

Lucas Villela Canôas

**Estudo Comparativo Entre Políticas
De Escalonamento Para Fog Computing**

São José do Rio Preto

2023

Lucas Villela Canôas

Estudo Comparativo Entre Políticas De Escalonamento Para Fog Computing

Trabalho de Conclusão de Curso (TCC) apresentado como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação, junto ao Conselho de Curso de Bacharelado em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista “Júlio de Mesquita Filho”, Campus de São José do Rio Preto.

Orientador: Prof. Dr. Aleardo Manacero Jr.

São José do Rio Preto

2023

Canôas, Lucas V.

Título do trabalho : Estudo Comparativo Entre Políticas De
Escalonamento Para Fog Computing / Lucas Villela Canôas. -- São José
do Rio Preto, 2023

XXX f. : il., tabs.

Orientador: Prof. Dr. Aleardo Manacero Jr.

Trabalho de Conclusão de Curso - Universidade Estadual Paulista

(Unesp),

Instituto de Biociências, Letras e Ciências Exatas, São José do Rio Preto

1. Fog Computing. 2. Escalonamento. 3. Simulação Computadores.
4. Computação em grade (Sistemas de computador).

CDU –

Lucas Villela Canôas

Estudo Comparativo Entre Políticas De Escalonamento Para Fog Computing

Trabalho de Conclusão de Curso (TCC) apresentado como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação, junto ao Conselho de Curso de Bacharelado em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista “Júlio de Mesquita Filho”, Câmpus de São José do Rio Preto.

Prof. Dr. Aleardo Manacero Jr. Lucas Villela Canôas

Banca Avaliadora:

Prof^a. Dr. Rodrigo Capobianco Guido

Prof. Dr. Wallace Correa de Oliveira Casaca

São José do Rio Preto

18 de novembro de 2023

*Para minha irmã Larissa,
a minha mãe Telma,
ao meu pai José.*

Agradecimentos

Gostaria de agradecer a minha mãe, minha irmã e meu pai, Telma, Larissa e José, vocês foram importantes nesse caminho da graduação, sem esta base de apoio nada disso seria possível.

Agradeço também ao Professor Doutor Aleardo Manacero Jr. que me ofereceu a oportunidade de conhecer melhor o mundo dos sistemas distribuídos e sua paciência, aos membros da banca Professor Doutor Rodrigo Capobianco Guido, pela confiança e por me proporcionar diversas oportunidades, ao Professor Doutor Wallace Correa de Oliveira Casaca, que nos motiva em conversas simples pelos corredores nos dando energia, ao T.L.A. pela ajuda na revisão deste trabalho e ao W.S.C. que prestou grande apoio nos momentos complicados e a B.G.S que me acompanhou nos momentos finais.

Impossível não lembrar de cada um que encontrei nos trabalhos que desenvolvi durante a graduação (dentro ou fora da universidade), cada experiência profissional (ou não) me proporcionou aprender com todas as pessoas que convivi, algo que levarei para sempre comigo como ensinamento de vocês, é que podemos e devemos aprender com todos.

A todas as pessoas incríveis que encontrei durante esta caminhada e a influência que tive ainda na adolescência de um grupo qualquer na internet, que me fez chegar até aqui também, espero que vocês estejam bem.

"Conhecimento é poder. A informação é libertadora. A educação é a premissa do progresso, em todas as sociedades, em todas as famílias."

Kofi Annan

Resumo

Com o advento de dispositivos cada vez menores, a computação conseguiu percolar em diversos locais, permitindo a automação e extração de dados em diferentes esferas da sociedade. Podemos aplicar os dispositivos IoT (*Internet of Things*) nesses setores, originando a agricultura inteligente, cidades inteligentes, mobilidade inteligente e outras. Os problemas solucionados por IoT, por vezes tem necessidade de conectividade e interoperabilidade, essa necessidade deriva dos sistemas que são criados tem a característica de que o tempo é crítico para lidar com os riscos. As redes de Fog Computing são utilizadas para ter justamente essas características, não utilizando a mais a arquitetura clássica de IoT. Nesse sentido, nos deparamos muitas vezes com problemas NP-Completo e por conta disso não é possível muitas vezes ter a melhor solução para um determinado problema, sendo necessário encontrar uma solução boa e não necessariamente ótima para solucionar a questão apresentada. Para isso, foi implementado os algoritmos *Load Balancer Ant Colony Optimization* (LBACO), *Ant Colony Optimization* (ACO), *Particle Swarm Optimization* (PSO) e *Artificial Bee Colony* (ABC), que podemos utilizar como escalonadores de tarefas em sistemas distribuídos que operam na camada de Fog Computing, notou-se o surpreendente desempenho do ABC, que obteve resultados superiores em eficiência, mesmo em ambientes mais adversos e com poucos parâmetros do ambiente utilizado.

Palavras-chave: Fog Computing. Otimização de Colônia de Formigas. IoT. Escalonamento. Sistemas Distribuídos. Balanceamento de carga. Colônia Artificial de Abelhas. Otimização por enxame de Partículas. Colônia de formigas com balanceamento de carga probabilístico.

Abstract

With the advent of increasingly smaller devices, computing has managed to permeate various locations, enabling automation and data extraction in different spheres of society. We can apply Internet of Things (IoT) devices in these sectors, giving rise to smart agriculture, smart cities, smart mobility, and more. The problems solved by IoT sometimes require connectivity and interoperability, stemming from systems that are created with the characteristic that time is critical to dealing with risks. Fog Computing networks are used to precisely meet these characteristics, no longer relying on the classical architecture of IoT. In this sense, we often encounter NP-Complete problems, and consequently, it is not always possible to find the best solution for a particular problem. It becomes necessary to find a good solution, not necessarily the optimal one, to address the presented issue. To tackle this, the Load Balancer Ant Colony Optimization (LBACO), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Artificial Bee Colony (ABC) algorithms were implemented. These algorithms can be used as task schedulers in distributed systems operating in the Fog Computing layer. The surprising performance of ABC was noted, achieving superior efficiency results even in more adverse environments and with few parameters in the utilized environment.

Keywords: Fog Computing. Ant Colony Optimization. Internet of Things. Scheduling. Distributed Systems. Load Balancing. Artificial Bee Colony. Particle Swarm Optimization. Probabilistic Load-balancing Ant Colony.

Lista de Ilustrações

Figura 1: Pseudocódigo do ACO (gerado pelo autor).....	30
Figura 2: Pseudocódigo do LBACO (criado pelo autor).....	33
Figura 3: Pseudocódigo do ABC (criado pelo autor).....	34
Figura 4: Pseudo código PSO (criado pelo autor).....	36
Figura 5: 10 máquinas com capacidade de processamento aleatorizado (criado pelo autor)	37
Figura 6: 50 máquinas com capacidade aleatorizada (criado pelo autor).....	38
Figura 7: 100 máquinas com processamento aleatorizado (criado pelo autor).....	39
Figura 8: 10 máquinas com capacidade de processamento e largura de banda aleatorizados (criado pelo autor).....	40
Figura 9: 50 máquinas com capacidade de processamento e largura de banda aleatorizados (criado pelo autor).....	41
Figura 10: 100 máquinas com capacidade de processamento e largura de banda aleatorizado (criado pelo autor).....	42
Figura 11: iSPD com 10 máquinas (criado pelo autor).....	43
Figura 12: Tempo médio da execução (em segundos) em ambiente com 10 máquinas com capacidade de processamento e largura de banda padrão (criado pelo autor).....	52
Figura 13: Tempo médio da execução (em segundos) em ambiente com 50 máquinas com capacidade de processamento e largura de banda padrão (criado pelo autor).....	52
Figura 14: Tempo médio da execução (em segundos) em ambiente com 100 máquinas com capacidade de processamento e largura de banda padrão (criado pelo autor).....	53
Figura 15: Tempo médio da execução (em segundos) em ambiente com 10 máquinas com capacidade de processamento e largura de banda heterogênea (criado pelo autor).....	54
Figura 16: Tempo médio da execução (em segundos) em ambiente com 50 máquinas com capacidade de processamento e largura de banda heterogênea (criado pelo autor).....	55
Figura 17: Tempo médio da execução (em segundos) em ambiente com 100 máquinas com capacidade de processamento e largura de banda heterogênea (criado pelo autor).....	56
Figura 18: Tempo total da simulação (em segundos) em ambiente com 100 máquinas com capacidade de processamento e largura de banda padrão (criado pelo autor).....	57
Figura 19: Tempo total da simulação (em segundos) em ambiente com 10 máquinas com capacidade de processamento e largura de banda aleatorizada (criado pelo autor).....	58

Figura 20: Tempo total da simulação (em segundos) em ambiente com 50 máquinas com capacidade de processamento e largura de banda aleatorizada (criado pelo autor).....	59
Figura 21: Tempo total da simulação (em segundos) em ambiente com 100 máquinas com capacidade de processamento e largura de banda aleatorizada (criado pelo autor).....	60
Figura 22: Eficiência em ambiente com 10 máquinas com capacidade de processamento e largura de banda aleatorizada (criado pelo autor).....	61
Figura 23: Eficiência em ambiente com 50 máquinas com capacidade de processamento e largura de banda aleatorizada (criado pelo autor).....	61
Figura 24: Eficiência em ambiente com 100 máquinas com capacidade de processamento e largura de banda aleatorizada (criado pelo autor).....	62
Figura 25: Eficiência em ambiente com 10 máquinas com capacidade de processamento aleatorizado e largura de banda padrão (criado pelo autor).....	63
Figura 26: Eficiência em ambiente com 50 máquinas com capacidade de processamento aleatorizado e largura de banda padrão (criado pelo autor).....	64
Figura 27: Eficiência em ambiente com 100 máquinas com capacidade de processamento aleatorizado e largura de banda padrão (criado pelo autor).....	64

Lista de Tabelas

Tabela 1: Eficiência média (%) LBACO em ambiente com capacidade de processamento heterogêneo e largura de banda homogênea.....	44
Tabela 2: Eficiência média (%) LBACO em ambiente com capacidade de processamento heterogêneo e largura de banda heterogênea.....	44
Tabela 3: Tempo médio simulação (segundos) LBACO em ambiente com capacidade de processamento heterogêneo e largura de banda homogênea.....	44
Tabela 4: Tempo médio simulação (segundos) LBACO em ambiente com capacidade de processamento heterogêneo e largura de banda heterogênea.....	45
Tabela 5: Tempo médio execução (segundos) LBACO em ambiente com capacidade de processamento heterogêneo e largura de banda homogênea.....	45
Tabela 6: Tempo médio execução (segundos) LBACO em ambiente com capacidade de processamento heterogêneo e largura de banda heterogênea.....	45
Tabela 7: Eficiência média (%) ACO em ambiente com capacidade de processamento heterogêneo e largura de banda homogênea.....	46
Tabela 8: Eficiência média (%) ACO em ambiente com capacidade de processamento heterogêneo e largura de banda heterogênea.....	46
Tabela 9: Tempo total simulação (segundos) ACO em ambiente com capacidade de processamento heterogêneo e largura de banda homogênea.....	46
Tabela 10: Tempo total simulação (segundos) ACO em ambiente com capacidade de processamento heterogêneo e largura de banda heterogênea.....	47
Tabela 11: Tempo médio de execução (segundos) ACO em ambiente com capacidade de processamento heterogêneo e largura de banda homogênea.....	47
Tabela 12: Tempo médio de execução (segundos) ACO em ambiente com capacidade de processamento heterogêneo e largura de banda heterogênea.....	47
Tabela 13: Eficiência do ABC em (%) ambiente com capacidade de processamento heterogêneo e largura de banda homogênea.....	48
Tabela 14: Eficiência do ABC em (%) ambiente com capacidade de processamento heterogêneo e largura de banda heterogênea.....	48
Tabela 15: Tempo total simulação (segundos) do ABC em ambiente com capacidade de processamento heterogêneo e largura de banda homogênea.....	48

Tabela 16: Tempo total simulação (segundos) do ABC em ambiente com capacidade de processamento heterogêneo e largura de banda heterogênea.....	49
Tabela 17: Tempo total execução (segundos) do ABC em ambiente com capacidade de processamento heterogêneo e largura de banda homogênea.....	49
Tabela 18: Tempo total execução (segundos) ABC em ambiente com capacidade de processamento heterogêneo e largura de banda heterogênea.....	49
Tabela 19: Eficiência do PSO (%) em ambiente com capacidade de processamento heterogêneo e largura de banda homogênea.....	49
Tabela 20: Eficiência do PSO (%) em ambiente com capacidade de processamento heterogêneo e largura de banda heterogênea.....	50
Tabela 21: Tempo total simulação (segundos) do PSO em ambiente com capacidade de processamento heterogêneo e largura de banda homogênea.....	50
Tabela 22: Tempo total simulação (segundos) do PSO em ambiente com capacidade de processamento heterogêneo e largura de banda heterogênea.....	50
Tabela 23: Tempo médio de execução (segundos) do PSO em ambiente com capacidade de processamento heterogêneo e largura de banda homogênea.....	50
Tabela 24: Tempo médio de execução (segundos) do PSO em ambiente com capacidade de processamento heterogêneo e largura de banda heterogênea.....	51

LISTA DE ABREVIATURAS E SIGLAS

LBACO	<i>Load Balancer Optimization</i>
ACO	<i>Ant Colony Optimization</i>
PSO	<i>Particle Swarm Optimization</i>
ABC	<i>Artificial Bee Colony</i>
CDN	<i>Content Distribution Network</i>
IoT	<i>Internet of Things</i>
IoV	Internet of Vehicles
IIoT	<i>Industrial Internet of Things</i>
SACO	<i>Smart Ant Colony Optimization</i>
iSPD	<i>iconic Simulator of Parallel and Distributed Systems</i>

Sumário

Estudo Comparativo Entre Políticas De.....	1
Escalonamento Para Fog Computing.....	1
.....	15
Capítulo 1.....	16
Introdução.....	16
1.1 - Motivação.....	17
1.2 - Objetivo.....	17
1.3 - Metodologia.....	17
1.4 - Estrutura do texto.....	18
Capítulo 2.....	19
Revisão Bibliográfica.....	19
2.1 IoT.....	19
2.1.1 Definição.....	19
2.1.2 Aplicações.....	19
2.1.3 Problemas no IoT.....	20
2.2 Fog Computing.....	20
2.2.1 História e definição.....	20
2.2.2 Diferenças entre Cloud, Edge e Fog.....	21
2.2.2 Aplicações.....	23
2.2.3 Problemas.....	23
.....	24
2.3 Soluções para escalonamento.....	24
2.3.1 Classificação de problemas.....	24
2.3.2 Algoritmos de escalonamento.....	24
2.4 Simulador.....	25
2.4.1 iSPD.....	25
2.4.2 iFogSim.....	26
2.4 Considerações Finais.....	26
Capítulo 3.....	27
Desenvolvimento.....	27
3.1 Colônia de formigas.....	28
3.1.1 Ant Colony Optimization.....	28
3.1.2 Load Balancer Ant Colony Optimization.....	30
3.2 Artificial Bee Colony.....	34
3.3 Particle Swarm Optimization.....	35
3.4 Características dos ambientes.....	36
Capítulo 4.....	44
Resultados.....	44
4.1.1 LBACO.....	44
4.1.2 ACO.....	46

4.1.3 ABC.....	48
4.1.4 PSO.....	49
4.2 Discussão dos resultados.....	51
Capítulo 5.....	65
Conclusões.....	65
5.1 Considerações.....	65
5.2 Dificuldades e desafios.....	66
5.3 Direções futuras.....	66
Referências Bibliográficas.....	67

Capítulo 1

Introdução

Nos últimos anos o crescimento da quantidade dos dispositivos conectados à internet foi vertiginoso e nada indica no horizonte que irá estacionar. A perspectiva é que em 2027 existam 29 bilhões de dispositivos conectados à internet, sendo que em 2022 eram 14 bilhões. Em 2023 a perspectiva (IOT-ANALYTICS, 2022) é de 16 bilhões de dispositivos IoT conectados, tornando-se o segmento de maior crescimento.

A utilização da tecnologia IoT (*Internet of Things*) se tornou a base da quarta revolução industrial (LEE; KAO, 2014), pois permite que os objetos sejam controlados e informação seja extraída do ambiente e dos próprios dispositivos. Sendo assim, a capacidade de controle atingiu um novo patamar, permitindo realizar a interoperabilidade de cadeias produtivas.

A primeira revolução industrial foi realizada com as máquinas a vapor, a segunda foi marcada pela utilização de energia elétrica, já a terceira foi definida pela utilização de sistemas eletrônicos e hoje nota-se a quarta revolução industrial, tendo como principal característica a fusão entre a indústria e os sistemas digitais hiperconectados (MAGALHÃES; VENDRAMINI, 2018).

Porém, com a expansão dos dispositivos IoT e um aumento dramático da produção de informação, o congestionamento da rede de computadores se tornaria uma possibilidade concreta e tornaria a implementação das soluções inviáveis. Para apresentar uma solução foi proposto o modelo de *Fog Computing* que permite uma comunicação separada da rede mundial de computadores.

1.1 - Motivação

Ambientes de *Fog Computing* tem muitos dispositivos diferentes, podem integrar o sistema carros, notebooks, sensores em geral, drones, semáforos e eventualmente uma cidade inteira, originando as *smart cities*. Neste cenário, a interoperabilidade do sistema é realizada de forma que uma tarefa pode ser processada paralelamente por toda a rede, porém a distribuição não pode ser homogênea, visto que carros, sensores e notebooks tem capacidade de processamento diferentes (IKRAM; MUHAMMED, 2019).

Nesta situação, é necessário realizar uma distribuição do fluxo de trabalho, condizente com as variáveis de cada dispositivo, como a capacidade de processamento de cada nó responsável por realizar o processamento. Por se tratar de uma infraestrutura muito grande e custosa, se faz necessário a utilização de softwares de simulação, sendo assim possível de realizar um estudo comparando algoritmos que realizam o escalonamento de recursos de forma economicamente viável para vários propósitos (SINGH, 2019).

1.2 - Objetivo

Este trabalho tem como objetivo implementar os algoritmos LBACO (GUPTA, 2017) e ACO (GHAFARI, 2019) para a realização de um estudo comparativo das políticas de escalonamento de tarefas em sistemas baseados em *Fog Computing*, utilizando o iSPD (DA SILVA et al, 2022).

1.3 - Metodologia

Para a realização dos experimentos será utilizada a linguagem Java 11.0.16 em um sistema operacional Linux 64 bits, executado em uma máquina com o processador i7-1165G7, com 16G de memória RAM, para a coleta de dados do iSPD para análise. Os algoritmos foram executados 20 vezes em cada situação específica, para que a média fosse observada.

1.4 - Estrutura do texto

No capítulo um, é realizada uma introdução ao assunto de *Fog Computing* e os problemas de escalonamento. No capítulo dois é realizada uma revisão bibliográfica com conceitos importantes e amparados por trabalhos relacionados e encontrados na literatura.

No capítulo três o desenvolvimento do trabalho realizado é descrito. No capítulo quatro são apresentados os resultados obtidos pelos experimentos e no capítulo cinco é realizada a conclusão e a reflexão para os próximos trabalhos.

Capítulo 2

Revisão Bibliográfica

2.1 IoT

Nesta seção será abordado o tema sobre IoT e sua relação com *Fog Computing*.

2.1.1 Definição

A tecnologia IoT (*Internet of Things*) permite que qualquer objeto do ambiente seja conectado em uma rede de computadores (KOPETZ; STEINER, 2022), sendo portanto um nome equivocado, visto que não necessariamente o dispositivo esteja conectado na rede pública. Já para Dorsemayne (2015), a definição de IoT é uma infraestrutura interconectada que possibilita a extração de dados do ambiente e da infraestrutura.

2.1.2 Aplicações

A capacidade de conectar qualquer objeto na rede permite que a utilização de sensores para extrair informações de ambientes diversos se torne possível, o que tem grande importância para monitoramento de umidade de solo na agricultura, extrair informações sobre linha de produção na indústria, verificar vibração nas estruturas no setor da construção ou verificar a situação de pacientes na saúde (KHELIFI, 2019), podendo ser utilizado para determinar a localização de um dispositivo, sua temperatura ou bem como controlar IoV (*Internet of Vehicles*), IIoT (*Industrial Internet of Things*) (JAMIL; RAHMAN, 2022). Nesse sentido é possível dizer que uma das características da tecnologia IoT é a possibilidade de controlar dispositivos diferentes de forma autônoma, permitindo uma interoperabilidade que antes não era concebível. Construindo uma oportunidade de solução de problemas que permitem a detecção de problemas de forma automatizada e sua solução também ser autônoma (ELKHODR; SHAHRESTANI; CHEUNG, 2016).

2.1.3 Problemas no IoT

Em sistemas críticos não é desejado ter situações adversas, para isso a cadeia de controle deve ter a possibilidade de comunicação rápida, um exemplo seria uma *smart cities*, tendo que o controle dos veículos é algo de suma importância, uma vez que um erro de comunicação provavelmente resultará em acidente. No caso de um sistema do tamanho de uma cidade, os dispositivos IoT teriam que se comunicar através da internet, gerando grande quantidade de tráfego de dados e aumentando a latência, em consequência de um caminho percorrido para se comunicar com um servidor remoto, necessitando utilizar uma quantidade significativa de recursos de comunicação e energia para realizar o processamento dos dados enviados pelos diversos dispositivos que temos. Ou seja, no paradigma de comunicação em Cloud Computing, a troca de informação tem um custo alto para o ecossistema como um todo, pois a carga de trabalho fica concentrada em uma camada de comunicação específica, que no caso são os data centers remotos (TANG, 2017), causando uma sobrecarga de comunicação.

2.2 Fog Computing

2.2.1 História e definição

A sugestão do modelo de comunicação utilizando *Fog Computing* foi criada em 2009 e popularizada pela Cisco pois consegue reduzir significativamente a latência de comunicação (WANG; GUO, 2018) e fazer com que seja possível administrar sistemas mais complexos. A ideia do novo formato de comunicação é fundamentada em processar dados em diferentes locais, sendo quatro camadas com responsabilidades diferentes. A primeira camada é a dos *data centers*, onde a informação será coletada da camada dois de comunicação, que é um nó intermediário e distribuído (é a camada *Fog*) se conecta de centenas a milhares de dispositivos, que por sua vez se conecta na camada três de comunicação, obtendo acesso aos dispositivos IoT (DIN, 2018). Este modelo de comunicação pode ser alterado e adicionar funcionalidades, é o caso das *smart cities*, que podem adicionar uma camada específica para análise de dados para detectar problemas e também para reduzir a quantidade de informações trocadas com o data center (TANG, 2017).

A camada *Fog* deve estar a exatamente um nó (hop) de distância da camada de dispositivos móveis, isso faz com que a camada se comporte como um *Data Center*, mas fisicamente mais próxima dos dispositivos finais, fazendo com que a latência de comunicação torne a interoperabilidade do sistema possível (SATYANARAYANAN et al, 2009). A Cisco vê a camada *Fog* como um sistema que oferece recursos como processamento, armazenamento e rede, tendo suporte a virtualização e multi-tenancy, que possibilita que diversos usuários utilizem o mesmo sistema estando individualmente credenciados (BONOMI, 2014). Para Vaquero, Rodero-Merino (2014) a camada é vista como uma grande coleção de dispositivos heterogêneos e descentralizados, que se comunicam entre si, armazenam dados e realizam processamento que podem ser utilizados para garantir o funcionamento de aplicações locais ou funcionamento básico da rede.

Portanto, é possível afirmar que o conceito básico de *Fog Computing* é a existência de uma camada intermediária na topologia da rede e que esta tem responsabilidades de controle, processamento e armazenamento, realizando uma função de reduzir a quantidade de informação, porém o projeto pode ser adaptado para as necessidades, criando mais camadas com outras funções e complementando a *Fog*.

2.2.2 Diferenças entre Cloud, Edge e Fog

Uma visão mais abstrata de *Fog* permite-nos deduzir que se trata de um intermediário na comunicação entre *Cloud* (Computação em Nuvem) e *Edge* (dispositivos de borda).

Para as características de recursos temos que a principal diferença é sobre a rede, visto que a camada *Fog* está mais próxima dos dispositivos de borda e também dos ambiente de *Cloud*, pois está localizada entre ambos, resultando em menos latência para os dois lados (SATYANARAYANAN et al, 2009). É necessário também que a camada *Fog* tenha acesso a uma conexão com alta capacidade de transmissão de dados (STOJMENOVIC, 2014), pois é necessário se comunicar com o *Data Center*, em contraposição a comunicação com os dispositivos de borda acabarão por ter acesso a uma largura de banda robusta (LUAN et al, 2015). Além disso, a camada *Fog* tem uma capacidade de reter informações e lê-las do

mesmo local de armazenamento por mais tempo (*data longevity*), mas tem menos capacidade de processamento do que o modelo em *Cloud*, sendo assim uma utilidade importante da camada *Fog* é a possibilidade de virtualização de sistemas e criação de cópias instantâneas de volumes, permitindo isolamento de aplicações do sistema (BITTENCOURT et al, 2015).

A camada *Fog* funciona também como um tipo de rede de entrega de conteúdo (CDN - *Content Distribution Network*) em ambas as direções (VAQUERO; RODERO-MERINO, 2014), uma vez que permite que os dispositivos de borda possam realizar o envio e também requisitar informações para a primeira camada (*Cloud*), salientando que por estar mais próximo geograficamente do que a *Cloud*, pode servir os dispositivos de forma mais rápida. A parte do envio é algo a se considerar, pois a camada *Fog* primeiro recebe os dados e posteriormente os envia para a camada de *Cloud*, fazendo com que o armazenamento funcione em camadas, permitindo que uma cópia esteja geograficamente perto e outras cópias espalhadas por servidores em locais geográficos diferentes e distantes uns dos outros. Além disso, antes de enviar as informações a camada pode ser responsável por realizar algum tipo de pré-processamento, por exemplo deduplicação (SATYANARAYANAN et al, 2015) e também permitindo que outros dispositivos de borda tenham acesso aos dados de outros dispositivos (LUAN et al, 2015).

Nesse contexto é possível deduzir que o número de *Fogs* irá impactar o número de servidores em *Cloud* necessários, mas surge um problema de gestão, pois o número de dispositivos de borda ainda pode chegar a bilhões. Mas gerir a camada *Fog* ainda é mais fácil do que a dos dispositivos de borda (LUAN et al, 2015).

As diferenças geográficas também são interessantes, enquanto os ambientes de nuvem são distribuídos geograficamente e com recursos centralizados, adotando modelos públicos, privados ou utilização por assinatura (LUAN et al, 2015). Enquanto isso, os dispositivos de borda e *Fog* ficam distribuídos (no caso deste último, serve para ficar próximo ao anterior a este), mas *Fog* não fica disperso como os dispositivos de borda.

2.2.2 Aplicações

A aplicação de modelo de comunicação é utilizada para controle de plantas de fábrica, sendo importante conceito para a indústria 4.0 (LEE; KAO; YANG, 2014). Por ter a característica de interoperabilidade, esse ambiente tem grande capacidade de sincronização de tarefas e por consequência, possibilidade de reduzir ao máximo o tempo de produção. Nas *smart cities*, a redução do tempo é contabilizada no fluxo de veículos, uma vez que podem andar mais próximos e adequar o a quantidade de veículos nos caminhos possíveis para o destino (HUANG; CHOO, 2017), além de poder colaborar com a capacidade de monitoramento urbano, com câmeras e outros sensores, algo que foi proposto de ser realizado em mais de 100 cidades da Índia (AMRUTUR et al, 2017). Outra aplicação possível é a implantação de sistemas de distribuição energética inteligentes, como o que foi implementado em Los Angeles (AMAN et al, 2015), sendo o maior projeto de utilidade pública da época nos EUA.

2.2.3 Problemas

Após os problemas de latência na comunicação entre dispositivos serem superados com a criação da camada *Fog*, um dos desafios emergentes foi o escalonamento de tarefas para os participantes da rede. Uma vez que os dispositivos que integram a camada intermediária são diversos e heterogêneos, não é ideal realizar a distribuição igualitária de tarefas para os integrantes da rede, pois o tempo de processamento seria demasiadamente dessincronizado, causando sobrecargas em determinados pontos. Para realizar o escalonamento existem algoritmos que realizam a distribuição de tarefas dependendo do tipo de ambiente utilizado (MATROUK; ALATOUN, 2021).

2.3 Soluções para escalonamento

2.3.1 Classificação de problemas

Um problema pode ser classificados em quatro categorias, P, NP, NP-completo e NP-difícil. A classe de problemas P são por exemplo a equação quadrática que pode ser resolvida usando um algoritmo polinomial, no caso do NP é possível realizar um algoritmo que constrói uma solução candidata e é possível validar em tempo polinomial. Nesse cenário devemos observar o problema de escalonamento de *Fog Computing* como problemas NP-Completo, mesmo quando se trata de apenas dois nós (YAN et al, 2021).

2.3.2 Algoritmos de escalonamento

A prática de escalonar algo é muito comum no nosso cotidiano, trata-se da distribuição de tarefas de modo que seja o mais otimizado possível, portanto é possível encontrar diversos problemas diferentes com soluções diferentes que dependem do objetivo. Sendo assim, um escalonamento pode ser realizado a partir da abstração da solução de um problema, sendo que a interpretação das métricas é subjetiva ao desejado, inexistindo uma solução fechada (BŁAŻEWICZ, 2001).

Um exemplo de escalonador tradicional é o *Round Robin* (ARPACI-DUSSEAU; ARPACI-DUSSEAU, 2018), o modo de realização é a execução da tarefa por um determinado tempo e realiza a mudança de fila de execução, não completando uma tarefa inteira de cada vez, realizando a intercalação entre filas de execução.

Para problemas de *Fog Computing* um dos algoritmos de escalonamento que pode ser utilizado é o PSO (*Particle Swarm Optimization*), seu funcionamento trata as soluções como se fossem partículas, sendo que estas se movimentam em um espaço determinado com uma fórmula que define seu deslocamento e as partículas são guiadas por sua melhor

posição naquele momento no espaço amostral e então o processo é repetido, porém não garante uma solução satisfatória, sendo necessário avaliar os ajustes necessários (KENNEDY; EBERHART, 1995). Este tipo de algoritmo é chamado de meta-heurístico, pois se propõe a encontrar um algoritmo ou parte de um, que funcione para realizar uma otimização que não seja necessariamente ótima, mas que seja uma solução possível de ser utilizada. Este tipo de resolução é empregada em problemas NP-completos, que não tem solução conhecida fechada (BIANCHI, 2009).

2.4 Simulador

2.4.1 iSPD

O iSPD é um simulador que possibilita a utilização de interface gráfica para criar modelos de sistemas distribuídos utilizando ícones, o que permite que não seja necessário o desenvolvimento de todo um sistema, além disso, permite que pessoas com pouco conhecimento de programação possam estudar os algoritmos presentes na plataforma (MANACERO et al, 2012). O iSPD pode ser separado em cinco partes:

- **Interface icônica:** O usuário pode interagir com o programa utilizando o mouse para selecionar o tipo de elemento que deseja inserir e o teclado para inserir informações, como por exemplo os parâmetros (quantidade de núcleos, memória, largura de banda conexão, latência e afins)
- **Interpretador de modelos internos:** transforma o modelo icônico em um modelo em que é possível realizar uma simulação, gerido pelo motor da plataforma
- **Interpretador de modelos externos:** Possibilita a importação de modelos que foram criados por outros simuladores, como por exemplo o GridSim (BUYAYA, MURSHED, 2002) e o SimGrid (CASANOVA, 2001).
- **Gerador e gerenciador de escalonadores:** módulo que permite o gerenciamento de escalonadores e criação de novas políticas de escalonamento utilizando algoritmos Java ou fórmulas matemáticas.

Motor de simulação: Realiza a função de simular os eventos discretos definidos pelo usuário e apresenta as métricas.

Uma das características fundamentais no processo de extração de dados do iSPD, é a possibilidade de executá-lo por linha de comando, possibilitando determinar quantas execuções do algoritmo devem ser realizadas e a opção de executá-lo de forma paralela, acelerando a conclusão do experimento.

2.4.2 iFogSim

O iFogSim é um simulador de redes *Fog*, que permite a criação de diversas configurações e simulação de situações especificadas (BUYA; SRIRAMA, 2019). Todas as etapas podem ser realizadas utilizando-se também de uma interface icônica. Sua utilização permite especificar a latência e tempo de processamento por dispositivo.

2.4 Considerações Finais

Com as informações apresentadas se torna possível ter embasamento teórico para realizar a busca da solução do problema de escalonamento de tarefas. A existência de simuladores nos permite avaliar sistemas complexos sem a necessidade de construí-los fisicamente, poupando recursos. Foi tratado sobre *Fog Computing*, as diferenças entre *Fog*, ambientes em nuvem e *Edge*, passando posteriormente sobre as possibilidades de aplicação e seus problemas, sendo que alguns podem ser solucionados com a utilização de escalonadores.

Capítulo 3

Desenvolvimento

Neste capítulo é apresentado os algoritmos LBACO (*Load Balancing Ant Colony Optimization*), ACO (*Ant Colony Optimization*), ABC (*Artificial Bee Colony Optimization*) e PSO (*Particle Swarm Optimization*), utilizados em ambientes de *Fog Computing* para a realização de escalonamento de tarefas utilizando meta-heurística. Os parâmetros passados para os algoritmos podem ser alterados, realizando extrações de informações e analisando o ambiente definindo a importância das informações.

O iSPD no caso nos fornece diversas informações sobre o ambiente, neste experimento foi utilizado as informações de tempo de execução e largura de banda, que podem ser recuperadas em tempo de execução do simulador.

O iSPD calcula o tempo de execução com a fórmula, utilizando as informações de poder computacional disponível por processador (PCDPP), poder computacional (PC), fator de carregamento (FDC) e número de processadores (NP) :

$$PCDPP = (PC - PC * FDC) / NP$$

O tempo de processamento (TP) pode ser obtido pelo tamanho da tarefa em Mflops dividido pelo PCDPP:

$$TP = Mflops / PCDPP$$

A eficiência no iSPD é medida pela capacidade do sistema (CS) dividido pela multiplicação do tamanho da tarefa (TT) pelo tempo de processamento (TP):

$$CS = TT / TP$$

3.1 Colônia de formigas

O algoritmo de colônia de formigas pode ser utilizado para resolver problemas NP-completos, visto que utiliza uma solução probabilística, empregando a lógica do funcionamento de um formigueiro, possibilitando a busca por uma solução possível de ser utilizada, sendo que as possibilidades são percorridas pelas formigas que liberam feromônios em seu caminho (DORIGO, 1992).

O funcionamento do algoritmo é para cada iteração é alterada uma formiga que escolhe um trabalho e um agente se propõe a realizar este trabalho. Caso a solução seja infactível, a formiga é reinicializada, caso contrário ocorre o decaimento do feromônio e verifica-se se todos os trabalhos foram alocados, caso a solução seja a melhor até agora é decaído todos os feromônios de todas as arestas, se não é aumentado o feromônio da aresta corrente e então a formiga é reinicializada (DORIGO; BIRATTARI; STUTZLE, 2006).

O ACO segue sendo um algoritmo que possibilita a resolução de problemas complexos na atualidade, mesmo sendo criado no século passado, para conseguir este feito, o algoritmo é alterado ou combinado com outros algoritmos, uma das alterações interessantes no nosso caso é a que resultou na implementação do LBACO, que será apresentado no decorrer do trabalho.

3.1.1 Ant Colony Optimization

O algoritmo ACO (DORIGO; BIRATTARI; STUTZLE, 2006) é um recorrentemente utilizado quando se trata de problemas relacionados a problemas NP-Completo, sua meta-heurística se baseia no cálculo da probabilidade de algo desejado, por exemplo economizar energia, executar tarefas mais rápido, ter uma latência menor ou algum outro parâmetro do ambiente. Por vezes as alterações do algoritmo ACO original, podem introduzir elementos no cálculo, porém essencialmente o algoritmo tem o parâmetro que corresponde a quantidade de feromônio depositado, a quantidade de formigas utilizadas, a taxa de evaporação do feromônio, *alpha* que corresponde ao peso matemático do feromônio na probabilidade e *beta* que considera o peso decisório do parâmetro que é observado.

Neste experimento, foi escolhido utilizar o tempo de execução como fator determinante para a abordagem do ACO. Portanto utiliza-se o poder computacional para definir o feromônio das formigas. Existindo apenas uma tabela de registro de tarefas e tempo execução correspondente por máquina.

Após extrair todos os dados necessários do ambiente de simulação, que no caso corresponde às tarefas que serão realizadas e seu tempo de execução, deve-se calcular a probabilidade da formiga escolher a máquina virtual em questão. Para isso devemos levar em consideração apenas o tempo de execução neste caso, que no algoritmo ACO seria equivalente ao tamanho do alimento (ASGHARI; NAVIMIPOUR, 2019).

```

Inicialização:
- Parâmetros: m (número de votos), Q (feromônio inicial), alpha, beta, rho
- Criação de uma população inicial de feromônios para cada máquina
- Criação de uma lista de tarefas e uma lista de máquinas

Para cada iteração até atingir o critério de parada:
  Para cada tarefa:
    Calcular os tempos de execução da tarefa em cada máquina

  Para cada máquina:
    Calcular o tempo de execução

  Para cada iteração t até um número máximo de iterações:
    Inicializar o tempo de execução total por máquina para zero

    Para cada tarefa:
      Calcular as probabilidades de seleção da máquina para a tarefa
      Realizar a votação para escolher a máquina
      Atualizar a frequência de probabilidade

    Determinar a máquina escolhida com mais frequência da probabilidade

    Atualizar o tempo de execução total da máquina
    Atualizar os feromônios

Retornar o mapeamento final de tarefas alocadas em máquinas

```

Figura 1: Pseudocódigo do ACO (gerado pelo autor)

3.1.2 Load Balancer Ant Colony Optimization

O algoritmo ACO (*Ant Colony Optimization*) por vezes precisa ser adaptado para que se possa ser utilizado em situações distintas, no caso a proposta é realizar um balanceador de carga com o ACO, portanto é necessário utilizar informações do ambiente para que seja possível ter uma análise do algoritmo que possibilite um escalonamento melhor do que os outros algoritmos utilizados. Neste caso, optou-se por utilizar o LBACO (GUPTA; GARG, 2017), a alteração é a criação de mais um parâmetro e a determinação de mais um peso a ser considerado, o *gamma*.

Para que a meta-heurística funcione, deve-se utilizar informações que são

definidoras do problema, no caso a capacidade computacional, que será definida pelo tempo de execução da tarefa somado com a largura de banda da conexão para cada máquina existente no ambiente utilizado (BAXODIRJONOVICH; CHOE, 2015), importante considerar que o poder computacional neste caso será o inicializador do feromônio.

$$PoderComputacional = TempoDeExecução + LarguraDeBanda$$

O tempo de execução é dado em segundos e é resultado da operação:

$$TempoDeExecução = Mflops / PoderComputacionalPorProcessador$$

Outra informação utilizada é também o próprio tempo de execução, após a extração das informações do ambiente, é utilizado uma par para indexar as informações necessárias com as tarefas que serão resolvidas, portanto existem duas tabelas de dados utilizados que serão consideradas por todas as formigas do algoritmo.

$$TempoDeExecuçãoTarefa = [Tarefa, TempoDeExecução]$$

$$PoderComputacionalTarefa = [Tarefa, PoderComputacional]$$

Nesse sentido, o tempo esperado de execução de uma tarefa é o somatório de todos os tempos de execução registrados e deste dado é subtraído o tempo restante de execução da tarefa sendo executada atualmente na máquina em questão.

O próximo passo é termos o balanceamento de carga calculado, para isso deve-se realizar a razão da carga de todo o sistema pela carga da máquina que está sendo avaliada, quanto mais alto o valor, significa que o processamento no dispositivo está menos utilizado do que o normal no ambiente.

De posse do fator de balanceamento de carga, é possível calcular a probabilidade de uma formiga escolher a máquina que está sendo analisada. Para chegar a este resultado, devemos elevar o feromônio ao expoente *alpha*, o poder computacional ao expoente beta e o fator de balanceamento de carga por *gamma*, que correspondem à máquina analisada.

Essa expressão deve ser dividida por sua correspondente para o sistema como um todo e o resultado será a probabilidade da formiga escolher o dispositivo.

A principal diferença entre o ACO e o LBACO, é a criação de outro fator de decisão e peso, no caso o poder computacional e o *gamma*, estes parâmetros podem ser experimentados e pode-se definir sua importância de acordo com as necessidades do ambiente. É interessante notar que a criação de novos fatores de decisão e novas matrizes, podem ser criadas neste algoritmo para que seja possível atender diversos objetivos diferentes, porém a implementação do LBACO, neste tipo de estrutura, faz com que muitos *loops* sejam gerados no código, causando um aumento no tempo de execução.

Porém, a proposta de Gupta (2017) para o LBACO, busca que as formigas realizem uma votação e seja realizada uma decisão pela maioria, que resulta em evitar as soluções locais ótimas, que o algoritmo ACO pode definir como solução. Uma possível saída para este problema de muitos loops, seria a utilização de apenas um parâmetro no LBACO, ao invés de dois parâmetros. Algo semelhante foi realizado no algoritmo SACO (*Smart Ant Colony Optimization*), que realiza a passagem de um parâmetro que normaliza diferentes características do ambiente, possibilitando que o tempo de execução seja menor (KISHOR, 2021).

O pseudocódigo do LBACO pode ser desenvolvido da seguinte forma:

```

Inicialização:
- Parâmetros: m (número de votos), Q (feromônio inicial), alpha, beta,
gamma, rho
- Criação de uma população inicial de feromônios para cada máquina
- Criação de uma lista de tarefas e uma lista de máquinas

Para cada iteração até atingir o critério de parada:
  Para cada tarefa:
    Calcular os tempos de execução da tarefa em cada máquina

  Para cada máquina:
    Calcular o tempo de execução

  Para cada máquina:
    Inicializar os feromônios da máquina com base no poder computacional

  Para cada iteração t até um número máximo de iterações:
    Inicializar o tempo de execução total por máquina para zero

    Para cada tarefa:
      Calcular as probabilidades de seleção da máquina para a tarefa
      Realizar a votação para escolher a máquina
      Atualizar a frequência de probabilidade

    Determinar a máquina escolhida com mais frequência da probabilidade

    Atualizar o tempo de execução total da máquina
    Atualizar os feromônios

Retornar o mapeamento final de tarefas alocadas em máquinas

```

Figura 2: Pseudocódigo do LBACO (criado pelo autor)

3.2 Artificial Bee Colony

O algoritmo ABC (GÖKALP, 2021) é bioinspirado no comportamento das abelhas, é sabido que o comportamento destes insetos se baseia em uma linguagem e a comunicação entre as abelhas é feita por desenhos que são realizados no ar, sendo assim a colmeia é informada de fontes de alimento existentes e localização, portanto o princípio é a imitação da procura por alimento por estes insetos (LUCIC; TEODOROVIC, 2002).

Existem dois tipos de implementações do ABC, é possível realizar considerando que cada abelha irá analisar todas as fontes de alimento disponíveis realizando uma solução parcial (utilizada neste experimento) ou então é pode-se considerar que as abelhas irão se cercar em grupos e se retirar com suas soluções parciais (HAJIMIRZAEI; NAVIMIPOUR, 2019), sendo respectivamente *forward-pass* e *backward-pass*.

Os parâmetros necessários para implementar o algoritmo, são o número de abelhas, que é dividido entre observadoras e trabalhadoras (NB), o número de fontes de alimento (NA), limite superior e inferior para abandonar uma fonte de alimento (Limite), um contador de avaliações da função objetivo, limites superior e inferior (lb e ub, sendo 0 e o número de máquinas neste contexto).

O pseudo-código do ABC pode ser representado como:

0. Determinação dos parâmetros
1. inicialização dos valores das fontes de comida
2. Armazenamento do melhor caminho
3. Enquanto o número de execuções for menor do que o determinado:
 - Enviar abelhas operárias
 - Calcular probabilidades
 - Enviar abelhas observadoras
 - Armazenamento do melhor caminho
 - Enviar abelhas escoteiras
4. Discretizar o resultado
5. Retorna a solução

Figura 3: Pseudocódigo do ABC (criado pelo autor)

O melhor caminho é determinado pelo mínimo global da distância até a fonte de comida, que são as máquinas que estão sendo utilizadas no contexto, sempre que uma solução é encontrada, o melhor caminho é armazenado e a determinação é realizada utilizando uma função de predição do tempo de resposta, que calcula o tempo máximo de resposta de uma máquina utilizando o tempo de execução da tarefa. Com essa informação é executada a função de aptidão, que determina a qualidade do resultado, caso a qualidade do resultado seja melhor do que a anterior, então é armazenada para as próximas execuções.

3.3 Particle Swarm Optimization

O PSO (KENNEDY; EBERHART, 1995) tem a proposta de criar um conjunto de partículas aleatórias, que se tratam de soluções potenciais e então as partículas se movem pelo espaço com uma velocidade definida, temos para cada iteração uma alteração da velocidade, levando em consideração o melhor caso da partícula e de todo o conjunto. A criação do algoritmo se baseia na observação do comportamento social de diversas espécies, inclusive a humana (SERAPIÃO, 2009).

Cada partícula tem a capacidade de influenciar todas as partículas e o conjunto total também pode influenciar as partículas individualmente, nesse caso há o parâmetro que define uma ligação entre todos os elementos existentes (*global best - gB*), uma segunda métrica cria uma relação entre a partícula e as de sua vizinhança (*position best - pB*), outra informação definida é a a cognição e social, sendo ϕ_1 e ϕ_2 . As definições do PSO também abarcam a idéia de inércia e limitação da velocidade, para que as partículas não saiam do domínio.

0. Determinar a quantidade de partículas P da população.
1. Inicializar a posição inicial (x) de cada partícula p de P aleatoriamente.
2. Atribua uma velocidade inicial (v) igual para todas as partículas.
3. Para cada partícula p em P faça:
 - (a) Calcule sua aptidão $f_p = f(p)$.
 - (b) Calcule e melhor posição da partícula até o momento (pB).
4. Descubra a partícula com a melhor aptidão de toda a população (gB).
5. Para cada partícula p em P faça:
 - (a) Atualizar a velocidade da partícula pela fórmula:

$$v = v + \phi_1 \times (pB - x) + \phi_2 \times (gB - x).$$
 - (b) Atualizar a posição da partícula pela fórmula:

$$x = x + v.$$
6. Caso a condição de finalização não for realizada, retorne ao passo 4.

Figura 4: Pseudo código PSO (criado pelo autor)

O algoritmo do PSO também utiliza uma função de aptidão para verificar a qualidade do resultado encontrado e compara com o próximo resultado que será encontrado, tentando encontrar o mínimo tempo de execução entre todos os resultados gerados aleatoriamente, que no caso são as partículas.

3.4 Características dos ambientes

Fog Computing tem um detalhe inerente no centro da questão a ser resolvida, os dispositivos são todos diferentes em si, podendo ter capacidade de processamento diferente e capacidade de comunicação não correspondente. A proposta se baseia em dois tipos de ambientes, onde o ACO, LBACO, ABC e PSO podem ser utilizados em contextos diferentes. No primeiro caso, é simulado um ambiente que poderia ser uma indústria que tem um ambiente muito controlado e que por isso varia apenas a capacidade de processamento, sendo que a capacidade de comunicação entre os dispositivos é igual. Para o ambiente do ACO, as máquinas variam de 10000 a 40000 Mflops de capacidade de processamento e sua capacidade de comunicação é padronizada em 1000 Mbits.

As máquinas criadas com capacidade de processamento aleatorizado, criando um ambiente condizente com a situação de uma camada de *Fog Computing*, é possível notar que quase todas as máquinas, geradas aleatoriamente por um código implementado em *Bash Script*, tem características diferentes entre si, mas mantendo a média (*Mean*) próxima.

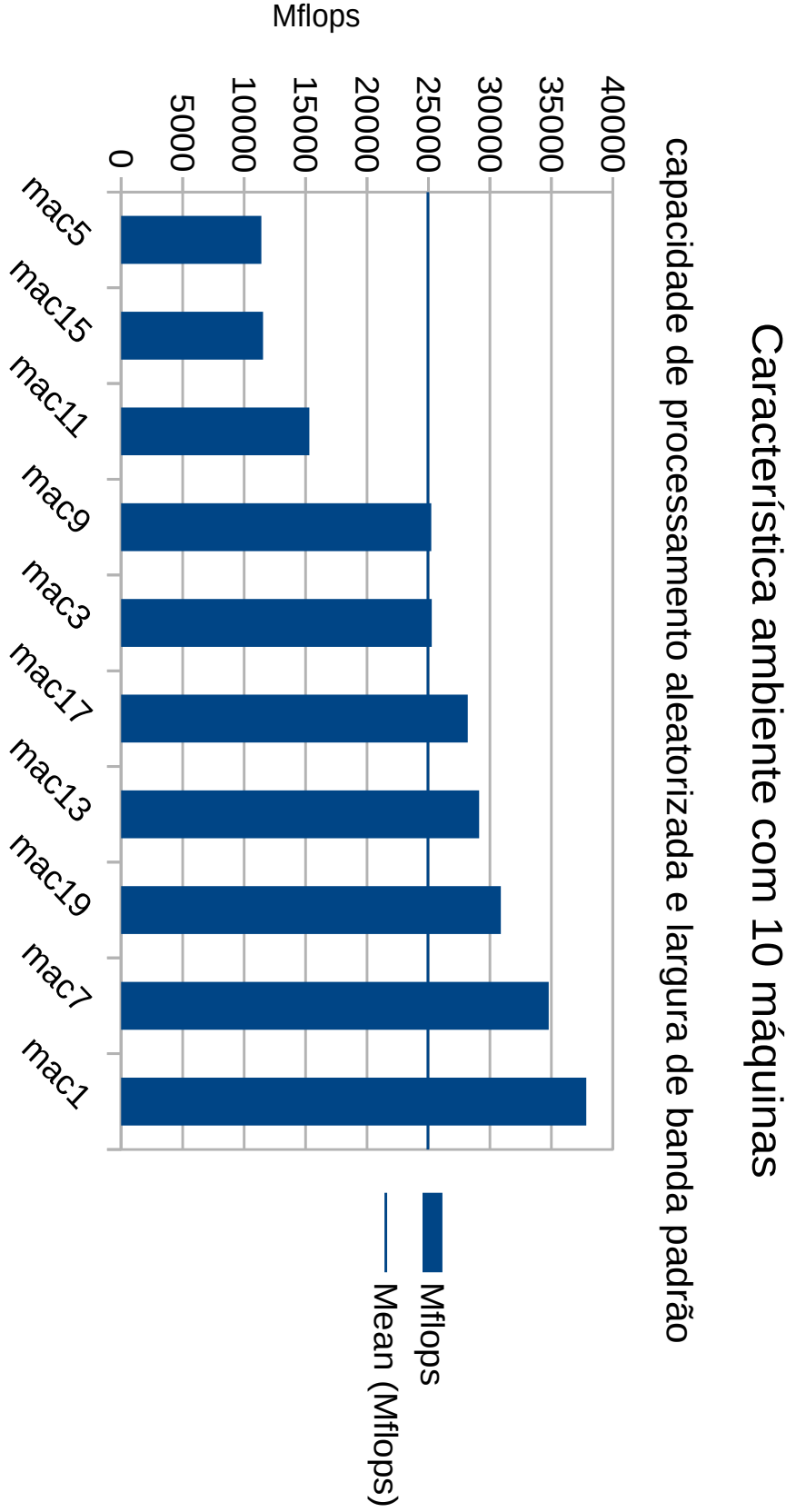


Figura 5: 10 máquinas com capacidade de processamento aleatorizado (criado pelo autor)

O ambiente com cinquenta máquinas conta com a aleatorização da capacidade de processamento, tendo uma média de processamento próxima a de 10 máquinas.

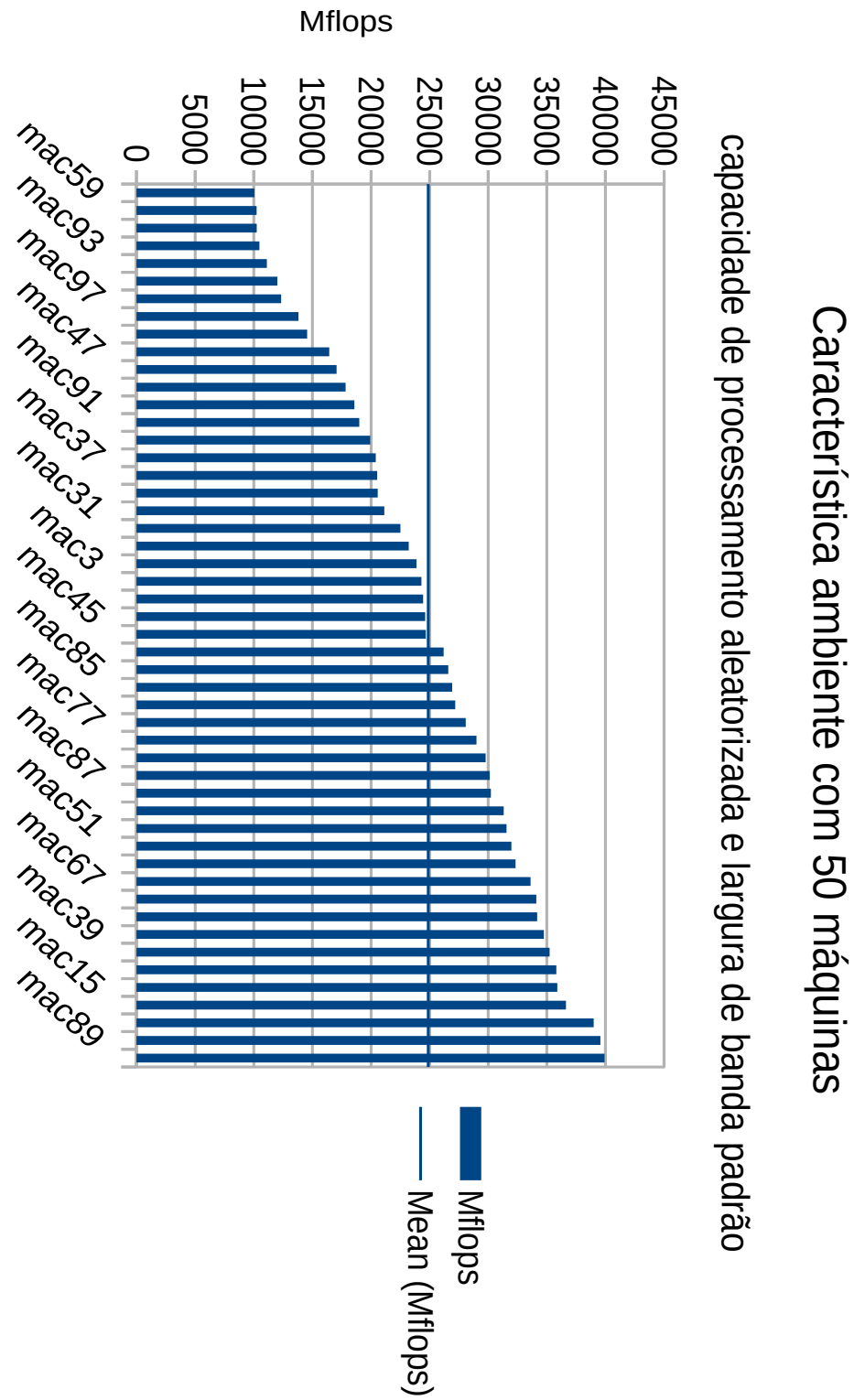


Figura 6: 50 máquinas com capacidade aleatorizada (criado pelo autor)

Características do ambiente com 100 máquinas

capacidade de processamento aleatorizada e largura de banda padrão

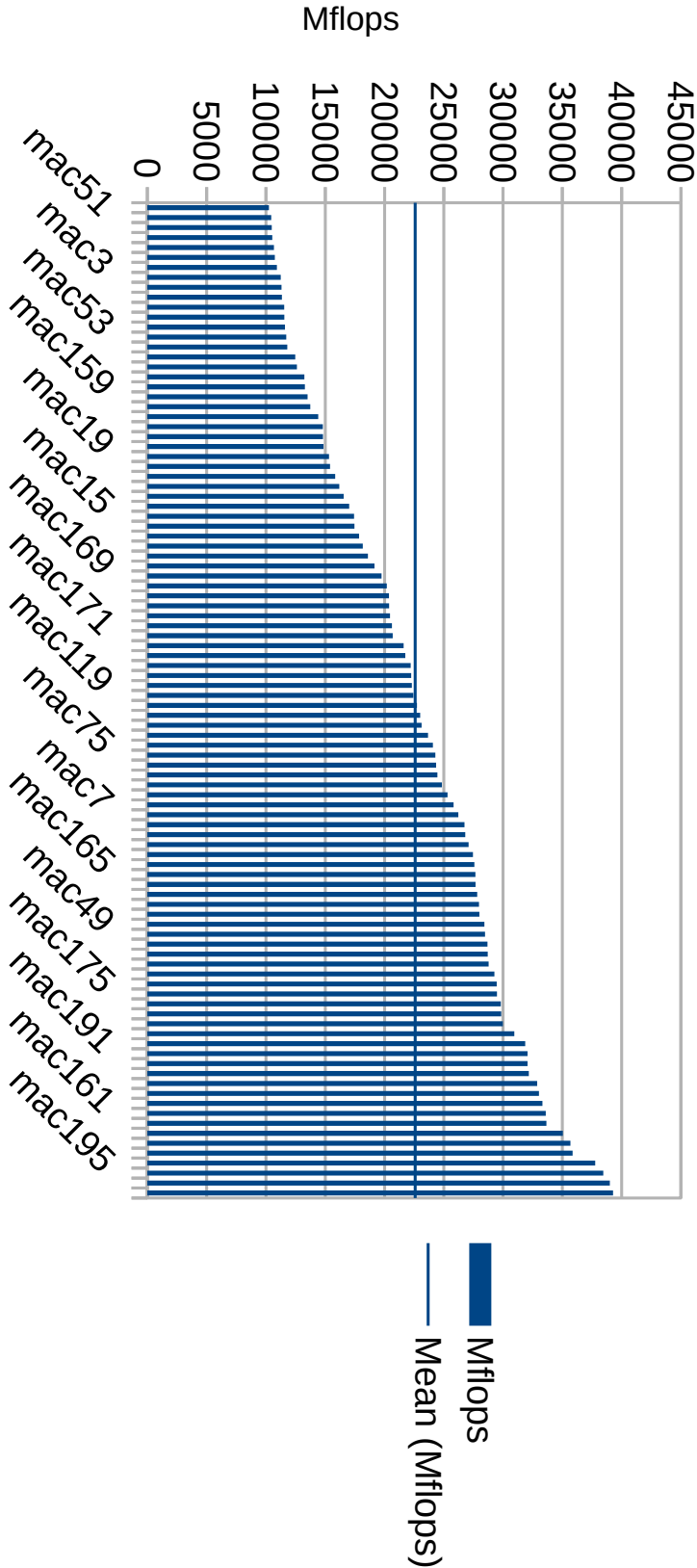


Figura 7: 100 máquinas com processamento aleatorizado (criado pelo autor)

No caso do ambiente que tem uma característica mais adversa, uma variação na capacidade de comunicação é aplicada, sendo que a largura de banda varia entre 1000 e 10000 Mbits, sendo a comunicação em um cenário como este um peso importante.

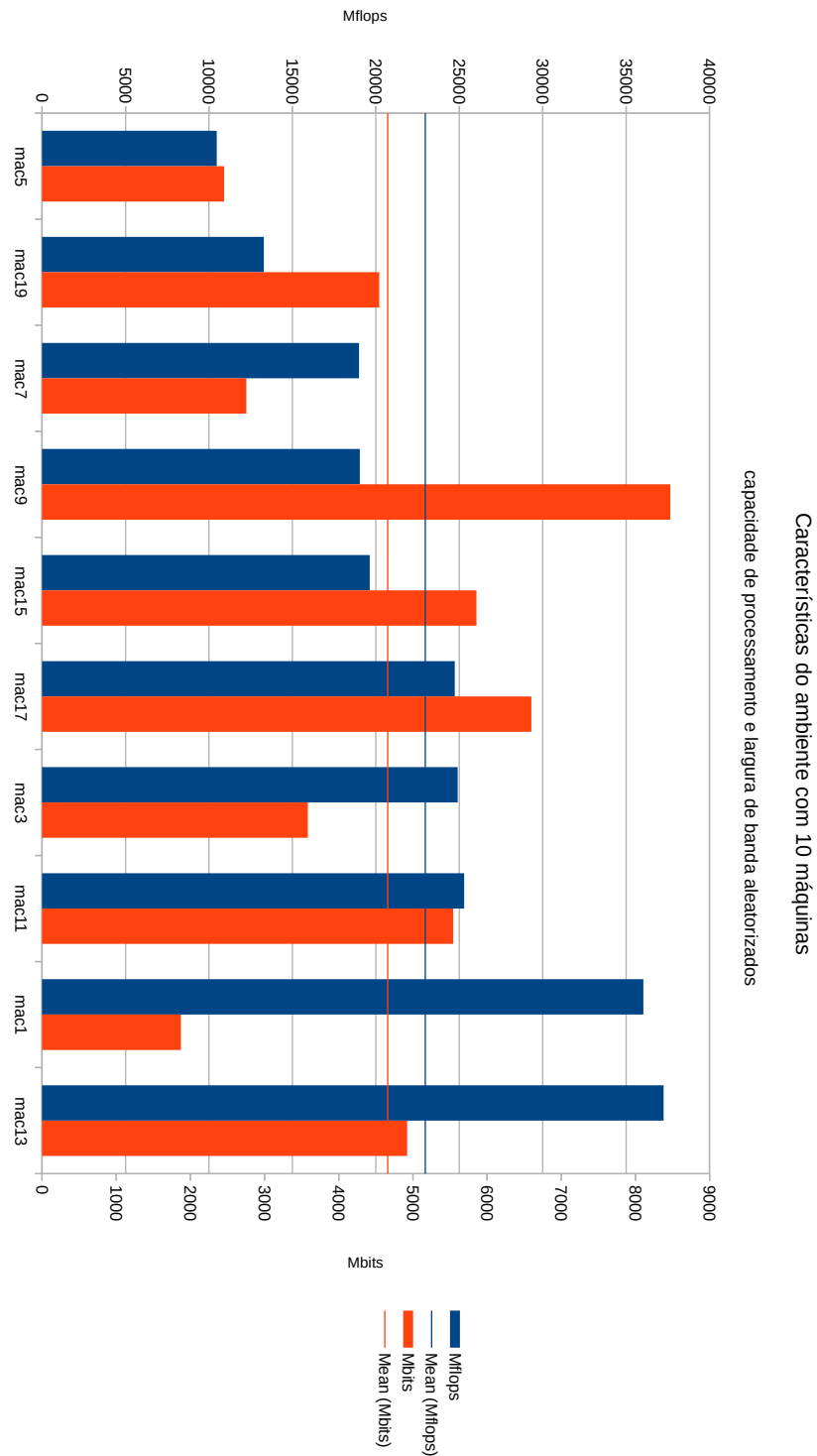
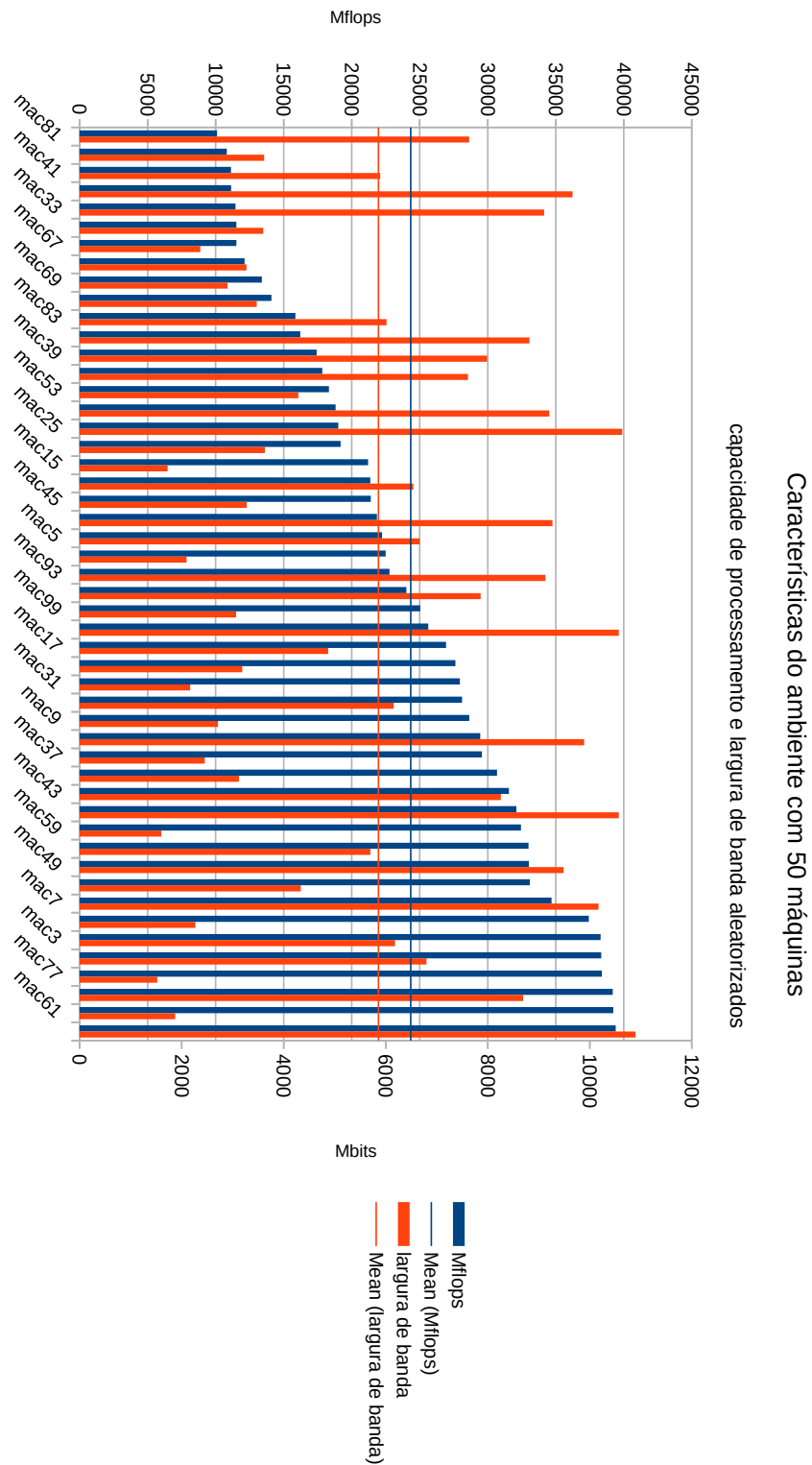


Figura 8: 10 máquinas com capacidade de processamento e largura de banda aleatorizados (criado pelo autor)

Figura 9: 50 máquinas com capacidade de processamento e largura de banda



Em ambos os casos a latência não é variada, pois para Fog Computing em ambientes críticos a proposta é que a latência seja algo que tenha no máximo 0.01s, tornando viável a existência de interoperabilidade do sistema crítico utilizado.

A carga das tarefas utilizadas variam entre 5000, 7500 e 1000 Mflops, condizente com ambientes de *Fog Computing*, pois o processamento de dados é oriundo de sensores e o tempo de chegada é de 1 segundo.

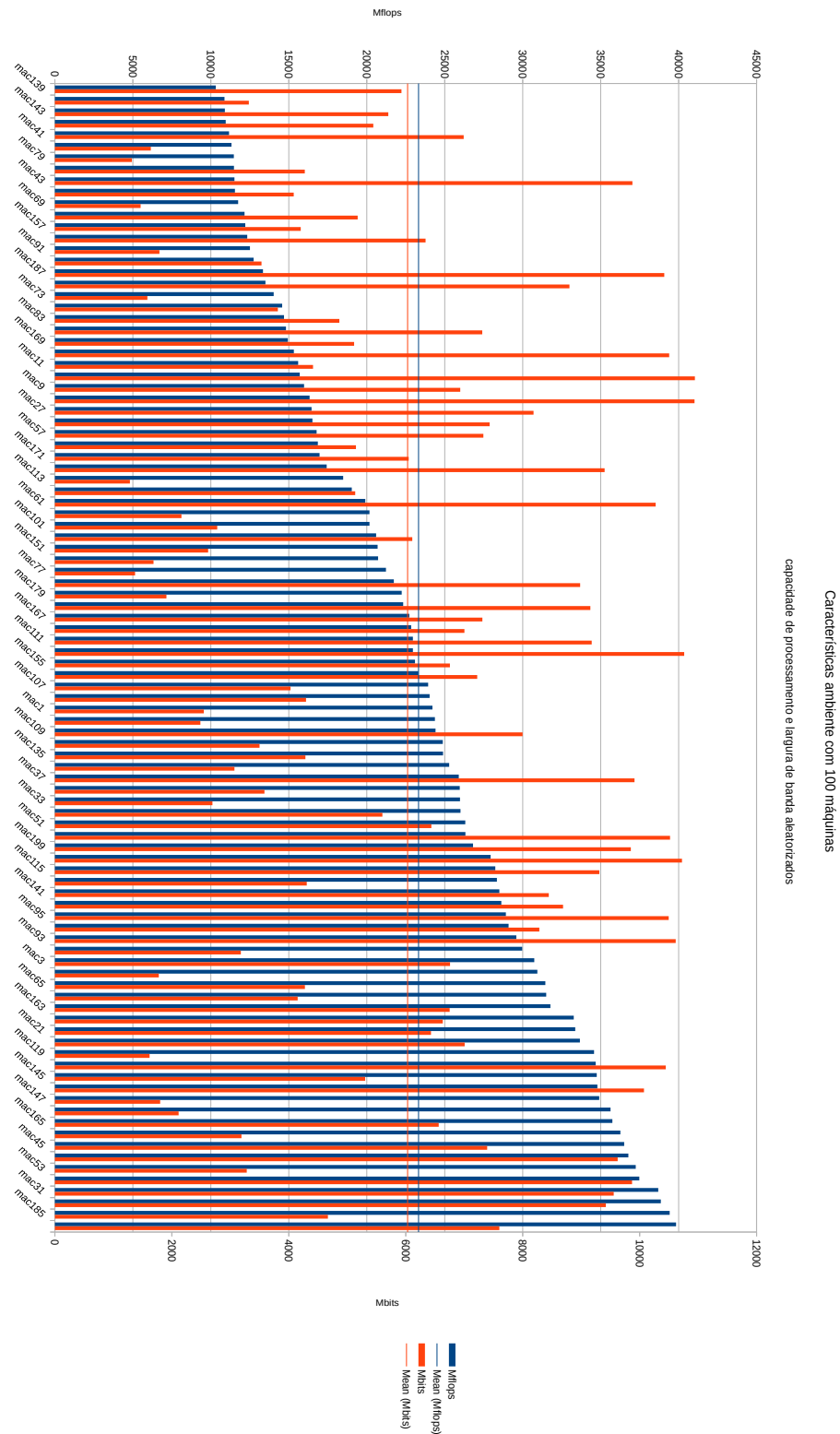


Figura 10: 100 máquinas com capacidade de processamento e largura de banda aleatorizado (criado pelo autor)

Para os experimentos foram utilizados ambientes com 10, 50 e 100 dispositivos, com uma quantidade de tarefas de 100, 1000 e 100000.

Para o LBACO, os valores de quantidade de formigas, quantidade de feromônio depositado, *alpha*, *beta*, *gamma* e *rho*, foram respectivamente 40, 1, 3, 2, 8 e 0.01. Para o ACO os parâmetros são os mesmos, com exceção do *gamma* que não está presente no cálculo.

Os parâmetros definidos no ABC correspondem a 80 no tamanho da colmeia, o limite para abandonar uma fonte de comida foi definida em 50 e o número máximo de iterações em 100. As definições no PSO correspondem a 100 para o tamanho do enxame, 1000 para o número máximo de iterações, o coeficiente social foi determinado em 2, a inércia em 0.9 e cognitivo em 2.

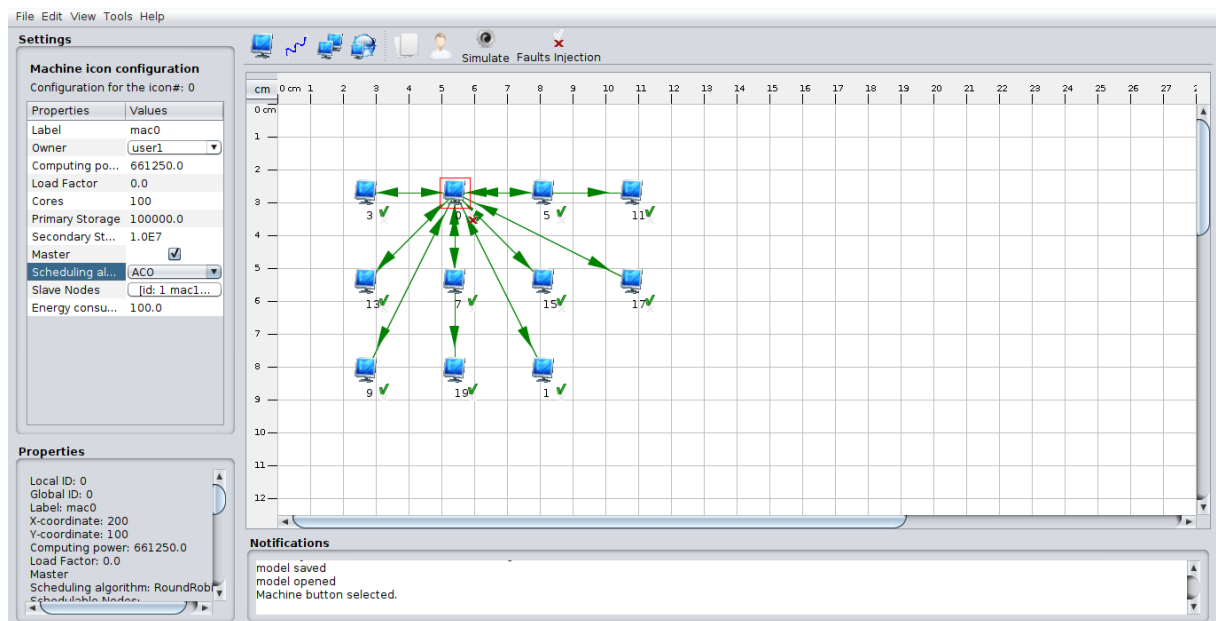


Figura 11: iSPD com 10 máquinas (criado pelo autor)

Capítulo 4

Resultados

Neste espaço será apresentado os resultados dos experimentos realizados com os algoritmos ACO, LBACO, ABC e PSO.

4.1.1 LBACO

O LBACO (Load Balancing Ant Colony Optimization) tem a proposta de utilizar a largura de banda e a capacidade de processamento para avaliar a possibilidade de escalonamento, em redes homogêneas, o algoritmo teve um bom desempenho. Isso se deve ao método de decisão para a largura de banda, que utiliza também a capacidade de processamento, fazendo com que em um ambiente homogêneo seu desempenho seja sensivelmente melhor.

Tabela 1: Eficiência média (%) LBACO em ambiente com capacidade de processamento heterogêneo e largura de banda homogênea

Número de tarefas	10 máquinas	50 máquinas	100 máquinas
100	24,73	28,47	28,35
1000	24,73	28,6	28,46
10000	24,77	28,67	28,44
100000	24,79	28,68	28,44

Já em ambientes com heterogeneidade, podemos observar que o desempenho é cerca de 1% pior, mas mantêm consistência no resultado.

Tabela 2: Eficiência média (%) LBACO em ambiente com capacidade de processamento heterogêneo e largura de banda heterogênea

Número de tarefas	10 máquinas	50 máquinas	100 máquinas
100	25,41	27,48	27,47
1000	25,63	28,15	27,63
10000	25,49	28,12	27,49
100000	25,5	28,13	27,5

Tabela 3: Tempo médio simulação (segundos) LBACO em ambiente com capacidade de processamento heterogêneo e largura de banda homogênea

Número de tarefas	10 máquinas	50 máquinas	100 máquinas
100	129,06	65,87	65,1
1000	1206,92	588,5	577,68

10000	11886,4	5769,46	5606,03
100000	117989	57730,5	56013,3

Na tabela 3 podemos observar que o tempo de simulação é significativamente maior do que na tabela 4, isso ocorre por causa da capacidade de comunicação dos ambientes heterogêneos, que é mais elevada.

Tabela 4: Tempo médio simulação (segundos) LBACO em ambiente com capacidade de processamento heterogêneo e largura de banda heterogênea

Número de tarefas	10 máquinas	50 máquinas	100 máquinas
100	78,93	23,23	8,92
1000	800,63	214,05	76,91
10000	8057,96	2124,45	775,59
100000	80669,8	21411,9	7709,28

Na tabela 5 podemos verificar o tempo de execução do LBACO em ambientes com largura de banda homogênea, o algoritmo não tem dificuldade de determinar a melhor máquina neste ambiente, pois todas as máquinas tem a mesma capacidade de comunicação, fazendo com que a execução do código de escalonamento seja também mais rápida.

Tabela 5: Tempo médio execução (segundos) LBACO em ambiente com capacidade de processamento heterogêneo e largura de banda homogênea

Número de tarefas	10 máquinas	50 máquinas	100 máquinas
100	0,06	0,51	1,05
1000	0,56	5,01	9,56
10000	7,1	49,28	90,08
100000	136,23	298,73	761,59

Em relação a tabela 5, a tabela 6 mostra um tempo de execução maior, isso ocorre justamente pela maior dificuldade de determinar qual máquina é a melhor, visto que o ambiente é mais adverso.

Tabela 6: Tempo médio execução (segundos) LBACO em ambiente com capacidade de processamento heterogêneo e largura de banda heterogênea

Número de tarefas	10 máquinas	50 máquinas	100 máquinas
100	0,06	0,57	1,04
1000	0,66	4,71	8,72
10000	11,27	47,07	82
100000	115,12	407,72	999,11

4.1.2 ACO

A tabela 7 demonstra a eficiência do ACO em um ambiente homogêneo, é possível notar que a eficiência aumenta quando se tem mais máquinas, a diferença em relação a tabela 1, mostra uma diferença significativa ao LBACO, que tem uma capacidade melhor de definir qual a melhor máquina. Isso se deve aos parâmetros utilizados, pois nesse contexto o LBACO tem mais peso no processamento da máquina, pelo ambiente ser homogêneo na largura de banda. Isso pode ser observado pelo fato de que a eficiência com 100 máquinas é próxima do LBACO.

Tabela 7: Eficiência média (%) ACO em ambiente com capacidade de processamento heterogêneo e largura de banda homogênea

Número de tarefas	10 máquinas	50 máquinas	100 máquinas
100	22,11	26,66	26,57
1000	22,09	27,12	26,37
10000	22,25	26,92	26,32
100000	22,23	26,9	26,29

Os resultados não mudam significativamente no quesito eficiência, mas pode-se notar uma inferioridade do ACO no ambiente heterogêneo em relação ao LBACO.

Tabela 8: Eficiência média (%) ACO em ambiente com capacidade de processamento heterogêneo e largura de banda heterogênea

Número de tarefas	10 máquinas	50 máquinas	100 máquinas
100	21,74	26,66	26,86
1000	22,08	27,12	26,18
10000	22,16	26,92	26,31
100000	22,14	26,94	26,39

O tempo total da simulação do ACO é influenciado também pelo código executado, uma vez que o ACO tem menos *loops* do que o LBACO, seu tempo de simulação é menor.

Tabela 9: Tempo total simulação (segundos) ACO em ambiente com capacidade de processamento heterogêneo e largura de banda homogênea

Número de tarefas	10 máquinas	50 máquinas	100 máquinas
100	95,64	51,87	49,32
1000	907,73	428,82	414,39
10000	9131,03	4347,1	3943,44
100000	91104,3	43733,5	39319,4

Porém o ACO não consegue ter a mesma performance em um ambiente com largura de banda heterogênea, tendo um resultado cerca de dez vezes pior do que o LBACO.

Tabela 10: Tempo total simulação (segundos) ACO em ambiente com capacidade de processamento heterogêneo e largura de banda heterogênea

Número de tarefas	10 máquinas	50 máquinas	100 máquinas
100	62,02	23,48	8,66
1000	627,92	212,58	72,38
10000	6331,87	2116,72	683,74
100000	63162,9	21194,9	63901,5

O tempo de execução do ACO também é diretamente relacionado a quantidade de *loops* que o código tem, mas também tem relação com sua capacidade de identificar o melhor cenário no tempo de convergência para a solução.

Tabela 11: Tempo médio de execução (segundos) ACO em ambiente com capacidade de processamento heterogêneo e largura de banda homogênea

Número de tarefas	10 máquinas	50 máquinas	100 máquinas
100	0,04	0,61	1,05
1000	0,32	4,23	9,43
10000	3,51	33,3	93,64
100000	35,38	338,05	725,36

Essa dificuldade do algoritmo pode ser observada em seu tempo de execução que tem um aumento significativo quando é executado em um ambiente mais adverso, a velocidade de convergência diminui com a piora com a maior dificuldade de encontrar a resolução do problema.

Tabela 12: Tempo médio de execução (segundos) ACO em ambiente com capacidade de processamento heterogêneo e largura de banda heterogênea

Número de tarefas	10 máquinas	50 máquinas	100 máquinas
100	0,07	0,5	1,01
1000	0,3	3,68	9,2
10000	3,29	32,83	80,28
100000	36,3	311,7	918,19

4.1.3 ABC

Tabela 13: Eficiência do ABC em (%) ambiente com capacidade de processamento heterogêneo e largura de banda homogênea

Número de tarefas	10 máquinas	50 máquinas	100 máquinas
100	28,43	31,2	28,8
1000	28,6	30,98	29,04
10000	28,53	31,08	29,1
100000	28,56	31,08	29,08

O algoritmo ABC demonstra um resultado muito consistente, mesmo colocando-o no ambiente com largura de banda heterogênea. A função de aptidão do algoritmo nesse sentido, consegue também melhorar a capacidade do algoritmo encontrar a solução.

Tabela 14: Eficiência do ABC em (%) ambiente com capacidade de processamento heterogêneo e largura de banda heterogênea

Número de tarefas	10 máquinas	50 máquinas	100 máquinas
100	26,47	29,13	29,17
1000	26,48	29,27	28,8
10000	26,33	29,21	28,77
100000	26,31	29,26	28,78

O tempo total de simulação do ABC vai decrescendo com a quantidade de máquinas disponíveis, pois o fluxo de comunicação aumenta, causando este efeito de forma natural. Importante notar que o desempenho do ABC não é um dos melhores no tempo de simulação, mas compensa por sua eficiência.

Tabela 15: Tempo total simulação (segundos) do ABC em ambiente com capacidade de processamento heterogêneo e largura de banda homogênea

Número de tarefas	10 máquinas	50 máquinas	100 máquinas
100	297,16	168,15	104,37
1000	2962,2	1662,78	990,34
10000	29661,5	16658,9	9971,57
100000	296545	166777	99677,1

Na rede com comunicação heterogênea, o ABC supera o ACO e é cerca de duas vezes pior quando temos 100 máquinas com 100000 tarefas, demonstrando uma consistência no resultado, mesmo com os poucos parâmetros do ambiente que foram passados.

Tabela 16: Tempo total simulação (segundos) do ABC em ambiente com capacidade de processamento heterogêneo e largura de banda heterogênea

Número de tarefas	10 máquinas	50 máquinas	100 máquinas
100	61,16	40,9	15,75
1000	610,12	409,28	148,44
10000	6104,58	4046,58	1458,98
100000	61028,3	40374,7	14562,4

Tabela 17: Tempo total execução (segundos) do ABC em ambiente com capacidade de processamento heterogêneo e largura de banda homogênea

Número de tarefas	10 máquinas	50 máquinas	100 máquinas
100	0,01	0,03	0,06
1000	0,03	0,04	0,2
10000	0,24	0,4	0,5
100000	3,64	3,45	3,8

O tempo total de execução do ABC não é significativamente influenciado pelo ambiente de execução deste experimento, diferente do comportamento do ACO e do LBACO.

Tabela 18: Tempo total execução (segundos) ABC em ambiente com capacidade de processamento heterogêneo e largura de banda heterogênea

Número de tarefas	10 máquinas	50 máquinas	100 máquinas
100	0,01	0,02	0,17
1000	0,03	0,17	0,19
10000	0,26	0,29	0,48
100000	3,66	3,56	3,97

4.1.4 PSO

Tabela 19: Eficiência do PSO (%) em ambiente com capacidade de processamento heterogêneo e largura de banda homogênea

Número de tarefas	10 máquinas	50 máquinas	100 máquinas
100	13,93	14,02	11,77
1000	14,11	13,96	11,52
10000	13,99	13,88	11,54
100000	14,02	13,88	11,48

É possível observar que o PSO sofre uma queda significativa quando existem poucas máquinas disponíveis, mas consegue ser consistente quando estamos no ambiente de 100 máquinas, a eficiência consegue se manter, mesmo tendo poucos parâmetros, pois assim como o ABC, o algoritmo também utiliza uma função de aptidão para avaliar a qualidade do

resultado encontrado

Tabela 20: Eficiência do PSO (%) em ambiente com capacidade de processamento heterogêneo e largura de banda heterogênea

Número de tarefas	10 máquinas	50 máquinas	100 máquinas
100	11,25	12,86	11,76
1000	11,32	12,91	11,8
10000	11,31	12,78	11,74
100000	11,3	12,81	11,72

O resultado do tempo total de simulação do PSO decai também com a quantidade de máquinas, isso ocorre porque o fluxo de comunicação aumenta, portanto é esperado este comportamento.

Tabela 21: Tempo total simulação (segundos) do PSO em ambiente com capacidade de processamento heterogêneo e largura de banda homogênea

Número de tarefas	10 máquinas	50 máquinas	100 máquinas
100	43,75	17,99	13,84
1000	338,67	94,93	56,79
10000	3169,76	702,77	375,29
100000	30579,8	6328,3	3245,99

O resultado do tempo total de simulação tem um resultado parecido quando é realizado até 100 tarefas nos ambientes, porém é possível observar que o decaimento do tempo não é equivalente ao homogêneo, isso ocorre porque o PSO não recebe como parâmetro a largura de banda.

Tabela 22: Tempo total simulação (segundos) do PSO em ambiente com capacidade de processamento heterogêneo e largura de banda heterogênea

Número de tarefas	10 máquinas	50 máquinas	100 máquinas
100	18,22	8,29	7,05
1000	164,26	47,23	30,77
10000	1617,97	401,03	241,19
100000	16133,2	3963,4	2357,29

Tabela 23: Tempo médio de execução (segundos) do PSO em ambiente com capacidade de processamento heterogêneo e largura de banda homogênea

Número de tarefas	10 máquinas	50 máquinas	100 máquinas
100	0,021	0,0375	0,051
1000	0,132	0,436	0,6
10000	1,054	2,32	2,94
100000	16,21	20,41	26,608

O tempo médio de execução do PSO não é influenciado pela heterogeneidade da comunicação do ambiente nas menores quantidades de tarefas de forma significativa, porém é possível observar que o algoritmo leva mais tempo de execução quando estamos com o máximo de tarefas e máquinas, demonstrando que mais iterações são realizadas para avaliação da qualidade da solução encontrada.

Tabela 24: Tempo médio de execução (segundos) do PSO em ambiente com capacidade de processamento heterogêneo e largura de banda heterogênea

Número de tarefas	10 máquinas	50 máquinas	100 máquinas
100	0,032	0,0415	0,057
1000	0,13	0,4195	0,4825
10000	1,047	2,0995	2,844
100000	15,401	21,166	29,654

4.2 Discussão dos resultados

O tempo médio de execução em redes com largura de banda homogênea pode ser observado com destaque ao ACO e ABC, tendo os piores tempos de execução para 100 máquinas, sendo o ACO o pior tempo de execução de todos, o que ocorre por conta dos vários laços for que são realizados em sua implementação, aumentando significativamente o tempo de execução do algoritmo, especialmente o LBACO, pois tem adicionado uma matriz a mais para a realização da otimização, fazendo com que o cálculo do feromônio final tenha uma influência significativa. Cabe ressaltar que é possível melhorar esse desempenho alterando os parâmetros passados anteriormente, por exemplo o número de formigas.

No quesito tempo de execução, o ABC tem o melhor desempenho entre todos, sendo o PSO o segundo melhor algoritmo nesse parâmetro, como esperado o ACO e o LBACO tem os maiores tempos de execução.

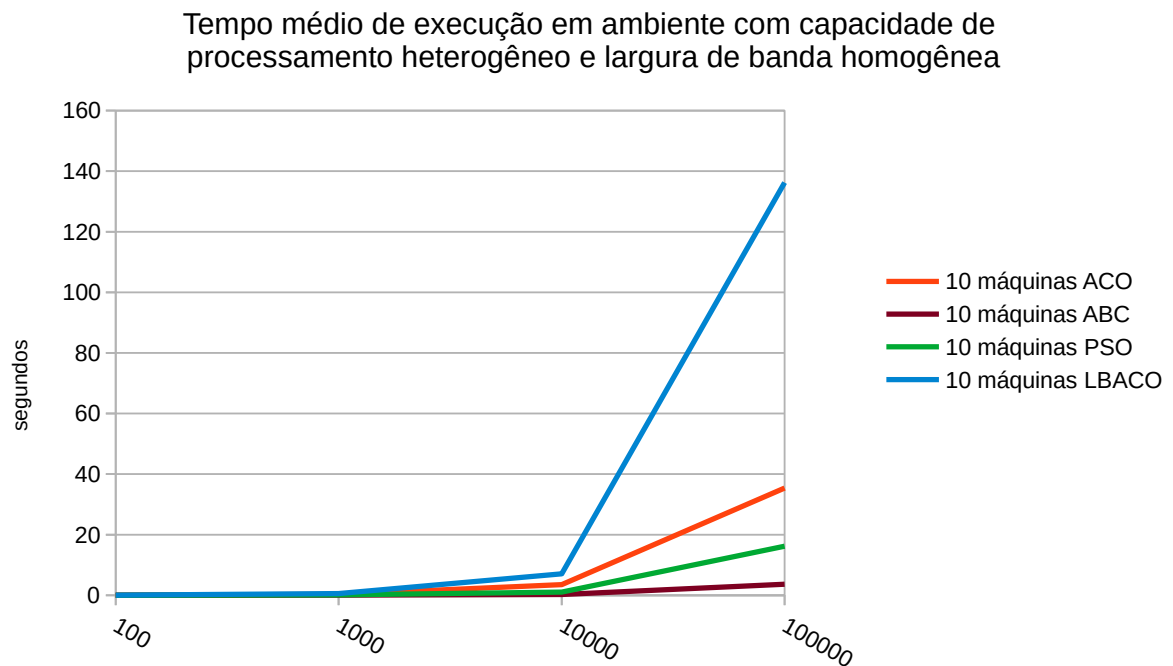


Figura 12: Tempo médio da execução (em segundos) em ambiente com 10 máquinas com capacidade de processamento e largura de banda padrão (criado pelo autor)

O tempo médio de execução com 50 máquinas sofre uma mudança que se aprofunda com o aumento de máquinas e número de tarefas.

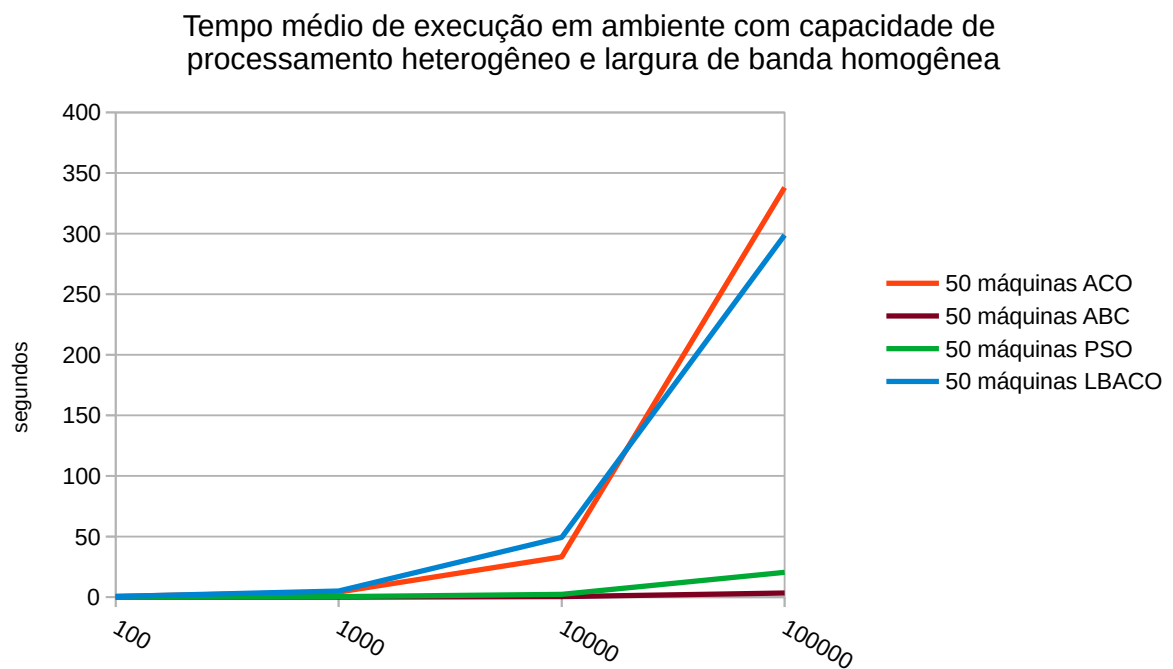


Figura 13: Tempo médio da execução (em segundos) em ambiente com 50 máquinas com capacidade de processamento e largura de banda padrão (criado pelo autor)

Com 100 máquinas é possível verificar que a convergência do resultado é que o tempo de execução do ACO é o pior de todos, seguido pelo ABC, LBACO e PSO.

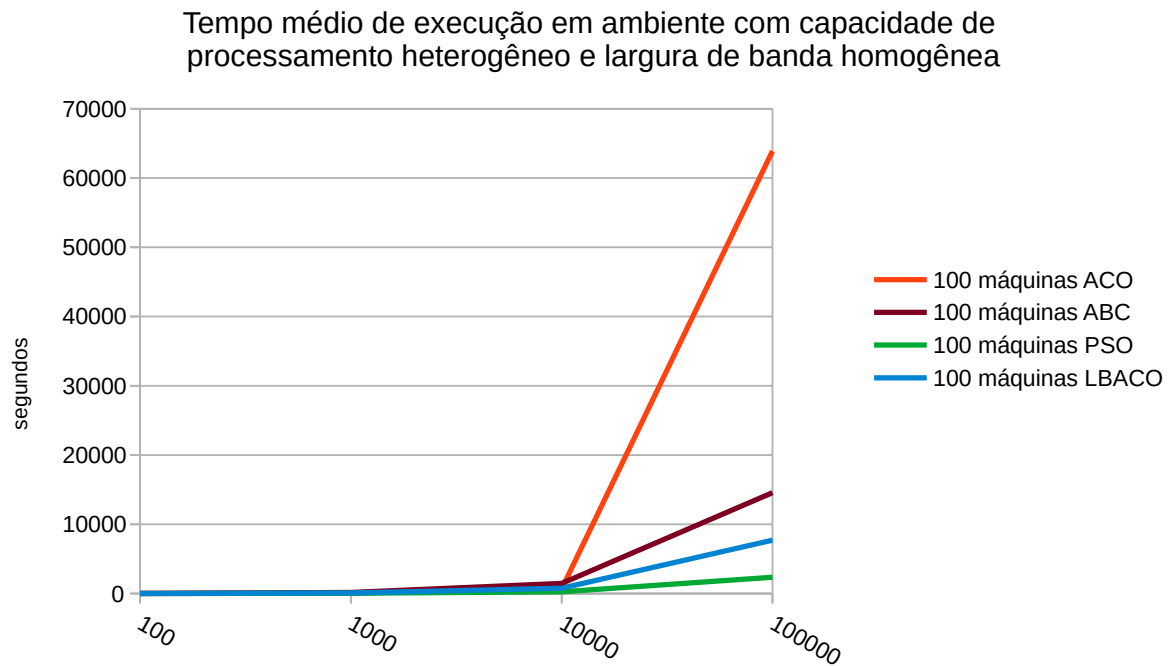


Figura 14: Tempo médio da execução (em segundos) em ambiente com 100 máquinas com capacidade de processamento e largura de banda padrão (criado pelo autor)

Ao observar uma rede heterogênea, é possível notar que o tempo de execução do LBACO ultrapassa o tempo do ACO, isso também se deve a necessidade de uma maior avaliação das informações recebidas pelo algoritmo, uma vez que em uma rede homogênea a largura de banda se torna algo não definidor, já os outros algoritmos não foram tão influenciados quanto o LBACO pela mudança:

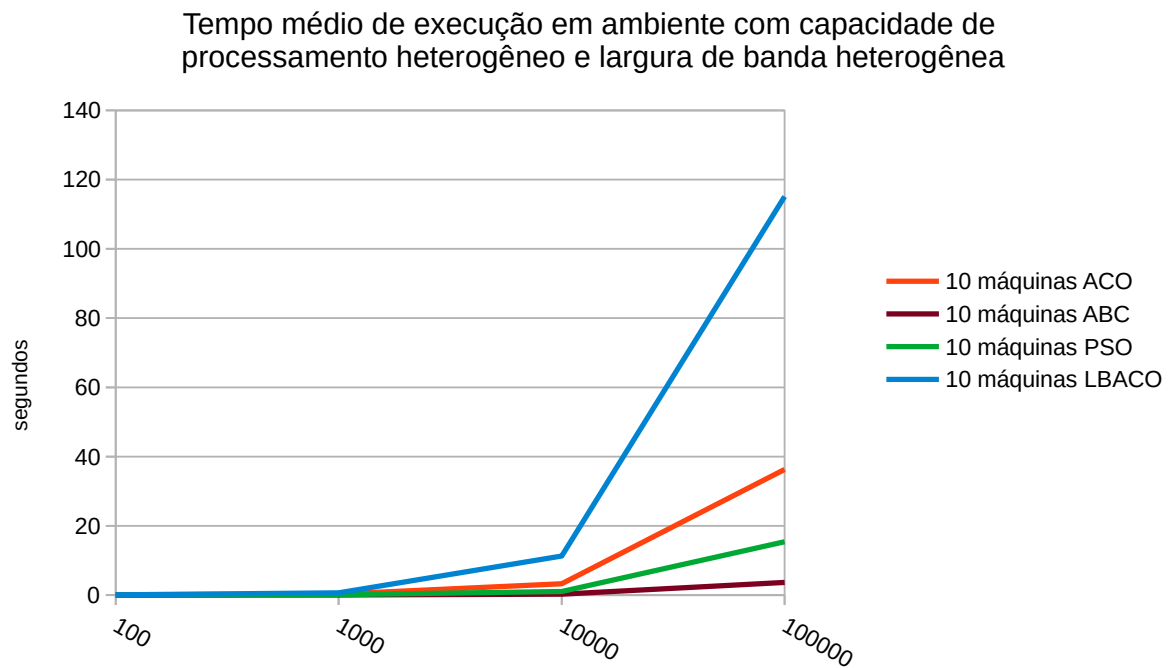


Figura 15: Tempo médio da execução (em segundos) em ambiente com 10 máquinas com capacidade de processamento e largura de banda heterogênea (criado pelo autor)

O tempo médio de execução é medido após 20 execuções do algoritmo, nestes testes nota-se que o ACO, com o aumento das máquinas, tende a se aproximar do LBACO, sendo o ABC o menos afetado, tanto nos ambientes heterogêneos, quanto com o aumento no número de máquinas.

Neste cenário é possível notar os mesmos resultados para o ABC e o PSO que Gökalp (2021) obteve em experimentos passados, os cenários testados referente ao estudo de Gökalp são diferentes, mas tem a mesma tendência de que o PSO tenha um tempo de execução muito mais elevado que o ABC. As diferenças ocorrem, pois o ambiente de experimento construído para o iSPD é mais aleatorizado, inclusive na capacidade de comunicação, algo que não foi realizado pelo referido autor.

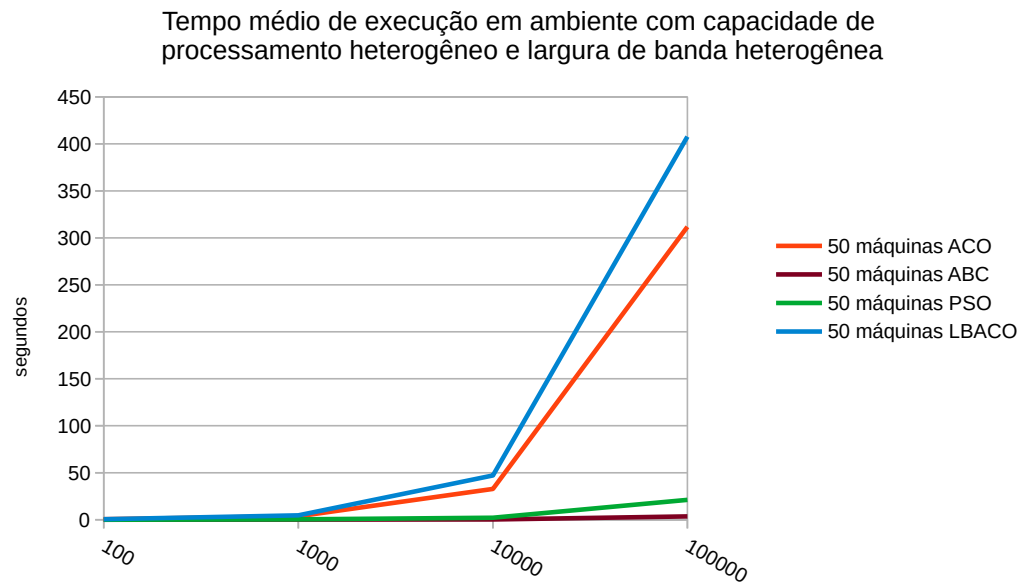


Figura 16: Tempo médio da execução (em segundos) em ambiente com 50 máquinas com capacidade de processamento e largura de banda heterogênea (criado pelo autor)

Com 100 máquinas podemos observar que o ACO continua a se aproximar do LBACO que o PSO e ABC continuam diminuindo o tempo de execução em relação aos outros algoritmos.

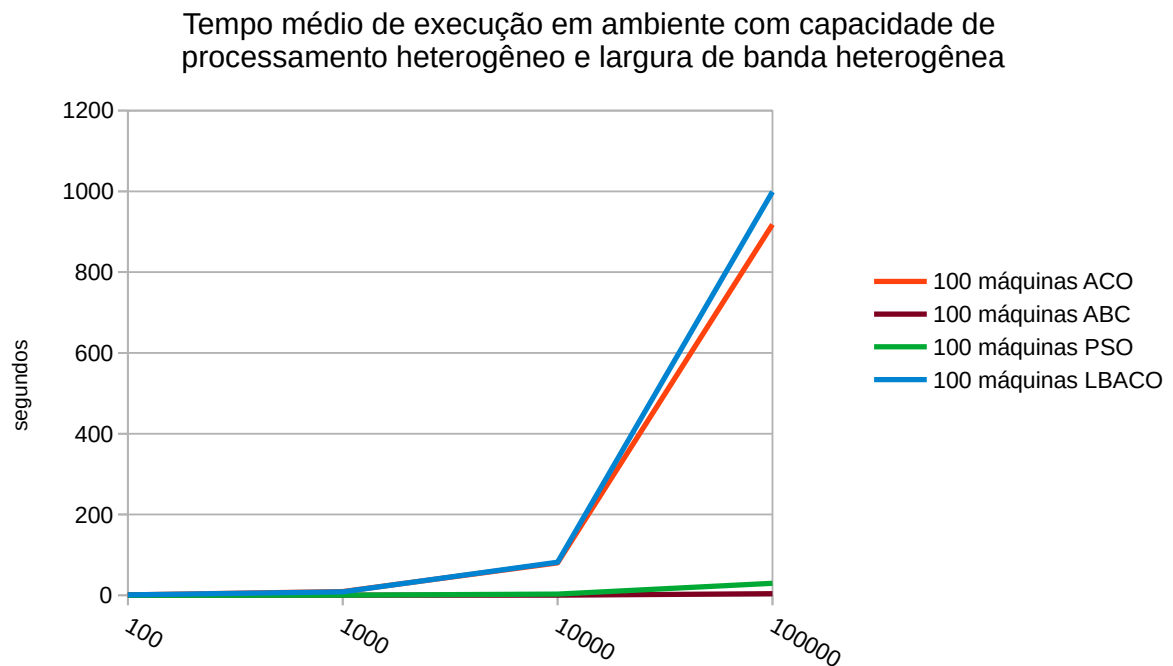


Figura 17: Tempo médio da execução (em segundos) em ambiente com 100 máquinas com capacidade de processamento e largura de banda heterogênea (criado pelo autor)

O tempo médio da simulação em ambientes com largura de banda homogênea é bastante penalizado pela baixa capacidade de comunicação em relação ao ambiente com comunicação heterogênea, que tem uma capacidade maior de transferência. Porém, mesmo nesse contexto o LBACO tem um desempenho melhor, pois o peso do *gamma* faz com que as máquinas com maior poder de processamento sejam um pouco mais privilegiadas. Já o PSO apresentou o pior resultado entre todos os algoritmos nesse contexto.

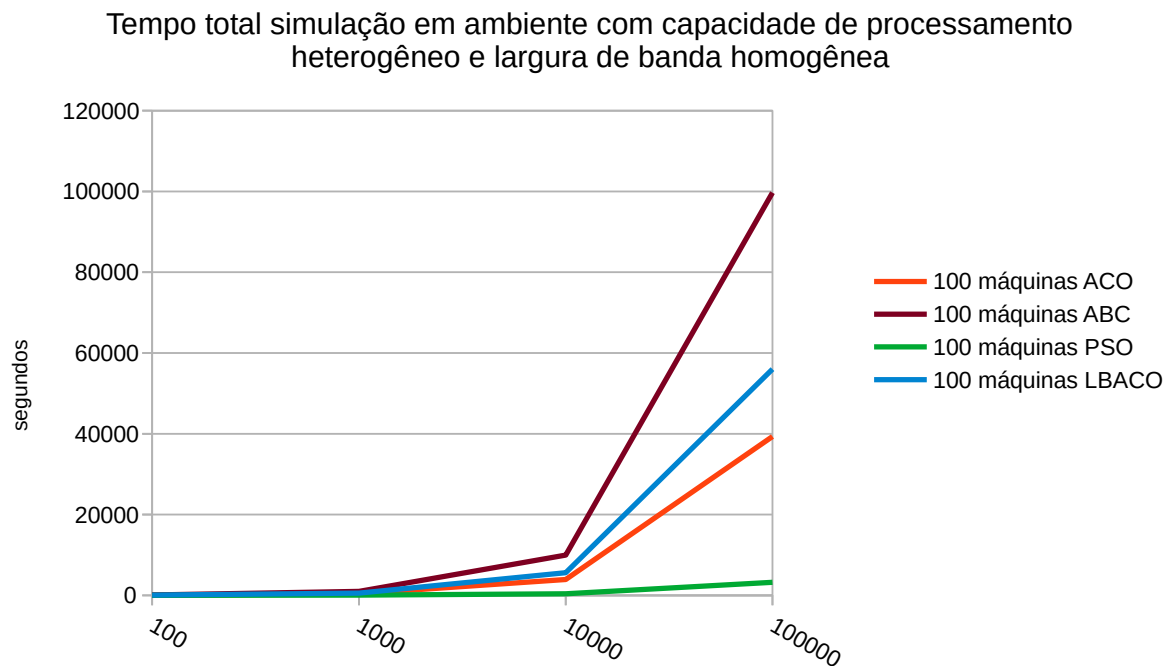


Figura 18: Tempo total da simulação (em segundos) em ambiente com 100 máquinas com capacidade de processamento e largura de banda padrão (criado pelo autor)

Nos ambientes heterogêneos o tempo médio da simulação cai porque a capacidade de comunicação geral é elevada, porém o PSO continua com o pior desempenho nesse quesito, já o LBACO superou todos os outros algoritmos nesse contexto mais adverso, uma vez que seus parâmetros de análise estão configurados para levar em consideração a largura de banda diferenciada.

O tempo total de simulação em redes heterogêneas é uma medida que pode nos ajudar a entender como os algoritmos atuam no fluxo de comunicação, para 10 máquinas, considera-se que existem poucas possibilidades de comunicação e por terem a aleatorização da capacidade de processamento e comunicação, a possibilidade de comunicação e processamento de forma combinadas, ficam escassas. Resultando em uma situação anômala, em que o LBACO tem o pior desempenho nesse ambiente, mesmo sendo o único que leva em consideração a largura de banda de cada dispositivo.

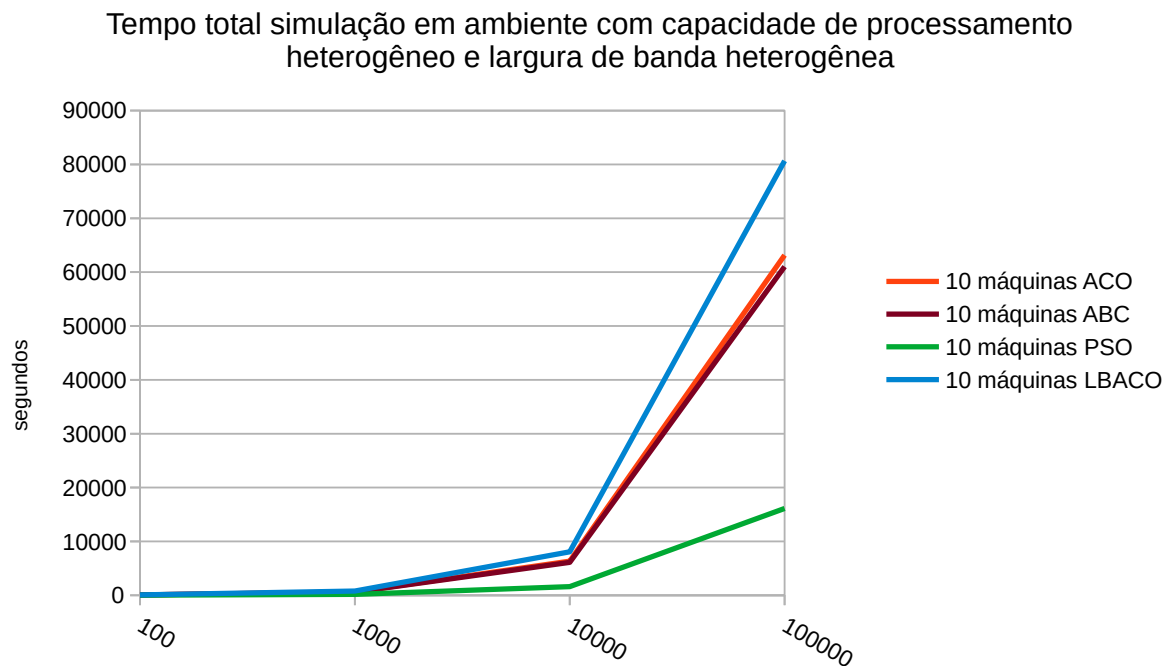


Figura 19: Tempo total da simulação (em segundos) em ambiente com 10 máquinas com capacidade de processamento e largura de banda aleatorizada (criado pelo autor)

Porém, com o aumento do número de máquinas e as possibilidades de combinação passam a ser mais possíveis, então com 50 máquinas o algoritmo LBACO consegue ficar muito próximo do ACO e o ABC passa a ser o pior nesse cenário., enquanto que o PSO tem resultados mais consistentes.

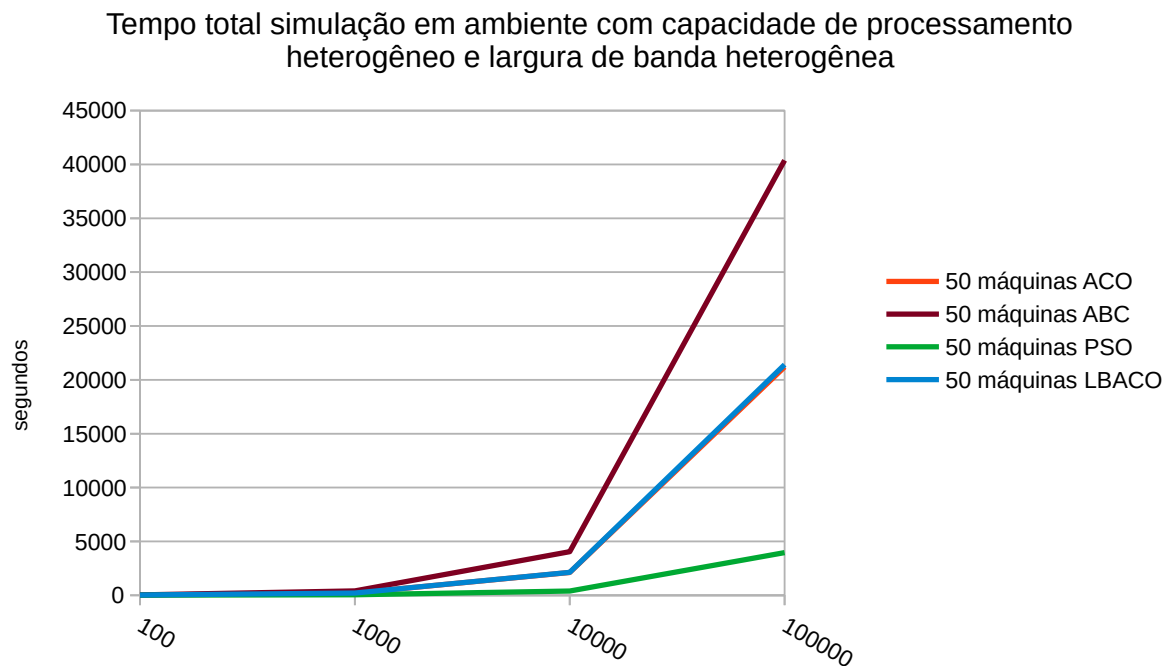


Figura 20: Tempo total da simulação (em segundos) em ambiente com 50 máquinas com capacidade de processamento e largura de banda aleatorizada (criado pelo autor)

Finalmente com 100 máquinas e 100000 tarefas podemos verificar que o tempo total de simulação é o pior com o ACO, que não leva em consideração a capacidade de comunicação dos dispositivos e também não tem a utilização de funções de aptidão para avaliar a qualidade da solução.

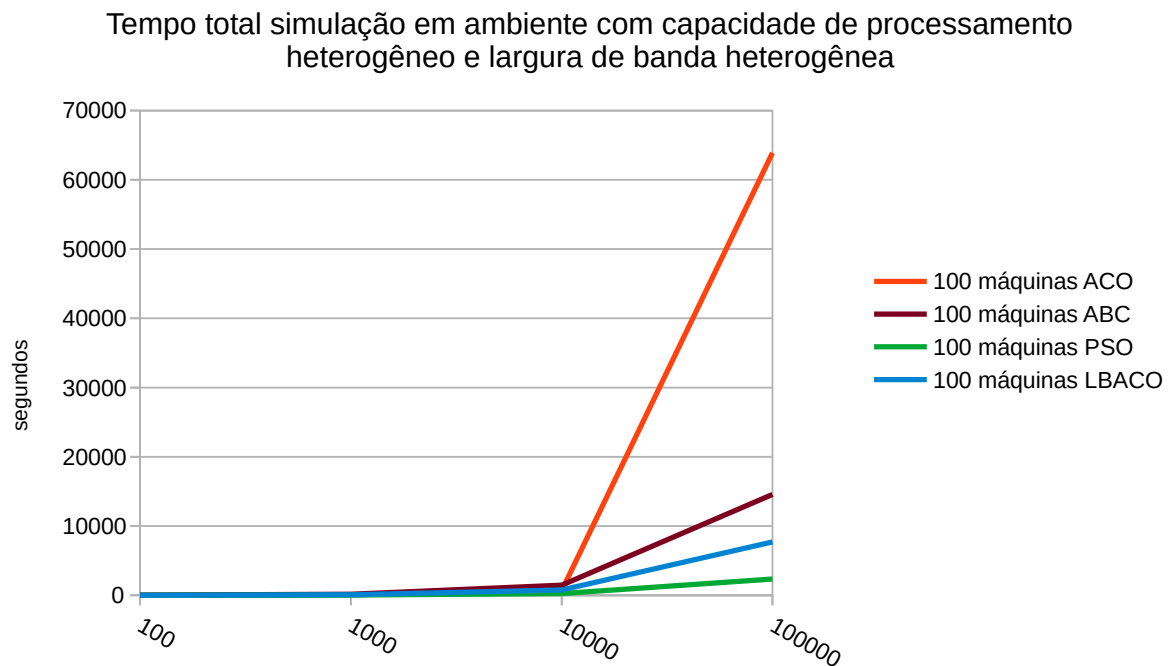


Figura 21: Tempo total da simulação (em segundos) em ambiente com 100 máquinas com capacidade de processamento e largura de banda aleatorizada (criado pelo autor)

No caso da eficiência o PSO teve o pior desempenho nos ambientes, muito por conta da dissonância mais acentuada em relação a máquinas com maior poder de processamento e menor capacidade de comunicação, a melhor eficiência é ocupada pelo ABC, seguido pelo LBACO. A diferença é que o Artificial Bee Colony utiliza uma função de aptidão para avaliar a qualidade do resultado, enquanto que o LBACO tem a capacidade de lidar com a variedade de comunicação, por receber como parâmetro também a largura de banda.

Neste cenário, podemos verificar que o experimento realizado por Gupta (2017), é válido, pois o algoritmo proposto por ele tem um tempo total de processamento das tarefas, substancialmente inferior em relação ao ACO, isso se deve a implementação baseada em votos das formigas, que busca evitar um resultado local ótimo.

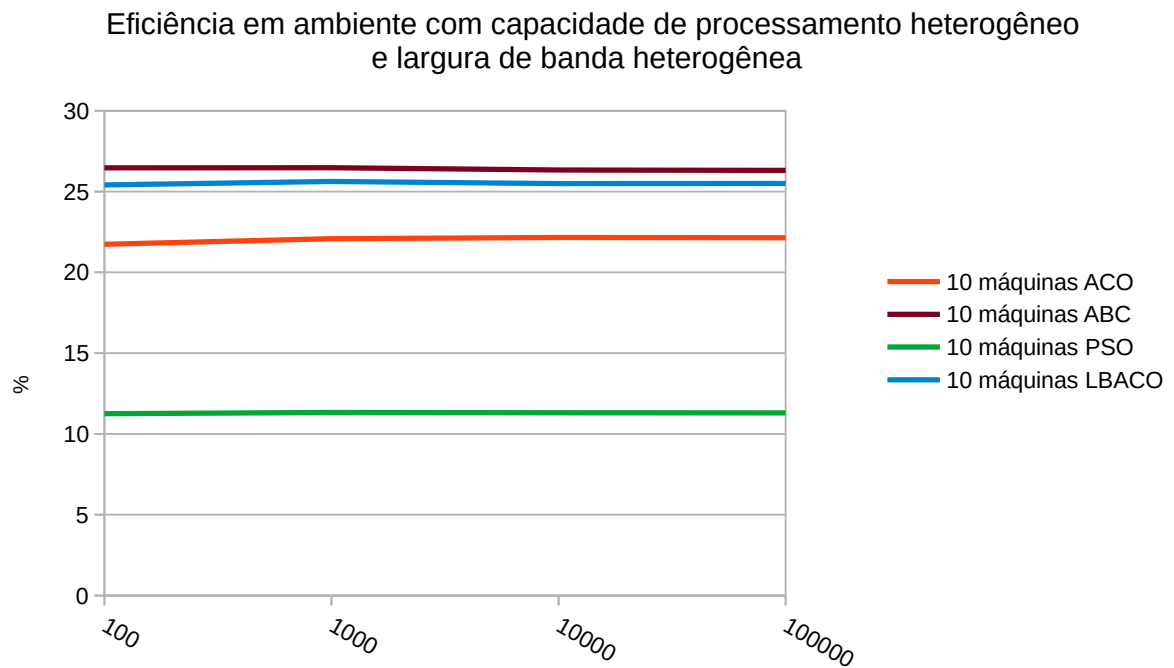


Figura 22: Eficiência em ambiente com 10 máquinas com capacidade de processamento e largura de banda aleatorizada (criado pelo autor)

O resultado do PSO não se altera de forma significativa com o aumento de máquinas, porém o LBACO consegue ter uma eficiência melhor do que o ACO, apesar que este tem uma eficiência melhor.

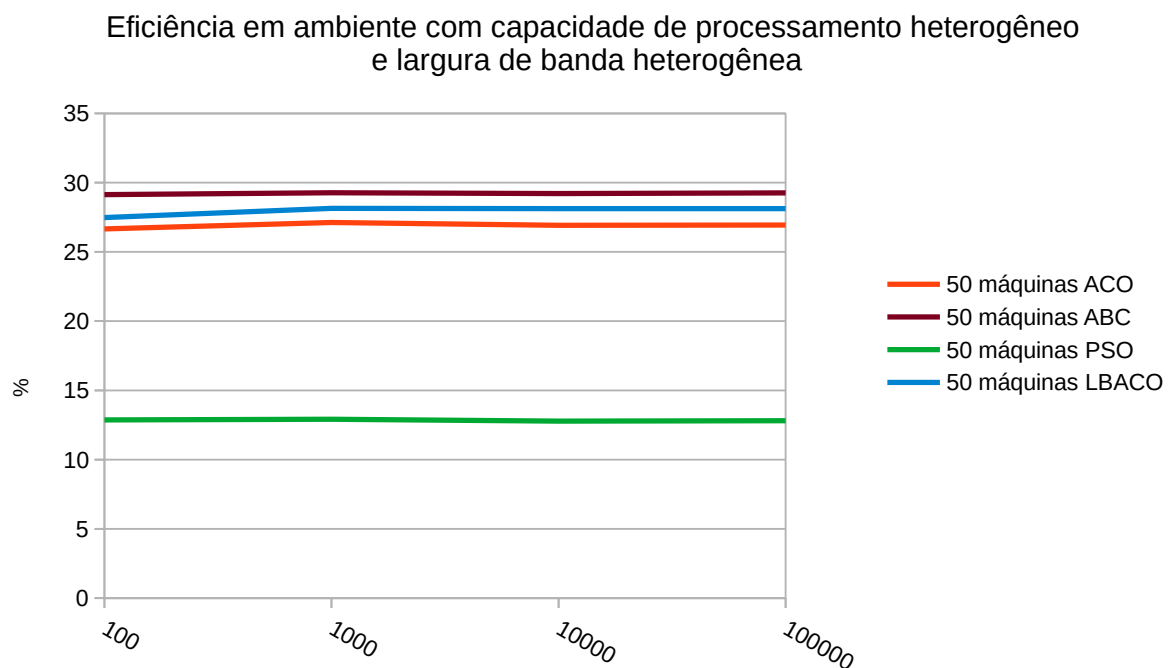


Figura 23: Eficiência em ambiente com 50 máquinas com capacidade de processamento e largura de banda aleatorizada (criado pelo autor)

Após 100 máquinas a eficiência praticamente não sofre alterações e o melhor algoritmo nesse quesito, surpreendentemente é o ABC, que mesmo sem receber como parâmetro a capacidade de comunicação, consegue sobressair em relação ao LBACO.

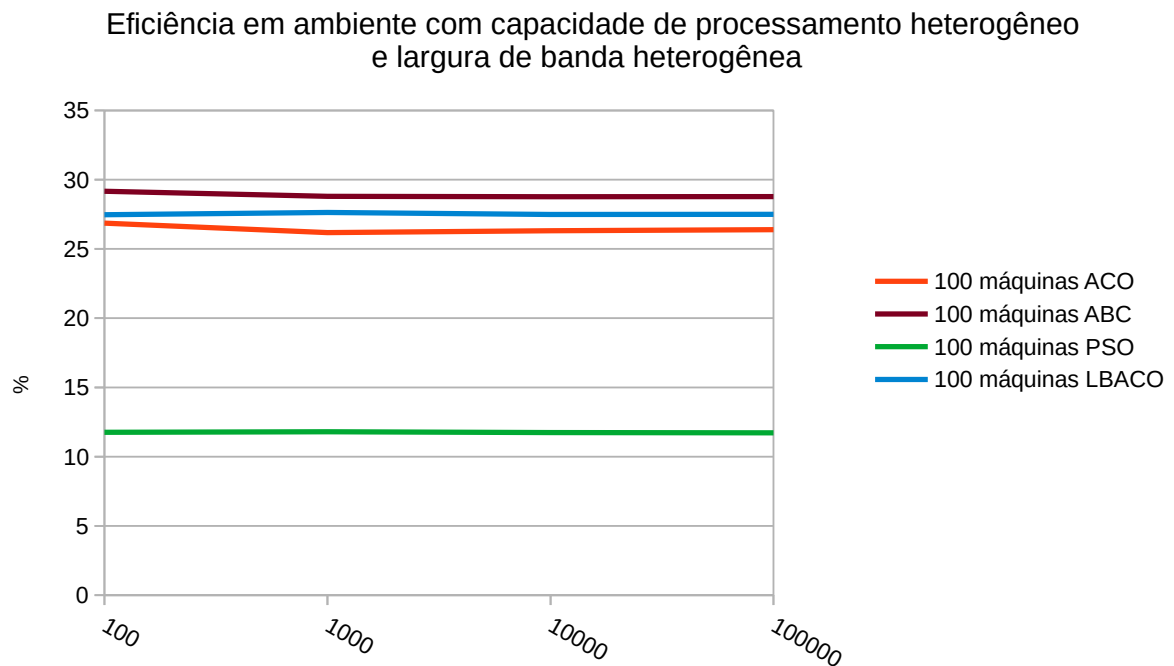


Figura 24: Eficiência em ambiente com 100 máquinas com capacidade de processamento e largura de banda aleatorizada (criado pelo autor)

A discrepância da eficiência em relação a ambientes com largura de banda padronizada fica mais evidente com o PSO que tem uma variação substancial em ambiente com 10 máquinas, sendo o ABC o mais eficiente nesse ambiente. No caso do LBACO e ACO, o primeiro consegue novamente se sobressair, também por conta do maior peso dado aos dispositivos que tem maior capacidade de processamento, uma vez que a largura de banda é padronizada, pode-se dizer que na prática o peso de decisão da capacidade processamento é maior no LBACO nesse tipo de ambiente.

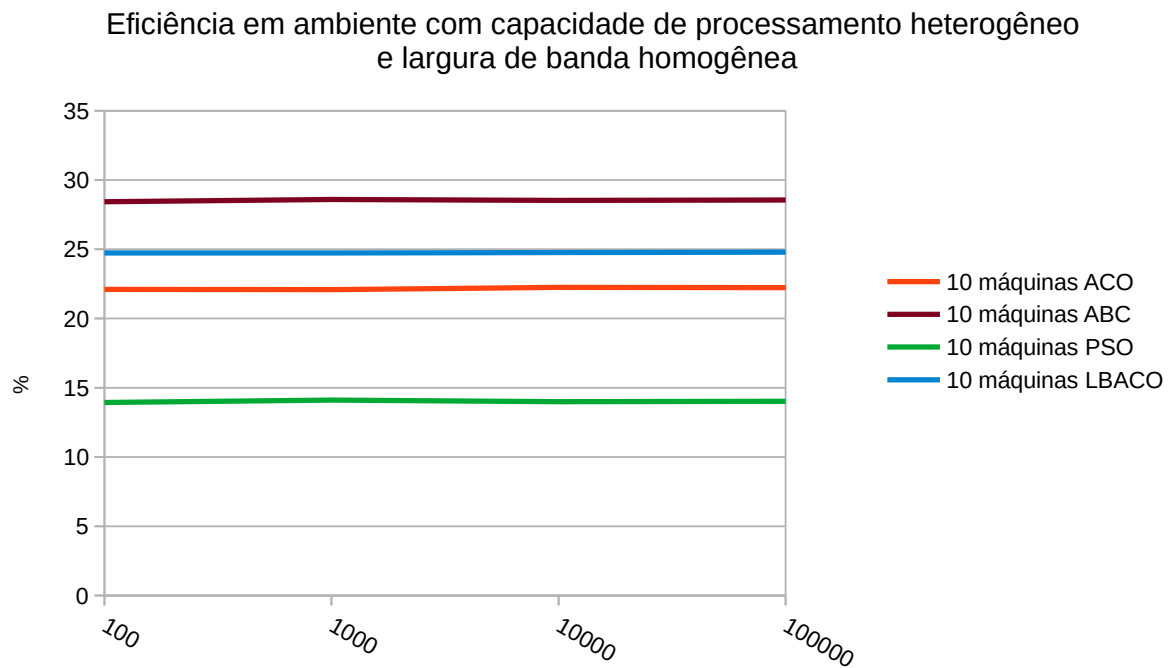


Figura 25: Eficiência em ambiente com 10 máquinas com capacidade de processamento aleatorizado e largura de banda padrão (criado pelo autor)

O PSO chega a variar positivamente quase três pontos percentuais em ambientes homogêneos, sendo uma diferença significativa. No caso do ACO a variação é por volta de um ponto percentual a mais, já o ABC se beneficia claramente de um ambiente com largura

de banda homogênea, especialmente no contexto de 50 máquinas.

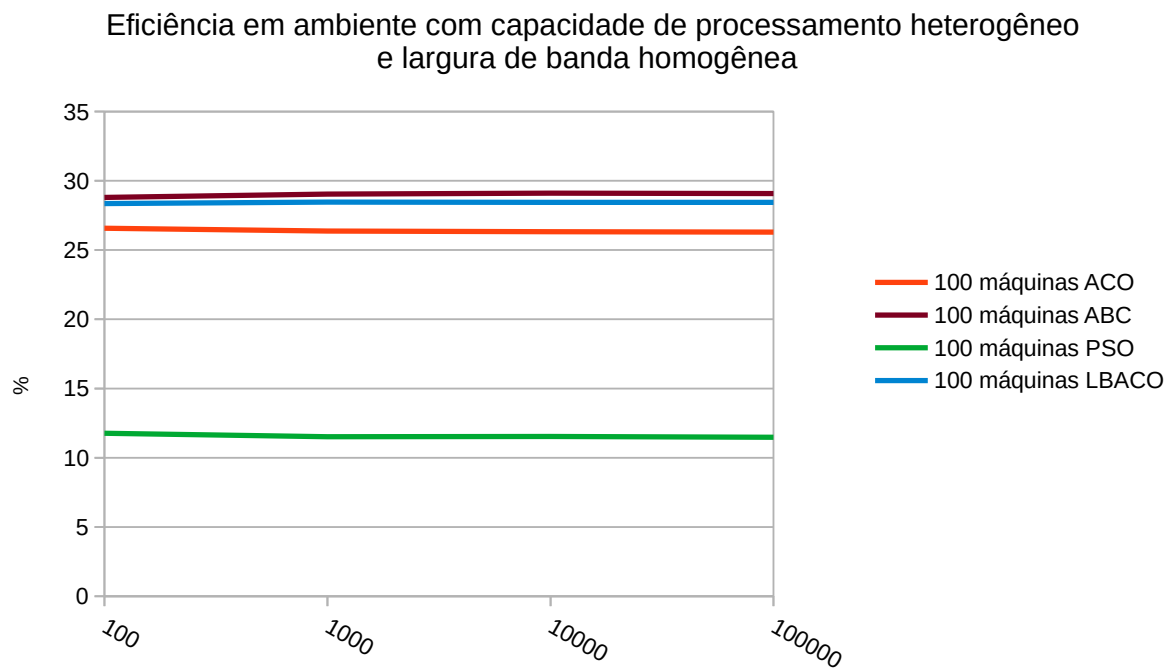


Figura 27: Eficiência em ambiente com 100 máquinas com capacidade de processamento aleatorizado e largura de banda padrão (criado pelo autor)

No contexto das 100 máquinas com largura de banda homogênea, é possível notar que a eficiência geral dos algoritmos tem um ganho geral, com exceção do PSO que pouco varia com as 100 máquinas.

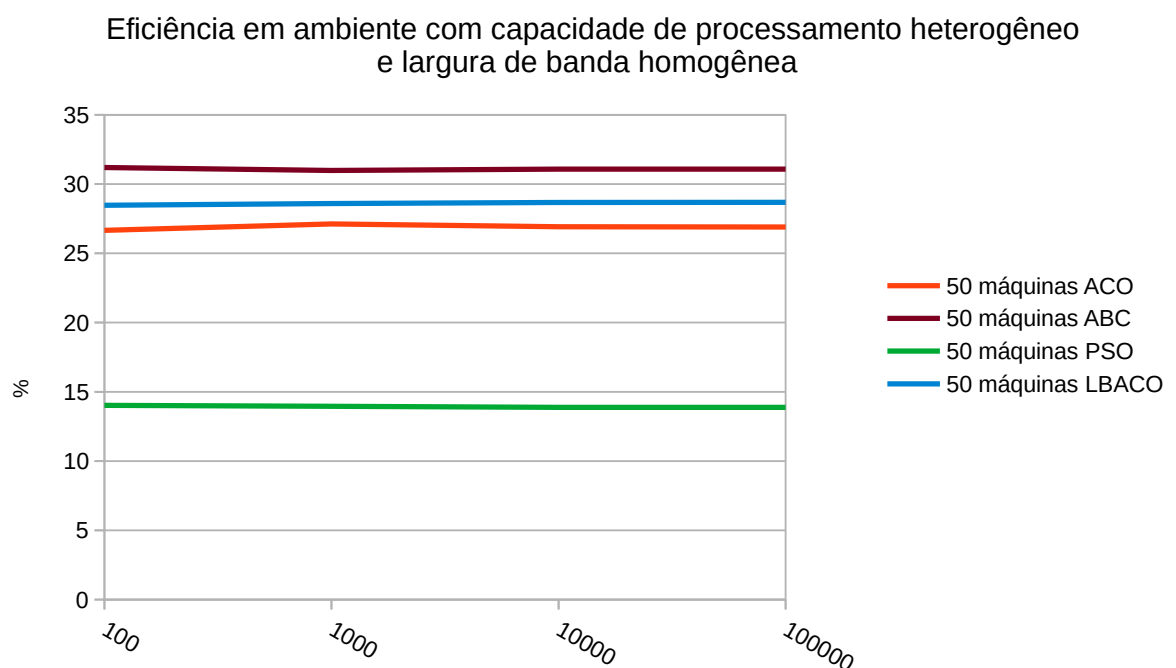


Figura 26: Eficiência em ambiente com 50 máquinas com capacidade de processamento aleatorizado e largura de banda padrão (criado pelo autor)

Capítulo 5

Conclusões

Ao longo deste trabalho foi realizado uma apresentação sobre ferramentas de simulação de ambientes e uma apresentação sobre os ambientes Fog Computing com o advento da tecnologia IoT, dimensionando seus impactos e maneiras de aproveitar melhor o processamento e controle dos dispositivos que integram a camada Fog.

5.1 Considerações

O iSPD se demonstrou uma ferramenta capaz de simular ambientes extramente complexos e extrair suas informações, mesmo que tais ambientes não existam no mundo físico, tornando possível que seja modelado um sistema complexo de forma barata e rápida, para detecção de problemas, redução de custo e estudo de algoritmos de escalonamento de diversos tipos de ambientes.

Os experimentos demonstram que dentre os algoritmos de *swarm*, o que tem o melhor desempenho geral é o ABC, mesmo com parâmetros menos significativos sobre o ambiente. Nesse sentido, a utilização do LBACO é exemplar de como a extração de informação do ambiente é algo fundamental, permitindo que seu desempenho fosse significativo em ambientes mais adversos quando se comparado com os outros no mesmo contexto, além de que a proposta deste algoritmo de utilizar um método de votação, mostra como a proposta do *Ant Colony* ainda pode ser mais utilizada e adaptada, nesse sentido é importante compreender que algoritmos podem ser modificados para atender situações específicas e por consequência, acabam por serem renovados. No caso do PSO demonstra ter uma convergência para o resultado mais otimizado para sua capacidade, demonstrando disparidade mais significativa quando colocado em um contexto mais problemático, mas ainda mantendo resultados que podem ser mais explorados.

Uma das palavras que mais foi utilizada neste trabalho é a probabilidade, fica explícito após o desenvolvimento deste estudo, que para entendermos o comportamento de

qualquer ser vivo, se trata de entender a probabilidade inerente à existência do ser analisado, portanto a compreensão deste assunto é um pilar importante para que seja possível a realização e compreensão dos algoritmos implementados.

O estudo dos algoritmos bioinspirados também nos leva a conclusão de que a fonte de conhecimento continua sendo a natureza, podemos extrair métodos de resolver problemas que foram criados ao longo da evolução, reforçando a importância de preservar a existência de todas as espécies conhecidas e desconhecidas, pois a resolução de um problema que até então não tem uma solução, pode ser resolvido por um algoritmo de um inseto que tem bilhões de anos de evolução.

5.2 Dificuldades e desafios

Durante a realização deste trabalho, as principais dificuldades encontradas foram:

- Compreensão do funcionamento do código do iSPD:
 - Foi possível pela utilização de *metaprogramming*, que possibilita a utilização da reflexão na linguagem Java e a análise do conteúdo de objetos.
- Alteração de partes do motor do iSPD para acessar informações que estavam inacessíveis (largura de banda de links) e correção de eventuais erros.
- Automação dos experimentos para extração em massa de dados:
 - Realizado utilizando *Bash Script*, possibilitando que o mínimo de 1440 execuções fossem realizadas de forma automatizada, cobrindo todos os ambientes necessários.

5.3 Direções futuras

Trabalhos que podem ser desenvolvidos futuramente:

- Implementação de redes neurais para otimização do *Ant Colony Optimization*
- Criação de algoritmos híbridos para serem executados em ambientes mais complexos, considerando não apenas um grupo de computadores *Fog*, mas a existência de *data centers* em distâncias continentais.

Referências Bibliográficas

AMAN, Saima et al. Prediction models for dynamic demand response: Requirements, challenges, and insights. In: **2015 IEEE International Conference on Smart Grid Communications (SmartGridComm)**. IEEE, 2015. p. 338-343.

AMRUTUR, Bharadwaj et al. An open smart city iot test bed: street light poles as smart city spines. In: **Proceedings of the Second International Conference on Internet-of-Things Design and Implementation**. 2017. p. 323-324.

ARPACI-DUSSEAU, Remzi H.; ARPACI-DUSSEAU, Andrea C. **Operating systems: Three easy pieces**. Boston: Arpaci-Dusseau Books LLC, 2018.

ASGHARI, Saied; NAVIMIPOUR, Nima Jafari. Cloud service composition using an inverted ant colony optimisation algorithm. **International Journal of Bio-Inspired Computation**, v. 13, n. 4, p. 257-268, 2019.

BAXODIRJONOVICH, Kamolov Nizomiddin; CHOE, Tae-Young. Dynamic task scheduling algorithm based on ant colony scheme. **International Journal of Engineering and Technology (IJET)**, v. 7, n. 4, p. 1163-1172, 2015.

BIANCHI, Leonora et al. A survey on metaheuristics for stochastic combinatorial optimization. **Natural Computing**, v. 8, n. 2, p. 239-287, 2009.

BITTENCOURT, Luiz Fernando et al. Towards virtual machine migration in fog computing. In: **2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)**. IEEE, 2015. p. 1-8.

BŁAŻEWICZ, Jacek et al. **Scheduling computer and manufacturing processes**. springer science & Business media, 2001.

BONOMI, Flavio et al. Fog computing: A platform for internet of things and analytics. In: **Big data and internet of things: A roadmap for smart environments**. Springer, Cham, 2014. p. 169-186.

BUYA, Rajkumar; MURSHED, Manzur. "Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing". **Concurrency and computation: practice and experience**, v. 14, n. 13-15, p. 1175-1220, 2002.

CASANOVA, Henri. "Simgrid: A toolkit for the simulation of application scheduling". In: **Proceedings First IEEE/ACM International Symposium on Cluster Computing and the Grid**. IEEE, 2001. p. 430-437.

DA SILVA, Diogo T. et al. Modeling and Simulation of Cloud Computing with iSPD. In: **Anais do XXIII Simpósio em Sistemas Computacionais de Alto Desempenho**. SBC, 2022. p. 217-228.

DIN, Ikram Ud et al. The Internet of Things: A review of enabled technologies and future challenges. **Ieee Access**, v. 7, p. 7606-7640, 2018.

DORIGO, Marco; BIRATTARI, Mauro; STUTZLE, Thomas. Ant colony optimization. **IEEE computational intelligence magazine**, v. 1, n. 4, p. 28-39, 2006.

DORIGO, Marco. Optimization, learning and natural algorithms. **Ph. D. Thesis**, Politecnico di Milano, 1992.

DORSEMAINE, Bruno et al. Internet of things: a definition & taxonomy. In: **2015 9th International Conference on Next Generation Mobile Applications, Services and Technologies**. IEEE, 2015. p. 72-77.

ELKHODR, Mahmoud; SHAHRESTANI, Seyed; CHEUNG, Hon. The internet of things: New interoperability, management and security challenges. **arXiv preprint arXiv:1604.04824**, 2016.

GHAFFARI, Elham. Providing a new scheduling method in fog network using the ant colony algorithm. 2019.

GÖKALP, Osman. Performance evaluation of heuristic and metaheuristic algorithms for independent and static task scheduling in cloud computing. In: **2021 29th Signal Processing and Communications Applications Conference (SIU)**. IEEE, 2021. p. 1-4.

GUPTA, Ashish; GARG, Ritu. Load balancing based task scheduling with ACO in cloud computing. In: **2017 International conference on computer and applications (ICCA)**. IEEE, 2017. p. 174-179.

HAJIMIRZAEI, Bahram; NAVIMIPOUR, Nima Jafari. Intrusion detection for cloud computing using neural networks and artificial bee colony optimization algorithm. **Ict Express**, v. 5, n. 1, p. 56-59, 2019.

HUANG, Cheng; LU, Rongxing; CHOO, Kim-Kwang Raymond. Vehicular fog computing: architecture, use case, and security and forensic challenges. **IEEE Communications Magazine**, v. 55, n. 11, p. 105-111, 2017.

JAMIL, Sonain; RAHMAN, MuhibUr. A Comprehensive Survey of Digital Twins and Federated Learning for Industrial Internet of Things (IIoT), Internet of Vehicles (IoV) and Internet of Drones (IoD). **Applied System Innovation**, v. 5, n. 3, p. 56, 2022.

KENNEDY, James; EBERHART, Russell. Particle swarm optimization. In: Proceedings of ICNN'95-international conference on neural networks. IEEE, 1995. p. 1942-1948.

KHELIFI, Fekher et al. A survey of localization systems in internet of things. **Mobile Networks and Applications**, v. 24, n. 3, p. 761-785, 2019.

KISHOR, Amit; CHAKARBARTY, Chinmay. Task offloading in fog computing for using smart ant colony optimization. **Wireless personal communications**, p. 1-22, 2021.

KOPETZ, Hermann; STEINER, Wilfried. Internet of things. In: **Real-time systems**. Springer, Cham, 2022. p. 325-341.

LEE, Jay; KAO, Hung-An; YANG, Shanhu. Service innovation and smart analytics for industry 4.0 and big data environment. **Procedia cirp**, v. 16, p. 3-8, 2014.

LUAN, Tom H. et al. Fog computing: Focusing on mobile users at the edge. **arXiv preprint arXiv:1502.01815**, 2015.

LUCIC, Panta; TEODOROVIC, Dusan. Transportation modeling: an artificial life approach. In: **14th IEEE International Conference on Tools with Artificial Intelligence, 2002. (ICTAI 2002). Proceedings**. IEEE, 2002. p. 216-223.

MAGALHÃES, Regina; VENDRAMINI, Annelise. Os impactos da quarta revolução industrial. 2018.

MANACERO, Aleardo et al. iSPD: an iconic-based modeling simulator for distributed grids. In: **SpringSim (ANSS)**. 2012. p. 5.

MATROUK, Khaled; ALATOUN, Kholoud. Scheduling Algorithms in Fog Computing: A Survey. **Int. J. Networked Distributed Comput.**, v. 9, n. 1, p. 59-74, 2021.

Rajkumar Buyya; Satish Narayana Srirama, "Modeling and Simulation of Fog and Edge Computing Environments Using iFogSim Toolkit" in **Fog and Edge Computing: Principles and Paradigms**, Wiley, 2019, pp.433-465, doi: 10.1002/9781119525080.ch17.

SATYANARAYANAN, Mahadev et al. Edge analytics in the internet of things. **IEEE Pervasive Computing**, v. 14, n. 2, p. 24-31, 2015.

SATYANARAYANAN, Mahadev et al. The case for vm-based cloudlets in mobile computing. **IEEE pervasive Computing**, v. 8, n. 4, p. 14-23, 2009.

SERAPIÃO, Adriane Beatriz de Souza. Fundamentos de otimização por inteligência de enxames: uma visão geral. **Sba: Controle & Automação Sociedade Brasileira de Automatica**, v. 20, p. 271-304, 2009.

SINGH, Simar Preet et al. Dynamic task scheduling using balanced VM allocation policy for fog computing platforms. **Scalable Computing: Practice and Experience**, v. 20, n. 2, p. 433-456, 2019.

STOJMENOVIC, Ivan. Fog computing: A cloud to the ground support for smart things and machine-to-machine networks. In: **2014 Australasian telecommunication networks and applications conference (ATNAC)**. IEEE, 2014. p. 117-122.

TANG, Bo et al. Incorporating intelligence in fog computing for big data analysis in smart cities. **IEEE Transactions on Industrial informatics**, v. 13, n. 5, p. 2140-2150, 2017.

VAQUERO, Luis M.; RODERO-MERINO, Luis. Finding your way in the fog: Towards a comprehensive definition of fog computing. **ACM SIGCOMM computer communication Review**, v. 44, n. 5, p. 27-32, 2014.

WANG, Yabin; GUO, Chenghao; YU, Jin. Immune scheduling network based method for task scheduling in decentralized fog computing. **Wireless Communications and Mobile Computing**, v. 2018, 2018.