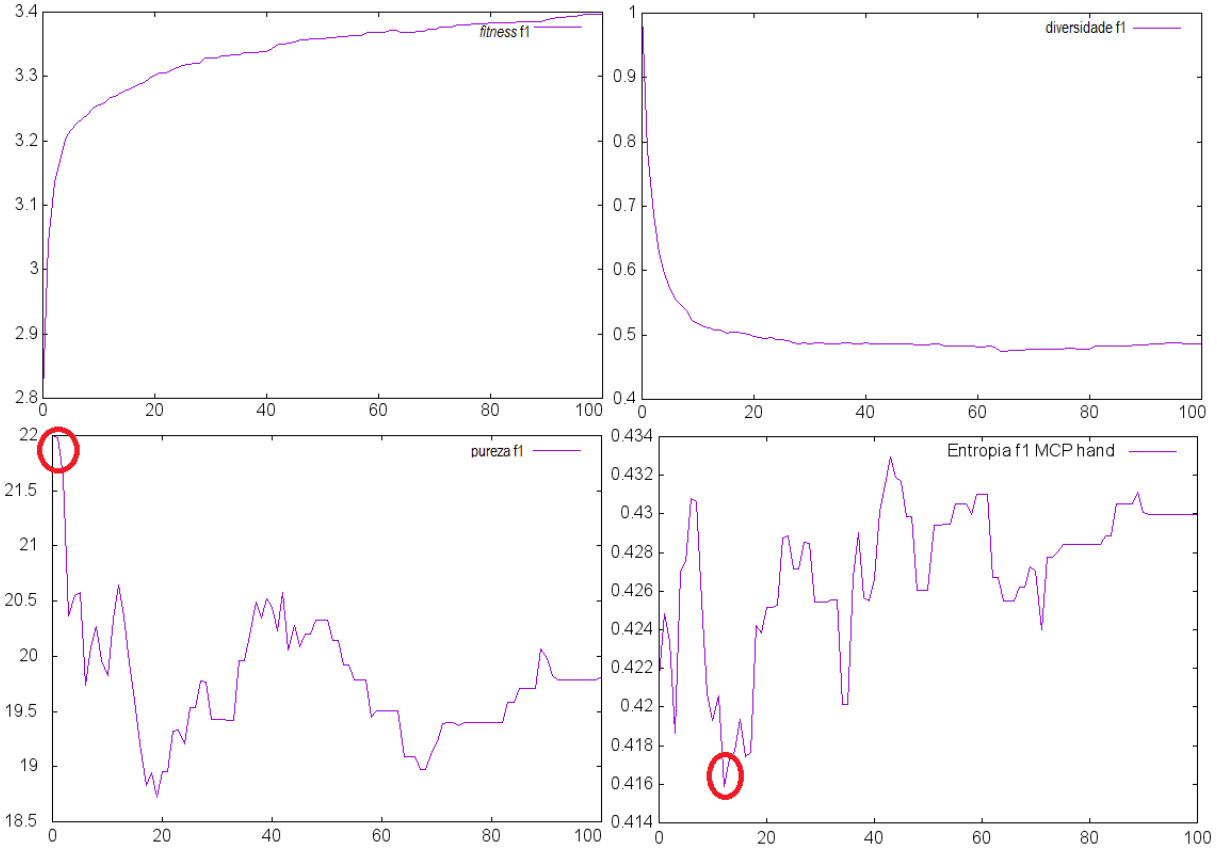

Figura 24 – Gráficos da evolução do *fitness*, diversidade genotípica, pureza e entropia para a função *f1* na base *Poker Hand*.

cálculo do item até o centroide mais próximo e distância com itens não-similares ao considerar a distância com o segundo centroide mais próximo. Atribui-se a adoção dessa estratégia como critério de qualidade de agrupamento para a não ocorrência da perda de pureza na obtenção de seus resultados finais. A influência da distância do item de dado ao segundo centroide mais próximo está presente apenas nessa função. Entende-se que a influência dessa distância contribuiu para que os resultados da versão *MRCSOS(f5)* sejam os melhores resultados de qualidade do que os resultados obtidos pelas outras versões do *MRCSOS*.

Visto que a análise de qualidade das funções *f1*, *f2*, *f3*, *f4* e *f5* foi realizada separadamente para a métrica de pureza e posteriormente para a métrica de entropia nas três bases de dados, a seguir será exibido o gráfico de Pareto para cada base de dados considerando as duas métricas em conjunto. No gráfico de Pareto o eixo-x representa a métrica de pureza e o eixo-y representa a entropia. Cada ponto no plano cartesiano representa a qualidade média da função em questão de acordo com os seus resultados obtidos de entropia e pureza. Dessa forma, analisando o gráfico de Pareto é possível identificar o conjunto de soluções não-dominadas. Esse conjunto é composto por funções cujas soluções não são piores que as outras tanto na pureza quanto na entropia.

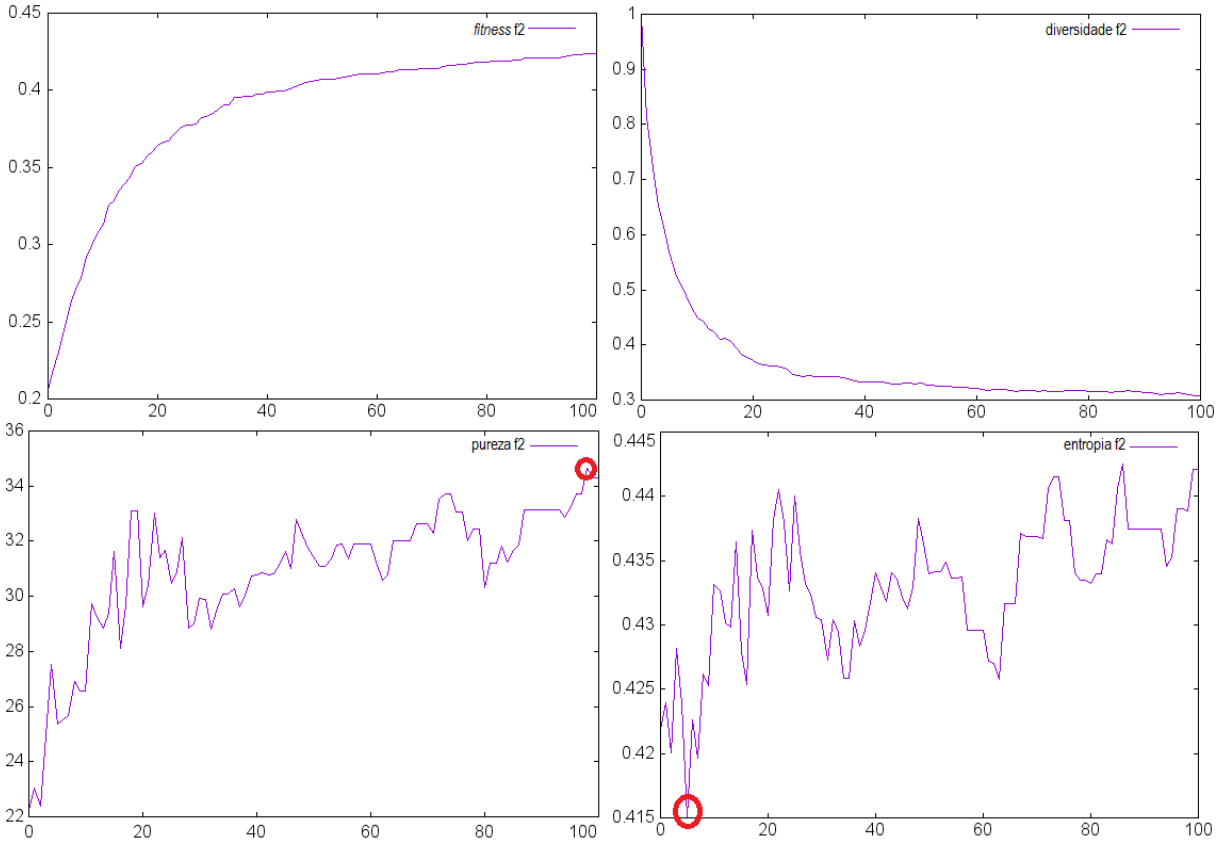

Figura 25 – Gráficos da evolução do *fitness*, diversidade genotípica, pureza e entropia para a função *f2* na base *Poker Hand*.

O gráfico de Pareto para a base *Magic*, exposto na Figura 29, demonstra que o conjunto de soluções não-dominadas para essa base é composta pelas funções *f2*, *f3* e *f5*. Esse conjunto pode ser representado por $P = \{f2, f3, f5\}$. Essas funções apresentam os melhores resultados de entropia e pureza que os obtidos pelas funções *f1* e *f4*. Apesar da função *f4* possuir melhor pureza que a função *f1* não é possível afirmar que a mesma domina a função *f1* pois a entropia dessas duas funções é a mesma.

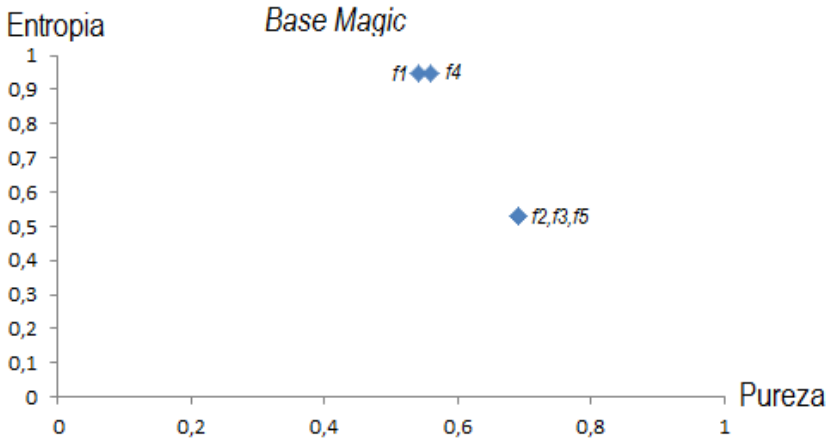

Figura 29 – Gráfico de Pareto para a base *Magic*.

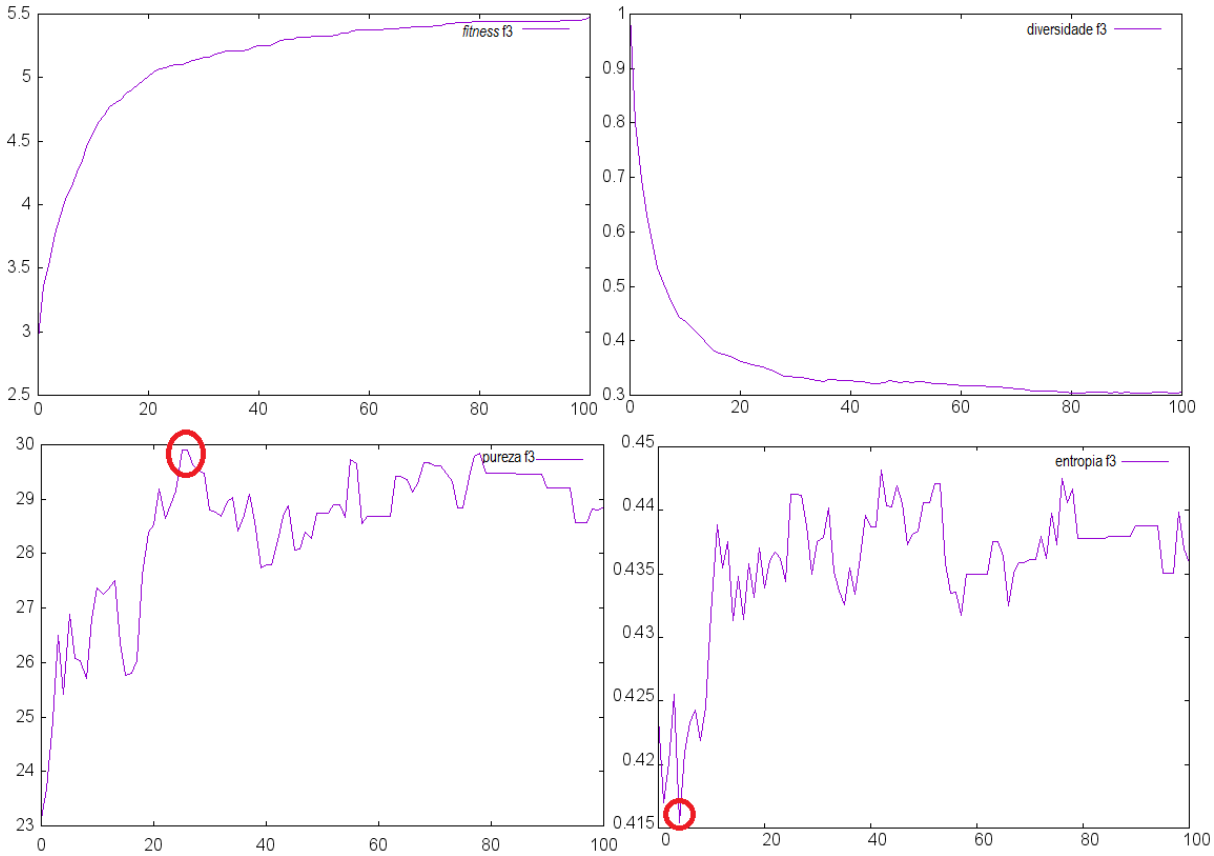

Figura 26 – Gráficos da evolução do *fitness*, diversidade genotípica, pureza e entropia para a função *f3* na base *Poker Hand*.

Na base *Electricity*, conforme Figura 30, o gráfico de Pareto explicita que a função *f5* é a única função que pertence ao conjunto de soluções não-dominadas, $P = \{f5\}$. Tanto a pureza quanto a entropia obtida pela função *f5* é melhor que os resultados obtidos pelas outras quatro funções. Apesar de existir diferença nos resultados de pureza, as funções *f1*, *f2*, *f3* e *f4* apresentam resultados de entropia bem semelhantes. Assim, não é possível afirmar dentre essas quatro funções qual domina as demais.

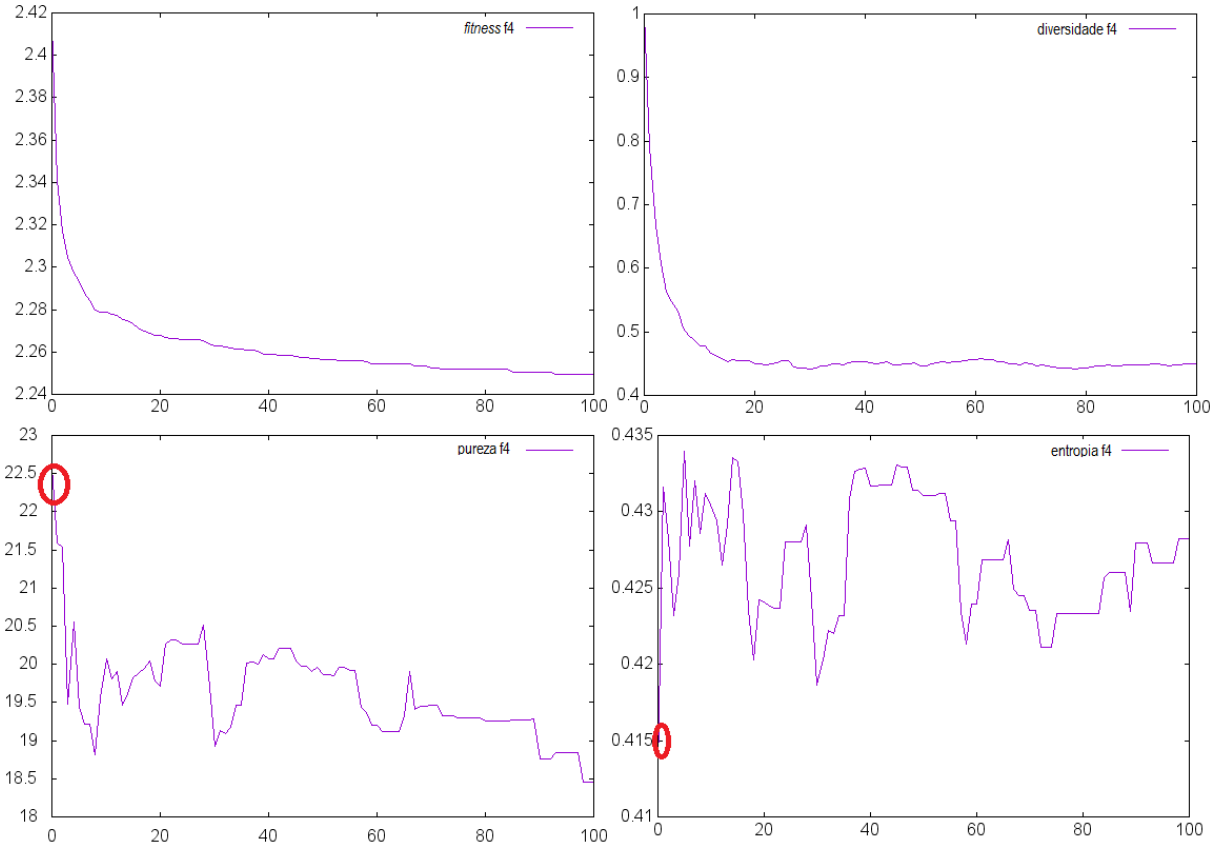

Figura 27 – Gráficos da evolução do *fitness*, diversidade genotípica, pureza e entropia para a função *f4* na base *Poker Hand*.

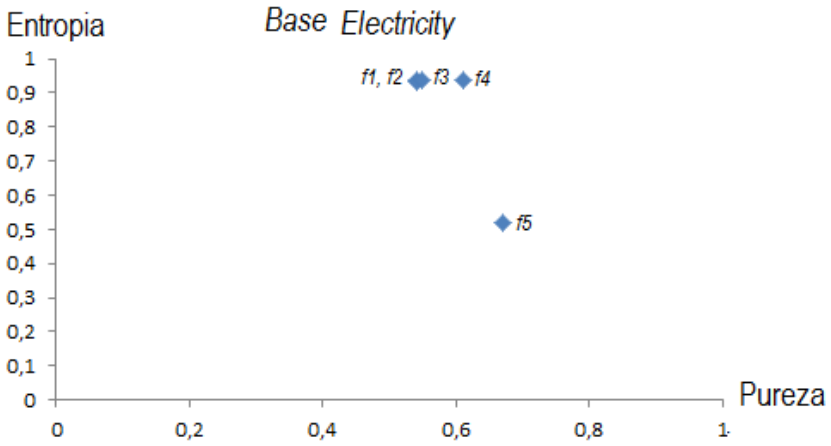

Figura 30 – Gráfico de Pareto para a base *Electricity*.

Baseado no gráfico de Pareto da base *Poker Hand* (Figura 31) é possível definir o conjunto de soluções não-dominadas como sendo apenas a função *f5*. Essa função apresenta o resultado de pureza e de entropia melhor que as outras quatro funções. No gráfico é claramente perceptível que a função *f5* possui o melhor resultado de pureza. A entropia

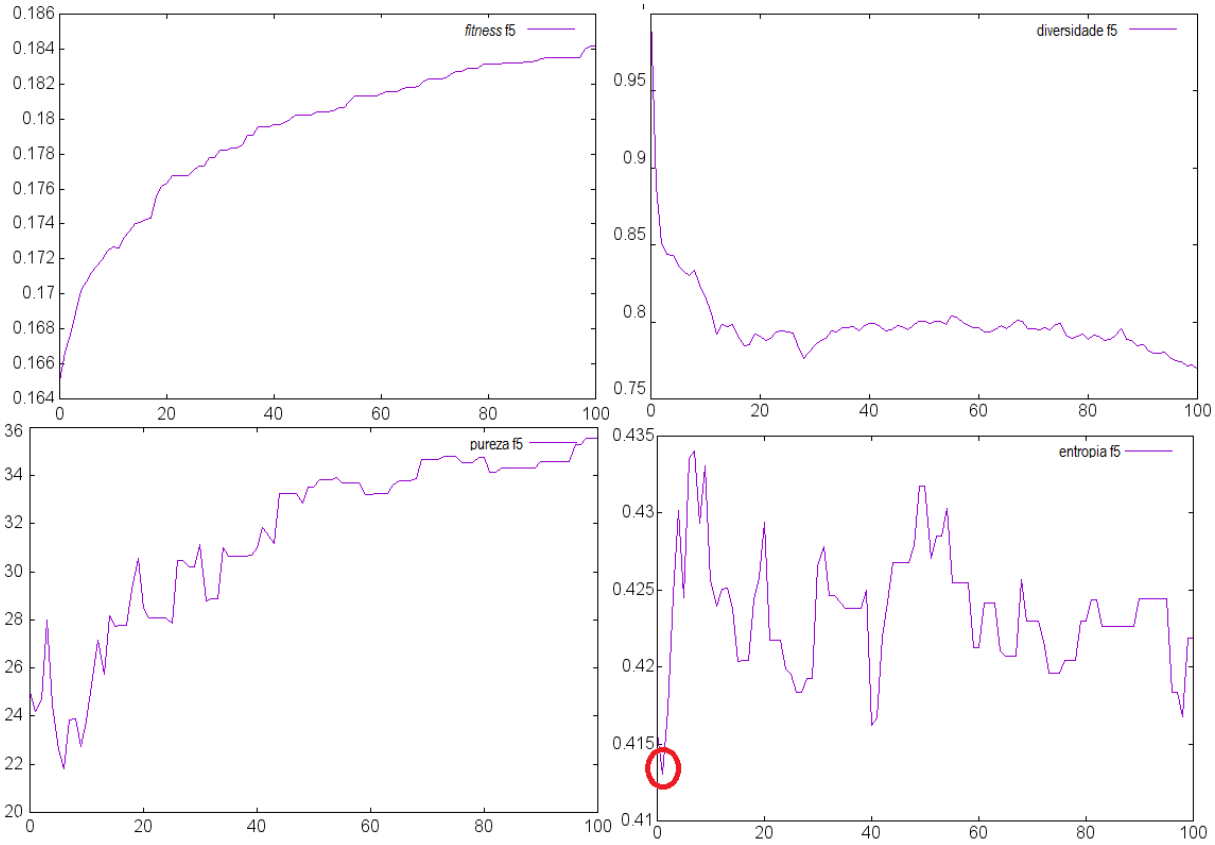

Figura 28 – Gráficos da evolução do *fitness*, diversidade genotípica, pureza e entropia para a função *f5* na base *Poker Hand*.

dessa função é um pouco melhor que a entropia das demais funções mas não é tão explícito perceber isso visualizando apenas o gráfico de Pareto. A percepção que a função *f5* possui o melhor resultado de entropia na base *Poker Hand* pode ser relembrada visualizando a Tabela 2. Logo $P = \{f5\}$ na base *Poker Hand*.

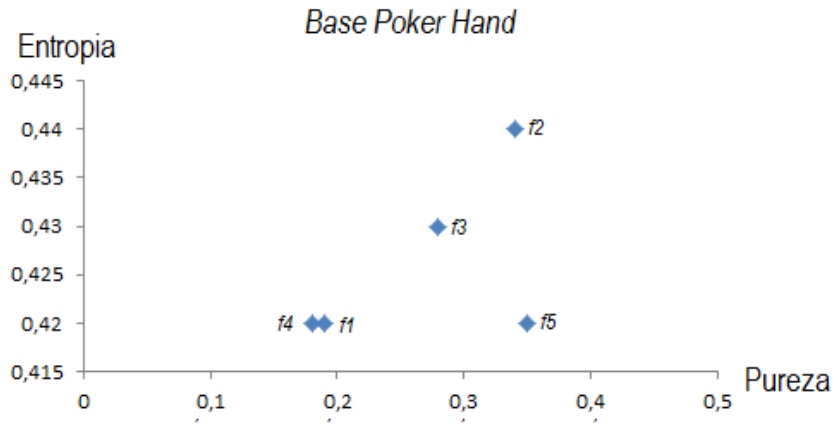

Figura 31 – Gráfico de Pareto para a base *Poker Hand*.

Tabela 3 – Resultado da média de tempo de execução, medido em minutos, da função *f5* nas bases *Magic*, *Electricity* e *Poker Hand*.

Número de computadores	*Magic*	*Electricity*	*Poker Hand*
1	256.7	271.2	1548.6
2	198.3	198.4	1053.6
4	174.4	159.3	811.9
8	161.9	147.2	723.7

4.2 ANÁLISE DE DESEMPENHO

Essa seção visa analisar o desempenho da abordagem proposta, *MRCSOS*, quando a mesma é executada em paralelo. *Speedup* e eficiência são as métricas de avaliação de desempenho utilizadas nessa análise. *Speedup* expressa quanto mais rápida é a execução de um programa em uma arquitetura paralela do que quando executado de maneira sequencial, e a mesma está definida na Equação 2.14 na Seção 2.1.1.2 do Capítulo 2 . A eficiência indica quanto do paralelismo foi explorado no algoritmo e essa métrica é definida na Equação 2.15 na Seção 2.1.1.2 do Capítulo 2. A versão *MRCSOS(f5)* é a versão analisada nesta Seção. A escolha é justificada pelo fato desta apresentar os melhores resultados de qualidade, considerando a pureza e entropia, nas bases analisadas. Conforme visto na seção anterior, a função *f5* é a única função que está no conjunto de soluções não-dominadas em todas as bases analisadas nesse trabalho.

Antes de realizar o cálculo das métricas de desempenho, o tempo de processamento, medido em minutos, gasto por *MRCSOS(f5)* considerando as três bases é apresentado a seguir. Os experimentos que geraram os resultados de desempenho foram executados em um *cluster Hadoop* com 1, 2, 4 e 8 *workers* ou *datanodes*. Os resultados de tempo com a média de 10 execuções estão dispostos na Tabela 3.

Percebe-se, de forma geral, que à medida que o número de computadores é incrementado o tempo de processamento é reduzido.

Na base *Magic*, conforme Figura 32, é possível visualizar a redução do tempo de execução quando a quantidade de computadores utilizados no *cluster Hadoop* é aumentada.

A redução de tempo é representada pela diferença de tamanho das colunas nos gráficos. A redução de tempo apresentada na base *Magic* é menor que a apresentada na base *Electricity*, conforme Figura 33.

Da mesma forma, o ganho de tempo de processamento utilizando a base *Poker Hand*, conforme Figura 34, é maior que o ganho apresentado utilizando a base *Electricity*.

Percebe-se que a redução do tempo de execução utilizando a maior base experimentada, base *Poker Hand*, foi a maior obtida nos experimentos. Isse fato vai de encontro com a recomendação da utilização do *Hadoop* em grandes bases de dados.

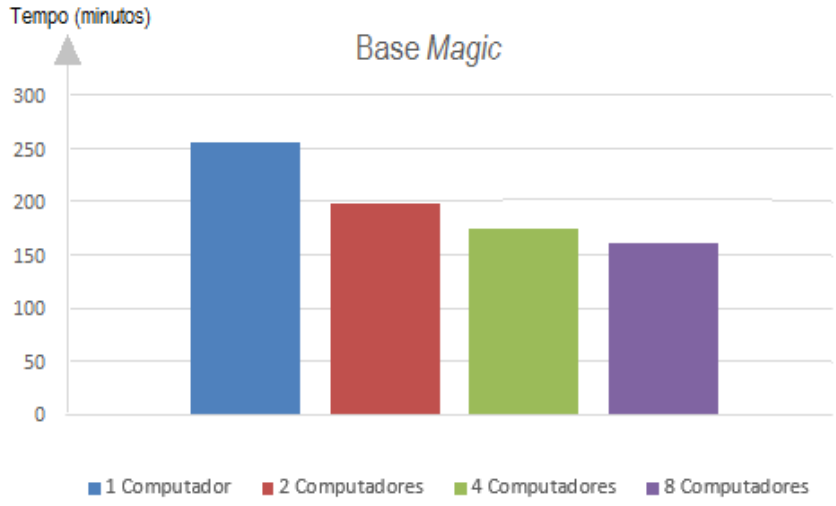

Figura 32 – Média de tempo de execução em minutos da função *f5* na base *Magic*.

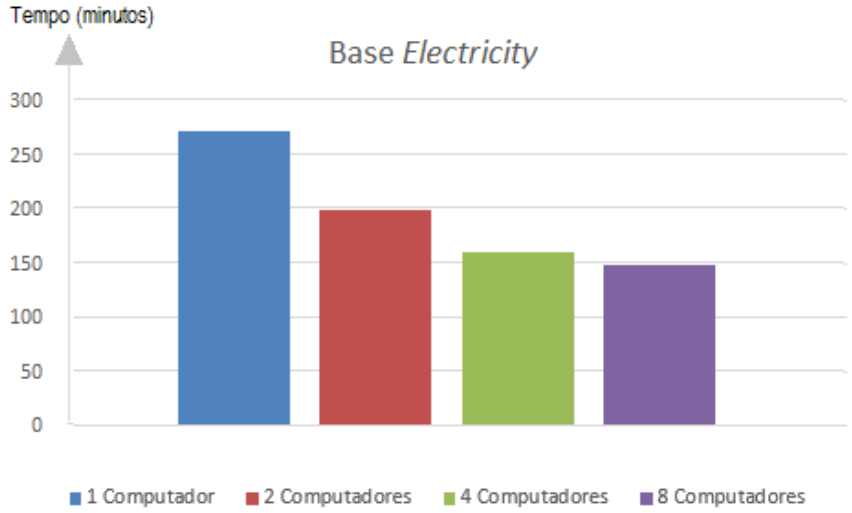

Figura 33 – Média de tempo de execução em minutos da função *f5* na base *Electricity*.

Tabela 4 – Tabela com resultados de *Speedup* da função *f5* nas bases *Magic*, *Electricity* e *Poker Hand*.

Número de computadores	*Magic*	*Electricity*	*Poker Hand*
2	1.29	1.36	1.48
4	1.47	1.70	1.90
8	1.59	1.84	2.14

Com os tempos de processamento obtidos por *MRCSOS(f5)* utilizando 1, 2, 4 e 8 computadores para cada uma das três bases foram caculados os *speedups* apresentados na Tabela 4.

Nas três bases de dados, à medida que os recursos são adicionados ao processamento de *MRCSOS(f5)* o valor do *speedup* sempre aumenta. Esse aumento indica que, com a adição de recursos, há um ganho de tempo de processamento em relação ao proces-

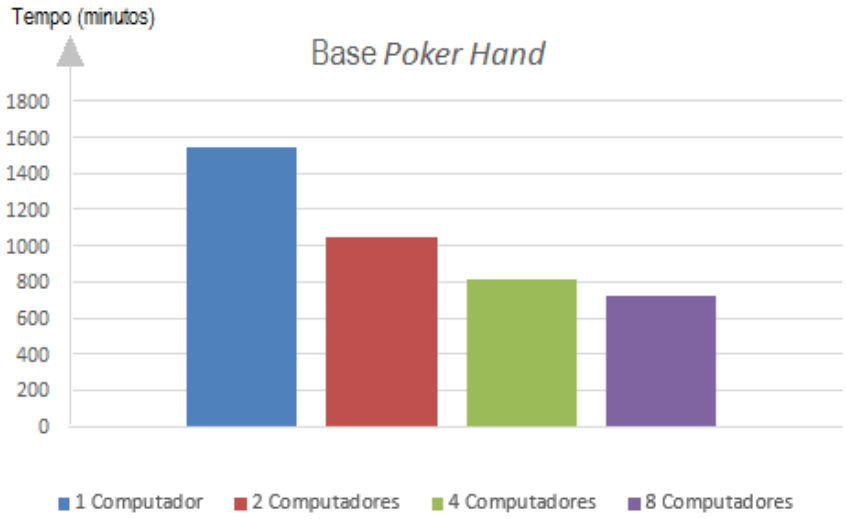

Figura 34 – Média de tempo de execução em minutos da função *f5* na base *Poker Hand*.

samento com menos recursos. Além disso, o fato de todos os valores de *speedup* exibidos na Tabela 4 possuírem valores maior que 1 evidencia que os experimentos em paralelo foram executados mais rápidos que de maneira sequencial.

Percebe-se ainda que a base *Poker Hand* obteve melhor *speedup* em todas as configurações de recursos utilizados.

A Figura 35 explicíta os resultados de *speedup* da Tabela 4 de forma visual. Essa figura facilita na percepção da superioridade do *speedup* obtido na base *Poker Hand* em relação às bases *Magic* e *Electricity*. O *speedup* teórico representa o máximo esperado por um sistema de múltiplos processadores em sua execução. No *speedup* teórico, quando o recurso utilizado para executar um processo é aumentado em 100% espera-se que o tempo de processamento para executar esse mesmo processo seja reduzido em 50%. O *speedup* obtido nos experimentos para as três bases não fica próximo do teórico. Na análise desse desempenho percebeu-se que no momento da execução do *Job Map Reduce* não utilizou-se 100% dos recursos disponíveis de todo o *cluster*. Apesar da configuração do *Hadoop* permitir a utilização de todo o recurso para processamento foi visto que quanto menor o *cluster* utilizado nos experimentos maior é o percentual de tempo da execução do *Job Map Reduce* utilizando 100% dos recursos do *cluster*. Por isso nas três bases de dados o maior *speedup* obtido é com 2 computadores.

Esse comportamento pode ser justificado pelo volume de dados. Devido a disponibilidade de recursos e tempo necessário para executar os experimentos no laboratório, não pode-se utilizar bases com maior volume de dados. Recomenda-se que o *Hadoop* seja utilizado com volume de dados maior para que se possa obter melhores resultados de *Speedup*.

Com os resultados de *speedup* e respectivos recursos utilizados foi calculada a efi-

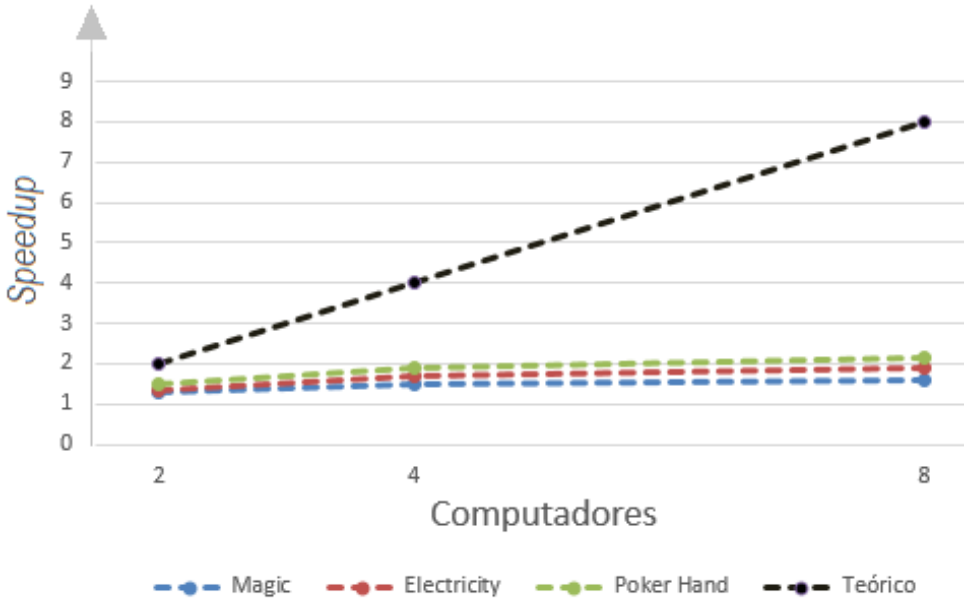

Figura 35 – Gráfico com Speedup de *MRCSOS(f5)* nas bases *Magic*, *Electricity* e *Poker Hand*

Tabela 5 – Tabela com resultados da eficiência da função *f5* nas bases *Magic*, *Electricity* e *Poker Hand*.

Número de computadores	*Magic*	*Electricity*	*Poker Hand*
2	0.64	0.68	0.74
4	0.36	0.42	0.47
8	0.19	0.23	0.26

ciência de *MRCSOS* utilizando as três base de dados. Os resultados da eficiência estão dispostos na Tabela 5.

A eficiência do *speedup* teórico é sempre 1 pois é a utilização de 100% dos recursos no processo. Dessa forma, quanto mais próximo o valor for de 1 melhor será a eficiência da aplicação com arquitetura de processamento distribuído.

Na Figura 36 pode-se perceber que a eficiência do sistema não ficou próxima do teórico, uma vez que a eficiência está diretamente relacionada com o *speedup*. Além disso, a eficiência apresentada na base *Poker Hand* é a mais acentuada. Em todas as bases a eficiência diminui a medida que o número de computadores do *cluster* aumenta, dado os mesmos motivos da análise realizada para o *speedup*.

Mesmo estando longe dos valores ideais teóricos de *speedup* e eficiência, a aplicação do algoritmo *SOS* na arquitetura paralela foi fundamental para a obtenção dos resultados.

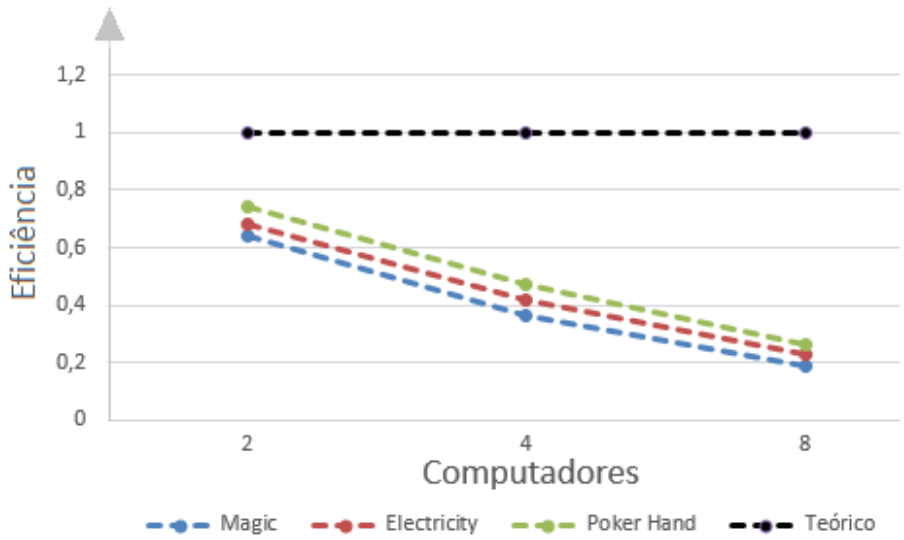

Figura 36 – Gráfico com eficiência de *MRCSOS(f5)* nas bases *Magic*, *Electricity* e *Poker Hand*

5 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Mineração de dados pode proporcionar vários benefícios a diversas áreas que contribuem com o desenvolvimento da sociedade. Desde a área da educação e dos negócios passando pela área de segurança cibernética e nacional, ajudando na detecção de atividade terroristas. Contudo, as características dos dados que podem ser minerados vêm se tornado cada vez mais complexos, como grandes volumes de dados e alta dimensionalidade. Dessa forma, o processo de mineração tende a tornar-se cada vez mais penoso se a tecnologia utilizada na mineração de dados não evoluir no decorrer dos tempos. Visando desenvolver a área de mineração de dados, este trabalho apresenta uma pesquisa que embasa a aplicação de uma abordagem de agrupamento de dados que utiliza o algoritmo bio-inspirado *SOS* projetado na infraestrutura *Hadoop MapReduce* (*MRCSOS*).

Para nivelar o conhecimento sobre essa abordagem foram detalhados neste trabalho os principais conceitos, técnicas e ferramentas utilizadas no desenvolvimento e análise dos resultados do *MRCSOS*. Uma revisão da literatura foi realizada identificando as principais abordagens de mineração de dados que utilizam algoritmos bio-inspirados conciliado com *Hadoop MapReduce*.

A partir da análise das abordagens revisadas, este trabalho contribui com a apresentação de três modelos de projeção de algoritmos bio-inspirados na arquitetura *Hadoop MapReduce*. Esses modelos podem ajudam no melhor entendimento de como alguns algoritmos já são projetados no *Hadoop MapReduce* e assim auxiliar na escolha de um desses modelos em um possível desenvolvimento. Assim, o desenvolvimento de *MRCSOS* adota o Modelo 1 pois o mesmo foi considerado o mais adequado. A vantagem desse modelo está embasada na simplicidade do mesmo em relação aos demais e por conseguir utilizar os benefícios do *framework MapReduce* no processamento dos dados.

Neste trabalho foram apresentados os experimentos realizados com *MRCSOS* utilizando cinco funções (*f1*, *f2*, *f3*, *f4* e *f5*) de *fitness* em três bases de dados no intuito de analisar a média final da pureza e entropia do agrupamento. Além disso, também é analisado o desempenho do algoritmo *SOS* em relação a utilização dos recursos fornecidos pelo *Hadoop MapReduce*. Contudo, a principal contribuição deste trabalho é a análise que correlaciona o gráfico de convergência (iterações vs *fitness*) das diferentes funções de agrupamento com seus respectivos gráficos de diversidade genotípica, pureza e entropia. Em relação a análise comparativa dos resultado finais de pureza, os resultados da versão *MRCSOS(f5)* foi a que obteve os melhores resultados nas bases *Magic* e *Electricity*. Na *Poker Hand*, a versão *MRCSOS(f5)* obteve resultados competitivos aos resultados obtidos pelas abordagens *MRCPSO* e *MRCPSO*. De forma geral, os resultados finais de entro-

pia da versão *MRCSOS(F5)* são os melhores obtidos dentre as outras quatro versões de *MRCSOS*. *MRCSOS(F5)* obteve o melhor resultado de entropia nas bases *Electricity* e *Poker Hand*. Na base *Magic*, a versões *MRCSOS(F5)* e *MRCSOS(F2)* obtiveram resultados equivalentes como sendo os melhores resultados. A superioridade da versão *MRCSOS(F5)* pôde ser evidenciada na análise do gráfico de Pareto construído para cada base de dados analisada onde o mesmo considera entropia e pureza. Nos três gráficos de Pareto a função *f5* é a única que está presente no conjunto de soluções não-dominadas.

A principal contribuição deste trabalho é a análise de correlação entre a função *fitness*, diversidade genotípica, entropia e pureza. Em relação ao comportamento do curso da pureza e entropia das diferentes funções analisadas espera-se que sempre ocorra melhoria contínua à medida que o processo de otimização vai evoluindo. Entretanto, o comportamento dos resultados de pureza das funções *f1*, *f2*, *f3* e *f4* demonstram a perda de pureza pois a pureza final é inferior a pureza obtida em iterações prévias. Esse comportamento evidencia que ocorre perda de pureza de agrupamento devido a não identificação da melhor solução encontrada por parte dessas funções ao longo de todo o processo. Contudo, o comportamento dos resultados de pureza obtidos pela função *f5* demonstram que a mesma consegue conservar o melhor resultado de pureza até o final do processo nas três bases analisadas.

O comportamento dos resultados de entropia das funções *f1*, *f2*, *f3* e *f4* evidencia a perda de entropia nas bases *Magic* e *Electricity*. Nessas bases, a função *f5* consegue conservar a entropia no decorrer do processo. Todas as funções não conservam a entropia na base *Poker Hand*. A conservação da qualidade do agrupamento, pureza e entropia, da função *f5* pode ser atribuída a valorização de agrupamentos que possuem itens de dados cobertos por um centroide que são mais afastados do segundo centroide mais próximo como também por apresentar maior sensibilidade a mudança de centroides dos grupos.

A análise da correlação entre a existência da diversidade genotípica e a convergência da função *fitness* possibilita identificar se a abordagem *MRCSOS* conseguiu utilizar toda a capacidade de otimização que a mesma possui no decorrer do processo de otimização. Com essa análise foi possível identificar que as execuções da versão *MRCSOS(f5)* utilizando 40.000 avaliações ainda possuía capacidade para continuar o processo de otimização. Dessa forma, foi possível aumentar a quantidade de avaliações para 80.000 e assim melhorar o resultado de pureza de 35% para 50%, ainda com possibilidade de melhoria.

O desempenho do algoritmo *SOS* na arquitetura *Hadoop MapReduce* não ficou próximo do linear. A melhor eficiência obtida foi com o *cluster* menor. Ou seja, o *cluster* com dois computadores além do servidor. Na análise de desempenho foi identificado que quanto menor o *cluster* maior é o tempo de utilização de 100% dos recurso disponíveis. Esse comportamento pode ser justificado pela utilização de volume da dados abaixo do

recomendado.

Em relação aos trabalhos futuros, pretende-se analisar a resiliência do sistema utilizando bases dinâmicas nos experimentos. Com isso a pesquisa irá considerar além do volume de dados minerados a velocidade que o volume dos dados aumenta. Além disso, também será desenvolvida uma abordagem do algoritmo *SOS* para realizar agrupamento de dados utilizando o *framework Spark*. Estudos indicam que o *Spark*, que trabalha no topo do sistema de arquivos *HDFS* do *Hadoop*, pode alcançar *speedups* ainda maiores. Além disso, pretende-se ampliar a quantidade de funções de agrupamentos analisadas e também aumentar o número de bases de dados com volume maior.

Referências

AL-MADI, N.; ALJARAH, I.; LUDWIG, S. A. Parallel glowworm swarm optimization clustering algorithm based on mapreduce. In: IEEE. *Swarm Intelligence (SIS), 2014 IEEE Symposium on*. [S.l.], 2014. p. 1–8.

AL-MADI, N.; LUDWIG, S. A. Scaling genetic programming for data classification using mapreduce methodology. In: IEEE. *Nature and Biologically Inspired Computing (NaBIC), 2013 World Congress on*. [S.l.], 2013. p. 132–139.

ALJARAH, I.; LUDWIG, S. A. Parallel particle swarm optimization clustering algorithm based on mapreduce methodology. In: IEEE. *Nature and Biologically Inspired Computing (NaBIC), 2012 Fourth World Congress on*. [S.l.], 2012. p. 104–111.

ALJARAH, I.; LUDWIG, S. A. Mapreduce intrusion detection system based on a particle swarm optimization clustering algorithm. In: IEEE. *Evolutionary Computation (CEC), 2013 IEEE Congress on*. [S.l.], 2013. p. 955–962.

ALJARAH, I. M. *MapReduce-enabled scalable nature-inspired approaches for clustering*. Tese (Doutorado) — North Dakota State University, 2013.

BHAVANI, R.; SADASIVAM, G. S. A novel ant based clustering of gene expression data using mapreduce framework. *International Journal on Recent and, Innovation Trends in Computing and Communication*, v. 2, p. 398–402, 2014.

BHAVANI, R.; SADASIVAM, G. S.; KUMARAN, R. A novel parallel hybrid k-means-de-aco clustering approach for genomic clustering using mapreduce. In: IEEE. *Information and Communication Technologies (WICT), 2011 World Congress on*. [S.l.], 2011. p. 132–137.

BINITHA, S.; SATHYA, S. S. A survey of bio inspired optimization algorithms. *International Journal of Soft Computing and Engineering*, v. 2, n. 2, p. 137–151, 2012.

CHENG, M.-Y.; PRAYOGO, D. Symbiotic organisms search: A new metaheuristic optimization algorithm. *Computers & Structures*, Elsevier, v. 139, p. 98–112, 2014.

CHENG, S.; SHI, Y.; QIN, Q.; BAI, R. Swarm intelligence in big data analytics. In: *Intelligent Data Engineering and Automated Learning–IDEAL 2013*. [S.l.]: Springer, 2013. p. 417–426.

CHUNG, P.-T.; CHUNG, S. H. On data integration and data mining for developing business intelligence. In: IEEE. *Systems, Applications and Technology Conference (LISAT), 2013 IEEE Long Island*. [S.l.], 2013. p. 1–6.

CORRIVEAU, G.; GUILBAULT, R.; TAHAN, A.; SABOURIN, R. Review of phenotypic diversity formulations for diagnostic tool. *Applied Soft Computing*, Elsevier, v. 13, n. 1, p. 9–26, 2013.

DAOUDI, M.; HAMENA, S.; BENMOUNAH, Z.; BATOUCHE, M. Parallel diffrential evolution clustering algorithm based on mapreduce. In: IEEE. *Soft Computing and Pattern Recognition (SoCPaR), 2014 6th International Conference of*. [S.l.], 2014. p. 337–341.

DAVIS, L. et al. *Handbook of genetic algorithms*. [S.l.]: Van Nostrand Reinhold New York, 1991. v. 115.

DEAN, J.; GHEMAWAT, S. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, ACM, v. 51, n. 1, p. 107–113, 2008.

DORIGO, M.; BLUM, C. Ant colony optimization theory: A survey. *Theoretical computer science*, Elsevier, v. 344, n. 2, p. 243–278, 2005.

DOSHI, K. A.; ZHONG, T.; LU, Z.; TANG, X.; LOU, T.; DENG, G. Blending sql and newsql approaches: reference architectures for enterprise big data challenges. In: IEEE. *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2013 International Conference on*. [S.l.], 2013. p. 163–170.

FABER, V. Clustering and the continuous k-means algorithm. *Los Alamos Science*, v. 22, n. 138144.21, 1994.

FAHAD, A.; ALSHATRI, N.; TARI, Z.; ALAMRI, A.; KHALIL, I.; ZOMAYA, A. Y.; FOUFOU, S.; BOURAS, A. A survey of clustering algorithms for big data: Taxonomy and empirical analysis. *Emerging Topics in Computing, IEEE Transactions on*, IEEE, v. 2, n. 3, p. 267–279, 2014.

FERRARI, D. G. et al. *Seleção de algoritmos para a tarefa de agrupamento de dados: uma abordagem via meta-aprendizagem*. Tese (Doutorado) — Universidade Presbiteriana Mackenzie, 2014.

FERRUCCI, F.; KECHADI, M. T.; SALZA, P.; SARRO, F. A framework for genetic algorithms based on hadoop. *CoRR*, abs/1312.0086, 2013. Disponível em: <http://arxiv.org/abs/1312.0086>.

GHOSH, A.; NATH, B. Multi-objective rule mining using genetic algorithms. *Information Sciences*, Elsevier, v. 163, n. 1, p. 123–133, 2004.

GRAMA, A.; GUPTA, A.; KUMAR, V. Isoe ciency function: A scalability metric for parallel algorithms and architectures. *IEEE Parallel and Distributed Technology, Special Issue on Parallel and Distributed Systems: From Theory to Practice*, v. 1, n. 3, p. 12–21, 1993.

GUYON, I.; ELISSEEFF, A. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, JMLR. org, v. 3, p. 1157–1182, 2003.

HAND, D. J.; MANNILA, H.; SMYTH, P. *Principles of data mining*. [S.l.]: MIT press, 2001.

HANS, N.; MAHAJAN, S.; OMKAR, S. Big data clustering using genetic algorithm on hadoop mapreduce. *International Journal of Scientific Technology Research*, v. 4, 2015.

HOLZINGER, A.; BLANCHARD, D.; BLOICE, M.; HOLZINGER, K.; PALADE, V.; RABADAN, R. Darwin, lamarck, or baldwin: Applying evolutionary algorithms to machine learning techniques. In: IEEE COMPUTER SOCIETY. *Proceedings of the 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)-Volume 02*. [S.l.], 2014. p. 449–453.

HRUSCHKA, E. R.; CAMPELLO, R. J.; CASTRO, L. N. D. Evolving clusters in gene-expression data. *Information Sciences*, Elsevier, v. 176, n. 13, p. 1898–1927, 2006.

HUANG, C.-S. J.; GRAMA, T. A.; GUPTA, A.; KARYPIS, G. et al. Introduction to parallel computing. 2006.

JAFAR, R. S. O. M. Ant-based clustering algorithms:a brief survey. In: . [S.l.]: International Journal of Computer Theory and Engineering, 2010.

JR, I. F.; YANG, X.-S.; FISTER, I.; BREST, J.; FISTER, D. A brief review of nature-inspired algorithms for optimization. *arXiv preprint arXiv:1307.4186*, 2013.

JUDITH, J.; JAYAKUMARI, J. An efficient hybrid distributed document clustering algorithm. *Scientific Research and Essays*, Academic Journals, v. 10, n. 1, p. 14–22, 2015.

KATAL, A.; WAZID, M.; GOUDAR, R. Big data: Issues, challenges, tools and good practices. In: IEEE. *Contemporary Computing (IC3), 2013 Sixth International Conference on*. [S.l.], 2013. p. 404–409.

KOTURWAR, P.; GIRASE, S.; MUKHOPADHYAY, D. A survey of classification techniques in the area of big data. *arXiv preprint arXiv:1503.07477*, 2015.

KOZA, J. R. *Genetic programming: on the programming of computers by means of natural selection*. [S.l.]: MIT press, 1992. v. 1.

KRISHNANAND, K.; GHOSE, D. Detection of multiple source locations using a glowworm metaphor with applications to collective robotics. In: IEEE. *Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE*. [S.l.], 2005. p. 84–91.

KUTLU, M.; AGRAWAL, G.; KURT, O. Fault tolerant parallel data-intensive algorithms. In: IEEE. *High Performance Computing (HiPC), 2012 19th International Conference on*. [S.l.], 2012. p. 1–10.

LESKOVEC, J.; RAJARAMAN, A.; ULLMAN, J. D. *Mining of massive datasets*. [S.l.]: Cambridge University Press, 2014.

LICHMAN, M. *UCI Machine Learning Repository*. 2013. Disponível em: <http://archive.ics.uci.edu/ml>.

LIU, H.; MOTODA, H. *Feature selection for knowledge discovery and data mining*. [S.l.]: Springer Science & Business Media, 1998.

MENEZES, S. L.; FREITAS, R. S.; PARPINELLI, R. S. Mineração em grandes massas de dados utilizando hadoop mapreduce e algoritmos bio-inspirados: Uma revisão sistemática. *Revista de Informática Teórica e Aplicada*, v. 23, n. 1, p. 69–101, 2016.

MENEZES, S. R. L. de; PARPINELLI, R. S. Análise da qualidade de diferentes métricas para agrupamento de dados utilizando algoritmo bio-inspirado e arquitetura mapreduce. p. 176–183, 2016.

MENEZES, S. R. L. de; PARPINELLI, R. S. Analise de escalabilidade da arquitetura hadoop mapreduce para agrupamento de dados usando algoritmo inspirado em organismos simbióticos. In: *XVI Escola Regional de Alto Desempenho do Estado do Rio Grande do Sul*. [S.l.: s.n.], 2016. p. 67–72.

MICHAEL, M.; MOREIRA, J. E.; SHILOACH, D.; WISNIEWSKI, R. W. Scale-up x scale-out: A case study using nutch/lucene. In: IEEE. *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*. [S.l.], 2007. p. 1–8.

MOHAMMED, E. A.; FAR, B. H.; NAUGLER, C. Applications of the mapreduce programming framework to clinical big data analysis: current landscape and future trends. *BioData mining*, BioMed Central Ltd, v. 7, n. 1, p. 22, 2014.

NARAYAN, S.; BAILEY, S.; DAGA, A. Hadoop acceleration in an openflow-based cluster. In: IEEE. *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:*. [S.l.], 2012. p. 535–538.

NURAIN, N.; SARWAR, H.; SAJJAD, M. P.; MOSTAKIM, M. An in-depth study of map reduce in cloud environment. In: IEEE. *Advanced Computer Science Applications and Technologies (ACSAT), 2012 International Conference on*. [S.l.], 2012. p. 263–268.

PARPINELLI, R. S.; LOPES, H. S. New inspirations in swarm intelligence: a survey. *International Journal of Bio-Inspired Computation*, Inderscience, v. 3, n. 1, p. 1–16, 2011.

PATHAN, A. A.; HASAN, M.; AHMED, M. F.; FARID, D. M. Educational data mining: A mining model for developing students' programming skills. In: IEEE. *Software, Knowledge, Information Management and Applications (SKIMA), 2014 8th International Conference on*. [S.l.], 2014. p. 1–5.

RAJARAMAN, S.; VAIDYANATHAN, G.; CHOKKALINGAM, A. Performance evaluation of bio-inspired optimization algorithms in resolving chromosomal occlusions. *Journal of Medical Imaging and Health Informatics*, American Scientific Publishers, v. 5, n. 2, p. 264–271, 2015.

SACHIN, R. B.; VIJAY, M. S. A survey and future vision of data mining in educational field. In: IEEE. *Advanced Computing & Communication Technologies (ACCT), 2012 Second International Conference on*. [S.l.], 2012. p. 96–100.

SIVARAMAN, E.; MANICKACHEZIAN, R. High performance and fault tolerant distributed file system for big data storage and processing using hadoop. In: IEEE. *Intelligent Computing Applications (ICICA), 2014 International Conference on*. [S.l.], 2014. p. 32–36.

SMUTNICKI, C. New trends in optimization. In: IEEE. *Intelligent Engineering Systems (INES), 2010 14th International Conference on*. [S.l.], 2010. p. 285–285.

STORN, R.; PRICE, K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, Springer, v. 11, n. 4, p. 341–359, 1997.

TALLON, P. P. Corporate governance of big data: Perspectives on value, risk, and cost. *Computer*, IEEE, v. 46, n. 6, p. 32–38, 2013.

THURAISINGHAM, B. Data mining for malicious code detection and security applications. In: IET. *Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT'09. IEEE/WIC/ACM International Joint Conferences on*. [S.l.], 2009. v. 2, p. 6–7.

YANG, X.-S.; CUI, Z.; XIAO, R.; GANDOMI, A. H.; KARAMANOGLU, M. *Swarm intelligence and bio-inspired computation: theory and applications*. [S.l.]: Newnes, 2013.

YANG, Y.; NI, X.; WANG, H.; ZHAO, Y. Parallel implementation of ant-based clustering algorithm based on hadoop. In: *Advances in Swarm Intelligence*. [S.l.]: Springer, 2012. p. 190–197.

YU, H.; MOREIRA, J. E.; DUBE, P.; CHUNG, I.-h.; ZHANG, L. Performance studies of a websphere application, trade, in scale-out and scale-up environments. In: IEEE. *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*. [S.l.], 2007. p. 1–8.

ZELINKA, I.; DAVENDRA, D.; LAMPINEN, J.; SENKERIK, R.; PLUHACEK, M. Evolutionary algorithms dynamics and its hidden complex network structures. In: IEEE. *Evolutionary Computation (CEC), 2014 IEEE Congress on*. [S.l.], 2014. p. 3246–3251.

ZHAO, Y.; KARYPIS, G. Evaluation of hierarchical clustering algorithms for document datasets. In: ACM. *Proceedings of the eleventh international conference on Information and knowledge management*. [S.l.], 2002. p. 515–524.