UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

MARIO CESAR L. BONICENHA
TAYSSA R. VANDELLI

PROMETHEUS: Um algoritmo de fusão de dados para múltiplas aplicações

RIO DE JANEIRO
2019

MARIO CESAR L. BONICENHA
TAYSSA R. VANDELLI

PROMETHEUS: Um algoritmo de fusão de dados para múltiplas aplicações

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Claudio Miceli de Farias

RIO DE JANEIRO
2019

Bonicenha, Mario Cesar Lestro
B715p PROMETHEUS: um algoritmo de fusão de dados para múltiplas aplicações / Mario Cesar Lestro Bonicenha, Tayssa Ribeiro Vandelli. – 2019.

51 f.

Orientador: Claudio Miceli de Farias.

Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Universidade Federal do Rio de Janeiro, Instituto de Matemática, Bacharel em Ciência da Computação, 2019.

1. Internet das coisas. 2. Redes de sensores sem fio. 3. Múltiplas aplicações. 4. Fusão de dados. I. Vandelli, Tayssa Ribeiro. II. Farias, Claudio Miceli de (Orient.). III. Universidade Federal do Rio de Janeiro, Instituto de Matemática. IV. Título.

MARIO CESAR L. BONICENHA
TAYSSA R. VANDELLI

PROMETHEUS: Um algoritmo de fusão de dados para múltiplas aplicações

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em ___ de _____ de _____

BANCA EXAMINADORA:

Claudio Miceli de Farias
DSC (PPGI-UFRJ)

Valeria Menezes Bastos
DSC (COPPE-UFRJ)

João Antônio Recio da Paixão
DSC (PUC-RIO)

AGRADECIMENTOS

Gostaríamos de agradecer primeiramente à nossa família, a qual nos apoiou e nos apoia sempre, nos ensina sobre respeito, dedicação, disciplina, nos ajuda com nossas batalhas diárias e que nos preenche todos os dias com seu amor; ao nosso professor orientador, Claudio Micelli, que nos confiou esse trabalho e nos orientou até aqui; à todos os professores que passaram por nós e que fizeram questão de nos mentorear e dar tudo de si para que aprendêssemos e fôssemos pessoas melhores; além dos professores que se dedicam à seus alunos e lutam pela educação dos mesmos, apesar dos problemas diários que precisam enfrentar e com toda dificuldade da prática da profissão nos dias de hoje; à todos os alunos que nos acrescentaram de alguma forma e nos fizeram crescer e amadurecer durante nosso período de graduação; e, finalmente, últimos mas não menos importantes, à todos os amigos que nos acrescentam todos os dias, nos influenciando de forma positiva, dando feedbacks construtivos e nos ajudando na nossa própria construção, tijolo por tijolo, à sermos pessoas melhores a cada dia.

"If you want to be successful in a particular field, perseverance is one of the key qualities."

George Lucas

RESUMO

No contexto de Internet das Coisas (IoT), algoritmos de fusão de dados podem ser importantes para diversas aplicações e para diferentes fins. Já existem algoritmos que realizam esse tipo de fusão de dados e, em particular, um deles - chamado Hephaestus - trabalha com heurísticas computacionalmente baratas (em troca de certa eficiência) para inferir determinadas respostas ao usuário final. Se fazem necessárias, então, otimizações no desenvolvimento de um algoritmo a ponto de conciliar performance, processamento e qualidade. Com o baixo poder de processamento dos dispositivos e sensores em IoT, foi criado o Prometheus, um algoritmo de fusão de dados, a fim de conseguir identificar e caracterizar eventos genéricos para auxiliar o poder de decisão do usuário final. Com o objetivo de otimizar o algoritmo já existente e conseguir resultados mais assertivos, esta proposta visa uma separação de eventos mais eficaz, definida de acordo com a análise dos dados.

Palavras-chave: internet das coisas. rede de sensores sem fio. múltiplas aplicações. fusão de dados.

ABSTRACT

In the context of the Internet of Things (IoT), data fusion algorithms are important for many applications and to different objectives. There are already some algorithms that make this fusion and particularly one of them - the Hephaestus - works with computationally cheap heuristics (in exchange for efficiency) to infer results to the final user. It is required then optimization in its development in order to conciliate performance, processing and quality. With a low device and IoT sensors' processing power, it was created a data fusion algorithm called Prometheus in order to identify and characterize generic and abstract events to assist the final user's decision power. With the goal of enhancing the already existing algorithm and getting to more assertive results, this proposal aim to a more effective segregation of the events, defined under the data analysis.

Keywords: internet of things. wireless sensor network. multiple applications. data fusion.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

IoT	Internet of Things
RSSF	Rede de Sensores sem Fio
FMR	Frequência Média Relativa
IDE	Integrated Development Environment
RAM	Random-Access Memory
NF	Nó de Fusão
NC	Nó Coletor
C1	Caso 1
C2	Caso 2
C3	Caso 3
C4	Caso 4
S	Assimetria
UFRJ	Universidade Federal do Rio de Janeiro

SUMÁRIO

1 INTRODUÇÃO

A Internet das Coisas (Internet of Things - IoT) se trata, basicamente, de dispositivos físicos capazes de coletar e transmitir dados através da internet. A utilização desse tipo de tecnologia ajuda os usuários a obter respostas cada vez mais rápidas (e muitas vezes em tempo real) a partir da aplicação, necessidade essa que vem aumentando de acordo com a complexidade e criticidade de cada aplicação na resolução de problemas. Exemplos comuns são: automatização de processos e serviços, sensores de monitoramento para datacenters e/ou detecção de incêndio, controle de consumo de energia, monitoramento de equipamentos industriais e de performance, etc.

Um dos principais componentes de sensoriamento no paradigma da Internet das Coisas (ATZORI; IERA; MORABITO, 2010; WHITMORE; AGARWAL; XU, 2015) é a Rede de Sensores Sem Fio (RSSF) (RAWAT et al., 2014; AKYILDIZ et al., 2002; POTTIE; KAISER, 2000), rede que possibilita o compartilhamento de recursos entre múltiplas aplicações, se utilizando de sensores que se conectam entre si de forma wireless, sem cabos, com essa capacidade de compartilhamento.

Quando os sensores foram criados, abriram-se grandes possibilidades para o mundo da tecnologia. Esses sensores foram desenhados com a capacidade de processamento e comunicação, além de conseguirem captar variáveis do ambiente. Quando vinculados, criavam essa rede sem fio na qual era limitada. Inicialmente as RSSFs não davam a possibilidade de reuso e compartilhamento de recursos entre múltiplas aplicações, pois os sensores até então eram desenhados para cumprir tarefas específicas, com intervalos bem definidos e com os requerimentos iniciais já conhecidos. Com os avanços tecnológicos e as necessidades do público, modernizações se fizeram necessárias, até que os sensores passaram a não precisar mais conhecer os requerimentos iniciais nem receber intervalos definidos (CALDAS et al., 2015).

Com novas possibilidades de compartilhamento, sensores poderiam ser postos juntos e formar uma rede. Uma grande e importante mudança ocorreu quando eles foram inseridos em uma única infraestrutura, sendo virtualizados. A virtualização de sensores (ISLAM et al., 2012; KHAN et al., 2015; SANTOS et al., 2015) possibilitou grandes avanços em tecnologia, explorando o máximo das RSSFs, facilitando o descobrimento de padrões em ambientes monitorados, fazendo predições e por consequência, melhorando o poder de decisão do usuário, reduzindo o tempo de resposta de decisões e possibilitando a percepção das situações de forma mais imediata e inteligente.

Com todas essas mudanças, a fusão de dados se torna viável e tem como objetivo derivar informações úteis que não são fornecidas pelos sensores individualmente, facilitando em sua manipulação (FACELI; CARVALHO; REZENDE, 2004).

A grande quantidade de dados registrados e consumidos em diversas aplicações tam-

bém pode gerar informações com valor agregado para o usuário final (AQUINO et al., 2016). Em termos de adaptação e de reutilização, se faz necessário considerarmos as aplicações com um contexto abstrato, para que o usuário possa escolher em qual situação se utilizar dos sensores.

Existem trabalhos relacionados de fusão de dados para múltiplas aplicações, porém ou não conseguem identificar os fenômenos corretamente ou apresentam restrições pois precisam da entrada de requerimentos iniciais das aplicações na rede. Em um ambiente com recursos limitados, conhecer os requisitos das aplicações, os tipos de dados coletados e os tipos de unidades de sensoriamento requeridos consomem recursos dos dispositivos. Conforme aumentam o número de aplicações, aumenta a necessidade de armazenamento dos requisitos de cada uma delas, característica essa que sensores típicos de IOT podem não suportar.

O Hephaestus (AQUINO et al., 2016), algoritmo de fusão de dados, foi criado, então, com o intuito de processar esses dados sem conhecer os requisitos iniciais. Este algoritmo consegue ainda inferir um resultado coerente sobre os fenômenos analisados, tornando implementações mais rápidas, eficientes e ainda barateando custos. Com o objetivo de atender múltiplas aplicações, o Hephaestus se utiliza de sensores para realizar a coleta e o processamento dos dados a partir de uma Rede de Sensores sem fio. Através de sua funcionalidade de classificação de eventos a partir de uma massa de dados, o algoritmo do Hephaestus opta por heurísticas computacionalmente baratas a fim de otimizar seu poder de processamento. Existe, contudo, situações e análises que podem ser prejudicadas pela escolha dessas heurísticas de baixo custo computacional.

Por esse motivo, criamos o Prometheus, algoritmo de fusão de dados que se utiliza de Internet das Coisas para transmitir e processar informações de diversos contextos, isto é, informações originadas de diferentes sensores ou aplicações. Através de implementações de heurísticas mais complexas, busca-se um modo de facilitar a tomada de decisão do usuário final, a partir do que chamamos de eventos, para qualquer quantidade de aplicações que se fizerem necessárias. Os eventos são uma importante parte do trabalho e significam um conglomerado de dados classificados como de um mesmo contexto pelo algoritmo, ou seja, dados que representam potencialmente um mesmo fenômeno de uma mesma aplicação. A partir do processamento na análise dos dados de entrada, o Prometheus se utiliza do grau de assimetria para decidir se há a necessidade da realização do processo de separação dos dados e também de algumas estruturas de dados específicas desenhadas para auxiliar no processo de separação em eventos, como o vetor de Pulo, a frequência média relativa (FMR), entre outros.

Estas estruturas de dados são a maior contribuição do Prometheus, as quais permitirão uma análise mais detalhada das massas de dados de entrada e que farão as saídas do algoritmo serem diferentes das do Hephaestus, que se utiliza apenas da média flutuante calculada a partir dos dados para sua análise. Todas estas estruturas referentes ao

Prometheus serão explicadas mais a fudo posteriormente no Capítulo 4.

1.1 MOTIVAÇÃO E DESAFIO

Com a crescente demanda por tecnologias inteligentes capazes de integrar diferentes espaços, a Internet das Coisas vem se popularizando e ganhando potencial na resolução de diversos paradigmas dentro e fora da computação. Assim, se faz necessário cada vez mais que o processamento de dados seja feito de forma eficiente, rápido e seguro. Diversos desafios são postos à prova quando se trata de processamento de dados em uma rede de sensores sem fio. Exploraremos alguns deles nos parágrafos seguintes.

Conseguir reduzir o tempo de resposta para aplicações complexas é um desafio que pode envolver diversos processos dentro de um serviço (AQUINO et al., 2016). Ao processar a massa de dados relevante nos próprios sensores em que os dados são coletados - ou mesmo separar uma certa quantidade de sensores para fazer apenas o processamento - pode agilizar a entrega de uma resposta do algoritmo para o usuário. Assim, procuramos dispensar a transmissão desnecessária desses dados para uma central de processamento, fazendo com que o tempo em que a tomada de decisão seja potencialmente menor.

Conseguir classificar os diferentes fenômenos dentro de uma massa de dados grande, de forma a resumir o conjunto inicial, é um problema que já foi resolvido pelo algoritmo base utilizado para esse trabalho, o Hephaestus. Porém, o desafio aqui se torna ainda maior quando a otimização desse processo visa uma melhor representatividade desses dados em busca de uma saída mais assertiva e clara para o usuário final.

A otimização do algoritmo para um sensor de baixo poder de processamento é vital para o sucesso do trabalho. Um algoritmo de baixo custo que roda em tempo viável de acordo com as aplicações e as necessidades do usuário se torna, também, um objetivo. No contexto de Internet das Coisas, há a limitação de recursos para atingir seu fim, como por exemplo a necessidade de realizar todo o processamento através de seus nós ao invés de ser realizada em um servidor, visando um melhor tempo de resposta.

Um outro desafio encarado neste trabalho foi a abstração de dados (NAKAMURA; LOUREIRO; FRERY, 2007). Alguns dos trabalhos relacionados lidam com dados de uma mesma aplicação (e, por vezes, com os requerimentos previamente conhecidos), porém essa é uma técnica limitada, pois a caracterização de cada métrica se torna complexa, ou seja, não se consegue processar quaisquer dados, limitando as variáveis de resposta daquele ambiente. Quando se abre a possibilidade de captar quaisquer dados de diferentes contextos, menos sensores são necessários (SANTOS et al., 2015), pois um sensor consegue captar diferentes tipos variáveis, barateando custos e ampliando a gama de variáveis do ambiente.

O algoritmo de fusão de dados proposto, caracterizado pela sua capacidade de integrar dados de um ou mais ambientes monitorados, procurará encontrar métricas desse ambi-

ente, analisar e processar variações entregando os resultados para o usuário final poder decidir o que fazer a partir disso. E, com a sua otimização, visamos alcançar resultados mais assertivos e facilitar na tomada de decisão do usuário final. Então, em suma, nossa motivação é a otimização desse algoritmo para permitir uma tomada de decisão mais rápida e eficiente.

1.2 PROPOSTA

A fim de desenvolver um algoritmo que tenha capacidade de fundir os dados provenientes do ambiente e refiná-los de forma a colaborar com a tomada de decisão do usuário, criamos o Prometheus, algoritmo que potencializa a análise feita sobre as informações e os eventos estudados.

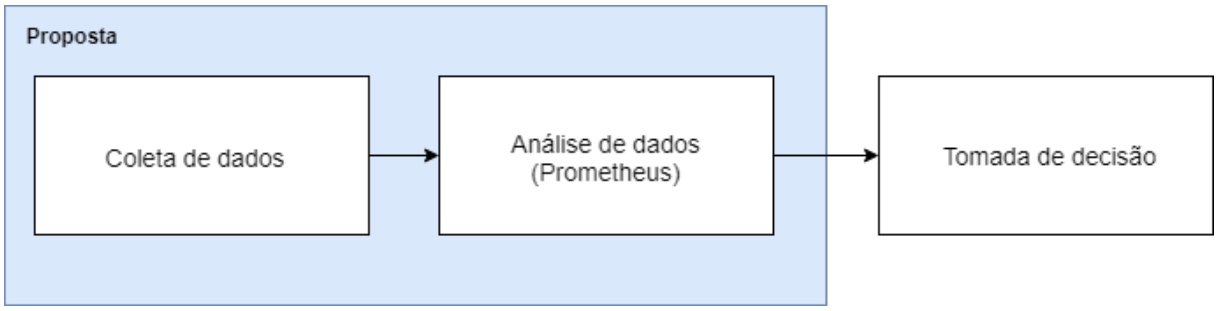

Figura 1 – Diagrama da proposta do Prometheus

A Figura 1 descreve a estrutura do projeto e o que ele abrange. A coleta de dados é realizada por sensores de coleta, transmitidos para sensores de fusão para que, após a execução do algoritmo e fora do escopo deste projeto, a tomada de decisão seja realizada por um profissional capacitado.

Os comportamentos padrões das aplicações não são conhecidos, a priori, pelo algoritmo. Ao conhecê-los, como alguns dos trabalhos relacionados citados no Capítlo 3, os projetos acabam por se tornar menos adaptáveis e reutilizáveis. Não tendo estas informações sobre os requisitos iniciais, o Prometheus não precisa armazená-las nos sensores de baixo poder de armazenamento.

No nosso trabalho, um fenômeno (ou evento) denota qualquer situação da área monitorada que gera dados em um intervalo específico que pode ou não ter correlação com um evento de interesse para uma aplicação no mundo real. Quanto mais fenômenos diferentes geram dados no ambiente monitorado, mais complexa é a entropia dos dados, que pode refletir na incerteza inferida (AQUINO et al., 2016).

Nosso algoritmo avalia as seguintes estatísticas dos dados brutos do ambiente: média, curtose e assimetria. Essas estatísticas são usadas por causa da sua simplicidade de serem implementadas em um sensor. Isso contribui para ajudar no processamento e, ainda, salvar energia gasta pelos sensores. Outra medida a fim de economizar no processamento envolve

a criação de um vetor de Pulo para resumir os dados da entrada caso a mesma seja grande o suficiente (explicaremos mais a fundo nos capítulos posteriores). No final, a informação será usada por outro método de fusão ou analisada por um expert no domínio de alguma aplicação para ajudar em um sistema ou numa tomada de decisão que, enfatizando, não entra no escopo deste projeto (Figura 2).

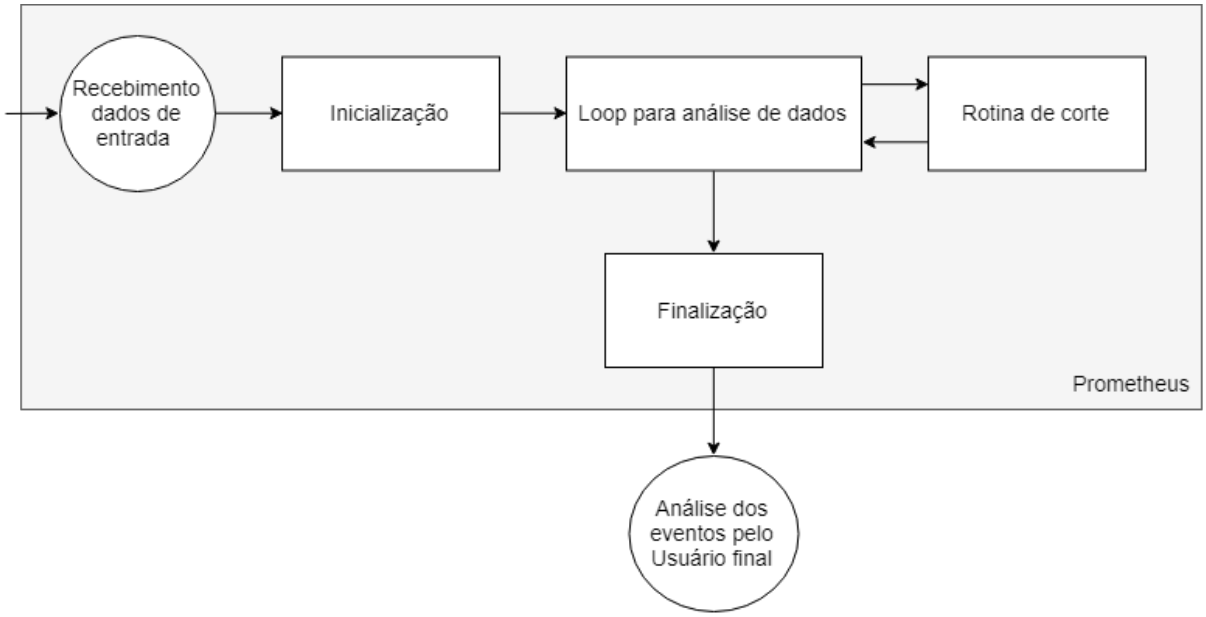

Figura 2 – Diagrama base do algoritmo Prometheus

Como mostraremos ao longo deste trabalho, a classificação e caracterização objetivada do algoritmo em que este foi baseado pode não ser tão eficiente na tradução dos dados a partir dos fenômenos do ambiente monitorado. Então buscamos desenvolver, a partir das mesmas premissas, ou seja, não se conhecendo os requisitos iniciais da aplicação, uma análise mais crítica das informações captadas a fim de conseguir gerar uma maior assertividade para o resultado final do Prometheus.

1.3 ORGANIZAÇÃO DO TRABALHO

O capítulo 2 descreve os conceitos básicos que o presente trabalho aborda e desenvolve. Dividido em subseções, as principais definições tratadas são sobre Rede de Sensores sem Fio, Fusão de dados em Redes de Sensores sem Fio e assimetria.

O capítulo 3 apresenta os trabalhos relacionados à este. O algoritmo de fusão de dados Hephaestus e seu funcionamento também são explicados.

O capítulo 4 descreve o algoritmo Prometheus, explicando como o mesmo funciona e realiza um de seus principais processos chamado de Rotina de corte. Processo esse que este trabalho visa otimizar e testar a fim de comprovar sua performance e viabilidade. Aponta também as mudanças e otimizações implementadas no algoritmo, mostrando principal-

mente as diferenças entre os processos, a análise realizada em cima dos dados coletados e a caracterização dos eventos.

O capítulo 5 trata de informações mais técnicas como o microcontrolador utilizado na implementação e utilização do algoritmo, tal como algumas de suas tecnologias empregadas, além de um diagrama explicando o fluxo de todo o Prometheus.

No capítulo 6 são apresentados testes realizados para a verificação tanto da viabilidade do algoritmo proposto quanto uma comparação com os resultados do Hephaestus. Explicitamos também como as simulações foram dadas e estudamos a aplicação em um caso real, averiguando sua efetividade.

O capítulo 7 mostra a forma na qual avaliamos a acurácia do algoritmo Prometheus, além do consumo de RAM e energia.

O capítulo 8 traz a conclusão do nosso trabalho e as discussões sobre os resultados e possíveis trabalhos futuros.

2 CONCEITOS BÁSICOS

Neste capítulo iremos descrever os conceitos básicos necessários para um melhor entendimento deste trabalho. Separamos suas definições e teorias em três macro seções, organizadas como a seguir: (i) seção 2.1 descreve Rede de Sensores sem Fio e técnicas relacionadas ao contexto de Internet das Coisas; (ii) seção 2.2 definirá o conceito de Fusão de Dados em Redes de Sensores Sem Fio e suas implicações; (iii) seção 2.3 abordará a teoria que envolve os conceitos estatísticos relacionados.

2.1 REDES DE SENSORES SEM FIO

Ao contrário das redes tradicionais analógicas, as RSSF são compostas de vários dispositivos de baixo custo sustentados por baterias de tamanho reduzido, capazes de captar, processar e transmitir informações através de uma Rede sem fio (DELICATO et al., 2005). Os sensores captam os dados de monitoramento (normalmente de estruturas físicas como prédios, construções, fábricas, etc.) e transmitem para os nós coletores para serem analisados e processados. Uma vantagem desse tipo de sensor é a vasta área coberta por diversos sensores de baixo-custo. Com os sensores trabalhando a partir de uma bateria, é de extrema importância que os protocolos e algoritmos tenham sido projetados para o uso eficiente da energia, maximizando o tempo de vida do sistema.

A natureza dinâmica da RSSF em termos tanto de organização como de topologia, requer uma capacidade de auto-organização e auto-configuração (DELICATO et al., 2006). Uma rede de sensores sem fio precisa se ajustar quando há uma perda ou inserção de sensores não somente quando acontece uma falha, mas também quando a bateria de algum sensor acaba, ou até mesmo quando o modo da bateria é alterado, como por exemplo ser colocada no modo de hibernação. Mecanismos para estender a vida útil de uma RSSF se tornam um desafio no campo, e que é presente em vários níveis da pilha de protocolos, onde podemos destacar dois: agendamento de tarefas e fusão de dados dentro da Rede, que será discutido mais a fundo na subseção seguinte.

A RSSF precisa ser adaptável e os motivos incluem principalmente as premissas de que tanto o ambiente onde ela é instalada quanto a aplicação sobre a qual ela está suportando podem mudar. A descentralização, nesse caso, é muito importante para manter a Rede adaptável e trabalhando de forma anônima, ou seja, sem assistência ou intervenção humana. Muitos fatores podem colaborar para esses fins, já que a quantidade de sensores presentes numa rede pode ser muito grande, como sua instalação pode ser feita, por exemplo, em lugares de difícil acesso.

Das vantagens de se empregar uma RSSF ao invés de redes tradicionais estão: (i) redução de custos ao diminuir o número de inspeções nas estruturas monitoradas; (ii)

possibilidade de instalar sensores em lugares de difícil acesso (sem a necessidade de se instalar fiação); (iii) melhor coleta de dados num curto período de tempo e sua melhor reutilização (GUNGOR; LU; HANCKE, 2010); (iv) vasta área coberta com um baixo custo. Porém, mesmo com muitas vantagens, a Rede de Sensores sem Fio também conta com alguns contras: o consumo de energia da bateria é um fator crucial e determinante para a vida útil desse tipo de Rede (DELICATO et al., 2003; DELICATO et al., 2005)).

Há uma abordagem para desenvolvimentos pequenos e rápidos que objetivam servir a apenas uma aplicação, que pode levar um ineficiente uso de recursos e resultados de baixo custo-benefício. Cada nova aplicação, nesse caso, necessitaria de uma nova infraestrutura sensorial, podendo gerar, assim, replicações estruturais desnecessárias.

2.2 FUSÃO DE DADOS EM REDES DE SENSORES SEM FIO

Um dos objetivos da técnica de fusão de dados é fazer com que um sensor tenha um menor consumo de energia. Uma forma de se fazer isso é eliminando dados redundantes. Essa eliminação pode ocorrer quando existe uma correlação entre os dados (FARIAS, 2014c), podendo ser espacial, quando os valores gerados por sensores próximos se relacionam; temporal, quando a leitura dos sensores muda aos poucos ao longo do tempo; ou semântica, quando a informação nos diferentes pacotes podem ser classificadas no mesmo grupo semântico (por exemplo, dados que são gerados por sensores inseridos em um mesmo ambiente fechado); e que acaba favorecendo na afirmação de acurácia dos dados.

Existem três conceitos muito importantes para a eficiência do mecanismo de fusão: a acurácia, o grau de agregação e a latência (FARIAS, 2014c). Quando a quantidade de dados é reduzida, existe o risco de perda de acurácia, que é definida como o grau de proximidade entre a mensuração observada e seu valor real esperado, porém com uma correlação eficiente é possível manter a mesma acurácia dos dados agregados. Além disso, para manter a eficiência na redução dos dados, precisa existir um equilíbrio entre o grau de agregação e a latência, pois o primeiro é definido como o número de pacotes de dados agregados em uma única transmissão, enquanto o segundo é o tempo entre pacotes recebidos por um nó coletor e os dados gerados pelos nós de origem.

Seus procedimentos são performados por um conjunto de sensores que o fazem descentralizado para as RSSFs. Se considera um número finito de nós e um ou mais sumidouros. A rede é considerada heterogênea já que seus sensores têm diferentes capacidades em termos de armazenamento, processamento, sensoriamento e comunicação.

Cada nó pode ter até dois papéis: podem performar como um Nó Coletor (NC), um Nó de Fusão (NF) ou os dois simultaneamente. A função do primeiro é detectar e coletar variáveis, enquanto o segundo é responsável por executar o algoritmo de fusão. A partir da coleta dos dados do ambiente, o Nó Coletor transfere esses dados para o nó de processamento, onde ocorrerá a fusão e, principalmente, a análise dessas informações a

partir de um algoritmo de fusão de dados.

2.3 ASSIMETRIA

Existe uma medida que traz informações sobre os dados: a assimetria. Essa é uma medida que diz se os dados são simétricos ou assimétricos quando comparados a uma distribuição normal.

A assimetria pode ser caracterizada em um gráfico como tendo dados homogêneos ou heterogêneos (PEARSON, 1895), tendo o primeiro caudas iguais ou parecidas e o segundo, caudas diferentes ou sendo a mistura de dois casos homogêneos. O Coeficiente de Assimetria de Pearson (2.1), que usa a Mediana da amostra dos dados (GOOS; MEINTRUP, 2015):

$$S = 3 * (media(amostra) - mediana(amostra))/(desvioPadrao(amostra)) \quad (2.1)$$

Um conjunto de dados pode ter (i) assimetria positiva (Figura 3), (ii) negativa (Figura 4), ou (iii) ser simétrico (BIRD; BEERS, 1993). Um conjunto de dados de assimetria positiva tem S com um valor positivo e sua cauda é mais longa na direita e a concentração de massa dos dados está do lado esquerdo da média. Um conjunto de dados com assimetria negativa é o oposto de (i), o que significa que o valor de S é negativo e a cauda do conjunto é na esquerda e a massa de dados está concentrada na direita. Um conjunto simétrico tem S=0 e os dados estão concentrados em seu centro de forma homogênea. Existe uma assimetria moderada quando $0.15<|S|<1$ e uma assimetria fortemente moderada quando $|S|>1$ (Figure 5).

Uma assimetria positiva moderada (Figura 3) ou uma assimetria negativa moderada (Figura 4) indica que há uma tendência de crescimento ou declínio. Um conjunto com assimetria fortemente moderada indica que o dataset é formado por dois ou mais grupos de dados heterogêneos. A estatística de assimetria pode ser usada para descrever padrões e tendências no dataset.

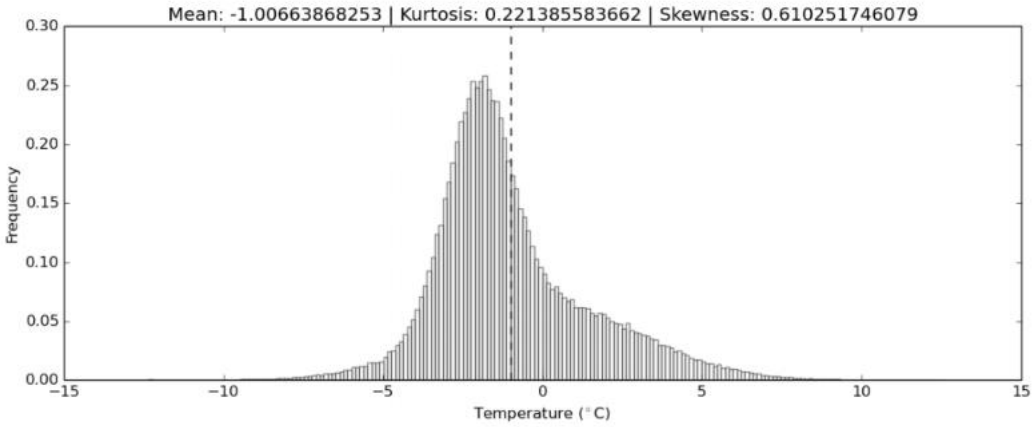

Figura 3 – Conjunto de dados com assimetria positiva

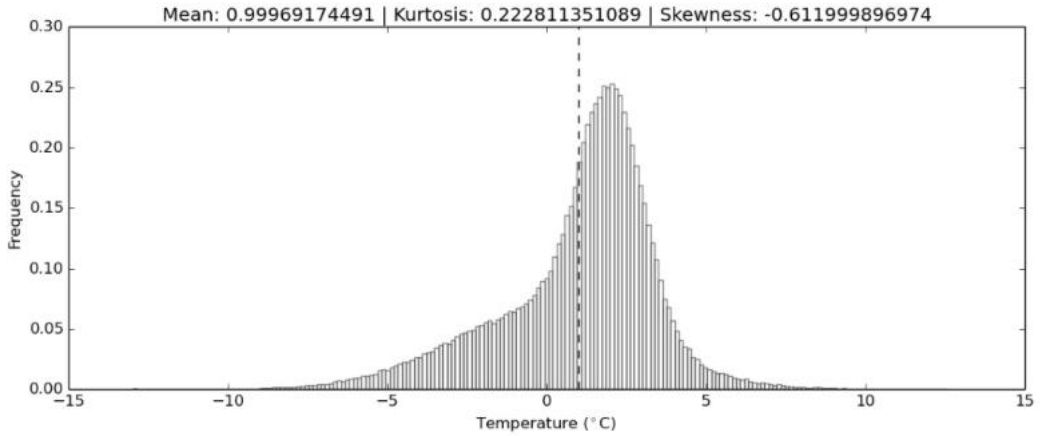

Figura 4 – Conjunto de dados com assimetria negativa

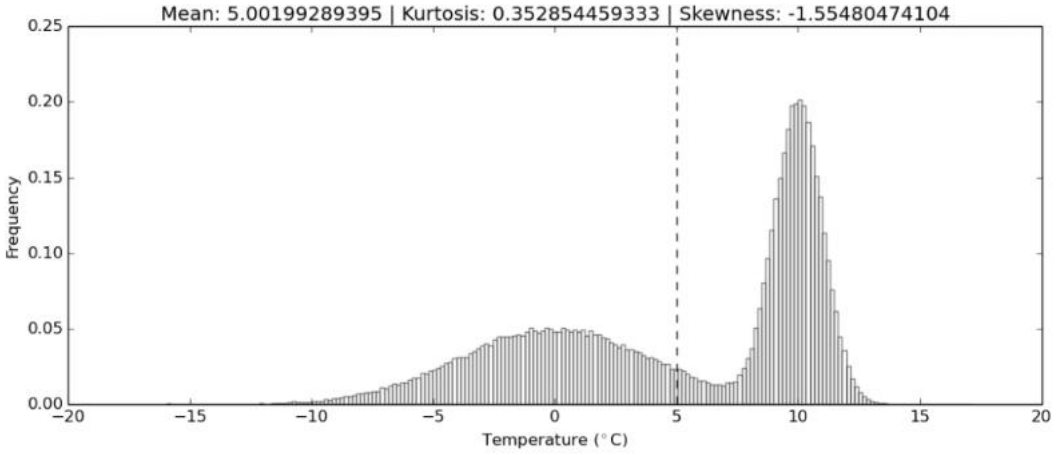

Figura 5 – Conjunto de dados fortemente assimétricos

3 TRABALHOS RELACIONADOS

Neste capítulo, introduziremos outros trabalhos presentes na literatura que possuem objetivos similares ao deste projeto. Ao desenvolverem processos que lidam com múltiplas aplicações ao mesmo tempo, sem conhecer os requerimentos iniciais destas, reconhece-se um nível de similaridade com a proposta deste trabalho. Diversas técnicas de análise são aplicadas na enorme quantidade de dados produzida pela RSSF em cada trabalho, resultando em diferentes agrupamentos de dados finais e característicos de cada aplicação. Para cada diferente trabalho, nós descrevemos suas características gerais e apontamos as principais diferenças quanto a proposta deste projeto. Analisaremos, então, os trabalhos de Dziengel et al. (2016), Safia et al (2016) e Mus and KR (2015) e de Aquino et al. (2016).

A proposta de um sistema classificador baseado na distribuição de detecção de eventos (DZIENGEL et al., 2016) vem a partir de dois frameworks: (i) um framework de avaliação, o qual entrega um modelo de classificação e permite a avaliação teórica de uma amostra de treino, e (ii) um framework de detecção de eventos distribuídos, que consequentemente se aplica aos modelos de classificação para avaliar, filtrar e classificar eventos dentro de uma rede. Dziengel et al. (2016) também desenvolve uma aplicação para cercas, que tem como motivação a necessidade de um sistema para proteger grandes áreas como lugares reservados para construção, aeroportos, etc. Uma cerca equipada com este sistema sem fios de detecção de eventos distribuídos é capaz de agregar e analisar os dados dos sensores baseados em múltiplos pontos de medidas a fim de classificar e detectar intrusos (eventos de intrusão). O sistema de vigilância de cercas citado segue uma abordagem orientada por dados.

Então, o objetivo geral é resolver o problema de classificação de diferentes eventos treinados no lugar de construção da cerca através de uma rede de sensores sem fio. Dziengel et al. (2016) introduz uma solução de processamento interno à rede que usa métodos de compressão de dados baseados em um sistema clássico de reconhecimento de padrões. A proposta de um sistema de detecção de eventos envolve múltiplos sensores para obter uma visão, de certa forma, geral e ampliada de um evento. Em princípio, observa eventos de diferentes perspectivas com nós sensores localizados em posições diferentes na cerca para complementar um ao outro. Um modelo de classificação integrado é usado para definir as diversas descrições de eventos. O processo colaborativo de detecção de evento final combina os dados do ambiente disponíveis na rede.

De um jeito similar ao nosso trabalho, Dziengel et al. (2016) não apresenta apenas um método de avaliação de informação local, o que poderia não ser efetivo em entender os eventos em sua totalidade. Também de forma similar ao nosso trabalho, Dziengel et al. (2016) usa características descritivas para distinguir os diferentes eventos detectados.

Contudo, diferentemente do nosso trabalho, a proposta de Dziengel et al. (2016) precisa de um treinamento para configurar seus classificadores (conhecidos como classificador de Naïve Bayes e K nearest neighbor classifier), enquanto o Prometheus infere de acordo com os dados para prover uma informação ao tomador de decisão (evitando, assim, a necessidade de um treinamento anterior).

A proposta de um algoritmo distribuído para detectar fenômenos (SAFIA; AGHBARI; KAMEL, 2016), como fogo, derramamento de óleo ou gases tóxicos é feita a partir de um ambiente de Rede de Sensores sem fio. Os autores assumiram que o ambiente não tem servidor centralizado para coletar e agregar os dados dos sensores. No algoritmo, os sensores se organizam em grupos desmembrados, elegendo alguns sensores para serem líderes do grupo e então os restantes se agrupam com o líder mais próximo. Sensores de cada grupo reportam seus dados sensoriados ao cabeça do grupo, o qual agrega os dados coletados e detecta o fenômeno na área abrangida pelos sensores membros do grupo. Então, baseados na ordem dos ID's dos líderes, cada líder reporta a informação do fenômeno local detectado ao próximo líder daquela ordem, o qual agrega a informação e manda para o próximo líder, e assim em diante. O último líder da sequência agrega a informação de todos os fenômenos locais detectados para descobrir o fenômeno global. Adicionalmente, o trabalho propõe a eleição de dois algoritmos de eleição de líderes, um elege líderes baseado na informação do fenômeno global e o outro, do fenômeno local. Além disso, o trabalho propõe uma técnica de otimização para diminuir custos de energia no relato da informação do fenômeno local, resumindo as informações resultantes do fenômeno para, então, reduzir o tamanho dos dados transmitidos entre líderes.

O trabalho apresenta algumas similaridades com a proposta deste trabalho, já que é um algoritmo de distribuição para detectar fenômenos em um ambiente de Rede de Sensores sem fio. Mas, diferentemente do nosso trabalho, ele segue algumas suposições: (i) para eles, sensores têm o mesmo poder de processamento, vida útil de bateria, armazenamento e alcance de comunicação em sua fase inicial; e (ii) sensores têm conhecimento prévio dos requisitos da aplicação. A respeito das duas suposições, as diferenças principais entre o Prometheus e o trabalho de Safia et al. (2016) são que o Prometheus não precisa ser rodado em uma rede onde sensores têm o mesmo poder de processamento, vida útil da bateria, armazenamento e alcance de comunicação em sua fase inicial e, principalmente, o Prometheus não precisa ter conhecimento prévio sobre nenhum requisito do ambiente como o Safia precisa.

Uma outra contribuição (MUSÍLEK; KRÖMER; BARTON, 2015) desenvolve um algoritmo de clustering de RSSF baseado na entropia de dados de sensores individuais. Reconhecido como E-BACH: Hierarquia de Clustering baseado em entropia para RSSFs, este algoritmo utiliza-se de métodos em que a qualidade de dados são o principal objetivo, procurando, então, uma maior otimização da rede. As medidas da entropia e suas aproximações podem ser usadas para identificar clusters de sensores. Dependendo das

medidas particulares usadas, resultados do procedimento propostos por Mus e Kr (2015) poderiam desenvolver uma hierarquia de clusters de acordo com a informação do seu conteúdo e identificar grupos de nós que provêem dados similares. Em suma, ao não conhecer os requerimentos iniciais das aplicações da rede, este trabalho propõe um algoritmo de processamento de dados que se utiliza da entropia dos dados (SHANNON, 1948) para facilitar na coleção e agrupamento dos dados obtidos. Como resultado do algoritmo de Mus e Kr (2015), a coleção de dados obtidas através do clustering não é adaptável independentemente do propósito da aplicação. A proposta do Prometheus, em contrapartida, pode ser aplicada genericamente para qualquer aplicação, ou seja, ela visa um auxílio na tomada de decisão, por parte do usuário final, para qualquer tipo aplicação.

O Hephaestus, algoritmo de fusão de dados (AQUINO et al., 2016), realiza um processamento de dados provindos de sensores em uma RSSF com o objetivo de auxiliar uma tomada de decisão por parte do usuário final. Após a análise de métricas específicas, como média flutuante, desvio padrão, assimetria, etc., o processamento dos dados de entrada é realizado através de diversas rotinas para a separação e a classificação dos dados em regiões. Foram desenvolvidas heurísticas para implementar a delimitação dessas regiões, que podem caracterizar, futuramente, potenciais eventos (AQUINO et al., 2016). Em suma, o algoritmo se assemelha à proposta do presente trabalho onde: (i) múltiplas aplicações podem ser utilizadas; (ii) grandes conjuntos de dados processados a partir de uma RSSF; (iii) resultado obtido procura ser aplicável de forma genérica. As diferenças básicas entre o algoritmo Hephaestus e a proposta deste trabalho - o Prometheus, estão em suas heurísticas e na forma como o algoritmo trabalha as mesmas. Através dos trabalhos relacionados acima, conseguimos ressaltar que existem propostas semelhantes ao Prometheus. Porém, como explicitado, diferenças em como os algoritmos processam e analisam a grande quantidade de dados tida inicialmente podem afetar de forma significativa o resultado final.

4 PROPOSTA: PROMETHEUS

Neste capítulo apresentamos o Prometheus. Prometheus pode ser classificado como um algoritmo de caracterização de fenômenos através do uso de fusão de dados. O principal objetivo é realizar essa caracterização dos fenômenos de forma a consumir poucos recursos computacionais.

Apresentamos a seguir, a partir das subseções, o algoritmo dividido em suas 3 partes principais: (i) inicialização e cálculos de auxiliares (média, mediana, desvio padrão e assimetria); (ii) criação das estruturas de dados especificamente desenhadas para auxiliar a análise dos dados (coordenadas, FMR e Pulo) e (iii) rotina de corte.

4.1 INICIALIZAÇÃO E CÁLCULO DE AUXILIARES

Diferentemente dos algoritmos que trabalham com requisitos de aplicação pré-definidos e conhecidos, o Prometheus não precisa da alimentação de nenhuma informação adicional para poder processar os dados coletados, fazendo com que a classificação desses dados por eventos seja uma tarefa de extrema importância e relevância para o sucesso do algoritmo.

Para a fusão dos dados, são calculados, inicialmente, a assimetria, média, mediana e o desvio padrão da massa de dados coletados, com o objetivo de caracterizar cada dado como potencialmente semelhante a um grupo. Abaixo, apresentamos um pseudocódigo que representa esta parte inicial do algoritmo. A Figura 6 demonstra o pseudocódigo desta rotina de inicialização.

O cálculo da assimetria é muito importante pois é utilizado como o principal fator na decisão da quebra dos dados. Essa assimetria pode ser classificada de duas maneiras: fortemente assimétrica e moderadamente assimétrica. Assumimos então que, quando a amostra é considerada suficientemente simétrica, ou seja, com baixo nível de assimetria (< 1.0), a gama de dados já se encontra em um mesmo contexto e, por isso, é atribuída a apenas um evento.

4.2 VETOR DE PULO, FMR E COORDENADAS

Visando evitar problemas na caracterização de eventos, a proposta do Prometheus se baseia na análise prévia dos dados para conseguir um corte mais coerente de acordo com cada contexto e sempre classificar uma região escolhida como um evento. Logo, ao invés de apenas calcular a média aritmética dos dados como é feito em outros algoritmos, a otimização começa calculando a FMR (frequência média relativa) de todo o dataset. A partir dessa métrica calculada, pode-se ter uma base do que está acontecendo dentro de todo o intervalo em análise.

```
RotinaPrincipal():

    OrdenarDadosDeEntrada()
    CalcularMetricas()

    Se assimetria > 1.0:
        Continuar com o processamento dos dados.


CalcularMetricas():

    media = CalcularMedia(dadosDeEntrada)
    mediana = CalcularMediana(dadosDeEntrada, media)
    desvioPadrao = CalcularDesvioPadrão(dadosDeEntrada, media)

    assimetria = CalcularAssimetria(dadosDeEntrada, media, mediana, desvioPadrao)


CalcularAssimetria(dadosDeEntrada, media, mediana, desvioPadrao):
    Se devioPadrao = 0, retornar 0.
    Senão retornar 3*(media - mediana)/desvioPadrao
```

Figura 6 – Cálculos iniciais do Prometheus

Com o objetivo final de separar os eventos da melhor forma possível, ou seja, caracterizar bem os diferentes tipos de fenômenos, identificamos a parte dos dados que são mais destoantes e relevantes, isso é, picos representados pelos dados que ocorrem com mais frequência em relação a própria FMR, como explicitado na Figura 7.

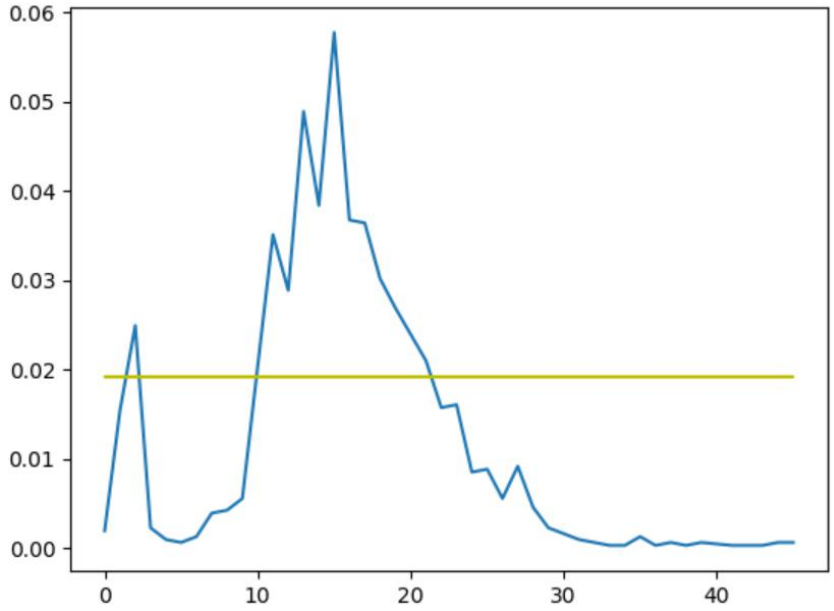

Figura 7 – Demonstração da FMR em um gráfico

Tendo em vista as diversas limitações de um sensor, como por exemplo o seu baixo

poder de processamento, a quantidade de dados que vão ser processados no algoritmo pode ser um fator determinante no tempo, custo e energia gastos na compilação e execução do algoritmo dentro do dispositivo.

Um vetor de tuplas é montado a partir das coordenadas do gráfico Frequência x Dado e será ordenado de acordo com os dados, de forma crescente. Desconhecendo, a priori, a quantidade de dados recebidos de entrada, foi desenvolvido um método para amenizar, de certa forma, a quantidade de dados a ser processada. Esse método é chamado de Pulo e tem como principal característica resumir os dados de entrada em um vetor menor que o anterior.

```
CriarVetorDePulo(dadosDeEntrada, dadoMaisFrequente):

    fatorDePulo = 10% * Tamanho(dadosDeEntrada)

    Se dadoMaisFrequente != primeiroDadoOrdenado:
        PercorrerDadosAEsquerda()

    Se dadoMaisFrequente != ultimoDadoOrdenado:
        PercorrerDadosADireita()

PercorrerDadosAEsquerda():
    Para cada dado a partir do primeiroDadoOrdenado até o dadoMaisFrequente, indo de fatorDePulo em fatorDePulo:
        Pulo.Adiciona(dado)

PercorrerDadosADireita():
    Para cada dado a partir do dadoMaisFrequente até o ultimoDadoOrdenado, indo de fatorDePulo em fatorDePulo:
        Pulo.Adiciona(dado)
```

Figura 8 – Criação do vetor de Pulo do Prometheus

O vetor de Pulo (Figura 8) calculado na aplicação leva em consideração, inicialmente, o dado de mais frequência dentro do vetor de entrada, ou seja, o pico mais alto no gráfico. A partir do registro desse dado no vetor de Pulo, são considerados os seus vizinhos de acordo com uma porcentagem equivalente a 0.1% da quantidade de dados inicial. Por exemplo: ao receber uma entrada de 10.000 dados, percorre-se o vetor de entrada de 10 em 10 posições para construir, então, o vetor de Pulo (Figura 9). Porém essa porcentagem pode ser mudada de acordo com a necessidade.

Existe um sacrifício feito ao se resumir os dados com o vetor de pulo estipulado. A perda de dados neste processo pode impactar na análise dos dados de maneira notável. Por esse motivo, adotamos a estratégia de sempre inicializar o vetor de pulo pelo dado mais frequente dentro da massa de dados, assim, sempre incluindo o dado mais representativo. Por este motivo, em certas situações específicas, o Prometheus pode vir a caracterizar eventos de forma menos intuitiva. Porém, nos nossos testes realizados, mostrados a partir do Capítulo 6, nenhum dos resultados pareceu ser influenciado negativamente por esta técnica.

Para se definir um evento, precisa-se caracterizar o grau de assimetria dos dados analisados. Temos a classificação do nível da assimetria como: moderada ou fortemente moderada (entre os módulos de 0.15 e 1.0 ou acima do módulo 1.0, respectivamente). Ao

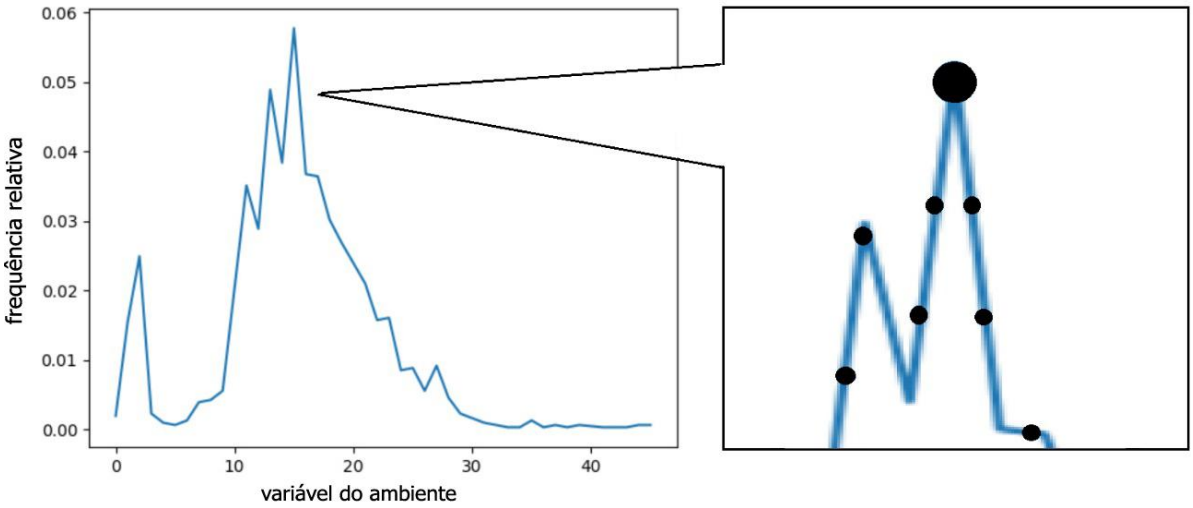

Figura 9 – Construção do vetor de Pulo a partir do dado mais frequente

verificar que a distribuição dos dados se encontra classificada como fortemente assimétrica, o algoritmo subentende que é necessário, então, realizar uma rotina de corte com o objetivo de separar os diferentes eventos ali encontrados.

4.3 ROTINA DE CORTE

A separação dos diferentes grupos de dados por meio da heurística de corte se torna vital para o sucesso de todo o algoritmo, pois a tomada de decisão se apoiará nas diferentes regiões identificadas (eventos). Portanto, a qualidade da escolha final do usuário pode ser diretamente afetada, previamente, por essa definição de corte. Com isso em vista, através de uma análise mais profunda do comportamento dos dados, foi desenvolvida uma rotina específica para lidar com todo esse processo.

A rotina de corte possui técnicas para conseguir separar os fenômenos de forma mais analítica. A partir da FMR, ou frequência média relativa, definimos um limite para a diferenciação e discrepância entre os dados estudados.

A definição do evento pode não ser tão confiável quando se deixa de analisar o comportamento da entrada mais a fundo. Ao se utilizar apenas do cálculo do primeiro momento do conjunto, ou seja, de uma média aritmética flutuante, a separação dos eventos se dá de forma simplória e nem sempre assertiva.

O algoritmo Hephaestus (AQUINO et al., 2016), neste ponto, quando detecta que os dados processados estão com um alto nível de assimetria (> 1.0) atrelado a eles, executa o processo que chamamos de "corte": a partir da média aritmética flutuante dos dados coletados, o grupo de dados inicial é separado em dois, um menor e outro maior que a média. São realizados novamente, então, todos os processos anteriores para esses dois grupos distintos de dados que foram originados a partir dessa rotina de corte.

Contudo, a escolha da média flutuante pode não ser tão eficiente (Figura 10). Para suportar diversas aplicações, diferentes alcances e diferentes características precisam ser considerados no momento de processar os dados de entrada.

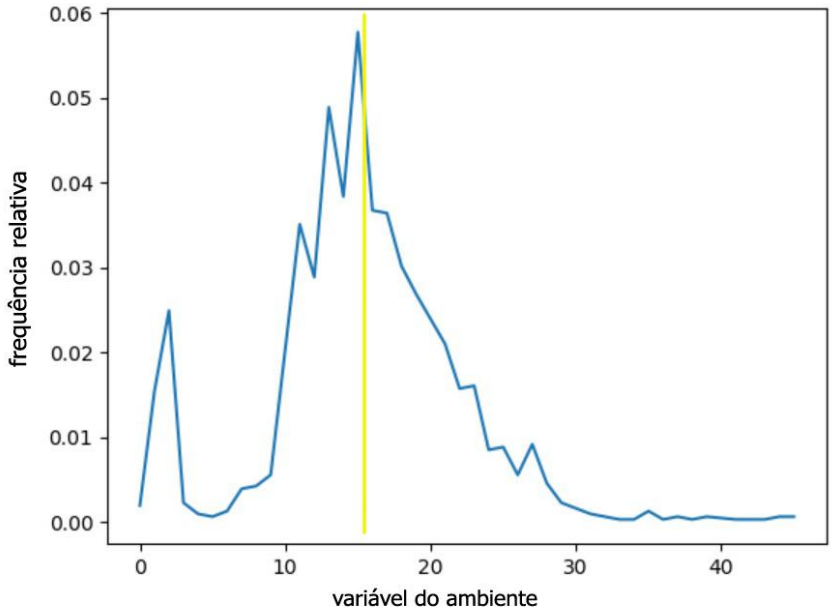

Figura 10 – Corte realizado pela Média aritmética

Como observado, um simples gráfico cortado pela heurística da média aritmética flutuante pode acabar sendo separado; mesmo que, visualmente e intuitivamente, se trate de um único evento. Esta escolha de corte não leva em consideração o objetivo final do processo (a definição dos próprios eventos), enquanto mostraremos adiante que as heurísticas escolhidas para compor o processo de corte do Prometheus leva em consideração, justamente, critérios que precisamos para definir um evento.

Para o Prometheus, a rotina de corte (Figura 11) começa percorrendo o vetor de pulo a procura do dado de maior frequência. A partir desta coordenada, o objetivo se torna encontrar o próximo turning-point, ou seja, quando a frequência do ponto anterior é menor do que a posterior, para os possíveis dois lados do pico. A cada turning-point encontrado, é verificado se a coordenada escolhida está acima ou abaixo da FMR. Caso esteja ainda acima, este é desconsiderado e o algoritmo volta a procurar por um outro candidato, já que ele ainda está sendo considerado parte do pico encontrado. Caso contrário, o algoritmo define o turning-point como um ponto de corte, isso é, um dos limites de um evento. Ainda há a possibilidade de não haver nenhum turning-point nessa busca, e nesse caso o algoritmo define o último ponto encontrado como o próprio limitador.

Com os dois pontos de corte encontrados, um evento é definido. Caracteriza-se esta região como um evento para, então, iniciar o processo de corte para os dados restantes. Esta recorrência possui um critério de parada além da assimetria testada, pois, teoricamente, nada garante que o restante dos dados consequentes ao corte convergirá para uma região

mais simétrica (por mais que, intuitivamente, estas tendam a ser menos assimétricas, já que estamos retirando uma parte mais destoante do gráfico assimétrico, caracterizando-a como um evento).

A partir da primeira iteração do algoritmo até a definição do primeiro evento, a sequência se dá na verificação de se o próximo pico está acima ou abaixo da FMR, pois a mesma é reutilizada em todas as iterações, ou seja, não é recalculada. Se acima, encontram-se então, os novos eventos a partir dessas novas análises, percorrendo os próximos picos mais frequentes, encontrando novos turning-points na medida em que o algoritmo avança (importante enfatizar que a FMR não é recalculada, e isso ajuda em termos de custo de processamento do algoritmo). Caso contrário, se o pico estiver abaixo, significa que não existem mais iterações e é definido um evento com o restante dos dados, que chamamos de "resto".

```
RotinaDeCorte(vetorDePulo, FMR):

    Se dadoMaisFrequente != vetorDePulo[Primeiro]:
        Para cada dado enrte o vetorDePulo[Primeiro] até dadoMaisFrequente:
            VerificarFMR()

    Se dadoMaisFrequente != vetorDePulo[Último]:
        Para cada dado enrte o dadoMaisFrequente até vetorDePulo[Último]:
            VerificarFMR()

VerificarFMR():
    Se frequencia do dado maior que FMR:
        Continua verificando
    Senão
        Define um lado do corte
```

Figura 11 – Processo simplificado da rotina de corte do Prometheus

Um certo critério de parada se faz necessário neste processo, tendo em vista que não há garantia de que o grau de assimetria convergirá para um nível menor que um. Ou seja, não podemos afirmar que a separação de dados classificados como assimétricos gerem regiões mais simétricas que o grupo anterior, o que na conjectura atual dos processos poderia gerar um loop infinito. É implementado, então, um número máximo de sete iterações para o processo de corte e separação dos dados no algoritmo, impedindo, assim, que o mesmo gere overhead ou entre em um loop infinito. Porém, como mostraremos nos resultados das simulações e casos de testes, não aconteceu nenhuma vez onde todas as iterações tenham sido executadas nos nossos experimentos. Indicando, assim, uma certa convergência para a simetria dos dados a partir da rotina de corte.

Com tudo isso feito, enviamos a saída no nosso algoritmo para um sistema de decisão, a fins de teste, para simular uma tomada de decisão de um usuário final. Basicamente,

este sistema é composto por testes ternários simples (if then else) que averiguam em qual situação pré definida o sistema se encontra naquele determinado evento.

5 IMPLEMENTAÇÃO

Neste capítulo serão apresentados os detalhes de implementação do Prometheus. O capítulo está dividido em: (i) microcontrolador utilizado, (ii) IDE utilizada e (iii) detalhes de implementação do algoritmo nas plataformas escolhidas.

5.1 MICROCONTROLADOR ESP8266

Com a finalidade de conseguir executar os casos de teste foi preciso um microcontrolador para compilar e executar o código construído a partir do algoritmo. O microcontrolador utilizado foi o ESP8266 (da plataforma NodeMCU), muito utilizado para desenvolvimentos IoT e difundido pela facilidade na sua utilização e instalação. A ESP8266 (Figura 12) possui um processador ESP8266-12E, que pode operar em 80MHz/160MHz, 4Mb de memória flash, 64Kb para instruções e 96Kb para dados.

Figura 12 – NodeMCU: ESP8266 microcontrolador

A implementação do algoritmo para esta placa foi dada em Micropython – uma implementação do software da linguagem de programação baseada em Python 3.0, escrita em C, otimizada para microcontroladores. Com a facilidade na adaptação da linguagem Python 3.0 para Micropython, a escolha da linguagem de programação para o começo da implementação era óbvia. O desenvolver do algoritmo e primeiras simulações se deram, então, a partir do Python 3.0, que tem como algumas das suas principais vantagens a facilidade na construção de códigos, portabilidade e extensibilidade. Foram utilizadas, também, algumas bibliotecas conhecidas auxiliares para o desenvolvimento do código e adaptação para o Micropython.

5.2 IDE ESPLORER

A partir da IDE ESPlorer, desenvolvida especialmente para esse tipo de plataforma, conseguimos compilar e rodar o algoritmo na placa citada. Encontramos complicações em diversas fases em busca da compilação do Prometheus no microcontrolador, como a instalação do SDK do Micropython na placa, instalação e configuração da IDE ESPlorer no computador e principalmente no gerenciamento, manutenção e otimização do código com o objetivo de compilar. Na Figura 13 abaixo mostramos como a IDE se parece.

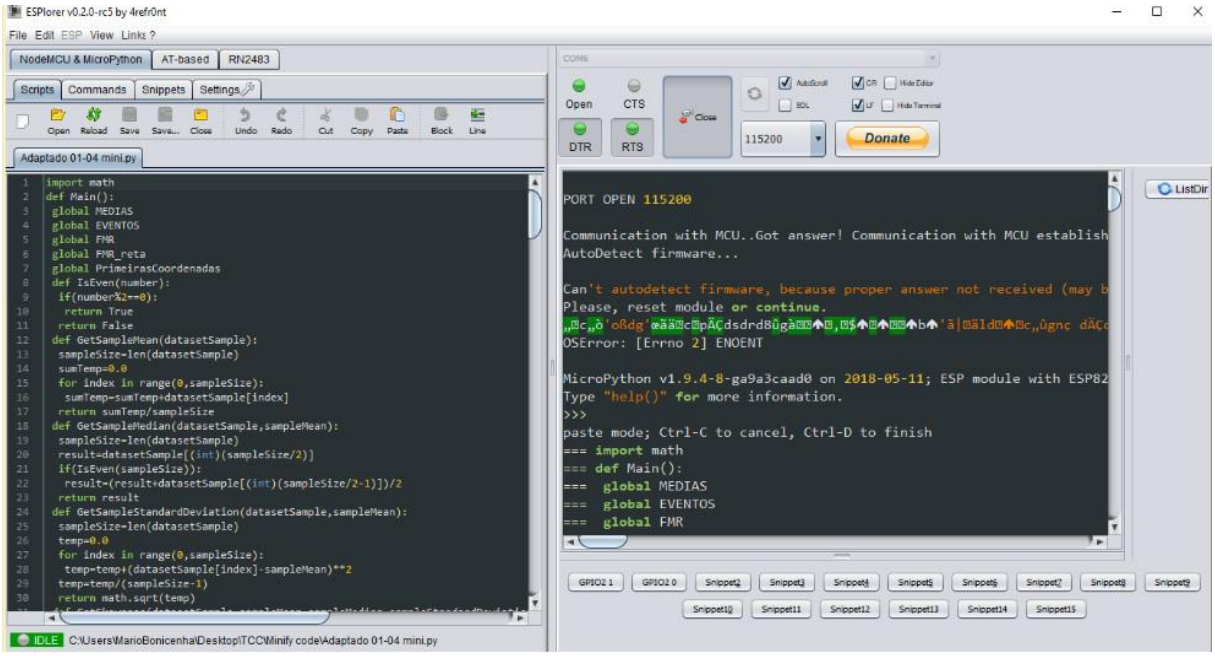

Figura 13 – IDE ESPlorer com microcontrolador conectado ao computador.

Uma das técnicas utilizadas para otimizar a memória consumida pelo algoritmo foi a minimização do algoritmo criado, onde espaços em branco desnecessários são removidos, nomes de variáveis e funções podem ser renomeadas por alguma nomenclatura que consuma menos memória no microcontrolador. Um dos maiores desafios se encontrou na otimização do algoritmo, pois o microcontrolador possui limitações em questão de memória.

5.3 IMPLEMENTAÇÃO DO ALGORITMO

Nesta subseção demonstraremos alguns códigos implementados para realizar alguns dos principais processos feitos pelo Prometheus. Abaixo, mostramos como funciona o fluxo final do algoritmo através de um diagrama de atividades (Figura 14).

A decisão de realizar ou não as rotinas para a classificação dos eventos, destacada em amarelo, é um ponto crucial do código. Ao verificar o grau de assimetria, o Prometheus finaliza seu processo de tratamento em relação àquela massa de dados específica caso o

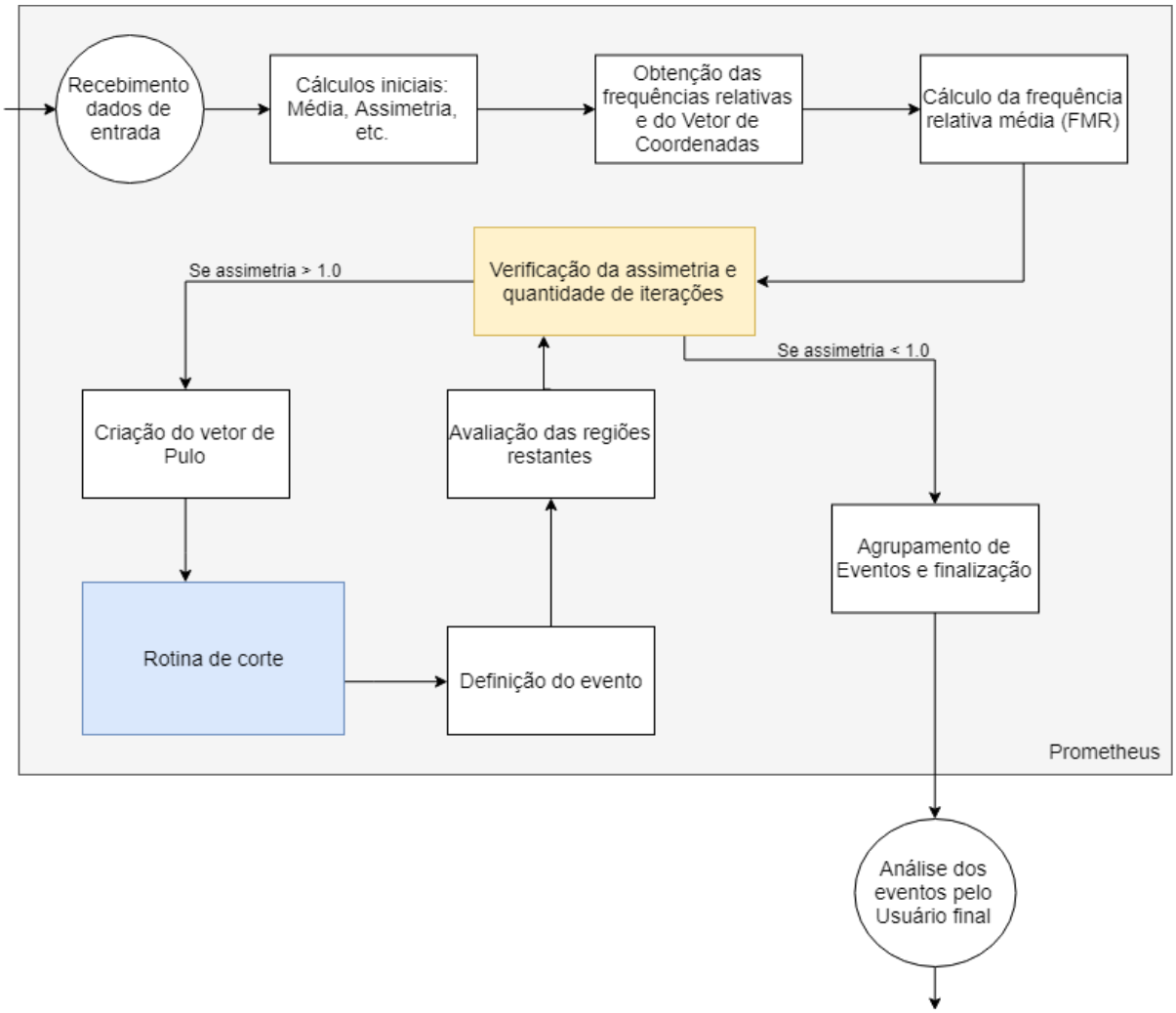

Figura 14 – Diagrama descrevendo o fluxo do código

grau de assimetria seja menor que 1.0, ou volta a percorrer o conjunto de processos que consiste a análise de dados caso contrário. Como explicado no capítulo 4, a quebra dos dados em regiões consiste em uma vital parte de todo o algoritmo, mas só acontece, de fato, depois de construídas estruturas auxiliares.

Sendo necessário analisar os dados dentro da região sensoriada, o algoritmo define algumas estruturas de dados para facilitar o manuseio dos mesmos. O vetor de pulo, lista de coordenadas (formada por tuplas como temperatura e frequência relativa, por exemplo, dependendo do tipo de dados tido como entrada) e a variável do tipo double chamada FMR são os exemplos que mais são citados neste trabalho por realizarem grande e importante parte do algoritmo Prometheus. A partir da obtenção destas estruturas, a quebra dos dados em regiões se dá início.

O processo de quebra (ou corte) (Figura 15) pode ser considerado como a parte mais importante do Prometheus. É através desse processo que há a maior diferenciação entre este algoritmo e os demais. Após a construção do vetor de pulo, explicado no capítulo

anterior, a rotina de corte dos dados se limita na FMR para conseguir, a partir dos picos dos gráficos - ou seja, dos dados mais frequentes -, classificar um evento. Na figura abaixo mostramos como implementamos a rotina de corte (destacada, em azul, no diagrama de fluxo do código) para nossas simulações em Python.

```python
@profile
    def CortarAmostraDeDados(pulo, index_max_pulo, FMR):
        tamanho_pulo = len(pulo)
        dado_max = pulo[index_max_pulo]
        index_corte_esquerda = -1
        index_corte_direita = -1
        corte_esquerda = -2
        corte_direita = -2

        if index_max_pulo != 0:
            for index in range(index_max_pulo, -1, -1):
                if (pulo[index][1] <= dado_max[1]):
                    dado_max = pulo[index]
                elif (pulo[index][1] > FMR):
                    dado_max = pulo[index]
                else:
                    index_corte_esquerda = index+1
                    corte_esquerda = pulo[index_corte_esquerda]
                    break;
        else:
            corte_esquerda = pulo[0]
        if corte_esquerda == -2:
            corte_esquerda = pulo[0]

        if index_max_pulo != tamanho_pulo:
            dado_max = pulo[index_max_pulo]
            for index in range(index_max_pulo, tamanho_pulo, 1):
                if (pulo[index][1] <= dado_max[1]):
                    dado_max = pulo[index]
                elif (pulo[index][1] > FMR):
                    dado_max = pulo[index]
                else:
                    index_corte_direita = index-1
                    corte_direita = pulo[index_corte_direita]
                    break;
        else:
            corte_direita = pulo[index]
        if corte_direita == -2:
            corte_direita = pulo[index]

        print("Corte Esquerda: ", corte_esquerda)
        print("Corte Direta: ", corte_direita)

        return corte_esquerda, corte_direita
```

Figura 15 – Implementação em Python da rotina de Corte

Este código foi representado, no capítulo anterior, pelo pseudocódigo de número 3, onde a FMR é dada de entrada como um dos parâmetros deste método. Ao realizar a separação da massa de dados em regiões, o algoritmo analisa, novamente, o restante dos dados que não foram classificados como eventos nesta iteração. O processo se repete até achar todos os eventos de dentro da entrada ou atingir um número máximo já predefinido de 7 iterações.

6 CASOS DE ESTUDO

Neste capítulo demonstraremos as métricas que usamos para avaliar o algoritmo e alguns dos testes realizados. O objetivo almejado aqui é que a fusão dos dados seja feita da melhor forma possível para que o usuário consiga entender o que está acontecendo no ambiente e tomar a melhor decisão sobre o que fazer a partir disso.

6.1 CASO DE TESTE: LINHA DE ENERGIA E BATERIA

Nossas simulações, neste caso, foram feitas para duas aplicações (DENHOLM; KULCINSKI; HOLLOWAY, 2005), simulação essa que já foi feita para o Hephaestus e conseguimos analisá-la a nível de comparação. O cenário envolve duas aplicações que compartilham a mesma infraestrutura de comunicação e sensoriamento, uma aplicação para o monitoramento de uma linha de transmissão de energia elétrica e outra para o monitoramento de uma bateria usada em torres de transmissão. As torres de transmissão são responsáveis por sustentar as linhas de energia, que são o meio mais comum de transmissão de energia, além de armazenarem energia usando baterias embutidas.

É importante ressaltar que as duas aplicações se correlacionam, o que significa que o comportamento de uma pode afetar o comportamento da outra, pois estamos analisando o compartilhamento do mesmo tipo de informação entre elas, que é a temperatura.

As especificações que precisam ser ditas sobre as aplicações são referentes às do trabalho feito sobre o Hephaestus (AQUINO et al., 2016): a temperatura a 75ºC pode causar danos à linha de energia. Se a linha for danificada não tem suprimento de energia. Consequentemente, não há energia a ser armazenada pela bateria mesmo que 75ºC represente uma temperatura normal para a aplicação do monitoramento da bateria (a temperatura normal da bateria é de 40ºC à 144ºC), mas a mesma temperatura representa que a linha pode ser danificada para a aplicação da linha de energia (a temperatura normal e segura dessa aplicação é entre 40ºC e 65ºC). Para avaliar o impacto de lidar com múltiplas aplicações simultaneamente, foram criadas quatro situações: C1, C2, C3 e C4. Cada situação representa um estado particular das aplicações ao longo do tempo, um evento particular a ser detectado. C1 representa condições ideais para as duas aplicações (condição segura, onde não é necessário ter ações preventivas para evitar danos na linha de energia e na bateria). C2 representa um aumento na temperatura da linha. C3 representa um aumento na temperatura da bateria. C4 representa um aumento tanto da bateria quanto da linha de energia. Os dados das situações estão resumidos na Tabela 1.

Durante o C1, a linha de energia está em sua condição normal e gera amostras de temperatura entre 40ºC à 65ºC e a bateria também está em suas condição normal e gera amostras de 40ºC à 144ºC. Durante o C2, a linha está sobrecarregada e gera amostras de

65ºC à 75ºC, enquanto a bateria está em suas condições normais e gera amostras de 40ºC à 144ºC. Durante C3, a linha de energia está em sua condição normal e gera amostras de temperatura entre 40ºC à 65ºC enquanto a bateria está sobrecarregada e gera amostras de mais de 144ºC. Finalmente, durante o C4, a linha de energia está sobrecarregada e gera amostras de 65°C à 75°C enquanto a bateria também está sobrecarregada e gera amostras de mais de 144ºC.

Casos	**Intervalo de temperatura**
C1	40°C à 65°C na linha de energia sobrecarregada 40°C à 144°C na bateria
C2	65°C à 75°C na linha de energia sobrecarregada 40°C à 144°C na bateria
C3	40°C à 65°C na linha de energia sobrecarregada Acima de 144°C na bateria
C4	65°C à 75°C na linha de energia sobrecarregada Acima de 144°C na bateria

Tabela 1 – Intervalos de temperatura: linha de energia e bateria

6.1.1 Caso 1

Na primeira simulação a massa de dados de entrada foi configurada atendendo bem as duas aplicações apresentadas, explicitadas por C1 na Tabela 1. Ao representar as duas aplicações funcionando de maneira usual, existe um intervalo comum entre as mesmas que ocorre de 40º C até 65º C e que não há distinção na realização das análises. Ou seja, o algoritmo não reconhece de qual aplicação estes dados são originados.

Contudo, podemos perceber que na realização da rotina de corte explicada nos capítulos anteriores, o algoritmo consegue reconhecer os diferentes eventos através do cálculo da FMR e da análise dos dados.

A saída obtida em 3 distintos eventos (Figura 16) pode evidenciar para o usuário final a distinção das aplicações. A futura análise dessa saída por um expert não é englobada no escopo desse trabalho, já que o objetivo deste consiste na abstração da massa de entrada e dos diferentes contextos que as mesmas podem servir.

Em comparação com o algoritmo primário do Hephaestus, a saída não muda de forma significativa. Porém, podemos identificar que, dependendo da situação, as saídas estão corretas; isto é, ela resume de forma sucinta a entrada dos dados para uma tomada de decisão.

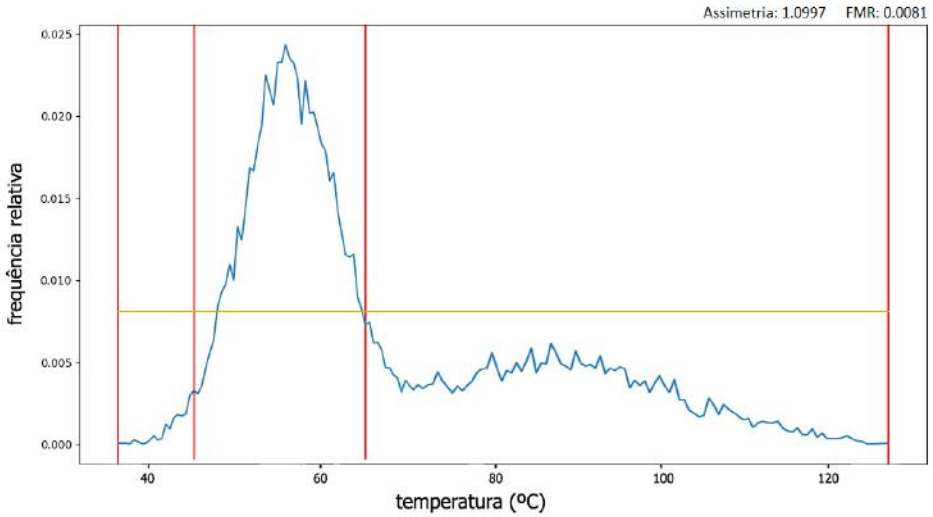

Figura 16 – Casos de Teste: 1ª simulação

6.1.2 Caso 2

Para o segundo teste (Figura 17) foi calculada uma assimetria absoluta menor que 1.0 e, por esse motivo, não houve necessidade de realizar qualquer rotina de corte. Como explicado anteriormente, ao se deparar com uma massa de dados considerada fortemente simétrica (< 1.0), o algoritmo designa apenas um evento a todo o conjunto de dados analisado. Com isso, as saídas tanto do Hephaestus original quanto a deste trabalho não seriam diferentes.

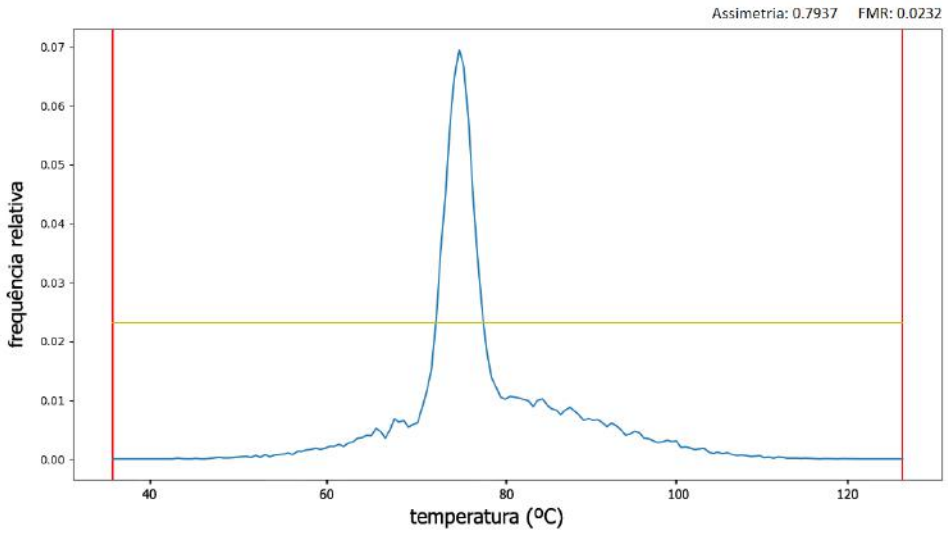

Figura 17 – Casos de Teste: 2ª simulação

Hipoteticamente, ao realizar a rotina de corte para esse gráfico no caso em que este fosse assimétrico, seria explicitado o comportamento não seguro da aplicação correspondente à linha de energia envolvendo as temperaturas 80º C e 95º C através de um evento.

6.1.3 Caso 3

Quando não existe um intervalo comum entre as aplicações em um determinado caso, a separação de eventos se torna mais evidente e clara. No terceiro teste realizado (Figura 18), as situações tanto do cabo de energia (segura) e da bateria (não segura) são resumidas em dois eventos onde cada um é representado por um pico diferente, como no mostrado no gráfico abaixo.

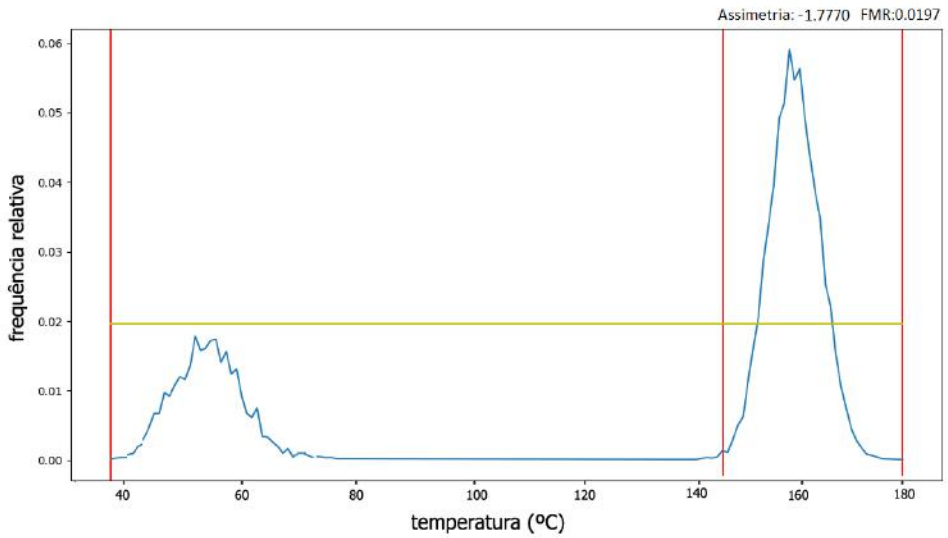

Figura 18 – Casos de Teste: 3ª simulação

6.1.4 Caso 4

Assim como no caso anterior, não existe um intervalo de temperaturas em comum (Figura 19). Assim, o algoritmo consegue identificar facilmente onde separar os eventos. Uma pequena diferença para a saída do Hephaestus se dá na quantidade de eventos classificados. Porém, se tratando do contexto geral dos dados de entrada, esta solução foi interpretada como igualmente cabível.

6.2 ESTUDO DE CASO: PROMETHEUS

Ao compararmos os resultados anteriores não vemos, a priori, uma diferença nítida e significativa entre a saída dos Hephaestus e do Prometheus, afinal a rotina de corte é uma das mais importantes rotinas do algoritmo. Com o intuito de demonstrar como o nosso algoritmo se diferencia do anterior (Figura 20 e Figura 21), simulamos um exemplo onde as saídas dos dois algoritmos diferem.

Realizando a rotina de corte através da média flutuante dos dados de entrada, obtém-se o corte dado pela linha central vertical do gráfico (20). Na iteração seguinte, o algoritmo encontra um grau de assimetria satisfatório (< 1.0) para as duas subpartes. Assim,

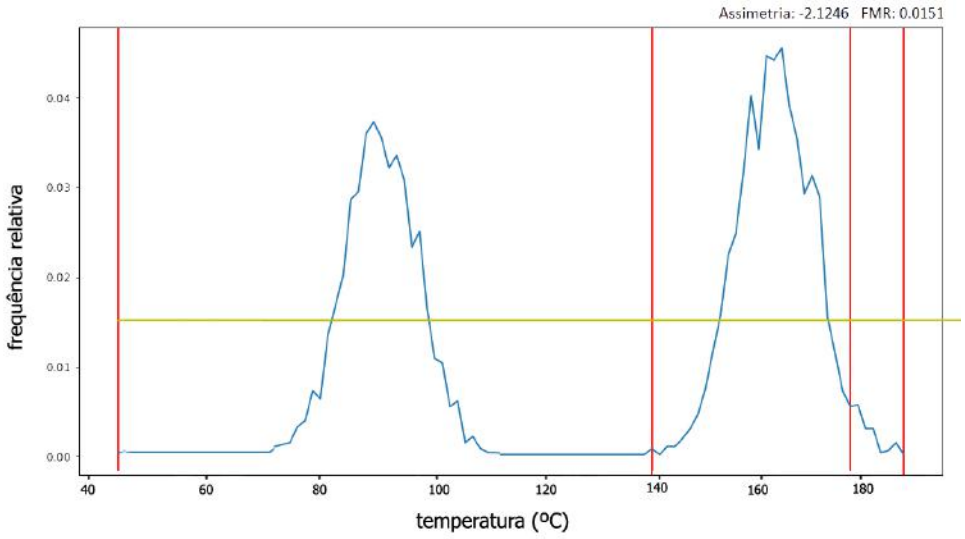

Figura 19 – Casos de Teste: 4ª simulação

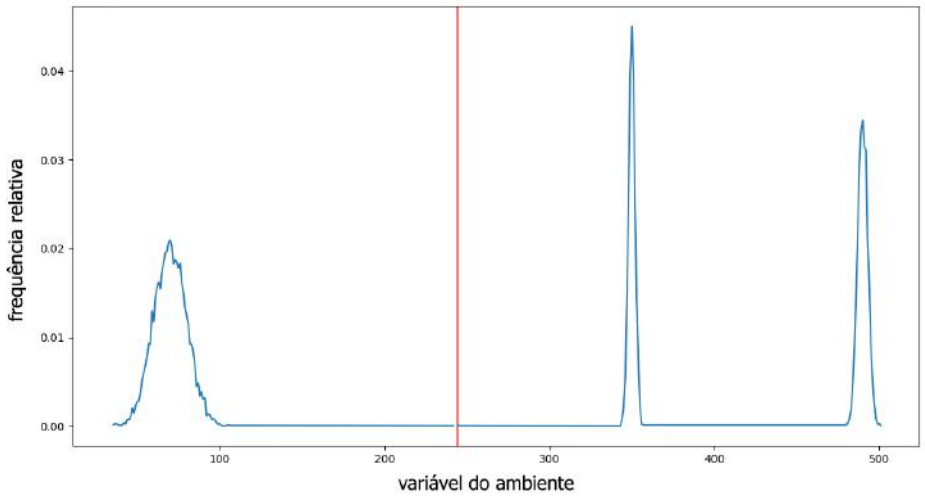

Figura 20 – Análise dos dados segundo a heurística do Hefestos

caracteriza-se um evento a partir de cada região especificada. Percebe-se então que os dois últimos picos destoantes do gráfico se encontram dentro de um mesmo evento. Intuitivamente, porém, não necessariamente se trataria da mesma situação.

Com a análise de dados realizada por este trabalho, o Prometheus, a rotina de separação de eventos designaria diferentes cortes para a mesma massa de dados, como mostra a Figura 21.

Ao procurar especificamente pelas partes que destoam do gráfico inicialmente assimétrico, o algoritmo consegue realizar a separação dos eventos de forma mais apurada. Separando os diferentes picos encontrados, pode-se perceber que, independente dos possíveis contextos, nosso algoritmo classifica os eventos, de forma mais eficaz, de acordo com

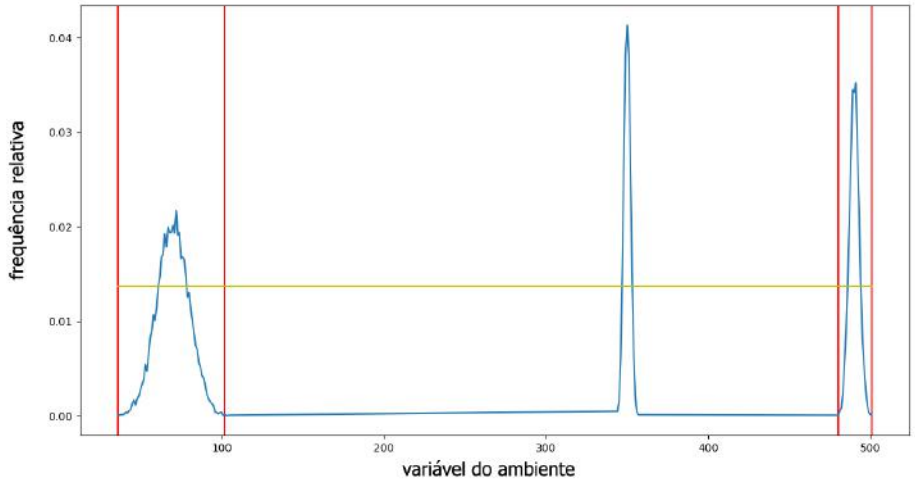

Figura 21 – Análise dos dados segundo a heurística do Prometheus

a análise da massa de dados realizada.

6.3 SINAL VITAL DE PACIENTE EM OBSERVAÇÃO: RESPIRAÇÃO

A partir de todo o esforço de criação do algoritmo, precisávamos aplicá-lo em casos reais para avaliar seu real funcionamento. Este capítulo se trata exatamente de um caso de estudo real e o comportamento do nosso algoritmo diante dos fatos e dados recebidos.

Nosso caso de estudo escolhido foi o de informações vitais de um paciente em observação através de uma máquina de sinais. Avaliamos a amostra de dados obtida em um determinado intervalo de tempo a partir de informações sobre sua respiração medida em respiração por minuto (rpm) [1].

O objetivo da análise do algoritmo sobre os dados de entrada era simplesmente identificar comportamentos adquiridos pela respiração do paciente e informar um expert na leitura da máquina a tomar uma decisão do que fazer com o paciente.

Com esta amostra, a assimetria dos dados foi acima de 1.0, indicando uma necessidade de separação de eventos para análise. Houve apenas duas iterações resultando em dois cortes e três eventos. Nosso algoritmo se comportou exatamente como esperávamos, identificando tanto os pontos de discrepância quanto os pontos considerados normais (a ver na Figura 22).

Foram identificados três eventos a partir da amostra enviada. Essas novas informações podem ser chamadas de resumos de todos os dados de entrada recebidos pelo algoritmo, os quais um expert teria que despender algum tempo analisando mais profundamente e que com a nova saída é mais intuitivo de se entender para, a partir disso, tomar suas decisões em cima desta análise.

[1] <https://physionet.org/cgi-bin/atm/ATM-MIMICDatabaseNumerics(mimicdb/numerics)>

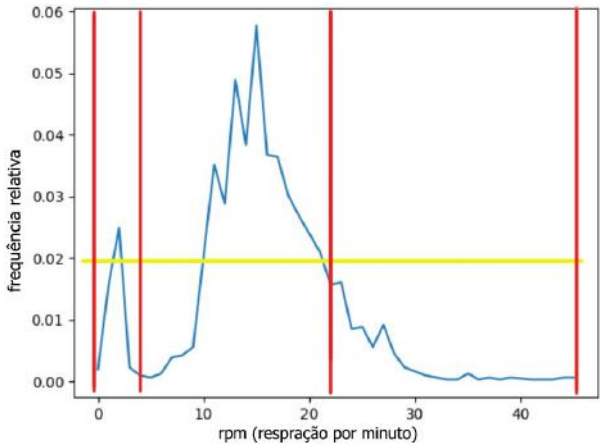

Figura 22 – Análise de respiração (rpm) versus frequência

7 EXPERIMENTOS

Nesta seção analisamos os resultados obtidos através dos algoritmos Hephaestus e Prometheus. Comparamos a capacidade de encontrar fenômenos na massa de dados (que chamaremos de acurácia), consumo de recursos em termos de memória RAM e também analisamos rapidamente a energia consumida dos dois algoritmos citados.

7.1 EXPERIMENTO ANALISANDO O CONSUMO DE MEMÓRIA RAM

Uma métrica escolhida para analisar a viabilidade do Prometheus é o consumo de RAM (*Random Access Memory*). Para isso, separamos algumas configurações de entrada para poder, de fato, medir o impacto que uma maior entrada pode ou não afetar no consumo de RAM. Utilizamos a biblioteca chamada Psutil [1] para nos ajudar a coletar a quantidade consumida desejada. Importante frisar que não é um número totalmente exato e, por isso, decidimos por descartar as maiores e menores quantidades registradas para o conjunto que apresentasse resultados exorbitantes que não retratam a realidade do consumo da aplicação.

A primeira configuração consistia em uma entrada com 50 dados apenas. Para esta, realizamos um conjunto de 10 testes. Obtivemos uma média de 45.2 Kb de memória RAM utilizada durante a execução do algoritmo. Não obtivemos nenhuma medida fora do esperado ou exorbitante, todos os 10 testes foram utilizados para esse resultado médio.

Para a segunda configuração designamos uma entrada de 200 dados. Com a mesma quantidade de testes realizados, 10, obtivemos uma média de 49,2 Kb de memória RAM utilizada. Neste conjunto de testes tivemos um consumo de RAM, para um dos experimentos, destoante, registrando um total de 116 Kb consumidos. Porém, por não considerarmos suficientemente destoante para descartar, continuamos contando com o mesmo para chegarmos no resultado de 49.2 Kb médio gasto de RAM para este conjunto.

Mesmo aumentando o número de dados iniciais sensoriados para 1000 por processo, o consumo de RAM continuou, em média, para os 10 testes, 51.9 Kb. Concluímos então que o consumo de RAM para este projeto se encontra suficiente para a proposta tida pelo trabalho, ou seja, sensores conseguem realizar as tarefas de modo adequado para uma quantidade razoável de entrada de dados.

7.2 EXPERIMENTO ANALISANDO A ACURÁCIA DE RESULTADOS

Para avaliarmos a eficiência do algoritmo Prometheus em termos de proporção, analisamos uma amostra contendo 90 diferentes entradas e comparamos estas com os resultados

[1] <https://pypi.org/project/psutil/>

obtidos a partir do Hephaestus. Para medir como o Prometheus se comporta em questão de acurácia, as análises se baseiam na quantidade de acertos na separação das curvas mais destoantes no gráfico temperatura vs frequência relativa. Essas regiões representam os eventos caracterizados por cada algoritmo e a partir dos picos mais destoantes na análise, podemos, em vários casos, definir qual deles representou melhor os mesmos.

A geração desses dados de entrada foi realizada com a ajuda de bibliotecas da linguagem Python, utilizada para desenvolver o algoritmo. Através da biblioteca Numpy[2], geramos diferentes funções através da distribuição gaussiana. Para esta análise, ignoramos os 48 gráficos tidos com um grau de assimetria insuficiente (ou seja, suficientemente simétricos), já que a diferença entre os dois algoritmos se baseia numa rotina apenas utilizada em dados assimétricos.

A Tabela 2 demonstra a classificação geral obtida pelo teste descrito, onde a quantidade absoluta representa quantas amostras cada classificação possui. No caso dos algoritmos, esta aponta a melhor caracterização de eventos em comparação ao outro.

Classificação/Quantidade	**Quantidade absoluta**
Resultados simétricos	58
Resultados similares	30
Prometheus	4
Hephaestus	2
Incertezas	6
Total	100

Tabela 2 – Resultados obtidos através da execução dos algoritmos Prometheus e Hephaestus

Como estamos medindo a eficiência do Prometheus através de suas rotinas de cortes e novas heurísticas, analisamos as porcentagens em relação aos resultados assimétricos obtidos. Das 100 saídas resultantes desta bateria de testes, obtivemos 42 saídas assimétricas, das quais aproximadamente 71% foram identificadas como potencialmente similares, pois a classificação dos eventos foi feita de modo similar e os pontos mais destoantes foram caracterizados; por volta de 14% consideramos como incertezas, pois suas rotinas de corte foram feitas de formas diferentes, resultando em diferentes saídas e que podem depender da escolha da aplicação; em aproximadamente 5% dos casos consideramos uma melhor caracterização do Hephaestus e nos outros 10%, uma melhor caracterização do Prometheus. Abaixo analisaremos um caso de cada para uma melhor exemplificação de cada interpretação (Incerteza, Hephaestus e Prometheus).

A Figura 23 é um exemplo de um caso de incerteza, onde os dois algoritmos obtêm diferentes caracterizações. O Hephaestus, por exemplo, faz a separação de um pico com dados de maior frequência juntamente à outro que poderia ser considerado como um

[2] <https://www.numpy.org/>

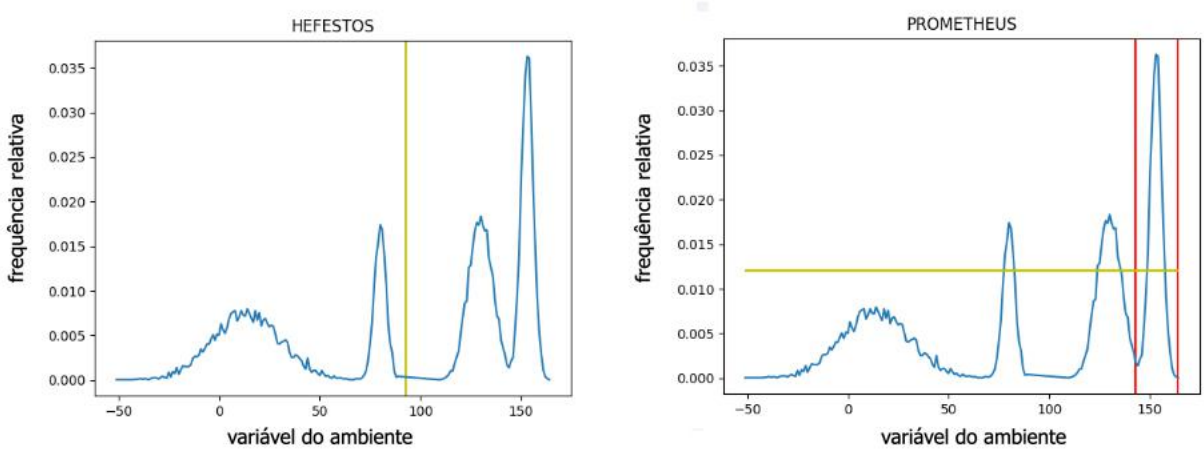

Figura 23 – Caracterizada como incerteza: dependente do objetivo da aplicação

outro evento, dependendo da aplicação. Já o Prometheus caracteriza esse maior pico separadamente de todos os outros, onde estes, menores, são considerados parte de um mesmo evento, quando poderiam ter sido considerados como diferentes. Então caracteriza-se o que chamamos de incerteza: diferentes aplicações poderiam se utilizar melhor de uma ou outra saída.

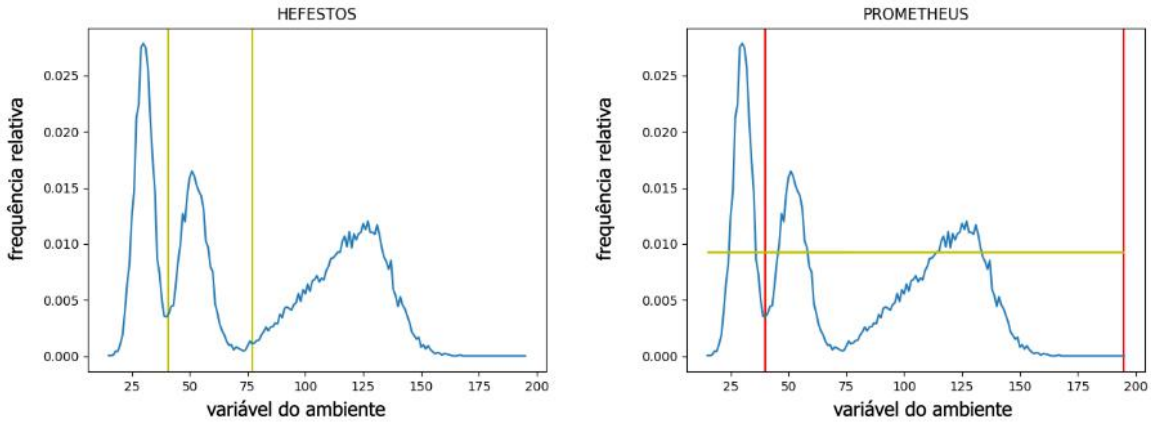

Figura 24 – Melhor caracterizada no Hephaestus

Para este exemplo (Figura 24), a interpretação em que o Hephaestus caracteriza melhor a massa de dados se dá a partir da separação de mais eventos destoantes do que o classificado pelo Prometheus. Porém, a interpretação tida pelo Prometheus não é falha e o maior destoante obtido é separado dos demais nos dois algoritmos.

Já na Figura 25, há uma maior caracterização de possíveis fenômenos feita pelo algoritmo Prometheus, enquanto o Hephaestus somente faz a separação do gráfico a partir da média calculada. Consideramos que há uma vantagem ao procurar especificamente eventos para fazer a separação das regiões em um gráfico. Este caso exemplifica como a separação do evento principal (pico centrado em 65) pode ser vantajosa no algoritmo do

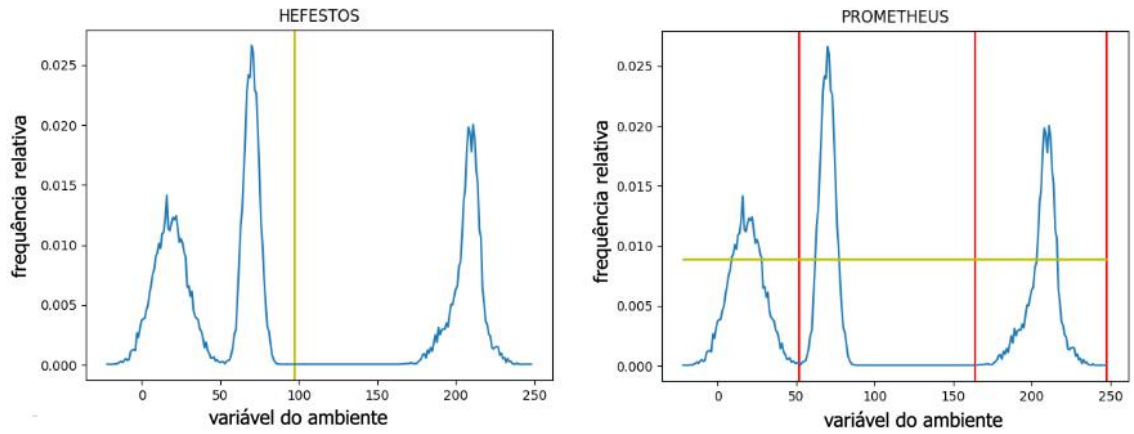

Figura 25 – Melhor caracterizada no Prometheus

Prometheus, enquanto o Hephaestus caracteriza as duas regiões separadas como suficientemente simétricas.

7.3 EXPERIMENTOS RELATIVOS AO CONSUMO DE ENERGIA

O consumo de energia do Prometheus é similar ao do Hephaestus visto que modifica-se somente o mecanismo de identificação de fenômenos. Desta forma podemos afirmar que o Prometheus é passível de ser implementado em sensores reais.

8 CONCLUSÃO

No contexto de Internet das Coisas, diversos dispositivos e sensores podem ser utilizados para automatizar e transformar dados em informações potencialmente valiosas. Através da utilização de sensores focados na coleta e processamento de uma grande quantidade de dados, foi desenvolvido o algoritmo chamado Prometheus para servir diferentes tipos de aplicações. Para uma tomada de decisão rápida e eficaz, estes dados são analisados e fundidos na caracterização de fenômenos.

Importante enfatizar que existiu uma grande dificuldade por parte dos autores de encontrar trabalhos relacionados que pudessem servir tanto como referência como comparação. Isso se deu principalmente pelos trabalhos existentes na literatura possuírem um objetivo voltado para uma única aplicação ao invés de múltiplas aplicações. E também por, em outros casos, terem que conhecer os requerimentos individuais iniciais de cada aplicação envolvida, restringindo assim sua área de atuação. Neste trabalho comparamos o Prometheus principalmente com o Hephaestus, já que os dois atuam para múltiplas aplicações e sem conhecer os requisitos iniciais de cada uma. O Prometheus busca ser de alguma forma adaptável e genérico, porém com uma análise mais complexa dos dados de entrada e uma melhor caracterização de eventos se comparado ao Hephaestus, por mais que seja computacionalmente mais custoso do que uma análise mais básica.

Estudos foram realizados e a partir de diversas análises foi desenvolvido o Prometheus, visando, principalmente, a otimização do processo chamado de corte. Sendo este o principal processo utilizado na caracterização de eventos, o mesmo impacta diretamente na decisão do usuário final. Com isso, novas heurísticas foram desenvolvidas a fim de inferir uma tomada de decisão mais rápida e eficiente.

Uma técnica mais diligente na separação e caracterização de eventos foi desenvolvida através dos cálculos da média, mediana, desvio padrão e assimetria. A utilização da FMR ajuda a reconhecer os dados destoantes e facilita a delimitação dos cortes. Já o procedimento de Pulo sumariza os pontos de interesse dentro da massa de dados quando há a necessidade de otimizar o algoritmo em termos de processamento.

Os casos de estudo se mostraram de grande importância na obtenção de resultados claros quanto a otimização proposta. Esses testes apontam similaridades nos casos também realizados pelo Hephaestus e, em algumas situações, uma melhora na construção dos eventos tidos como saída. Ao comparar apenas resultados elegíveis como melhor interpretado por um dos algoritmos, obtivemos então 66.6% de resultados favorecendo a saída do Prometheus em relação ao Hephaestus.

Este trabalho conta com uma solução capaz de ser processada em pequenos microcontroladores sem comprometer sua eficácia. É notável sua melhora em relação à versão anterior onde apenas se utilizava de um filtro de média flutuante como delimitador. Existe

uma diferença importante em parte da proposta do Prometheus versus a proposta do Hephaestus: o primeiro busca uma solução computacionalmente viável para apresentar os resultados ao usuário final, procurando sempre achar a melhor caracterização de eventos possíveis; já o segundo, Hephaestus, busca uma das soluções mais computacionalmente baratas (e, de certa forma, benéficas computacionalmente) na caracterização dos eventos. Com isso, tendo em vista a melhor caracterização dos eventos e resultados finais do algoritmo, a tomada de decisão do usuário final é potencialmente aprimorada, como pretende a proposta do Prometheus.

8.1 TRABALHOS FUTUROS E DESAFIOS

A fim de melhorar o trabalho aqui apresentado, os trabalhos futuros visados possuem três frentes: processamento, verificação de possíveis falhas na classificação de eventos, análise dinâmica da FMR.

Quanto ao processamento dos dados, o algoritmo precisa, ainda, ser otimizado para conseguir ser executado em sensores de pequeno porte. Ao ser utilizado no ESP8266, com apenas 4Mb de memória flash, 64Kb para instruções e 96Kb para dados, a pouca flexibilidade resultante nos impossibilitou de fazer testes mais profundos na prática.

Classificamos certas regiões obtidas, resultantes do Prometheus, como falhas menores, chamadas de "resto". Os restos são eventos classificados ao final do processamento dos dados da iteração da vez. Uma análise mais profunda pode eliminar esses restos, potencialmente não desejados, com o objetivo de obter um resultado mais claro e menos confuso para o usuário final.

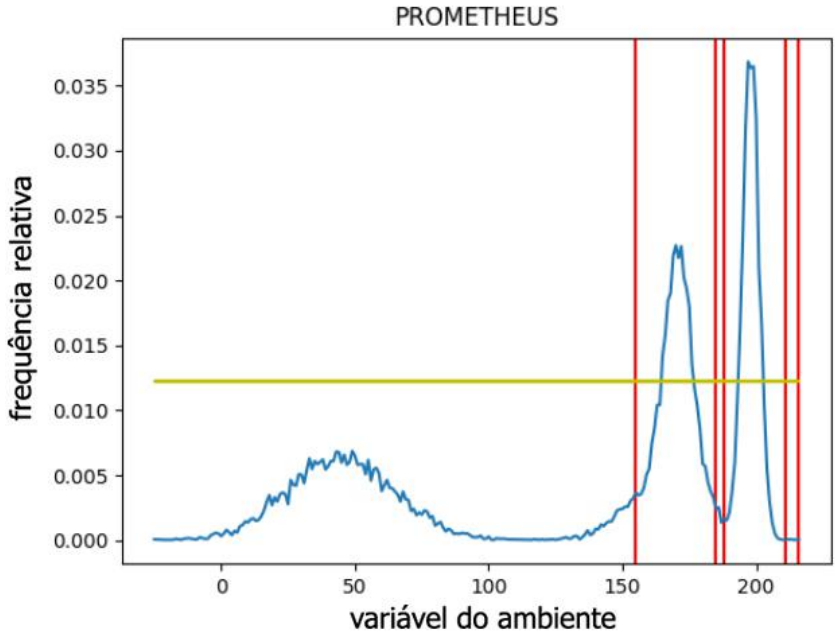

Figura 26 – Restos obtidos em uma das análises do Prometheus

Na Figura 26 conseguimos identificar as regiões classificadas como Resto entre os picos mais destoantes, por volta do intervalo entre 180 e 185, e no final da massa de dados, entre 215 e 225.

Como último trabalho futuro temos análise dinâmica da FMR, que consiste na verificação da FMR de acordo com algumas métricas referentes à massa de dados. Nota-se que ao adotar um multiplicador para a FMR em alguns casos pode vir a ser tornar benéfico para a análise geral da entrada. Na Figura 27, temos a diferença entre duas análises do Prometheus quando a FMR possui multiplicador 1 e 2/3 respectivamente.

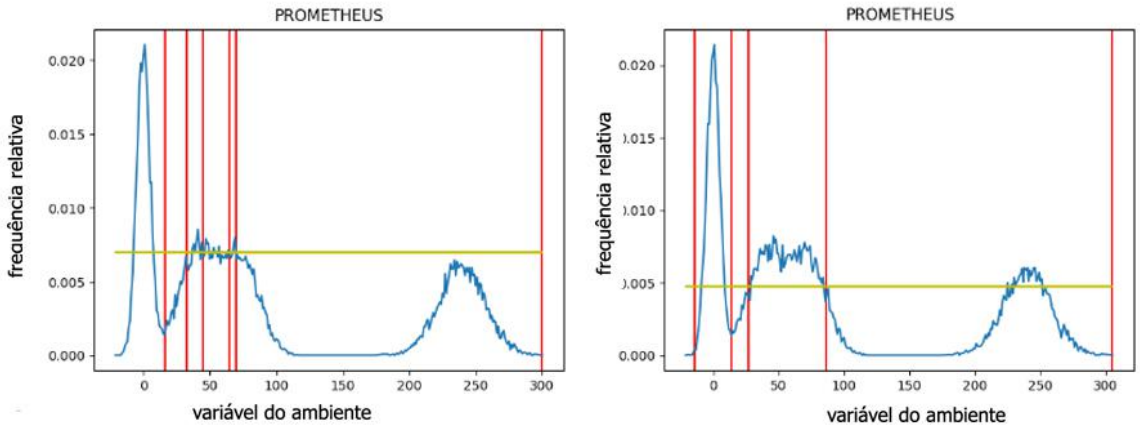

Figura 27 – Exemplificação da diferença de análise ao se utilizar de uma FMR com multiplicador dinâmico

REFERÊNCIAS

AKYILDIZ, I. F. et al. Wireless sensor networks: a survey. **Computer networks**, Elsevier, v. 38, n. 4, p. 393–422, 2002.

AQUINO, G. et al. Hephaestus: A multisensor data fusion algorithm for multiple applications on wireless sensor networks. In: IEEE. **2016 19th International Conference on Information Fusion (FUSION)**. [S.l.], 2016. p. 59–66.

ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. **Computer networks**, Elsevier, v. 54, n. 15, p. 2787–2805, 2010.

BIRD, C. M.; BEERS, T. C. Astronomical applications of distribution shape estimators. **The Astronomical Journal**, v. 105, p. 1596–1606, 1993.

CALDAS, G. et al. S-leach: A leach extension for shared sensor networks. In: THE STEERING COMMITTEE OF THE WORLD CONGRESS IN COMPUTER SCIENCE, COMPUTER .... **Proceedings of the International Conference on Wireless Networks (ICWN)**. [S.l.], 2015. p. 121.

DELICATO, F. et al. An efficient heuristic for selecting active nodes in wireless sensor networks. **Computer Networks**, Elsevier, v. 50, n. 18, p. 3701–3720, 2006.

DELICATO, F. C. et al. A flexible web service based architecture for wireless sensor networks. In: IEEE. **23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings.** [S.l.], 2003. p. 730–735.

DELICATO, F. C. et al. A service approach for architecting application independent wireless sensor networks. **Cluster Computing**, Springer, v. 8, n. 2-3, p. 211–221, 2005.

DENHOLM, P.; KULCINSKI, G. L.; HOLLOWAY, T. Emissions and energy efficiency assessment of baseload wind energy systems. **Environmental science & technology**, ACS Publications, v. 39, n. 6, p. 1903–1911, 2005.

DZIENGEL, N. et al. Deployment and evaluation of a fully applicable distributed event detection system in wireless sensor networks. **Ad Hoc Networks**, Elsevier, v. 37, p. 160–182, 2016.

FACELI, K.; CARVALHO, A. C. D.; REZENDE, S. O. Combining intelligent techniques for sensor fusion. **Applied Intelligence**, Springer, v. 20, n. 3, p. 199–213, 2004.

FARIAS, C. M. A framework for developing smart space applications using shared sensor networks. In: **D. Sc. Dissertation, Federal University of Rio de Janeiro, Brazil**. [S.l.: s.n.], 2014c.

GOOS, P.; MEINTRUP, D. **Statistics with JMP: graphs, descriptive statistics and probability**. [S.l.]: John Wiley & Sons, 2015.

GUNGOR, V. C.; LU, B.; HANCKE, G. P. Opportunities and challenges of wireless sensor networks in smart grid. **IEEE transactions on industrial electronics**, IEEE, v. 57, n. 10, p. 3557–3564, 2010.

ISLAM, M. et al. A survey on virtualization of wireless sensor networks. **Sensors**, Molecular Diversity Preservation International, v. 12, n. 2, p. 2175–2207, 2012.

KHAN, I. et al. Wireless sensor network virtualization: A survey. **IEEE Communications Surveys & Tutorials**, IEEE, v. 18, n. 1, p. 553–576, 2015.

MUSÍLEK, P.; KRÖMER, P.; BARTON, T. E-bach: Entropy-based clustering hierarchy for wireless sensor networks. In: IEEE. **2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)**. [S.l.], 2015. v. 3, p. 231–232.

NAKAMURA, E. F.; LOUREIRO, A. A.; FRERY, A. C. Information fusion for wireless sensor networks: Methods, models, and classifications. **ACM Computing Surveys (CSUR)**, ACM, v. 39, n. 3, p. 9, 2007.

PEARSON, K. X. contributions to the mathematical theory of evolution.—ii. skew variation in homogeneous material. **Philosophical Transactions of the Royal Society of London.(A.)**, The Royal Society London, n. 186, p. 343–414, 1895.

POTTIE, G. J.; KAISER, W. J. Wireless integrated network sensors. **Communications of the ACM**, ACM, v. 43, n. 5, p. 51–58, 2000.

RAWAT, P. et al. Wireless sensor networks: a survey on recent developments and potential synergies. **The Journal of supercomputing**, Springer, v. 68, n. 1, p. 1–48, 2014.

SAFIA, A. A.; AGHBARI, Z. A.; KAMEL, I. Phenomena detection in mobile wireless sensor networks. **Journal of Network and Systems Management**, Springer, v. 24, n. 1, p. 92–115, 2016.

SANTOS, I. L. et al. Olympus: The cloud of sensors. **IEEE Cloud Computing**, IEEE, v. 2, n. 2, p. 48–56, 2015.

SHANNON, C. E. A mathematical theory of communication. **Bell system technical journal**, Wiley Online Library, v. 27, n. 3, p. 379–423, 1948.

WHITMORE, A.; AGARWAL, A.; XU, L. D. The internet of things—a survey of topics and trends. **Information Systems Frontiers**, Springer, v. 17, n. 2, p. 261–274, 2015.