Uma proposta de algoritmo de escalonamento de aplicações móveis sensíveis ao contexto para o paradigma *fog computing*

Celestino Barros[1], Vítor Rocio[2], André Sousa[3], Hugo Paredes[4]

[1]Faculdade de Ciência e Tecnologia da Universidade de Cabo Verde, Praia, Cabo Verde, celestino.barros@docente.unicv.edu.cv
[2]INESC TEC e Universidade Aberta de Portugal, Lisboa, Portugal, vitor.rocio@uab.pt
[3]Critical TechWorks, Porto, Portugal, asousa@roundstone.pt
[4]INESCT TEC e Universidade de Trás-os-Montes e Alto Douro, Vila Real, Portugal, hparedes@utad.pt

Resumo

Escalonamento na arquitetura *cloud* e no paradigma *fog* continuam a apresentar alguns desafios aliciantes. Na *cloud,* segundo o conhecimento dos autores, ela é amplamente estudada e em muitas pesquisas é abordada na perspetiva de provedores de serviço. Na *fog*, é muito complexo e, existem poucos estudos. Procurando trazer contributos inovadores nas áreas de escalonamento de tarefas, neste artigo, propomos uma solução para o problema de escalonamento de aplicações móveis sensíveis ao contexto para o paradigma *fog computing* onde diferentes parâmetros de contexto são normalizados através da normalização *Min-Max*, as prioridades são definidas através da aplicação da técnica da Regressão Linear Múltipla (RLM) e o escalonamento é feito recorrendo a técnica de Otimização de Programação Não Linear Multi-objetivo (MONLP).

Palavras-chave: qualidade de experiência, *cloud* e *fog computing*, escalonamento na arquitetura *cloud* e no paradigma *fog*, escalonamento sensível ao contexto.

Title: A proposal of context-aware scheduling of mobile applications for the fog computing paradigm

Abstract: Scheduling in cloud architecture and in the fog paradigm continue to present some exciting challenges. In the cloud, according to the authors' knowledge, it is widely studied and in many researches, it is addressed from the perspective of service providers. In fog, it is very complex and there are few studies. Trying to bring innovative contributions in the areas of task scheduling, in this paper we propose a solution to the problem of context-aware scheduling of mobile applications for the fog computing paradigm, where different context parameters are normalized through Min-Max normalization, priorities are defined by applying the Multiple Linear Regression (MLR) technique and scheduling is performed using Multi-Objective Nonlinear Programming Optimization (MONLP) technique.

Keywords: quality of experience, cloud and fog computing, scheduling in cloud architecture and fog paradigm, context-aware scheduling.

1. Introdução

O aumento dos dispositivos móveis e a evolução da *Internet of Things* (IoT), tem instigado o incremento dos dispositivos conectados à Internet e os números tendem a aumentar de forma expressiva. Por outro lado, vários desses dispositivos, executam aplicações que requerem que parte do processamento sejam executados em grandes *datas centers*, conhecidos como *cloud*. No entanto, a *cloud* está fisicamente distante dos dispositivos dos utilizadores finais, esta divisão provoca altas latências na comunicação e prejudicam aplicações que requerem respostas em tempo real. Diferentes técnicas que minimizam o processamento na *cloud* através do processamento local e que permitem resolver as limitações da *cloud* têm sido propostas. Uma dessas técnicas passa pela utilização da arquitetura *fog* [Musumba, Nyongesa 2013].

Segundo [Gupta *et al.* 2016], *fog computing* é uma extensão não trivial da *cloud*, que disponibiliza serviços de processamento, armazenamento e rede próximo da borda da rede. Avançam ainda que devido à densidade e à heterogeneidade de dispositivos nessa arquitetura, o escalonamento é muito complexo e, na literatura, existem poucos estudos.

Procurando trazer contributos inovadores nas áreas de escalonamento de tarefas e computação distribuída, neste artigo, fizemos uma revisão da literatura sobre os principais algoritmos de escalonamento na arquitetura *cloud* e *fog*, sugerimos algumas perspetivas de melhorias e com base nestas sugestões, propomos um algoritmo de escalonamento sensível ao contexto para o paradigma *fog.* Está dividida em seis secções sendo a primeira, a introdução, a segunda, a contextualização, a terceira, o levantamento do estado da arte sobre os algoritmos de escalonamento na arquitetura *cloud* e no paradigma *fog*. Na quarta, são descritos os contextos previstos, o modelo e as arquiteturas propostas. Na quinta, é apresentado um exemplo ilustrativo que ajuda a entender as funcionalidades do modelo proposto e na sexta secção, é feita a conclusão.

Como resultado, propomos um algoritmo de escalonamento de tarefas sensível ao contexto para a arquitetura *fog* onde a normalização *Min-Max* é utilizado para normalizar os diferentes parâmetros de contexto, as prioridades são definidas através da aplicação da técnica da RLM e o escalonamento otimizado é feito através da técnica de MONLP.

2. Contexto

A computação móvel disponibiliza aos utilizadores vários utilitários, permitem a portabilidade e suportam aplicações de variados interesses, possuem também várias limitações como: escassez de recursos; autonomia reduzida da bateria; entre outros. Nos últimos anos têm sido propostas várias arquiteturas que visam resolver essas limitações, sendo a *cloud computing* uma delas. Não obstante as vantagens, em algumas situações, não é benéfica a utilização da *cloud*. Ela é centralizada e consequentemente o processamento é feito em *data centers* concentrados, para otimização de custos energéticos e de comunicações. Visando resolver essas inconveniências, vários paradigmas têm sido propostos. Entre eles se destaca a *fog computing,* que visa disponibilizar os serviços oferecidos pela *cloud* na extremidade da rede [Stojmenovic 2014].

Antes da revisão bibliográfica sobre os algoritmos de escalonamento, nesta secção, fizemos a contextualização onde abordamos tópicos como a conceção de algoritmos de escalonamento e a sensibilidade ao contexto.

2.1. Conceção de algoritmos de escalonamento

Segundo [Swaroop 2019], o escalonamento é a atribuição de recursos necessários para a execução de uma tarefa. Assume como um processo fundamental para melhorar a confiabilidade e a flexibilidade dos sistemas. Exige algoritmos avançados capazes de escolher o recurso mais adequado para executar a tarefa. Avança ainda, que na sua conceção, devemos ter em conta algumas restrições como: custo das tarefas, dependências entre tarefas e a localização. As decisões de escalonamento podem ser estáticas - onde as decisões sobre o escalonamento são tomadas durante a compilação. Ou dinâmicas - onde são utilizadas informações sobre o estado do fluxo das tarefas num determinado instante durante a execução para a tomada de decisões de escalonamento [Swaroop 2019].

A decisão de escalonamento dinâmica constitui a melhor abordagem, porque possibilita que vários problemas tenham solução que podem ser representados por uma árvore de pesquisa. No entanto, esses problemas são exigentes computacionalmente, requerem estratégia de paralelização e balanceamento de carga dinâmico.

2.2. Sensibilidade ao contexto

[Bazire & Brézillon 2005], examinaram 150 definições de contexto, nas diferentes áreas de investigação e concluíram que a criação de uma única definição é um esforço árduo e provavelmente, impossível visto que a mesma varia com a área científica e depende principalmente do campo em que ela está a ser aplicada. No entanto, avançam que contexto é um conjunto de restrições que influenciam o comportamento relativo a uma determinada tarefa. Para [Musumba & Nyongesa 2013], em computação móvel, contexto refere-se ao ambiente de processamento, ambiente do utilizador, ambiente físico, relevante para a interação entre um utilizador e uma aplicação, incluindo o utilizador e as próprias aplicações.

As expetativas do utilizador sobre um sistema e a antecipação da reação do sistema com o qual o utilizador está a interagir é altamente dependente da situação e do ambiente. No que diz respeito em como fazer a utilização do contexto, a solução passa pela utilização de aplicações e sistemas sensíveis ao contexto. Isto é, que utiliza o contexto para disponibilizar informações e serviços relevantes ao utilizador, onde a relevância depende da tarefa do utilizador [Dey *et al.* 1999]. Por exemplo, quando um sistema sensível ao contexto detetar que um utilizador nunca atende chamadas telefónicas enquanto está a conduzir, ele propõe automaticamente a transferência de chamadas recebidas para o correio de voz do utilizador.

3. Revisão sobre o escalonamento de tarefas na arquitetura *cloud* e *fog*

Nesta secção, descrevemos a metodologia utilizada para a revisão bibliográfica, analisámos os algoritmos selecionados e propomos algumas sugestões de melhorias aos algoritmos analisados.

3.1. Metodologia utilizada para a revisão de literatura

Com o objetivo de alcançar maior de rigor científico no levantamento feito, assegurámos o processo de investigação com base nas perspetivas da revisão sistemática da literatura, que segundo [Kitchenham 2004], identifica, avalia e interpreta todas as pesquisas disponíveis relevantes para uma determinada área temática, uma questão específica, ou um fenómeno de interesse. Avança ainda, que a sua concretização deve considerar as seguintes fases: preparação da revisão; definição de metodologias para a execução e extração da revisão sistemática e procedimentos para a criação de relatórios da revisão efetuada.

A revisão feita nesta secção teve em conta a aplicação dos seguintes passos metodológicos:

1. Foi criada uma base de dados de propostas científicas utilizando os seguintes parâmetros: artigos completos disponíveis *online*, e publicados no período de 2015 a 2020 que tratam tópicos relevantes como: *task scheduling or scheduling algorithm and scheduling cloud computing or scheduling fog computing and cloud-fog computing.* Utilizaram-se bases de dados *ScienceDirect*, 371 artigos, *IEEE Explore*, 108 artigos, *Google Scholar*, 278 artigos, ACM, 302 artigos e SCOPUS, 200 artigos, totalizando 1259 artigos. Os resultados foram importados para o software *Publish or Perish* [Harzing 2020].

2. Após a limpeza dos duplicados de 143 artigos, compilou-se uma lista de 1116 artigos, dos quais foram removidos 68 por não possuirmos o acesso ao documento.

3. Com base na lista de 1048 artigos, procedemos à exclusão de propostas não essencialmente relevantes por "não serem propostas de escalonamento de tarefas, ou algoritmos de escalonamentos de tarefas" e "não serem relacionadas com *cloud* ou *fog computing*". Esta exclusão foi realizada por análise de títulos e resumos dos artigos e, em caso de dúvida, por análise dos artigos. Foram removidos os artigos de tipo: *Survey* (53); *position papers* (78); Não serem uma proposta de escalonamento de tarefas, ou algoritmos de escalonamentos de tarefas (184); Não relacionadas com *cloud* ou *fog computing* (450). Após esta análise, obteve-se uma lista de 283 artigos.

4. Seleção, por inspeção manual, de um conjunto reduzido de propostas, a partir de 283 artigos, com base nos seguintes critérios:

    - Ordenação por número de citações na literatura ajustada ao ano (calculada pelo *software Publish or Perish*), permitiu obter um conjunto de propostas que foi analisado e validado por outros autores e remover o enviesamento provocado pelo tempo de publicação;

    - A relevância para a temática, possibilitou obter um conjunto de propostas que corresponde às necessidades inseridas dentro da temática dos algoritmos de escalonamento de tarefas na arquitetura *cloud* ou paradigma *fog*. Foram excluídos 195 artigos por não serem relevantes e seis por não possuirmos acesso.

Com base nesta metodologia de seleção, obteve-se uma amostra de 30 propostas, que foram categorizados, considerando suas características, em: básicos; baseados em

prioridade e orientado a qualidade de serviço (QoS) e sensíveis ao contexto, conforme apresentada e analisada na secção 3.2.

3.2. Análise dos algoritmos selecionados

Alguns autores (cf. [Sheikhalishahi *et al.* 2016], [Lawanyashri, Balusamy & Subha 2017], [Tiwary *et al.* 2018] e [Gawali & Shinde 2018]), analisam o escalonamento sob a perspetiva dos provedores de serviços e ignoram questões contextuais dos utilizadores finais.

[Wang, Wang & Cui 2016], apresentaram o *Energy-Aware Multi-Job Scheduling Model* que se baseia no *Map Reduce* e tem por objetivo melhorar a eficiência energética através do ajuste do CPU do servidor.

[Qiu *et al.* 2015], propuseram o *Energy-Aware Dynamic Task Scheduling* que visa minimizar o consumo de energia dos sistemas ciberfísicos que, devido as suas limitações, têm a necessidade de fazer muitas e frequentes interações com a *cloud* a fim de executarem tarefas como: solicitar recursos, enviar requisições e receber respostas.

Os autores em [Deng *et al.* 2016], [Li *et al.* 2017], [Lawanyashri, Balusamy & Subha 2017] e [Yang *et al.* 2018] desenvolveram abordagens que enfatizam a redução do consumo de energia dos *data centers* e nota-se uma grande preocupação com a eficiência energética.

Em [Shi *et al.* 2016), é apresentado um escalonador probabilístico adaptativo, que otimiza o consumo de energia de tarefas com restrição de tempo.

[Sahoo & Dehury 2018] conceberam um algoritmo que permite escalonar tarefas relacionadas com a saúde, armazenadas e geridas em *data center*.

[Bitam, Zeadally & Mellouk 2018], apresentaram o *Bees Life Algorithm* (BLA) que visa encontrar o equilíbrio entre o tempo de execução de tarefas no CPU e a sua alocação na memória. O algoritmo proposto alcançou o objetivo de minimizar o tempo de execução comparado com os algoritmos *Particle Swarm Optimization* e *Genetic Algorithm* proposto em [Woo, Jung & Kim 2017].

Em [Zhou *et al.* 2015], é proposto um algoritmo que se baseia na QoS para a *Mobile Cloud Server*, onde os atributos de tarefas como: privilégios dos utilizadores; tamanho da tarefa; expetativa e o tempo suspenso na fila são utilizados para calcular a prioridade.

[Sarkar, Chatterjee & Misra 2018], avaliaram a aplicabilidade da arquitetura *fog* com o propósito de responder as demandas das aplicações sensíveis à latência no contexto da IoT e concluíram que o objetivo da *fog*, que visa proporcionar respostas às requisições em tempo real e com baixa latência, favorece a sua utilização no contexto da IoT.

[Cardellini *et al.* 2015], propuseram um escalonador distribuído sensível a QoS para o processamento de fluxo de dados na arquitetura *fog*. No entanto, em topologias que envolvem muitos operadores funciona de forma instável.

Em [Stavrinides, Karatza 2019], é proposto um algoritmo leva em consideração o custo da comunicação decorrente da transferência de dados dos sensores e dispositivos da camada da *fog* durante o processo de escalonamento e em [Aazam *et al.* 2016], é proposto um algoritmo que visa otimizar a utilização dos recursos e QoS.

O *Offline Multi-Level Scheduling* foi concebido por [Seddik & Hanzálek 2017] e visa resolver o problema do nível de criticidade de cada tarefa com base na sua prioridade, leva em consideração o tempo total de processamento das tarefas tendo em vista o objetivo final.

[Skarlat *et al.* 2017], definiram uma política de escalonamento que visa escalonar aplicações sensíveis à QoS em nós da *fog*. Em [Intharawijitr, Iida & Koga 2016], tentam garantir a melhor utilização da largura de banda disponibilizada.

Em [Sheikhalishahi *et al.* 2016], [Deng *et al.* 2016], [Lawanyashri, Balusamy, & Subha 2017], há uma evidente preocupação dos autores com a eficiência energética dos recursos. Os algoritmos propostos em [Shojafar *et al.* 2015] e [Fun *et al.* 2017], pretendem cumprir o *deadline* das tarefas e aumentar os lucros do provedor de serviços da *fog*. Nestas cinco últimas propostas, há uma evidente preocupação dos autores em focar e defender principalmente os interesses dos provedores de serviços, ao invés de considerarem os contextos dos utilizadores finais e as suas experiências de utilização.

O *Hybrid Fog and Cloud-Aware Heuristic* (Hybrid-EDF) foi proposto e descrito em [Stavrinides & Karatza 2019]. Esta abordagem utiliza espaços no escalonamento das máquinas virtuais da *cloud* e *fog* para escalonar tarefas exigentes com baixos requisitos de comunicação na *cloud* e tarefas intensivas de comunicação com baixos requisitos computacionais na *fog*. Durante o escalonamento, essa abordagem considera o custo da comunicação decorrente da transferência de dados dos sensores e dispositivos da camada da *fog*. A performance da heurística proposta foi avaliada e comparada com o algoritmo *Baseline Cloud-Unaware Strategy*, *Fog*-EDF.

[Li *et al.* 2017], desenvolveram o *Cooperative-Based Model for Smart–Sensing Tasks* (CMST) um algoritmo de escalonamento cooperativo focado em melhorar as recompensas associado as recolhas de dados nas regiões suburbanas.

[Ghouma & Jaseemuddin 2015], propuseram o *Context Aware Cloud System*, política de escalonamento e afetação de recursos sensível ao contexto que monitoriza e utiliza as informações de contexto do *smartphone* do utilizador, para tomar decisões inteligentes sobre a afetação de recursos na *cloud* e escalonar tarefas. Enquanto que o proposto por [Zhou *et al.* 2017], exploram os níveis da conectividade da rede e os recursos de Maquinas Virtuais (MV) alugados para disponibilizar decisões de descarga de códigos.

[Mahmud *et al.* 2016], propõem uma política de escalonamento de aplicação sensíveis ao contexto para *Mobile Cloud Computing* que é executado em uma *Cloudlet*.

O algoritmo proposto em [Zhu *et al.* 2015], é utilizado em tarefas agrupadas com objetivo de minimizar o tempo de execução. E o proposto em [Oueis, Strinati & Barbarossa 2015], apesar de proporcionar ganho de latência e baixo consumo de energia, tal como os

propostos em [Cardellini *et al.* 2015], e [Zhu *et al.* 2015], o desempenho degrada-se quando é utilizado em arquitetura *fog* de grande escala.

Em [Gill, Garraghan & Buyya 2019], é proposta uma técnica de gestão de recursos para ambientes *cloud* e *fog* sensíveis ao QoS que, aproveita da otimização *Particle Swarm* para otimizar em simultâneo vários parâmetros do contexto.

Em [Aazam *et al.* 2016], é proposto um algoritmo que objetiva estimar os recursos da *fog* com intuito de otimizar a qualidade de experiência (QoE). Em vez disso, [Skarlat *et al.* 2017], propuseram uma abordagem que visa escalonar aplicações sensíveis à QoS em recursos virtualizados da *fog*. Contrariamente, em [Zhu *et al.* 2015], [Oueis, Strinati & Barbarossa 2015], [Mahmud *et al.* 2016] e [Aazam *et al.* 2016], foram tidas em conta as questões relacionadas com o aprimoramento da QoE do utilizador final.

[Fan *et al.* 2017], apresentaram o *Deadline-Aware Task Scheduling Mechanism,* através dele, os provedores de serviços exploram a colaboração entre os nós da *fog* e os recursos da *cloud* alugados para executar com eficiência as tarefas dos utilizadores em grande escala geográfica. E como principal objetivo visa maximizar os lucros do provedor de serviços *fog* através da satisfação do tempo definido para a execução das tarefas.

Muitas propostas, como as descritas em [Sheikhalishahi *et al.* 2016], [Lawanyashri, Balusamy & Subha 2017], [Tiwary *et al.* 2018] e [Gawali & Shinde 2018], [Fan *et al.* 2017], abordam o problema da otimização sob a perspetiva dos provedores de serviço e ignoram questões contextuais dos utilizadores finais e as suas experiências de utilização. Outras, como as definidas em: [Cardellini *et al.* 2015], [Aazam *et al.* 2016], [Skarlat *et al.* 2017] e [Gill, Garraghan & Buyya 2019], pretendem sobretudo otimizar os níveis de QoS da aplicação e alguns concentram apenas no escalonamento de tarefas na arquitetura *cloud* e no paradigma *fog*. Outros, ainda, preocupam-se com a eficiência energética.

Um resumo dos algoritmos de escalonamento estudados, elucidando a sua categoria e as suas principais vantagens e inconveniências é descrita na Tabela 1.

Tabela 1. Síntese dos algoritmos de escalonamento analisados

Autores	Algoritmos	Vantagens	Inconveniências	Categoria
[Gawali & Shinde, 2018]	*BATS+ BAR*	Facilita o aluguer de recurso na *cloud*.	Escalonamento e afetação de recursos são muito complexos	Básico
[Wang, Wang, & Cui, 2016]	*Energy-Aware Multi Job Scheduling*	Melhora a eficiência energética do servidor na afetação de dados	Possui limitações no escalonamento e otimização de tarefas;	Básico
[Bitam Zeadally, & Mellouk, 2018]	*Bees Life Algorithm* (BLA)	Maximiza o tempo de execução das tarefas;	Existem poucos estudos relativamente a sua utilização em ambientes densamente distribuídas	Básico
[Aazam *et al.,* 2016]	*MEdia Fog Resource Estimation (MeFoRE)*	Estima os recursos da *fog* com base no QoE do utilizador; permite otimizar QoS e a utilização e gestão de recursos é descentralizada	Não respeita a expectativas do utilizador no acesso ao serviço nem no tempo de processamento; Velocidade de processamento não satisfaz	Baseado em prioridade e orientado a QoS

(continuação)

Autores	Algoritmos	Vantagens	Inconveniências	Categoria
[Woo, Jung, & Kim, 2017]	*Genetic Algorithm*	Permite reduzir o tempo de execução de tarefas	Possui pouco improviso o que complica o seu uso em ambientes *fog*	Básico
[Sarkar, Chantterjee, & Misra, 2018]	*Fog Computing in the Context of Internet of Things*	Permite melhorar o desempenho de aplicações sensíveis à latência	Possibilita grande tráfego de dados	Básico
[Deng *et al.,* 2016]	*Fog –Cloud Architecture*	Permite melhorar a mobilidade, a distribuição geográfica e localização	A latência nas respostas podem levar à privação de serviços	Básico
[Lawanyashri, Balusamy, & Subha, 2017]	*Fruity Fly Algorithm*	Permite melhorar a disponibilidade, escalabilidade, redução do consumo de energia custos	A afetação de recursos não é feita de forma equilibrada	Básico
[Intharawijitr, Iida, & Koga, 2016]	*Mathematical model of a Fog network*	Permite o escalonamento eficiente e ótimo das tarefas;	Utiliza as politicas aleatória; menor latência e capacidade máxima disponível o que torna o modelo muito complexo.	Baseado em prioridade e orientado a QoS
[Tiwary *et al.,* 2018]	*Non-Cooperative Game Model*	Melhora o tempo de resposta das requisições	Executa tarefas com tamanhos limitados	Básico
[Sheikhalishahi *et al.,* 2016]	*Multi-Capacity Aware Resource Scheduling*	Permite melhorar a utilização de recursos e prima pela eficiência energética	Utiliza menos CPU, o que faz com que o processamento das tarefas seja muito complexo	Sensível ao contexto
[Seddik & Hanzálek, 2017]	*Offline Multi-Level Scheduling*	Permite melhorar a utilização de recursos	A definição do tempo de processamento de uma tarefa é complexa	Baseado em prioridade e orientado a QoS
[Sahoo & Dehury, 2018]	*CPU Intensive Scheduling Data-Intensive Scheduling*	Permite melhorar a infraestrutura dos servidores e da rede	Pode ocasionar indisponibilidade de recurso e ter impacto negativo ao nível do QoS	Básico
[Oueis, Strinati, & Barbarossa, 2015]	*Reduced Complexity Task Scheduling Algorithm for Fog*	Proporciona satisfação dos utilizadores em termos de baixa e consumo de energia	Algoritmos utilizados alcançam bons resultados apenas em infraestruturas de computação de baixa densidade	Baseado em prioridade e orientado a QoS
[Skarlat *et al.,* 2017]	*QoS-Aware Application Placement on Virtualized Fog Resources*	Observa as expectativas em termos de equilíbrio de recursos e tempo de processamento. Satisfaz o *status* de instâncias quanto à disponibilidade de recursos e velocidade de processamento;	Não respeita as expectativas em termos de acesso ao serviço; Não atende ao *status* de instância quanto à proximidade ou taxa de resposta	Baseado em prioridade e orientado a QoS

(continuação)

Autores	Algoritmos	Vantagens	Inconveniências	Categoria
[Zhou *et al.,* 2015]	*QoE-Driven Video Cache Allocation Scheme for Mobile Cloud Server*	Observa as espectativas em termos do acesso ao serviço, satisfaz o *status* de instâncias quanto à proximidade ou taxa de resposta e disponibilidade de recursos;	Não respeita as expetativas em termos de equilíbrio de recursos e tempo de processamento; Não atende ao *status* de instância quanto a velocidade de processamento	Sensível ao contexto
[Cardellini *et al.,* 2015]	*Quality of Service Distributed Scheduler with QoS Awareness*	Em ambientes pequenos, melhora o desempenho da aplicação e aprimora o sistema com recursos de adaptação em tempo de execução	Em topologias *fog* complexas, com muitos operadores causam instabilidade o que diminui os níveis da disponibilidade da aplicação	Baseado em prioridade e orientado a QoS
[Mahmud *et al.,* 2016]	*QoE and Context-Aware Scheduling (QCASH)*	Observa as espectativas em termos de acesso ao serviço e tempo de processamento;	A gestão é centralizada, o que dificulta a sua implementação em ambientes densamente distribuídos como a *fog*	Sensível ao contexto
[Shi *et al.,* 2016]	*Adaptive Probabilistic Scheduler*	Consegue reduzir o consumo médio de energia e mantêm uma elevada taxa de execução da tarefa.	Pode causar desequilíbrios na utilização de recursos	Sensível ao contexto
[Zhu *et al.,* 2015]	*Priority-Based Two-Phase Min–Min Scheduling Policy (PTMM)*	Pode ser aplicado em tarefas agrupadas por forma a minimizar o tempo de execução esperado;	Possui um mau desempenho caso as requisições forem complexas o que pode resultar numa fraca QoE dos utilizadores	Baseado em prioridade e orientado a QoS
[Qiu *et al.,* 2015]	*Energy-Aware Dynamic Task Scheduling*	Reduz o consumo de energia dos sistema ciberfísico, em comparação com o método de escalonamento do caminho crítico e o do escalonamento baseado no paralelismo	Abordagem ainda pouco estudada. Foi testada apenas nos dispositivos moveis Android. Há a necessidade de se fazer mais testes e combinar mais técnicas *Mobile Edge Computing*.	Básico
[Li *et al.,* 2017]	*Cooperative-Based Model for Smart-Sensing Tasks* (CMST)	Experiências extensivas mostram que o CMST melhora a cobertura e qualidade de dados e permite reduzir os custos.	Possui um bom desempenho, apenas em casos específicos como recolhas de dados com objetivo de alimentar plataformas.	Básico

(continuação)

Autores	Algoritmos	Vantagens	Inconveniências	Categoria
[Shojafar *et al.,* 2015]	*FUzzy GEnetic Task Scheduling* (FUGE)	Resultados das experimentações mostram a eficiência dessa abordagem em termos de tempo, custo de execução e grau médio de desequilíbrio.	Não considera o consumo de energia e a migração de maquinas virtuais; Não é eficiente em termos energéticos; No escalonamento considera apenas o tempo de execução de uma tarefa.	Básico
[Gill, Garraghan, & Buyya, 2019]	ROUTER	Permitiu a redução na utilização da largura de banda da rede, no tempo de resposta e latência e no consumo de energia.	Foi testada em *Smart Home* de pequena escala; Não atende aos parâmetros como: escalabilidade, confiabilidade, custo e disponibilidade.	Baseado em prioridade e orientado a QoS
[Yang *et al.,* 2018]	*Delay Energy Balanced Task Scheduling* (DEBTS)	Minimiza o consumo de energia e reduz o atraso médio do serviço e o atraso na entrega de dados na rede; Possibilita melhor desempenho que os algoritmos tradicionais *Random e least busy scheduling.*	Apenas foi testado em arquitetura *fog* simples, há a necessidade de ser testado em arquiteturas *fog* heterogêneas e complexas.	Básico
[Fan *et al.,* 2017]	*Deadline-Aware Task Scheduling Mechanism*	Melhora o desempenho do sistema quando comparado com os algoritmos *Min-Min* e *First Come First Served.*	Em ambientes como a *fog*, o desempenho degrada e há a necessidade de se considerar algoritmos distribuídos e leves para a otimização.	Básico
[Stavrinides & Karatza, 2019]	*Hybrid Fog and Cloud-Aware Heuristic (Hybrid-EDF)*	Proporciona diminuição em relação a falhas na entrega porque utiliza recursos da *cloud*. Sobretudo quando esta for composta por maquinas virtuais *on-demand multi-tenant*.	Quando a camada da *cloud* for constituída por maquinas virtuais em *hosts* reservados e dedicados tem um alto custo.	Básico
[Zhou *et al.*, 2017]	*mcloud: Context-Aware Offloading Framework*	Disponibiliza decisões de descarregamento com base no contexto atual dos dispositivos o que permite diminuir o custo do tempo de execução e o consumo de energia.	Por forma a ser mais eficiente e confiável, há necessidade de ampliar a estrutura por forma a que os recursos da *cloud* tenham a capacidade de intercomunicarem com base nas mudanças de contexto	Sensível ao contexto

(conclusão)

Autores	Algoritmos	Vantagens	Inconveniências	Categoria
[Ghouma & Jaseemuddin, 2015]	*Context Aware Cloud System*	Reduz o custo de execução das tarefas e, ao mesmo tempo, satisfaz a duração da execução definida.	A elevada latência associada a *cloud* impossibilita a sua utilização em aplicações que requerem baixas latência.	Sensível ao contexto

3.2.1. Perspetivas de melhoria dos algoritmos analisados

Apesar de existirem vários escalonadores para a arquitetura *cloud* e paradigma *fog*, que permitem resolver muitos problemas, alguns aspetos ainda podem ser explorados a fim de se alcançar melhorias em relação às estratégias existentes.

Seguidamente, sugerimos algumas melhorias aos escalonadores analisados:

- *Consciencialização do contexto no escalonamento de tarefas:* várias pesquisas asseguram que os escalonadores sensíveis ao contexto são eficientes no aperfeiçoamento da QoS e otimizam os custos, tanto do ponto de vista dos utilizadores como dos provedores de serviço.

- *Priorização de tarefas sensíveis ao contexto:* devem ser introduzidos modelos de escalonamento onde as prioridades são definidas com base no contexto das requisições.

- *Preservação da restrição de energia:* a restrição energética deve ser levada em consideração durante o escalonamento de tarefas. Os valores dos níveis de bateria de uma requisição devem ser superiores ou iguais ao nível da bateria limite definido.

- *Conservação da força do sinal da rede:* a intensidade do sinal associada a uma requisição deve assegurar a força mínima do sinal de forma a possibilitar ao utilizador solicitar recursos e obter respostas em tempo hábil.

- *Salvaguarda da QoS:* atendendo que as tarefas descarregadas na *fog* são heterogéneas, ao escalonar tarefas devemos considerar que o tempo necessário para a obtenção da resposta deve estar confinado dentro dos limites temporais previstos.

- *Redução do tempo médio de espera:* um mecanismo de compensação relativamente ao tempo médio de espera deve ser introduzido pelo escalonador de tarefas de forma a que o aumento do tempo de espera seja proporcionalmente menor em relação à chegada das tarefas.

- *Otimização da QoE:* os algoritmos de escalonamentos devem também concentrar em otimizar tanto a QoE dos utilizadores finais como a QoS.

Propomos na secção 4, um algoritmo de escalonamento, que incorporam estas sugestões.

4. Modelo e arquitetura proposta

Nas subsecções seguintes, descrevemos os contextos previstos, ilustramos e discutimos o modelo e a arquitetura da solução proposta.

4.1. Contextos previstos e assunções

Assumimos que uma técnica de descarregamento do código adequado (por exemplo, MAUI definido em [Bahl *et al.* 2010], COMET apresentado em [Gordon *et al.* 2012], entre outros) esteja a ser executada nos dispositivos móveis a fim de tomar a melhor decisão em relação ao descarregamento ou não de códigos e em quais nós da *fog* [Berg, Durr & Rothermel, 2014].

Consideramos que uma requisição descarregada inclui o atraso máximo permitido para executar a aplicação, o nível de bateria e os valores da força do sinal da rede do dispositivo. Também assumimos, tal como em [Luan *et al*. 2016], que a *fog* disponibiliza capacidades computacionais maiores que os dispositivos móveis e consegue extrair os contextos associados às requisições e tomar a decisão de escalonamento em conformidade.

[Musumba e Nyongesa 2013], definiram os principais contextos que podem ser explorados em qualquer ambiente de computação móvel como sendo: a ligação a rede; processadores disponíveis; nível de bateria; localização; largura de banda da rede; tráfego na rede; MV alugadas e requisitos de QoS da aplicação.

No nosso domínio do problema, os contextos dos provedores de serviços foram ignorados. Também, depois de descarregada as tarefas na *fog*, torna-se desnecessário considerar os processadores do dispositivo móvel. A localização do dispositivo também não afeta o escalonamento, bem como, o tráfego na rede e a largura de banda que são iguais para todos os utilizadores. Com base nesses critérios levamos em consideração três parâmetros de contexto: nível da bateria; relação sinal-interferência-ruído da rede (SIN) e QoS da aplicação.

4.2. Modelo proposto

Os nós da *fog,* com a nossa proposta ativada, é constituída por três unidades: *Unidade de recuperação de informações de contexto,* compreende uma arquitetura, conforme definido em [La e Kim 2010]. Ela recupera as informações de contexto (C_i) de cada requisição ($r \in R$).

As informações de contexto recuperadas são encaminhadas para a *unidade de priorização de tarefas sensíveis ao contexto* que, estima o valor da prioridade de contexto (P_r) para cada requisição individual $r \in R$ e a encaminha para a *unidade de QoE e escalonamento sensível ao contexto* que*,* escalona as tarefas para serem executadas em *MVs* de forma que a QoE seja otimizada.

A Figura 1, ilustra as diferentes unidades do modelo proposto.

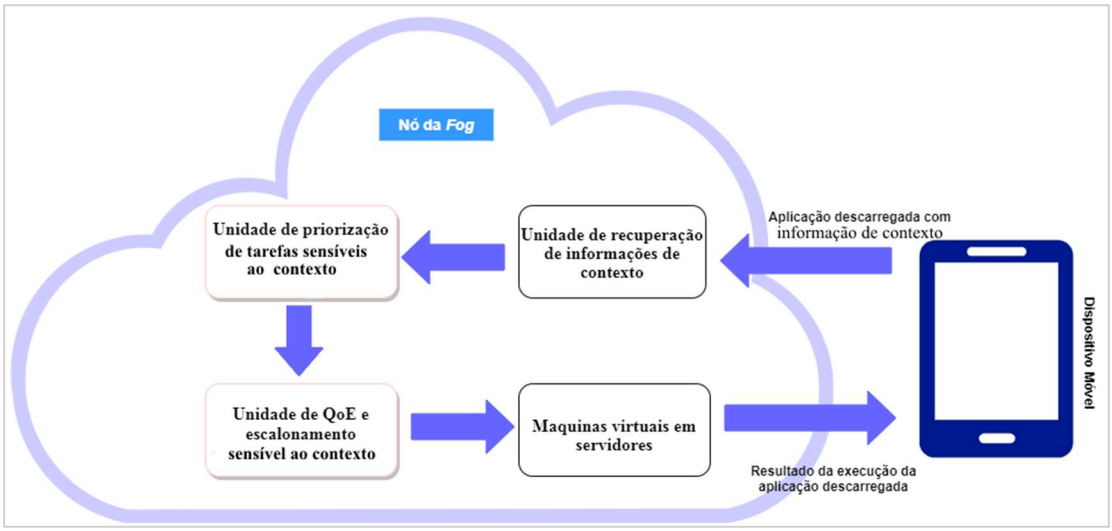

Figura 1. Trocas de informações entre as diferentes unidades do modelo proposto.

As notações relevantes utilizadas nas diferentes unidades estão listadas na Tabela 1.

Tabela 2. Notações e definições da arquitetura do modelo proposto.

Símbolo	Definição
C	Conjunto de todos os parâmetros de contexto
α_i	Quantidade de diferentes tipos de níveis abstratos de um parâmetro de contexto, $C_i \in C$
γ_i	Diferença lógica entre dois níveis abstratos $C_i \in C$
η_i	Intervalo normalizado de valores $C_i \in C$
L_i	Conjunto de todos os níveis concretos de $C_i \in C$ no instante t
θ_i	Conjunto de valores tendenciosos $C_i \in C$
F	Conjunto de todas as combinações possíveis de um elemento de cada L_i
M	Conjunto múltiplo de valores mínimos $\forall \mu \in F$
R	Conjunto de execuções de aplicações em um dado instante t
V	Conjunto de máquinas virtuais disponíveis em um dado instante t
β	Conjunto dos coeficientes de RLM
P_r	Prioridade de qualquer requisição $r \in R$
$\psi_{r,v}$	Tempo necessário para a execução de uma requisição $r \in R$ em uma máquina virtual $v \in V$
I_r	Quantidade de interrupções de uma requisição $r \in R$
r	Requisição
Q_r	Tempo de espera na fila da requisição r
T_r	Corresponde à QoS da aplicação
z	Índice dos níveis abstratos de um parâmetro de contexto, $C_i \in C$
$\theta_{i,j}$	Valor mínimo de enviesamento de C_i

Assumimos que algumas MVs já estejam criadas com diferentes configurações o que permite minimizar as sobrecargas relativas aos processos de criação e eliminação de MVs, conforme referido em [Hong *et al.* 2013] e [Skarlat *et al.* 2017]. A provisão ótima de MVs para requisições de aplicações e as suas afetações energeticamente eficientes estão fora do escopo deste artigo. Foram, no entanto, aprofundadas em [Li *et al.* 2017, 2018] e [Yang *et al.* 2018].

4.3. Arquitetura do modelo proposto

O fato dos parâmetros de contextos associados a uma requisição serem heterogéneas, torna difícil a exploração das informações do contexto no escalonamento. Para resolver este problema, em [Han 2011], foi proposto um resolvedor de heterogeneidade de contexto, que processa vários parâmetros, num intervalo normalizado, através da normalização *Min-Max,* onde cada requisição é priorizada com base nos seus valores de contexto.

Nas subsecções seguintes são definidas a arquitetura do modelo proposto, a começar pela unidade de priorização das tarefas sensíveis ao contexto.

4.3.1. Unidade de priorização de tarefas sensíveis ao contexto

Esta unidade é composta por: *Repositório de contexto,* que armazena informações de contexto das tarefas atuais e anteriormente recebidas e *Unidade de previsão de contexto*, explora a informação de contexto num determinado instante de tempo e alimenta a *tabela de previsão*. Assim, conseguimos eliminar a heterogeneidade das informações de contexto na alimentação da *tabela de previsão.*

A *tabela de previsão*, disponibiliza um conjunto de dados para a análise da RLM que visa definir a prioridade das requisições. A Figura 2, ilustra arquitetura da unidade de priorização de tarefas sensíveis ao contexto do modelo proposto.

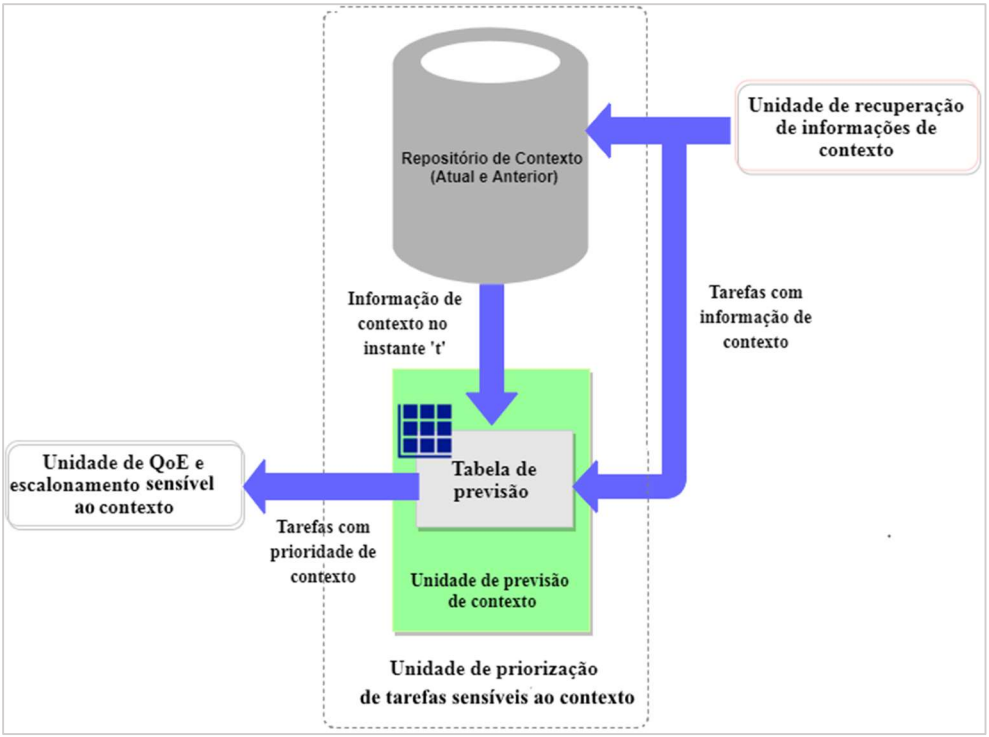

Figura 2. Arquitetura da unidade de priorização de tarefas do modelo proposto.

4.3.2. Criação da tabela de previsão de prioridade de contexto

Para a resolução de problemas de heterogeneidade de informações do contexto, com base na normalização *Min-Max*, foi concebido um modelo que alimenta a tabela de previsão. O repositório de contexto disponibiliza informações do contexto de tarefas previamente recebidas num determinado instante *t*. $\forall C_i \in C$, todas as informações do contexto são normalizadas em relação aos extremos, os valores ficam concentrados num intervalo entre

0 e 1. Esta abordagem permite minimizar a heterogeneidade dos diferentes parâmetros do contexto em termos de valores e unidades. Definimos o intervalo normalizado, η_i, para um parâmetro de contexto, $C_i \in C$, como:

$$\boldsymbol{\eta_i = \frac{\max(C_i) - \min(C_i)}{\max(C_i)}}. \quad \textbf{(1)}$$

Dentro deste intervalo normalizado $C_i \in C$, assumimos que existem α_i níveis abstratos. Estes níveis abstratos representam a medição qualitativa do parâmetro de contexto correspondente. Permitem que as variações de valores de um determinado parâmetro de contexto possam ser classificadas internamente. As diferenças lógicas entre dois níveis abstratos consecutivos, γ_i, de um contexto $C_i \in C$ são definidas como:

$$\boldsymbol{\gamma_i = \frac{\eta_i}{\alpha_i}}. \quad \textbf{(2)}$$

A representação numérica desta abstração compreende um conjunto em concreto de níveis, L_i, para qualquer $C_i \in C$. Esta abordagem transforma a medição qualitativa em quantitativa, compatível para cálculos posteriores. O L_i é definido como:

$$\boldsymbol{L_i = \left\{x: x = \frac{\min(C_i)}{\max(C_i)} + z_i\gamma_i\right\}, \forall C_i \in C}. \quad \textbf{(3)}$$

Onde $z = 0,1,2,\ldots,(\alpha_i - 1)$.

Calculando o conjunto de produtos cartesianos, são definidos o conjunto combinatório de todos os níveis de contexto a partir dos diferentes parâmetros de contexto. Este produto cartesiano é formulado conforme indicado na equação 4.

$$\mathbf{F} = \prod_{i=1}^{|C|} \mathbf{L_i}. \quad \textbf{(4)}$$

Todas as combinações possíveis dos diferentes parâmetros do contexto de uma requisição $(r \in R)$ são determinadas.

Também é criado um multiconjunto *M* com os valores mínimos de cada combinação de *F*, definido conforme a equação 5.

$$\mathbf{M = \{q : q = \min(\mu)\},\ \forall \mu \in F}. \quad \textbf{(5)}$$

Este identifica o estrangulamento de todas as possíveis combinações do contexto de uma requisição $r \in R$. Este parâmetro de estrangulamento influência a priorização das combinações simbólicas.

Além dos parâmetros de estrangulamento, os valores de enviesamento, associados aos diferentes níveis dos vários parâmetros de contexto, utilizados na priorização, também influenciam a priorização.

Este valor de enviesamento permite enfatizar os outros parâmetros. Basicamente, é um mapeamento *um-para-um* entre os elementos de L_i e o conjunto enviesado, θ_i para um determinado parâmetro de contexto, $C_i \in C$.

Seja, $\theta_{i,j}$ refere ao valor mínimo de enviesamento de C_i, no seu *j-ésimo* nível associado a uma combinação candidata em *F*. Para mapear, θ_i para L_i, $\theta_{i,0}$, presume-se, $\forall C_i \in C$. $\forall C_i \in C, \theta_{i,j}$ deve satisfazer a seguinte condição:

$$\boldsymbol{\theta_{i,j} * max(q \in M) < \theta_{i,j+1} * min(q \in M).} \quad \textbf{(6)}$$

As prioridades destas combinações são definidas de forma que a informação do contexto de qualquer requisição possa ser mapeada sobre ela mesma a fim de prever a prioridade dessa requisição.

O cálculo da prioridade é feito utilizando os valores relevantes de enviesamento $\forall C_i \in C$ e o seu parâmetro de contexto de estrangulamento. Assumimos que $\delta_i\ (\mu_k)$ define a prioridade do contexto $\mu_k \in F$ enviesado em $C_i \in C$. $\delta_i\ (\mu_k)$ é representado conforme a equação 7.

$$\boldsymbol{\delta_i\ (\mu_k) = \frac{\theta_{ij} * q_k}{\sum q}, \forall q \in M.} \quad \textbf{(7)}$$

Onde, $0 \leq k\ \leq (|F| - 1)$ e $q_k \in M$ está associada à μ_k.

A prioridade estimada, $\hat{\delta}_i(\mu_k)$ de μ_k é calculada através da equação 8.

$$\boldsymbol{\hat{\delta}_i(\mu_k) = \sum_{C_i \in C} \delta_i\ (\mu_k).} \quad \textbf{(8)}$$

μ_k e $\hat{\delta}(\mu_k)$, $\forall \mu_k \in F$ vão alimentar a tabela de previsão com um conjunto de dados. Estes dados da tabela de previsão permitem prever a prioridade de qualquer requisição em fila de espera.

4.3.3. Previsão da prioridade do contexto

A RLM é um dos métodos estatísticos multivariados cuja a principal preocupação consiste em estabelecer as relações entre várias variáveis independentes ou preditoras e uma variável dependente ou critério. Ao identificar como estas múltiplas variáveis independentes se relacionam com a variável dependente, as informações sobre as variáveis independentes podem ser utilizadas para fazer previsões precisas e poderosas [Quinn & Keough 2002]. Consideremos, *m* observações de um conjunto de *p* número do variável preditor *Xs* e um variável critério *Y* associado a elas. O modelo de RLM, que se ajusta a este cenário, é definido conforme a equação 9.

$$\boldsymbol{y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_j x_{ij} + \cdots + \beta_p x_{ip} \in_i,} \quad \textbf{(9)}$$

onde, para a *i-ésima* observação, $y_i = Y$ e $x_{ji} = X_j$, $\forall X_j \in X$. O coeficiente do RLM, β_0, é a interceção da população e β_j, é a variação de *Y* em uma unidade de variação em X_j, $\forall X_j \in X\ e\ 1 \leq j \leq p$, mantendo as outras variáveis independentes constantes, ϵ_i é o erro aleatório ou inexplicável associado à *i-ésima* observação. Os valores estimados dos coeficientes da RLM são calculados através dos valores conhecidos da equação (8) de

cada observação, e resolvidos algebricamente. Estimando $\beta = \{\beta_0, \beta_1, \ldots, \beta_p\}$, $\forall\beta_0 \in \beta$ e σ_ϵ^2 (erro da variância), a reta de regressão ajustada, que prevê Y para qualquer observação desconhecida, é expressa conforme a equação 10:

$$\hat{\boldsymbol{y}}_{\boldsymbol{i}} = \boldsymbol{b_0} + \boldsymbol{b_1 x_{i1}} + \cdots + \boldsymbol{b x_{ij}} + \cdots + \boldsymbol{b_p x_{ip}} \in_{\boldsymbol{i}} \quad \textbf{(10)}$$

onde, $\hat{y}_i$ é o valor previsto para qualquer observação desconhecida, b_j é a estimativa da amostra de $\beta_j\ \forall\beta_j\ \in\ \beta$.

Ao invés de calcular as regressões para cada variável preditora individualmente, a RLM utiliza informações de todas as variáveis independentes simultaneamente para prever uma única variável critério. Como resultado, ela é naturalmente mais rápida do que os outros métodos de análise multivariada [Mertler & Reinhart, 2016]. Por conseguinte justifica a nossa opção por esse método estatístico multivariado.

Fazemos o mapeamento da tabela de previsão de contexto para o modelo RLM, considerando cada *tuplo* da tabela de previsão como uma observação de um conjunto de dados da RLM em que, o conjunto de variáveis independentes, $X = C_i,\ \forall C_i \in C$ de cada combinação $\mu \in F$, a variável dependente, $Y = \hat{\delta}_i(\mu_k)$ é a prioridade prevista de qualquer requisição desconhecida, $P_r = \hat{y}_i$.

Para qualquer requisição $r \in R$, quanto menor for o valor P_r, maior será a prioridade. A tabela de previsão de prioridade de contexto é criada através dos valores quantitativas $\forall C_i \in C$ e dos parâmetros de contexto que são independentes entre si.

Após a definição da prioridade do contexto, as informações do contexto desta tarefa são armazenadas no repositório de contexto.

4.4. Otimização no escalonamento das aplicações

A *unidade de QoE e escalonamento sensível ao contexto* do sistema proposto escalona as tarefas prioritárias para serem executadas em MVs na *fog* após um intervalo de escalonamento (IE), τ.

De modo a otimizar a QoE, o escalonador proposto explora a prioridade de contexto (P_r) da requisição $r \in R$ e a sua duração estimada do tempo de execução ($T_{r,v}$) para definir o escalonamento dessa requisição $r \in R$ em uma MV, $v \in V$. Se em virtude da quantidade limitada de MVs e à baixa prioridade, uma requisição $r \in R$ não poder ser escalonada num intervalo de escalonamento, ela deverá ser escalonada nos próximos intervalos. Na nossa proposta, a execução de tarefas com menor prioridade é interrompida em virtude da chegada de tarefa com maior prioridade. Considerando que essa preempção pode propiciar a espera por tempo indeterminado para a execução de uma requisição $r \in R$, exploramos também o número de intervalos de escalonamento (I_r), em que uma requisição de escalonamento é adiado desde a sua chegada, com o objetivo de evitar a condição de *starvation*.

Com vista a otimizar a QoE foi utilizada a técnica MONLP para escalonar as requisições.

Segundo [Miettinen 1998], a técnica MONLP é geralmente aplicada aos problemas de otimização onde existem mais de uma função objetivo (FO) não lineares e conflituantes a serem otimizada simultaneamente. Ela possui alguns elementos: conjunto de funções objetivo não lineares que devem ser otimizadas; variáveis de decisão que constituem o domínio no qual cada FO deve operar; restrições que delimitam o espaço de pesquisa.

Nas secções 4.4.1 e 4.4.2, são definidas as funções objetivos e as restrições para o escalonamento otimizado de aplicações.

4.4.1. Definição da função objetivo

Um dos objetivos deste artigo consiste em escalonar requisições, $r \in R$ em MV, $v \in V$, com o objetivo de otimizar a QoE para todas as requisições num determinado intervalo de escalonamento.

Assumindo que a execução de todas as requisições, $r \in R$ não são preemptivas e que todas as MVs, $v \in V$ estão criadas na *fog*, a FO é definida conforme a equação 11.

$$\boldsymbol{min}_{r,v} \sum_{r \in R} \sum_{v \in V} \frac{\boldsymbol{P_r} * \boldsymbol{\psi_{r,v}}}{\boldsymbol{I_r}} \quad \textbf{(11)}$$

E está sujeita a algumas restrições, (inequações (12) à (17)).

Esta equação indica que a QoE de todas as requisições de aplicações dos utilizadores podem ser otimizadas através da minimização da soma dos seus tempos de execução. Ela também leva em consideração a execução prioritária das tarefas com maiores prioridades através da minimização da soma das prioridades de todas as requisições, dado que, quanto menor for o resultado, maior será a prioridade obtida. Além disso, a soma dos valores inversos do $(I_r), \forall r \in R$ mostra que as requisições, em que foram adiados os seus escalonamentos num determinado intervalo, terão maior prioridade para serem escalonadas nos atuais intervalos, atenuando assim a situação de *starvation*.

4.4.2. Definição das restrições

Para o escalonamento otimizado definimos as seguintes restrições:

- Restrição de capacidade, é apresentada como a inequação 12.

$$\sum_{v \in V} \boldsymbol{\eta_v} \leq \boldsymbol{\eta} \quad \textbf{(12)}$$

  onde, η_v representa o tamanho da máquina virtual e η é a capacidade total do nó da *fog*.

- Restrição de afetação de MV, é apresentada conforme a inequação 13.

$$\sum_{v \in V} \boldsymbol{K_{v,r}} \leq \boldsymbol{1}, \forall \boldsymbol{v} \in \boldsymbol{V}, \forall \boldsymbol{r} \in \boldsymbol{R} \quad \textbf{(13)}$$

  onde, $K_{v,r}$ é uma variável booleana, que é igual a um, se uma MV $v \in V$ for afetada a uma requisição $r \in R$; caso contrário é zero.

- Restrição de escalonamento das requisições, é escrita conforme a inequação 14.

$$\sum_{r \in R} \chi_{r,v} \leq 1, \forall r \in R, \forall v \in V. \quad (14)$$

onde, $\chi_{r,v}$ é uma variável booleana, é igual a um, se uma requisição $r \in R$ estiver escalonada em uma MV $v \in V$; caso contrário, é zero.

- Restrição de QoS, é representada conforme a inequação 15.

$$\psi_{r,v} + Q_r \leq T_r, \forall r \in R. \quad (15)$$

onde, $\psi_{r,v}$ representa o tempo necessário para executar uma requisição $r \in R$ em uma máquina virtual $v \in V$, Q_r é o tempo de espera na fila da requisição *r*. e T_r corresponde à QoS da aplicação.

- Restrição de consumo de energia, é apresentada conforme a inequação 16:

$$B_r \geq B_{th}. \quad (16)$$

onde, B_r representa o nível de bateria do dispositivo do utilizador final associado a uma requisição, $r \in R$ e B_{th} indica o nível mínimo de bateria, para que o dispositivo requisitante se permaneça ligado até ao fim da execução.

- Restrição da qualidade do sinal, é escrita conforme a inequação 17:

$$SIN_r \geq SIN_{th}, \quad (17)$$

onde, SIN_r representa a intensidade do sinal associado a uma requisição $r \in R$ e SIN_{th} indica a intensidade mínima do sinal necessário para a submissão de uma requisição.

5. Exemplo ilustrativo

Nesta secção apresentamos um exemplo ilustrativo que ajuda a entender as funcionalidades do sistema proposto.

5.1. Construção da tabela de contexto

Seja o valor de níveis abstratos para diferentes contextos, onde:

$\alpha_{\text{Bateria}} = 3, \alpha_{\text{Sin}} = 2, \alpha_{\text{QoS}} = 4.$

Num determinado período de tempo *t*, o repositório de contexto disponibiliza os valores *Min-Max* e as equações (1) e (2) calculam o intervalo normalizado η_i e as diferenças lógicas entre dois níveis abstratos consecutivos, γ_i para os diferentes parâmetros de contexto $C_i \in C$, como ilustra a Tabela 2.

Tabela 3. Intervalo normalizado e diferenças gerais das informações de contexto

Parâmetros	Valores *Min-Max*	η_i	γ_i
$C_{bateria}$	Max - 99,75 %; Min - 16,65 %	0,83	0,28
C_{Sin}	Max - 59,83 dB; Min - 23,63 dB	0,61	0,3
C_{QoS}	Max - 12,71s; Min - 3,17s	0,75	0,19

Utilizando a equação (3), o conjunto dos níveis quantitativos, L_i para os diferentes parâmetros de contexto $C_i \in C$, é calculado:

$$L_{bateria} = \{0{,}17;\ 0{,}44;\ 0{,}72\}, L_{Sin} = \{0{,}39;\ 0{,}7\}, L_{QoS} = \{0{,}25;\ 0{,}44;\ 0{,}62;\ 0{,}81\}$$

Em seguida, calculamos o produto cartesiano *F* para os diferentes níveis quantitativos, L_i através da equação (4).

O multiconjunto correspondente, *M*, é calculado conforme a equação 5.

$$M = \{0{,}17; \ldots; 0{,}17;\ 0{,}25;\ 0{,}39;\ \ldots; 0{,}25;\ 0{,}44;\ 0{,}62;\ 0{,}7\}$$

Considerando os valores iniciais do θ_i, como: $\theta_{\text{Bateria},0} = 3$, $\theta_{Sin,0} = 2$ e $\theta_{QoS,0} = 8$, o conjunto enviesado $\theta_i, \forall C_i \in C$, que satisfaz a inequação (6), é indicado a seguir:

$$\theta_{Bateria} = \{3;\ 13;\ 55\}, \theta_{Sin} = \{2; 9\}, \theta_{QoS} = \{8;\ 34;\ 143; 598\}$$

Ao aplicar a equação (7) a qualquer elemento candidato $\mu(0{,}17;\ 0{,}39;\ 0{,}25) \in F$, a prioridade de contexto estimada, $\delta, \forall C_i \in C$, é calculada como:

$$\delta_{Bateria}(\mu) = 0{,}06;\ \delta_{Sin}(\mu) = 0{,}04;\ \delta_{QoS}(\mu) = 0{,}17$$

Através da equação (8), obtemos a prioridade estimada, $\hat{\delta}_i(\mu), \forall \mu \in F$, conforme descrito na Tabela 3.

Tabela 4. Tabela de previsão

Série	$C_{Bateria}(\mu)$	$C_{Sin}(\mu)$	$C_{QoS}(\mu)$	$\hat{\delta}_i(\mu_k)$
1	0.17	0,39	0,25	0,27
2	0,17	0,39	0,44	0,83
…	…	…	…	…
24	0,72	0,70	0,81	53,3

5.2. Previsão da prioridade do contexto

Na *unidade de previsão do contexto* é aplicada a análise de RLM na tabela de previsão de contexto, Neste exemplo, consideramos $\epsilon_i = 0$, os seguintes coeficientes de regressão são calculados através da equação (9). $\beta_0 = -3{,}21;\ \beta_1 = 2{,}24;\ \beta_2 = 1{,}1;\ \beta_3 = 5{,}09$.

Estes coeficientes são utilizados para prever a prioridade do contexto para qualquer tarefa, através da equação (10), durante um período de tempo *t,* neste exemplo igual a três segundos, em um nó da *fog,* composta por dois MVs com as mesmas configurações. Durante este tempo, $|R| = 8$ requisições chegam. As informações do contexto são normalizadas de acordo com os valores máximos e mínimos de $C_i \in C$ e, através da

equação (9), a previsão da prioridade $P_r, \forall r \in R$, é calculada conforme apresentado na Tabela 4, com valores da informação do contexto atual (A) e o seu respetivo valor normalizado (N).

Tabela 5. Previsão de prioridade de contexto

ID (r)	*Tempo de chegada (r)*	$C_{Bateria}$	C_{Sinr}	C_{QoS}	P_r
1	0	A:52,60; N:0,53	A:33,48; N:0,56	A:3,56; N:0,28	0,02
2	0	A:18,42; N:0,18	A:57,58; N:0,97	A:5,75; N:0,45	0,57
3	0	A:35,05; N:0,35	A:41,34; N:0,69	A:11,56; N:0,91	2,97
4	0,22	A:84,88; N:0,85	A:35,56; N:0,59	A:8,31; N:0,65	2,68
5	0,72	A:39,73; N:0,40	A:47,69; N:0,80	A:7,94; N:0,63	1,74
6	0,88	A:32,22; N:0,32	A:29,45; N:0,49	A:10,1; N:0,79	2,10
7	1,63	A:39,28; N:0,39	A:38,92; N:0,65	A:6,95; N:0,55	1,17
8	1,77	A:44,00; N:0,44	A:47,54; N:0,79	A: 5,50; N:0,43	0,86

5.3. Otimização do escalonamento de tarefas

De acordo com a FO, equação (11), as tarefas são escalonadas na *unidade de QoE e escalonamento sensível contexto*. As tarefas que tenham satisfeito as restrições são escalonadas na *fog*. A Tabela 5, apresenta o resultado do calculo da FO na *unidade de QoE*.

Tabela 6. Escalonamento de tarefas

Linha de tempo	ID & Saída da FO	Informações sobre o escalonamento
0,0	*r1 - 0,015; r2 - 0,573; r3 - 2,971*	*r*1 → MV0; *r*2 → MV1
1,0	*r3 - 1,485; r5 - 1,743; r6 - 2,103; r4 - 2,681*	*r*3 → MV0, *r*5 → MV1
2,0	*r8 - 0,857; r6 - 1,051; r7 - 1,173; r4 - 1,340*	*r*8 → MV0, *r*6 → MV1
3,0	*r7 - 0,586; r4-0,893*	*r7* → MV0, *r4* → MV1

6. Conclusões

Neste artigo, fizemos uma revisão da literatura sobre os principais algoritmos de escalonamento na arquitetura *cloud* e no paradigma *fog*, explorámos e sugerimos algumas perspetivas de melhorias e com base nestas, propomos um algoritmo de escalonamento sensível ao contexto para a arquitetura *fog*.

O algoritmo proposto utiliza a normalização *Min-Max*, para resolver o problema da heterogeneidade dos diferentes parâmetros de contexto, o método da RLM, para a definição das prioridades das requisições e a técnica MONLP, para o escalonamento ótimo visando otimizar a QoE dos utilizadores.

As principais conclusões alcançadas foram que:

- Na *cloud,* o escalonamento foi amplamente estudado. Enquanto que na *fog*, devido densidade e heterogeneidade de dispositivos, o escalonamento é complexo e segundo o conhecimento dos autores, existem poucos estudos.

- Muitos algoritmos estudados não descrevem a forma como a prioridade é definida, não explicam o método utilizado para a priorização das tarefas, nem definem a

prioridade com base em informações do contexto. Outros ainda, ignoram a forma como a heterogeneidade de diferentes contextos são tratadas e muitos descartam o aprimoramento da QoE.

- A maioria analisa as politicas na perspetiva dos provedores de serviços. Outros são aplicados em tarefas agrupadas para diminuir o tempo de execução. Alguns otimizam apenas a QoS.

- Aspetos como: utilização do contexto no escalonamento; priorização de tarefas sensíveis ao contexto; consideração da restrição energética no escalonamento; preservação da força do sinal da rede; preservação da QoS; redução do tempo médio de espera e otimização da QoE podem ser explorados e melhorados pelos escalonadores atuais.

Todos os objetivos propostos foram alcançados. Consideramos vários parâmetros de contextos. No entanto, outros ainda podem ser ponderados por forma a serem analisados as suas influências no escalonamento. Ainda, prevemos implementar a nossa proposta em ambiente *fog* real ou no simulador *iFogSim*, analisar o desempenho e comparar os resultados com as outras propostas de escalonamento não sensíveis ao contexto: *First Come First Served, Shortest Job First e QoS based Priority Scheduling*. Com base em algumas métricas da avaliação como: percentagem de execução das requisições bem-sucedidas; tempo médio de espera e QoE dos utilizadores em relação ao aumento das requisições, de MVs e de requisitos QoS da aplicação.

Agradecimentos
Os autores agradecem à *Fundação Calouste Gulbenkian* pelo financiamento desta investigação através da bolsa de Doutoramento sob a referência n.º 234242, 2019-Bolsas de Pós-Graduação para estudantes dos PALOP e de Timor-Leste.

Referências

Aazam M., Hilaire M. St., Lung Ch, Lambadaris I. (2016). MeFoRE: Resource Estimation QoE based at Fog to Enhance QoS in IoT, in: Proc. of the 23rd International Conference on Telecommunications, ICT '16, IEEE, pp. 1-5, https://doi.org/10.1109/ICT.2016.7500362.

Bahl P., Cuervo E., Balasubramanian A., Cho D. k, Wolman A., Saroiu S. and Chandra, R. (2010). Maui: making smartphones last longer with code offload., in Proceedings of the 8th international conference on Mobile systems, applications, and services. pp. 49-62, https://doi.org/10.1145/1814433.1814441.

Bazire, M., & Brézillon, P. (2005). Understanding Context Before Using It. Em A. Dey, B. Kokinov, D. Leake, & R. Turner (Eds.), *Modeling and Using Context* (pp. 29–40). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/11508373_3

Berg F., Durr F., Rothermel K. (2014). Increasing the efficiency and responsiveness of mobile applications with preemptable code offloading., IEEE International Conference on Mobile Services (MS), IEEE, pp. 76–83, https://doi.org/10.1109/MobServ.2014.20.

Cardellini V., Grassi V., Presti F.L., and Nardelli M. (2015). On QoS-Aware Scheduling of Data Stream Applications over Fog Computing Infrastructures, IEEE Symposium on Computers and Communication (ISCC), pp. 271-276, https://doi.org/10.1109/ISCC.2015.7405527.

Das A., Adhikary T., Razzaque M. , and Hong C. S. (2013). An intelligent approach for virtual machine and QoS provisioning in cloud computing. in Information Networking (ICOIN), International Conference, pp. 462-467 https://doi.org/10.1109/ICOIN.2013.6496423.

Deng R., Luan T. H, Lu R., Liang H., and Lai C. (2016). Optimal Allocation Workload in Fog-Cloud Computing Towards Balanced Delay and Power Consumption, IEEE Internet Things J., vol. X, no. X, pp. 1171-1181, https://doi.org/10.1109/JIOT.2016.2565516.

Dey A.K., Abowd G.D., Brown P.J., Davies N., Smith M., Steggles P. (1999) Towards a Better Understanding of Context and Context-Awareness. In: Gellersen HW. (eds) Handheld and Ubiquitous Computing. HUC 1999. Lecture Notes in Computer Science, vol 1707. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-48157-5_29.

Fan J., Wei X., Wang T., Lan T., Subramaniam S. (2017). Deadline-aware task scheduling in a Tiered IoT Infrastructure, GLOBECOM 2017 - 2017 IEEE Global Communications Conference, Singapore, pp. 1-7, https://doi.org/10.1109/GLOCOM.2017.8255037.

Ghouma H. and Jaseemuddin M. (2015). Context aware resource allocation and scheduling for mobile cloud, 2015 IEEE 4th International Conference on Cloud Networking (CloudNet), Niagara Falls, ON, pp. 67-70, https://doi.org/10.1109/CloudNet.2015.7335282.

Gill S. S., Garraghan P., Buyya R. (2019). ROUTER: Fog enabled cloud based intelligent resource management approach for smart home IoT devices, Journal of Systems and Software, Volume 154, pp. 125-138, https://doi.org/10.1016/j.jss.2019.04.058.

Gordon M., Jamshidi D., Mahlke S., Mao Z., Chen X. (2012). COMET: code offload by migrating execution transparently, Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation, pp. 93-106, https://dl.acm.org/doi/10.5555/2387880.2387890.

Gupta H., Dastjerdi A. V., Ghosh S. K., and Buyya R. (2016). iFogSim: A toolkit for modeling and simulation of resource management techniques in internet of things, edge and fog computing environments. CoRR, vol. abs/1606.02007.

Han J. (2011). Data Mining: Concepts and Techniques (3th Ed.). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc..

Intharawijitr K., Iida K., and Koga H. (2016). Analysis of Fog Model considering Computing and Communication Latency in 5G Cellular Networks, IEEE

International Conference on Pervasive Computing and Communication Workshops (Workshops Percom), pp. 1-4, https://doi.org/10.1109/PERCOMW.2016.7457059.

Kitchenham, B. (2004). Procedures for Performing Systematic Reviews Kitchenham, B., 2004. Keele, UK, Keele University (Vol. 33). Obtido de https://www.inf.ufsc.br/~aldo.vw/kitchenham.pdf em 22 de maio de 2020.

Lawanyashri M., Balusamy B., and Subha S. (2017). Energy-Aware fruitfly hybrid optimization for load balancing in cloud environments is EHR applications, Informatics Med. Unlocked, vol. 8, no. March, pp. 42-50, https://doi.org/10.1016/j.imu.2017.02.005.

Li Q., Novak E., Yi S. and Hao Z. (2017). Challenges and Software Architecture for Fog Computing, in IEEE Internet Computing, vol. 21, no. 2, pp. 44-53, https://doi.org/10.1109/MIC.2017.26.

Li T., Liu Y., Gao A. L. and Liu. (2017). A for Cooperative - based Smart Sensing Tasks in Fog-Computing in IEEE, Access, vol. 5, pp. 21296-21311, https://doi.org/10.1109/ACCESS.2017.2756826.

Mahmud M. R., Afrin M., Razzaque M. A. , Hassan M. M., Alelaiwi A., Alrubaian M. (2016). Maximizing Quality of Experience through Context-Aware Mobile Application Scheduling in Cloudlet Infrastructure, Software: Practice and Experience 46 (11), pp. 1525-1545, https://doi.org/10.1002/spe.2392.

Mertler, C., & Reinhart, V. (2016). Advanced and Multivariate Statistical Methods (6th Editio). CA, USA: Routledge. https://doi.org/10.4324/9781315266978.

Musumba G.W., Nyongesa H. O. (2013). Context awareness in mobile computing: a review, International Journal of Machine Learning and Applications Vol. 2 (1): pp. 1-5, https://doi.org/10.4102/ijmla.v2i1.5.

Oueis J., Strinati E. C. and Barbarossa S. (2015). The Fog Balancing: Load Cell Distribution for Small Cloud Computing, IEEE 81st Vehicular Technology Conference (VTC Spring), Glasgow, 2015, pp. 1-6, https://doi.org/10.1109 / VTCSpring.2015.7146129.

Qiu, M., Chen, Z., Yang, L. T., Qin, X., & Wang, B. (2012). Towards Power-Efficient Smartphones by Energy-Aware Dynamic Task Scheduling. Em 2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems (pp. 1466–1472). IEEE. https://doi.org/10.1109/HPCC.2012.214.

Quinn, G. P., & Keough, M. J. (2002). Experimental Design and Data Analysis for Biologists. Cambridge University Press. https://doi.org/10.1017/CBO9780511806384.

Seddik Y. and Hanzálek Z. (2017). Match-up scheduling of mixedcriticality jobs: Maximizing the probability of execution jobs, Europen Journal of Operational Research., Vol. 262, no. 1, pp. 46-59, https://doi.org/10.1016/j.ejor.2017.03.054.

Sheikhalishahi M., Grandinetti L., Guerriero F., Wallace RM, and Vazquez-Poletti JL. (2016). Multi-dimensional job scheduling, Future Generation Computer Systems, Vol. 54, pp. 123-131, https://doi.org/10.1016/j.future.2015.03.014.

Shinde S.K. and Gawali M. B. (2018). Task scheduling and resource allocation in the cloud using heuristic approach, Jornal Cloud Computing, Vol. 7, no. 1, https://doi.org/10.1186/s13677-018-0105-8.

Shojafar M., Javanmardi S., Abolfazli S. (2015). FUGE: The joint meta-heuristic approach to cloud job scheduling algorithm using fuzzy theory and the genetic method, Cluster Computing Vol. 18, pp. 829-844, https://doi.org/10.1007/s10586-014-0420-x.

Skarlat O., Nardelli M., Schulte S., Dustdar S., (2017). Towards QoS-aware Service Placement Fog, in: Procedure of the First IEEE International Conference on Fog and Edge Computing, ICFEC '17, IEEE, https://doi.org/10.1109/ICFEC.2017.12.

Stavrinides G. L., Karatza H. D. (2019). A hybrid approach to real-time scheduling IoT workflows in fog and cloud environments, Multimedia Tools and Applications Vol. 78, pp. 24639-24,655, https://doi.org/10.1007/s11042-018-7051-9.

Stojmenovic, I. (2014). Fog computing: a cloud to the ground support for smart things and machine-to-machine networks. Telecommunication Networks and Applications Conference (ATNAC), pp 117–122, https://doi.org/10.1109 / ATNAC.2014.7020884.

Swaroop P. (2019). Cost Based Job Scheduling In Fog Computing, PhD thesis, DTU, India, Available at: http://dspace.dtu.ac.in:8080/jspui/handle/repository/16722, (Accessed 6 july 2020).

Tiwary M. Sahoo B., LT, Yang D., Puthal K. S. (2018). Response time for optimization cloudlets in Mobile Computing Edge, Jornal of Parallel Distributed Computing, Vol. 119, pp. 81-91, https://doi.org/10.1016/j.jpdc.2018.04.004.

Wang X., Wang Y., and Cui Y. (2016). An energy-aware bi-level optimization model for multi-job scheduling problems under cloud, Soft Computing, Vol. 20, no. 1, pp. 303-317, https://doi.org/10.1007/s00500-014-1506-3.

Yang Y., Zhao S., Zhang W., Chen Y., Luo X. and Wang J. (2018). DEBTS: Delay Balanced Energy Task Scheduling in Homogeneous Fog Networks, in IEEE Internet of Things Journal, vol. 5, no. 3, pp. 2094-2106, https://doi.org/10.1109/JIOT.2018.2823000.

Zhou B., Dastjerdi A. V., Calheiros R. N., Srirama S. N., Buyya R. (2017). mcloud: The Context-Aware Offloading Framework for Heterogeneous Mobile Cloud, in IEEE Transactions on Services Computing, vol. 10, no. 5, pp. 797-810, https://doi.org/10.1109/TSC.2015.2511002.

Zhou X., Sun M., Wang Y., Wu X., (2015). The New QoE-driven Video Cache Allocation Scheme for Mobile Cloud Server, in: Procedure of the 11th Conference on International Heterogeneous Networking for Quality, Reliability, Security and Robustness, QSHINE 15, IEEE, pp. 122-126.

Zhu C., Li X., Leung V., Hu X., Yang T.L. (2015). Towards Integration of Wireless Sensor Networks and Cloud Computing, IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom), IEEE, Singapore, pp. 62-69, https://doi.org/10.1109/CloudCom.2015.27.

Luan T.H., Gao L., Li Z., Xiang Y., We G., Sun L. (2016). A View of Fog Computing from Networking Perspective, https://arxiv.org/abs/1602.01509v3.

La H. J., Kim S. D. (2010). A conceptual framework for provisioning context-aware mobile cloud services, in Cloud Computing (CLOUD), IEEE 3rd International Conference on. IEEE, pp. 466-473, https://doi.org/10.1109/CLOUD.2010.78.

Celestino Barros, Professor assistente na Faculdade de Ciências e Tecnologia (FCT) da Universidade de Cabo Verde (Uni-CV). Licenciado em Informática em 2006 pelo Instituto Superior de Educação. Obteve o grau Mestre em Engenharia Electrónica e Telecomunicações pela Universidade de Aveiro em 2010. Possui o certificado de Estudos Avançados e é Doutorando em Ciência e Tecnologia pela UaB e UTAD. Tem como áreas de interesse, a *Cloud computing* e as suas paradigmas.

Vitor Rocio, Professor Associado da Universidade Aberta (UAb). Doutorado em Informática pela Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa (2002), e Licenciado em Engenharia Informática pela FCT-UNL (1993). É diretor do Departamento de Ciências e Tecnologia da UAb. Os seus principais interesses são as tecnologias das linguagens humanas, o processamento automático de línguas naturais, os sistemas de análise sintáctica evolutivos, e as tecnologias de elearning.

André Filipe Sousa é Licenciado em Informática (2009) e Doutorado em Informática (2017) pela Universidade de Trás-os-Montes e Alto Douro (UTAD). Foi professor auxiliar convidado na UTAD entre 2017-2019. Atualmente é *developer* na empresa Critical Techworks, onde desenvolve *software* na área de *automotive, IoT e smart devices*. Os seus interesses de investigação são na área de *Smart devices*, IoT e inteligência artificial para melhoria da interação com o utilizador.

Hugo Paredes é Licenciado em Engenharia de Sistemas e Informática (2000) e Doutorado em Informática (2008) pela Universidade do Minho, e possui o título de Agregado (2016) pela Universidade de Trás-os-Montes e Alto Douro (UTAD). Atualmente é Professor Auxiliar com Agregação na UTAD e Diretor do Mestrado em Engenharia Informática. É Coordenador Adjunto do Centro de Sistemas de Informação e Computação Gráfica no INESC TEC. Os seus interesses de investigação são na área de Interação Pessoa-Computador, particularmente em Crowd computing, sistemas colaborativos e acessibilidade.