UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Escalonamento de workflow com anotações de tarefas sensitivas para otimização de segurança e custo em nuvens

Henrique Yoshikazu Shishido

Tese de Doutorado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP SÃO CARLOS

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Henrique Yoshikazu Shishido

Escalonamento de workflow com anotações de tarefas sensitivas para otimização de segurança e custo em nuvens

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Doutor em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Júlio Cézar Estrella
Coorientador: Prof. Dr. Claudio Fabiano Motta Toledo

USP – São Carlos
Fevereiro de 2019

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

Shishido, Henrique Yoshikazu
SS558e Escalonamento de workflow com anotações de tarefas sensitivas para otimização de segurança e custo em nuvens / Henrique Yoshikazu Shishido; orientador Júlio Cezar Estrella; coorientador Claudio Fabiano Motta Toledo. -- São Carlos, 2018.
186 p.

Tese (Doutorado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2018.

1. COMPUTAÇÃO EM NUVEM . 2. COMPUTAÇÃO EVOLUTIVA . 3. SEGURANÇA DE REDES . I. Estrella, Júlio Cezar, orient. II. Toledo, Claudio Fabiano Motta, coorient. III. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Henrique Yoshikazu Shishido

Workflow scheduling with sensitive task annotations for security and cost optimization in clouds

Doctoral dissertation submitted to the Institute of Mathematics and Computer Sciences – ICMC-USP, in partial fulfillment of the requirements for the degree of the Doctorate Program in Computer Science and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Júlio Cézar Estrella
Co-advisor: Prof. Dr. Claudio Fabiano Motta Toledo

USP – São Carlos
February 2019

Dedico esse trabalho à minha esposa Gabrielle e ao meu fiel companheiro Tenma 10.

AGRADECIMENTOS

Embora este seja um dos primeiros elementos deste documento, deixei a sua elaboração ao final da escrita acreditando que seria algo fácil e rápido. No entanto, me enganei. Aos que me apoiaram nessa caminhada e não tenham sido mencionados, peço minhas desculpas, mas sintam-se incluídos!

Em primeiro lugar, agradeço à Força desconhecida que rege o universo que conhecemos até então.

À minha querida esposa, Gabrielle, por todo apoio, carinho, incentivo, companheirismo e amor. Mesmo eu estando ausente durante as semanas nesses últimos quatro anos, sempre me impulsionou para que eu pudesse manter meus esforços e concentração no desenvolvimento desta tese.

Aos meus pais, Américo e Rosa, e minha irmã Évelin pelo incentivo, carinho, confiança e apoio durante todo o transcorrer de minha vida, principalmente neste período árduo e desafiador.

Ao meu orientador Júlio Cezar Estrella, pela orientação e confiança em minha capacidade para o desenvolvimento deste doutorado. Obrigado Júlio por ter auxiliado no aperfeiçoamento da minha auto-confiança profissional e pessoal.

Ao meu co-orientador Claudio Fabiano Motta Toledo, pela co-orientação, pela confiança, pelo respeito e pensamento positivista mesmo em momentos “nebulosos”.

Aos professores do Laboratório de Sistemas Distribuídos e Programação Corrente. Ao Paulo Sérgio Lopes de Souza pelos ensinamentos de resiliência e por me apresentar uma ótica diferente de diversos assuntos. À Sarita Mazzini Bruschi pelas indagações e sugestões acerca do trabalho, e pelas diversas conversas no dia-a-dia.

Aos colegas do Laboratório de Sistemas Distribuídos , Leonildo Azevedo, Edvard de Oliveira, Luiz Henrique Nunes, Helder Luz, Lourenço Alves Pereira Jr., Danilo Costa, Carlos Henrique Ferreira, Carlos Emílio Cacho, George Dourado, Rafael Libardi, Zhang Yifei, Dionisio Leite, Bruno Guazzelli Batista, Edwin Choquehuanca Manani e Luis Nakamura, Gesiel Rios Lopes, Vinicius Aires, Welington Martins, Davi Conti, Guilherme Martins (bumbum), Ana Spengler e os demais que conheci.

Aos meus companheiros de moradia Diego Camargo, Pedro Gimenes e Nathália Gardin Pessoa pelo abrigo oferecido e pelos momentos de descontração, ajuda médica e procrastinação.

Aos colegas da UTFPR por proporcionar a licença de minhas atividades acadêmicos que

foi crucial para o desenvolvimento deste trabalho.

À Secretaria de Pós-graduação pela ajuda com os trâmites burocráticos.

Aos órgãos de fomento CNPq, FAPESP e CAPES pelo auxílio para a aquisição de equipamentos e publicação de trabalhos.

“Na ciência, o crédito vai para o homem
que convence o mundo de uma ideia, não para
aquele que a teve primeiro.”
(William Osler)

RESUMO

SHISHIDO, H.Y. **Escalonamento de workflow com anotações de tarefas sensitivas para otimização de segurança e custo em nuvens**. 2019. 186 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2019.

A evolução dos computadores tem possibilitado a realização de experimentos *in-silico*, incluindo aplicações baseadas no modelo de *workflow*. A execução de workflows é uma atividade que pode ser computacional dispendiosa, onde grades e nuvens são adotadas para a sua execução. Inserido nesse contexto, os algoritmos de escalonamento de workflow permitem atender diferentes critérios de execução como o tempo e o custo monetário. Contudo, a segurança é um critério que tem recebido atenção, pois diversas organizações hesitam em implantar suas aplicações em nuvens devido às ameaças presentes em um ambiente aberto e promíscuo como a Internet. Os algoritmos de escalonamento direcionados à segurança consideram dois cenários: (a) nuvens híbridas: mantêm os tarefas que manipulam dados sensitivos/confidenciais na nuvem privada e exporta as demais tarefas para nuvens públicas para satisfazer alguma restrição (ex.: tempo), e; (b) nuvens públicas: considera o uso de serviços de segurança disponíveis em instâncias de máquinas virtuais para proteger tarefas que lidam com dados sensitivos/confidenciais. No entanto, os algoritmos de escalonamento que consideram o uso de serviços de segurança selecionam as tarefas de forma aleatória sem considerar a semântica dos dados. Esse tipo de abordagem pode acabar atribuindo proteção a tarefas não-sensitivas e desperdiçando tempo e recursos, e deixando dados sensitivos sem a proteção necessária. Frente a essas limitações, propõe-se nesta tese duas abordagens de escalonamento de workflow: o *Workflow Scheduling - Task Selection Policies* (WS-TSP) e a *Sensitive Annotation for Security Tasks* (SAST). O WS-TSP é uma abordagem de escalonamento que usa um conjunto de políticas para a proteção de tarefas. O SAST é outra abordagem que permite utilizar o conhecimento prévio do Desenvolvedor de Aplicação para identificar quais tarefas devem receber proteção. O WS-TSP e a SAST consideram a aplicação de serviços de segurança como autenticação, verificação de integridade e criptografia para proteger as tarefas sensitivas do workflow. A avaliação dessas abordagens foi realizada através de uma extensão do simulador WorkflowSim que incorpora a sobrecarga dos serviços de segurança no tempo, do custo e do risco de execução do workflow. As duas abordagens apresentaram menor risco de segurança do que as abordagens da literatura, sob um custo e *makespan* razoáveis.

Palavras-chave: *workflow*, algoritmo, escalonamento, segurança, nuvem.

ABSTRACT

SHISHIDO, H.Y. **Workflow scheduling with sensitive task annotations for security and cost optimization in clouds**. 2019. 186 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2019.

The evolution of computers has enabled *in-silico* experiments to take place, including applications based on the workflow model. The execution of workflows is an activity that can be computationally expensive, where grids and clouds are adopted for its execution. In this context, the workflow scheduling algorithms allow meeting different execution criteria such as time and monetary cost. However, security is a criterion that has received attention because several organizations hesitate to deploy their applications in clouds due to the threats present in an open and promiscuous environment like the Internet. Security-oriented scheduling algorithms consider two scenarios: (a) hybrid clouds: holds tasks that manipulate sensitive data in the private cloud and export the other tasks to public clouds to satisfy some constraints (eg, time); (b) public clouds: considers the use of security services available in instances of virtual machines to protect tasks that deal with sensitive data. However, scheduling algorithms that consider the use of security services randomly select tasks without considering data semantics. This type of approach may end up assigning protection to non-sensitive tasks and wasting time and resources and leaving sensitive data without the necessary protection. In view of these limitations, two workflow scheduling approaches are proposed: Workflow Scheduling (WS-TSP) and Sensitive Annotation for Security Tasks (SAST). WS-TSP is a scheduling approach that uses a set of policies for task protection. SAST is another approach that allows using the Application Developer's prior knowledge to identify which tasks should be protected. WS-TSP and SAST consider implementing security services such as authentication, integrity verification, and encryption to protect sensitive tasks. The evaluation of these approaches was carried out through an extension of the simulator WorkflowSim that incorporates the overhead of security services in the execution time, the cost and the risk of execution The two approaches presented a lower security risk than the literature approaches, at a reasonable cost and *makespan*.

Keywords: *workflow*, algorithm, scheduling, security, cloud.

LISTA DE ILUSTRAÇÕES

Figura 1 – Codificação adotada pelas abordagens SCAS (LI *et al.*, 2016), FFBAT (ARUNARANI; MANJULA; VIJAYAN, 2017) e (SHISHIDO *et al.*, 2017). 31
Figura 2 – Ciclo de vida de um *workflow*. 40
Figura 3 – Exemplo de um *workflow*. 40
Figura 4 – Modelo organizacional de Nuvem de Computadores. 44
Figura 5 – Visão simplificada de uma solução baseada na Computação em Nuvem. . . 47
Figura 6 – Visão simplificada de uma solução baseada na Computação em Nuvem. . . 48
Figura 7 – Número de objetos armazenados na Amazon S3 entre o quarto trimestre de 2006 ao segundo trimestre de 2013. 49
Figura 8 – Métodos clássicos de otimização. 55
Figura 9 – Método da roleta. 60
Figura 10 – Dois cenários de mutação de um indíviduo filho. 62
Figura 11 – Representação do conceito de Dominância de Pareto. 64
Figura 12 – Conjunto de soluções e a Fronteira de Pareto. 65
Figura 13 – Diagrama da revisão do mapeamento sistemático. 69
Figura 14 – Publicações de algoritmos de escalonamento de workflows. O rótulo sobre cada barra representa as publicações por ano. 70
Figura 15 – Publicações de algoritmos de escalonamento de workflows voltados para grades ou nuvens. 71
Figura 16 – Publicações de algoritmos de escalonamento com foco na minimização de *makespan* ou satisfação de prazo. O rótulo sobre cada barra representa o número de publicações por ano. 71
Figura 17 – Publicações de algoritmos de escalonamento com foco na minimização de *custo* ou satisfação de orçamento. O rótulo sobre cada barra representa o número de publicações por ano. 72
Figura 18 – Publicações de algoritmos de escalonamento com foco na otimização do consumo de energia, confiabilidade e segurança. O rótulo sobre cada barra representa o número de publicações por ano. 73
Figura 19 – Países que realizam pesquisas em algoritmos de escalonamento de workflows. Países que apresentaram mais de seis publicações foram destacados em vermelho. 77
Figura 20 – Exemplo de um workflow. 90

Figura 21 – Visão geral do WorkflowSim. A área cercada por linhas tracejadas é suportada pelo CloudSim. 97
Figura 22 – Fluxo da simulação de execução de *workflows* adicionando *overhead* de segurança. 99
Figura 23 – Processo de aplicação dos serviços de segurança em uma tarefa. 100
Figura 24 – Exemplo de codificação de escalonamento para uso de otimização baseada em meta-heurísticas. 100
Figura 25 – Diagrama de classes da extensão WfSecurity. As classes na cor azul representam o núcleo do WorkflowSim, enquanto as brancas representam a extensão do WfSec. 102
Figura 26 – Métricas de execução de workflow. (a) Custo monetário aplicando serviços de segurança nas tarefas do workflow; (b) Tempo de execução aplicando serviços de segurança nas tarefas do workflow; (c) Risco de aplicação de serviços de segurança nas tarefas do workflow. 104
Figura 27 – Exemplo de codificação de escalonamento de um workflow de quatro tarefas. 109
Figura 28 – Operadores de mutação. (a) mutLimit e (b) mutStrong. 114
Figura 29 – Estrutura e migração da população. 115
Figura 30 – Custo total de execução para diferentes restrições de risco. 119
Figura 31 – Tempo de otimização das meta-heurísticas AG, MPGA e PSO na otimização do custo de execução do workflow Cybershake. 122
Figura 32 – Tempo de resposta das meta-heurísticas AG, MPGA e PSO na otimização do custo de execução do workflow Epigenomics. 122
Figura 33 – Tempo de resposta das meta-heurísticas AG, MPGA e PSO na otimização do custo de execução do workflow SIPHT. 123
Figura 34 – Ciclo de vida da execução de um workflow. 127
Figura 35 – Políticas de seleção de tarefas: (a) Entry e End; (b) Gather; (c) Scatter; (d) Gather e Scatter; (e) All. 129
Figura 36 – Processo de evolução do algoritmo NSGA-II. 131
Figura 37 – Processo de otimização do escalonamento de tarefas do workflow apoiado pela anotação de tarefas. 133
Figura 38 – Codificação adotada pela abordagem SCAS (LI *et al.*, 2016) e FFBAT (ARUNARANI; MANJULA; VIJAYAN, 2017). 133
Figura 39 – Codificação utilizada na abordagem baseada em anotações. 134
Figura 40 – Workflow Cybershake. Círculos representam as diferentes tarefas do workflow. As setas representam a dependência de dados entre as tarefas. 138
Figura 41 – Workflow Montage. Círculos representam as diferentes tarefas do workflow. As setas representam a dependência de dados entre as tarefas. 139
Figura 42 – Workflow Epigenomics. Círculos representam as diferentes tarefas do workflow. As setas representam a dependência de dados entre as tarefas. 140

Figura 43 – Workflow SIPHT. Círculos representam as diferentes tarefas do workflow. As setas representam a dependência de dados entre as tarefas. 141
Figura 44 – Makespan de diferentes políticas de seleção de tarefas em workflows usando as abordagens Fastest, Slowest, SCAS e WS-TSP. 147
Figura 45 – Custo para diferentes políticas de seleção de tarefas e workflows usando as abordagens Fastest, Slowest, SCAS e WS-TSP. 147
Figura 46 – Percentual de risco de execução para diferentes políticas de seleção de tarefas e workflows, considerando o algoritmo SCAS. 148
Figura 47 – O tempo de resposta de diferentes políticas de seleção de tarefas. 149
Figura 48 – Fronteiras de Pareto encontradas em diferentes condições de parada. 150
Figura 49 – Fronteiras de Pareto considerando custo e makespan. 151
Figura 50 – Custo monetário considerando diferentes quantidades de dados (por painel - bytes) e proporção de tarefas sensíveis (eixo X - %) usando as políticas *SAST*, *SCAS* e *ALL*. 154
Figura 51 – Tempo de sobrecarga acumulada das tarefas protegidas pelas diretivas *SAST*, *SCAS* e *ALL* considerando diferentes quantidades de dados (por painel - bytes) e proporção de tarefas sensíveis (eixo X - %). 155
Figura 52 – Tempo gasto na transferência de dados entre máquinas virtuais considerando três quantidades de dados. 156
Figura 53 – Makespan considerando diferentes quantidades de dados (por painel - bytes) e proporção de tarefas sensíveis (eixo X - %) usando as políticas *SAST*, *SCAS* e *ALL*. 158
Figura 54 – Desperdício para proteger tarefas não sensíveis. 158

LISTA DE ARQUIVOS

Arquivo 1 – Exemplo do uso do atributo secRequired na definição de tarefa sensível. . 99
Arquivo 2 – Exemplo de *Hello World* em um arquivo DAX 126
Arquivo 3 – Exemplo de uma descrição de um workflow no formato DAX usando o atributo *sensitive* para anotação de tarefa. 131

LISTA DE TABELAS

Tabela 1 – Instruções básicas para definição de um *workflow*. 41
Tabela 2 – Artigos de algoritmos de escalonamento de workflows mais citados (ordenado pelo número de citações). 74
Tabela 3 – Ranking de autores (ordenado pelo número de publicações). 75
Tabela 4 – Os dez periódicos com maior número de publicações sobre algoritmos de escalonamento de workflows (ordenada por ordem decrescente do número de publicações). N_1 : quantidade de publicações; N_2 : citações; N_2/N_1 : média de citações por artigo. 75
Tabela 5 – As dez conferências que mais publicaram artigos sobre algoritmos de escalonamento de workflows (ordenada por ordem decrescente do número de publicações). N_1 : quantidade de publicações; N_2 : citações; N_2/N_1 : média de citações por artigo. 76
Tabela 6 – Comparativo entre os algoritmos de escalonamento de *workflows* que consideram segurança. r-restrição, c-critério, M-Meta-heurística, H-Heurística, E-Estático, D-Dinâmico. 86
Tabela 7 – Algoritmos de segurança. 92
Tabela 8 – Lista de algoritmos de segurança adicionados à extensão WfSecurity. 103
Tabela 9 – Algoritmos de criptografia. 108
Tabela 10 – Parâmetros de geração da carga de trabalho de workflows. 117
Tabela 11 – Parâmetros usados nas meta-heurísticas. 118
Tabela 12 – Teste de Friedman para avaliação da otimização do custo obtido por cada método. 121
Tabela 13 – Teste estatístico de Nemeneyi para prover um ranking geral entre os métodos de otimização. A nota *A* representa a melhor posição no ranking, enquanto D a pior. 121
Tabela 14 – Visão geral dos perfis de execução dos workflows. Essa tabela resume as informações do perfil em nível de workflow. 141
Tabela 15 – Especificações dos nós do *Cluster* Halley 142
Tabela 16 – Configurações de máquinas virtuais consideradas no processo de otimização do escalonamento das tarefas do workflow. 142
Tabela 17 – Parâmetros usados nas meta-heurísticas. 142
Tabela 18 – Lista de algoritmos de segurança disponíveis para a simulação do overhead de serviços de segurança. 143

Tabela 19 – Descrição da carga de trabalho adotada na avaliação da abordagem WS-TSP. Fator: “Workflow” Níveis: Cybershake, Montage, Epigenomics e SIPHT. . . 143
Tabela 20 – Descrição dos tipos de tarefas em que há dados sensíveis na avaliação da abordagem WS-TSP. Fator: “Tipo de tarefa”. Níveis: No, Entry/End, Gather, Scatter, Gather/Scatter e All. 144
Tabela 21 – Descrição das abordagens de escalonamento. Fator: “Abordagem”. Níveis: WS-TSP, SCAS, Slowest e Fastest. 144
Tabela 22 – Desenho de experimentos para avaliação da abordagem WS-TSP. 144
Tabela 23 – Cenários de avaliação. 145
Tabela 24 – Trade-off entre o risco de vazamento de dados sensíveis e o custo monetário. 156

LISTA DE ABREVIATURAS E SIGLAS

ACM	*Association for Computing Machinery*
AE	Algoritmo Evolutivo
AG	Algoritmo Genético
Amazon EC2	*Amazon Elastic Cloud Computing*
Amazon S3	*Amazon Simple Storage Service*
CCGRID	*International Symposium on Cluster, Cloud and Grid Computing*
CE	Computação Evolutiva
CPU	*Central Processing Unit*
DAG	*Directed Acyclic Graph*
DBaaS	*Database as a Service*
DVS	*Dynamic Voltage Scaling*
EMO	*Evolutionary Multi-objective Optimization*
FGCS	*Future Generation Computer Systems*
GA	*Genetic Algorithm*
HEFT	*Heterogeneous Earliest Finish Time*
IaaS	*Infrastructure as a Service*
IC-PCP	*IaaS Cloud Partial Critical Paths*
IDC	*International Data Corporation*
IEEE	*Institute of Electrical and Electronics Engineers*
IF	*Impact Factor*
IO	*Input and Output*
MCDM	*Multi-Criteria Decision Making*
MCP	*Minimum Critical Path*
MPGA	Multi-population Genetic Algorithm
NFS	*Network File System*
NSGA-II	*Non-dominated Sorting Genetic Algorithm II*
PaaS	*Platform as a Service*
PRISMA	*Preferred Reporting Items for Systematic Reviews and Meta-Analyses*
PSO	*Particle Swarm Optimization*
PVFS	*Parallel Virtual File System*
QoS	*Quality of Service*

SaaS	*Software as a Service*
SAST	*Security-aware Sensitive Tasks*
SecaaS	*Security as a Service*
SGWf	Sistema Gerenciador de Worflows
STaaS	*Storage as a Service*
T.I.	Tecnologia da Informação
TPDS	*Transactions on Parallel and Distributed Systems*
VM	*Virtual Machine*
WaaS	*Workflow as a Service*
WS-TSP	*Workflow Scheduling Task Selection Policies*
XaaS	*Everything as a Service*
XML	*eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO 29
1.1 Contextualização 30
1.2 Motivação e objetivos 32
1.3 Estrutura da tese 33

2 REFERENCIAL TEÓRICO 35
2.1 Considerações iniciais 35
2.2 E-science 35
2.3 Workflows 38
2.4 Computação em nuvem 44
2.5 Escalonadores 50
2.6 Computação evolutiva 54

3 REVISÃO DA LITERATURA 67
3.1 Considerações iniciais 67
3.2 Mapeamento sistemático 67
3.3 Trabalhos relacionados 76
3.4 Considerações finais 86

4 MODELOS DO SISTEMA E FORMULAÇÃO DO PROBLEMA . . 89
4.1 Considerações iniciais 89
4.2 Modelo de workflow 89
4.3 Modelo de cloud datacenter 90
4.4 Modelo de overhead de segurança 90
4.5 Modelo de execução de tarefa 92
4.6 Modelo de análise de risco e desperdício de segurança 93
4.7 Formulação do problema 93
4.8 Considerações finais 94

5 SIMULADOR PARA AVALIAÇÃO DE ALGORITMOS DE ESCALONAMENTO 95
5.1 Considerações iniciais 95
5.2 Segurança em execução de *workflows* em nuvens 95
5.3 Estudo de caso: WorkflowSim 97

5.4 Requisitos para simulação de segurança de workflows 98
5.5 Extensão WfSecurity 101
5.6 Exemplo de uso 102
5.7 Considerações finais 104

6 ESTUDO EXPLORATÓRIO DE METAHEURÍSTICAS 107
6.1 Considerações iniciais 107
6.2 Estratégia de codificação 107
6.3 Geração do escalonamento 108
6.4 Função *fitness* 111
6.5 Enxame de partículas 111
6.6 Algoritmo genético 112
6.7 Algoritmo genético multi-populacional 114
6.8 Comparação PSO vs. AG vs. MPGA 116
6.9 Comparação do custo de execução 119
6.10 Validação estatística 120
6.11 Análise do tempo de resposta 121
6.12 Considerações finais 123

7 MÉTODOS DE ESCALONAMENTO PARA A PROTEÇÃO DE DADOS SENSÍVEIS 125
7.1 Considerações iniciais 125
7.2 Ciclo de vida 126
7.3 Políticas de seleção de tarefas 128
7.4 Otimização multi-objetivo 130
7.5 Anotação de tarefas sensíveis 131
7.6 Codificação do escalonamento 132
7.7 Decodificação 134
7.8 Métricas de avaliação 134
7.9 Considerações finais 136

8 AVALIAÇÃO EXPERIMENTAL 137
8.1 Considerações iniciais 137
8.2 Caracterização da carga de trabalho 137
8.3 Ambiente de execução 141
8.4 Desenho de experimentos - WS-TSP 143
8.5 WS-TSP vs. SCAS 146
8.5.1 Avaliação Mono-Objetivo *146*
8.5.2 Otimização multi-objetivo *149*
8.6 SAST vs. SCAS 153

9 CONCLUSÃO 161
9.1 Argumentação geral 161
9.2 Contribuições 162
9.3 Publicações 163
9.4 Dificuldades 164
9.5 Trabalhos futuros 165

REFERÊNCIAS 167

CAPÍTULO

1

INTRODUÇÃO

A evolução do poder de processamento dos computadores nas últimas décadas tem possibilitado a realização de experimentos *in-silico*, oferecendo várias vantagens tais como: (a) maior precisão e qualidade dos dados experimentais; (b) acesso a amplo conjunto de dados experimentais; (c) simulações mais precisas considerando modelos mais sofisticados; (d) experimentos individuais mais rápidos, e; (e) ambiente controlado permitindo melhor reprodutibilidade dos experimentos. No entanto, a experimentação *in-silico* sofre com o aumento da complexidade de configurar, manter e fazer mudanças nos sistemas de simulação experimental.

A *e-Ciência* ou *e-Science* se estabeleceu como uma ramificação na ciência que vai além da teoria e experimentação. Seu principal objetivo é promover a colaboração interdisciplinar no ciclo de vida de pesquisas que envolvem processamento intensivo ou armazenamento de grande volume de dados oferecendo métodos de manipulação, extração, análise, reuso e compartilhamento de dados (HEY; TREFETHEN, 2002a). Contudo, esse processo recorrente torna-se custoso e, muitas vezes, inviável e de difícil reprodução.

Diante desse contexto, o conceito de workflow passou a ser aplicado em atividades de *e-Science*. Um workflow é uma especificação de um conjunto de tarefas e dependências entre elas que devem ser satisfeitas para atingir um determinado fim (DEELMAN *et al.*, 2009), e podem ser representados por um Grafo Acíclico Direcionado. Workflows podem ser classificados em aplicações de negócio ou de cunho científico. A primeira sofre poucas modificações e requerem segurança e integridade das transações. Por outro lado, workflows científicos passam por frequentes adaptações para atender as necessidades de diferentes organizações, podendo envolver um grande volume de dados (BARGA; GANNON, 2007).

Tecnicamente, um workflow pode ser desenvolvido em qualquer linguagem de programação moderna. Entretanto, a codificação de um programa em linguagens como Java, Python ou Perl para orquestrar a execução de um conjunto de tarefas em um ambiente distribuído vai além do conhecimento da maioria dos cientistas.

A execução de workflows científicos é uma atividade, na maioria das vezes, computacionalmente dispendiosa. Utilizar uma única estação de trabalho para a execução de um workflow científico pode ser inviável em função do longo tempo necessário para a execução de centenas e até milhares de tarefas envolvidas em uma única aplicação. Desse modo, arquiteturas distribuídas como grades e nuvens têm sido empregadas para a execução de workflows.

Considerando a dificuldade de programação e execução de workflows científicos, sistemas gerenciadores de workflows (SGWf) oferecem um ambiente de programação para facilitar a orquestração de experimentos computacionais baseados em workflows (DEELMAN; JUVE; BERRIMAN, 2012). Entre as diversas funções de um SGWf, pode-se citar a capacidade de definição, modificação, monitoramento e compartilhamento de workflows. A distribuição das tarefas de um workflow para os recursos computacionais existentes é realizado por meio do módulo de escalonamento de um SGWf de modo a atender critérios e restrições. Um critério ou objetivo é uma necessidade que influencia diretamente na execução das tarefas de um workflow. A minimização do tempo ou custo de execução, bem como a maximização do uso geral dos recursos computacionais são exemplos de critérios de escalonamento. A restrição de escalonamento é uma limitação imposta ao escalonador que precisa ser satisfeita como, por exemplo, executar uma aplicação dentro de um prazo ou orçamento. Logo, um algoritmo de escalonamento de workflow procura otimizar um conjunto de n critérios e m restrições, onde $n \geq 1$ e $m \geq 0$. Por exemplo, minimizar o tempo de execução, considerando um limite de orçamento disponível.

1.1 Contextualização

Diante de tantos cenários de execução de workflows, várias abordagens de escalonamento têm sido propostas na literatura como as identificadas em Rodriguez e Buyya (2016). O *makespan* ou tempo total de execução de um workflow é um critério dominante na maioria dos algoritmos de escalonamento desde a era da computação em grade. Já o custo é um critério relevante no escalonamento de workflow em nuvens cujo o uso de máquinas virtuais é pago (ZHENG; SAKELLARIOU, 2013a). Em um caso mais flexível em que o tempo e o custo podem ser relaxados, é possível obter soluções que apresentam múltiplos *trade-off* entre os dois critérios como proposto em Grandinetti, Pisacane e Sheikhalishahi (2013).

Organizações hesitam em implantar e armazenar suas aplicações e dados em nuvens, sendo a segurança o principal motivo (HASHIZUME *et al.*, 2013). Várias empresas relutam em confiar na computação em nuvem para transferir seus ativos digitais para serviços terceirizados (LATIF *et al.*, 2014). As organizações admitem confiar no provedor de serviços em nuvem, mas não confiam nos outros usuários que compartilham recursos no mesmo provedor (ALI; KHAN; VASILAKOS, 2015). Assim, muitas dessas organizações preferem manter seus ativos digitais no domínio administrativo de sua própria infraestrutura de T.I.

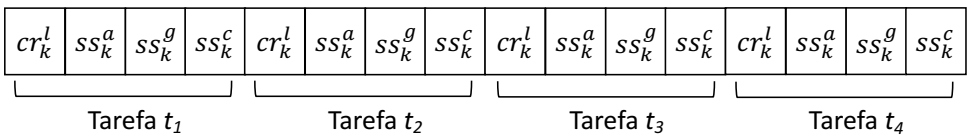
Diante destas ameaças, abordagens de escalonamento de workflows em nuvem consi-

derando a segurança têm recebido atenção nos últimos anos (SHISHIDO; ESTRELLA, 2017). Esses algoritmos contemplam dois cenários. O primeiro considera organizações que dispõem de sua própria nuvem (privada) para a execução de workflows. Caso a complexidade de execução do workflow inviabilize a execução dentro de um prazo determinado, parte das tarefas são exportadas para nuvens públicas para satisfazer tal prazo. O cerne dessa ideia é manter as tarefas sensíveis (que manipulam dados sensíveis) na nuvem privada da organização, enquanto as não sensíveis são processadas em nuvem pública. De acordo com a revisão realizada nesta tese, a maioria dos algoritmos de escalonamento com base em segurança consideram essa abordagem como em Watson (2011), Sharif *et al.* (2013), Sharif *et al.* (2014), Zeng, Veeravalli e Li (2015), Chen *et al.* (2017), Wen *et al.* (2018).

O segundo cenário consiste em organizações que não possuem uma nuvem privada e, assim, têm a necessidade de instanciar uma infraestrutura em nuvem pública para a execução de seus workflows. Entretanto, como discutido anteriormente, nuvens públicas estão sujeitas a ataques que podem comprometer a segurança de workflows. Falsificação de autenticidade, corrupção/alteração de dados e interceptação de dados são exemplos de ataques na nuvem (TEP *et al.*, 2015).

Neste contexto, algoritmos de escalonamento utilizam instâncias de VMs com serviços de segurança para a proteção de dados sensíveis de workflows. No estudo de Li *et al.* (2016) foi proposto um algoritmo de escalonamento para minimizar o custo de execução e atender a um requisito mínimo de segurança e restrição de prazos em nuvem pública. Eles propuseram uma otimização baseada em Enxame de Partículas para encontrar o escalonamento apropriado otimizando os tipos de VM e os serviços de segurança para cada tarefa. A codificação adotada segue o padrão apresentado na Figura 1. Quatro parâmetros são considerados na codificação de cada tarefa: (a) tipo de VM; (b) autenticação; (c) verificação de integridade e; (d) confidencialidade. Logo, o tamanho da codificação de um workflow com n tarefas é dado por $D = 4 \cdot n$. Por exemplo, um workflow com 4 tarefas tem 16 parâmetros. Os parâmetros de 0 a 3 correspondem à tarefa 1, os parâmetros de 4 a 7 correspondem à tarefa 2 e assim por diante.

Figura 1 – Codificação adotada pelas abordagens SCAS (LI *et al.*, 2016), FFBAT (ARUNARANI; MANJULA; VIJAYAN, 2017) e (SHISHIDO *et al.*, 2017).



Fonte: Elaborada pelo autor.

Shishido *et al.* (2017) realizou um estudo comparativo entre o método de otimização proposto em Li *et al.* (2016) (Enxame de Partículas) e a sua proposta baseada em Algoritmos Genéticos e Algoritmos Genéticos de Múltipopulações. O mesmo problema foi otimizado por meio da combinação das meta-heurísticas de Vagalumes e Morcegos em Arunarani, Manjula e

Vijayan (2017).

Entretanto, ao utilizar essa codificação, esses três algoritmos não garantem que serviços de segurança serão atribuídos a todas as tarefas sensíveis do workflow, pois a semântica das tarefas não é levada em consideração. A métrica de segurança é calculada sobre a quantidade de tarefas protegidas e não em relação a quantidade de tarefas sensíveis protegidas. Por exemplo, supondo um workflow de quatro tarefas *A* - *B* - *C* - *D*, onde as tarefas *A* - *B* - *C* não são sensíveis e a tarefa *D* manipula dados sensíveis. Esses algoritmos podem produzir uma solução s_1 que aplica serviços de segurança nas tarefas *A* - *B* - *C* e não associar nenhum serviço de segurança para *D*. Uma outra solução s_2 pode contemplar a associação de serviços de segurança para as tarefas *A* - *C* - *D*, enquanto que *B* não recebe qualquer proteção.

Logo, nota-se que os algoritmos de escalonamento existentes para escalonar workflows em nuvens públicas deixam a seleção de tarefas para um método de otimização sem considerar a semântica das tarefas. Isso pode produzir escalonamentos em que tarefas sensíveis são associadas a instâncias de VMs que não apresentem qualquer serviço de proteção.

1.2 Motivação e objetivos

A consolidação da *e-Science* e a colaboração interdisciplinar no ciclo de vida de pesquisas que envolvem o processamento intensivo ou armazenamento de grande volume de dados tem fomentado cada vez mais o uso e o desenvolvimento de workflows. A possibilidade de redução de custos, isenção da manutenção da infraestrutura, escalabilidade quase ilimitada e acesso a uma ampla rede de servidores de alta disponibilidade contribuem para a execução de workflows científicos. Contudo, o uso de nuvens públicas para a execução de workflows que envolvem dados sensíveis pode resultar em perdas irreparáveis.

Nesse sentido, algoritmos de escalonamento com atenção na segurança de dados sensíveis têm sido propostos nos últimos anos. Porém, as abordagens podem realizar a proteção de tarefas incorretas, facilitando o êxito de ataques maliciosos. Além disso, pode haver o desperdício de processamento, prolongamento do tempo de execução e aumento do custo de execução ao aplicar serviços de segurança em tarefas não sensíveis.

Diante do contexto apresentado, a hipótese desta tese é:

> A identificação das tarefas sensíveis no processo de modelagem do workflow aliada ao uso de uma meta-heurística para otimização das instâncias de máquinas virtuais pode reduzir o risco de segurança e contribuir na redução do tempo e do custo total de execução.

Os objetivos desta tese são os seguintes:

- Proposta de políticas de seleção de tarefas para proteger dados sensíveis;

- Introdução do conceito de anotação de tarefas sensíveis na fase de modelagem do workflow;
- Elaboração de um algoritmo de escalonamento de workflow que minimize o custo de execução e satisfaça um prazo definido pelo usuário em nuvens públicas.

O processo de identificação de lacuna, proposição e avaliação desse projeto seguiu os seguintes passos:

- Mapeamento bibliográfico dos algoritmos de escalonamento de workflows;
- Identificação de um critério de escalonamento pouco explorado na literatura;
- Extensão do ambiente de simulação para fornecer meios para avaliar algoritmos de escalonamento com foco em segurança;
- Comparação de meta-heurísticas (Enxame de Partículas, Algoritmo Genético e Algoritmo Genético Multi-populacional) para definir qual deles se adequa ao algoritmo de escalonamento proposto;
- Otimização das abordagens propostas utilizando meta-heurística multiobjetivo.

1.3 Estrutura da tese

Este trabalho está organizado em mais oito capítulos:

- O Capítulo 2 apresenta a base teórica utilizada neste estudo;
- O Capítulo 3 retrata uma análise bibliométrica de algoritmos de escalonamento de workflows e apresenta brevemente os principais artigos de acordo com cada critério de escalonamento;
- O Capítulo 4 descreve os modelos do sistema e a formulação do problema desta tese;
- O Capítulo 5 descreve a proposta e o desenvolvimento de uma extensão do simulador WorkflowSim para a avaliação das abordagens de escalonamento baseadas em serviços de segurança;
- O Capítulo 6 compara as meta-heurísticas de Enxame de Partículas, Algoritmo Genético e Algoritmo Genético Multi-populacional na otimização de escalonamento de workflows;
- O Capítulo 7 apresenta a metodologia de identificação de tarefas sensíveis WS-TSP e SAST e o algoritmo para decodificação de escalonamento;
- O Capítulo 8 avalia as abordagens WS-TSP e SAST com os algoritmos existentes na literatura;

- Por fim, o Capítulo 9 realiza as considerações finais, as contribuições, as publicações derivadas desta pesquisa, as dificuldades enfrentadas e os trabalhos futuros.

CAPÍTULO

2

REFERENCIAL TEÓRICO

2.1 Considerações iniciais

O escalonamento de workflows científicos em nuvem é um tema multidisciplinar inserido no contexto de e-ciência. A definição de *workflow* e os demais conceitos relacionados em sua representação e execução são abordados neste capítulo. Além disso, a descrição de sistemas de gerenciamento de *workflows*, a teoria de escalonamento, aspectos de segurança e alguns métodos de otimização do escalonamento de workflows são apresentados. Esses conceitos são fundamentais para a compreensão dos próximos capítulos, incluindo a proposta da tese.

2.2 E-science

A *e-science* é um nome que foi proposto por John Taylor em 2001, o qual era diretor do Conselho de Pesquisa do Centro de Ciências e Tecnologia do Reino Unido. A ideia desse termo foi contemplar todas as pesquisas que são realizadas em todas as áreas do conhecimento e que têm a necessidade de lidar com métodos computacionais avançados e grandes volumes de dados (HEY; TREFETHEN, 2002b). O forte apelo da ciência para simular fenômenos complexos alavancou o interesse pela *e-science*. A pesquisa em *e-science* abrange todo o processo de pesquisa, desde o desenvolvimento de ferramentas que auxiliem a formulação de problemas de pesquisa, coleta e análise de dados, até a modelagem, a simulação, a divulgação, a reprodutibilidade e o reuso dos resultados. Pesquisas em *e-science* podem ser descritas pelas seguintes características:

- colaboração entre cientistas da Computação e pesquisadores de outras áreas do conhecimento;
- criação de tecnologias computacionais para lidar com grande volume de dados e para

executar simulações e programas que exigem sistemas computacionais complexos.

A *e-science* dá atenção tanto para o gerenciamento de dados quanto para o desenvolvimento de software. O desafio no gerenciamento de dados está na solução do gargalo presente na pesquisa científica: organizar, classificar, selecionar, compartilhar e garantir acesso ao grande volume de dados gerados nos últimos anos em todas as áreas do conhecimento. O cruzamento dessa grande quantidade de dados proporciona análises mais abrangentes e acelera as descobertas na ciência, tanto que, comumente, a *e-science* é também conhecida como *Data-driven Science*. É a análise computacional de grandes quantidade de dados que possibilita a descoberta de associações entre fatos produzindo, consequentemente, conhecimento. Por outro lado, há a ênfase no projeto de algoritmos que permitam acelerar ou desenvolver simulações que favoreçam novas descobertas científicas. Esses algoritmos precisam ser capazes de explorar o hardware de forma adequada, pois não basta ter máquinas avançadas se o algoritmo não é capaz de tirar proveito do potencial oferecido pelo hardware.

As simulações geralmente são caracterizadas como *data-intensive*, pois envolvem o processamento e/ou armazenamento de uma grande quantidade de dados (HEY; TANSLEY; TOLLE, 2009; HEY, 2012). Esse grande volume de dados é originado a partir de diversas tecnologias e colaborações entre pessoas. O uso de equipamentos de leituras automatizadas, sensores, experimentos com dados retroalimentados, bases de dados alimentadas por vários cientistas são exemplos de fontes de dados experimentais. A formulação de modelos e abordagens para análise desses dados também aumenta a colaboração entre instituições de pesquisa e cientistas. Contudo, o armazenamento e a análise dos dados exigem uma infraestrutura computacional poderosa, distribuída e escalável para que vários usuários possam analisar dados a partir de qualquer local com acesso à Internet.

A partir dos conceitos apresentados, há ao menos três necessidades para o avanço da *e-science*:

- a implantação de uma infraestrutura computacional distribuída com acesso ubíquo capaz de realizar processamento em larga escala;
- a geração e o uso intensivo de dados; e
- oferecer ambientes de colaboração entre pesquisadores, grupos de pesquisa ou instituições por meio do compartilhamento de esforços, dados e recursos computacionais.

Essa concepção é reforçada por Schroeder (2008) ao descrever que a *e-science* utiliza tecnologias computacionais para o compartilhamento de recursos digitais na pesquisa acadêmica ou científica e a promoção da colaboração global.

Infraestrutura computacional para e-science

A infraestrutura computacional para *e-science* pode ser constituída de diferentes soluções como grades, nuvens, *middleware*, *webservices*, *workbenches*, tecnologias de notação, armazenamento de dados como XML e outras estruturas.

A computação em grade ou *grid computing* emergiu na década de 90 como um novo campo que se distinguia da computação distribuída convencional pelo compartilhamento de recursos em grande escala, aplicações inovadoras e, em alguns casos, capacidade de alto desempenho (FOSTER; KESSELMAN; TUECKE, 2001). O termo *grid* faz analogia com a rede de energia elétrica de modo que esta poderia estar disponível para qualquer pessoa em qualquer lugar (HEY; TREFETHEN, 2005). As grades computacionais oferecem infraestrutura de hardware e software que fornece acesso a recursos de alto poder de processamento confiáveis e de baixo custo (FOSTER; KESSELMAN, 1999). A confiabilidade está relacionada à velocidade de transmissão de dados, ao poder de processamento e aos serviços de software e à segurança. A consistência expressa padronização das interfaces e operações de modo que todos os usuários sejam capazes de utilizar os recursos oferecidos pela grade. A abrangência determina que os serviços da grade possam ser acessados a partir de qualquer sistema que deseja implementá-lo. O baixo custo é um fator que evolui conforme a consolidação das tecnologias.

O *middleware* consiste no software para disponibilizar os recursos oferecidos nas grades. O *middleware* faz a comunicação entre a infraestrutura computacional das grades e as aplicações individuais de pesquisadores ou instituições. Além disso, permite que um usuário compartilhe e acesse recursos de outros usuários. Para facilitar o uso de serviços oferecidos pelas grades, ambientes de trabalho (*workbench*) têm sido propostos. Estes ambientes propiciam a criação de projetos e a automatização de experimentos por meio da integração de sensores, dispositivos, recursos e serviços dentro de uma grade (FOSTER; KESSELMAN, 2004).

Um dos possíveis métodos para a acoplagem a um *middleware* é por meio de um serviço web. O serviço web é uma tecnologia de computação distribuída que oferece interoperabilidade entre sistemas por meio do encapsulamento de recursos internos dentro de um serviço. Um serviço web provê uma camada lógica de aplicação entre recursos e os consumidores. Assim, os proprietários do serviço ficam livres para evoluir sua estrutura interna ao longo do tempo (por ex.: melhorar o desempenho ou confiabilidade), sem fazer modificações nos padrões de troca de mensagens usados pelos consumidores de serviço. Isso possibilita o fraco acoplamento entre consumidores e provedores de serviço, fator essencial para a construção de sistemas corporativos robustos, pois nenhum dos envolvidos está sob controle total de todas as partes da aplicação distribuída.

Grande volume de dados em e-science

O processamento de grande volume de dados gerados por atividades de pesquisa é um dos motivos para a implantação de infraestrutura computacional descrita na seção anterior. A produção de grandes quantidades de dados é um fato recorrentemente observado e identificado como o movimento *Big Data*, em que o volume de dados supera a capacidade do processamento convencional. Esses dados podem variar em volume, velocidade de crescimento e variabilidade, dificultando seu processamento e armazenamento em sistemas de banco de dados tradicionais (WALKER, 2014). Para resolver este problema, há a necessidade de recorrer a métodos alternativos de processamento e armazenamento.

O termo *e-science* por si só denota o uso intensivo de dados e seu objetivo é exclusivamente voltado à pesquisa científica. Um aspecto que diferencia a *e-science* do simples processamento em *big data* é a colaboração entre cientistas e a utilização de recursos compartilhados para o processamento de dados. Três fatores são considerados primordiais quanto ao uso de dados na *e-science*: captura, curadoria e análise (HEY; TANSLEY; TOLLE, 2009). A captura é o processo em que os dados são gerados por equipamentos de captura ou por meio de simulações computacionais prévias, processados por software e armazenados em banco de dados. Neste processo, vários pesquisadores ainda precisam codificar suas informações para que possam ser disponibilizadas com outros usuários. Para isso é necessário representar os dados de acordo com os padrões dos objetos em análise como, por exemplo, genes, temperaturas, galaxias, etc. A curadoria consiste na administração para preservar os recursos digitais durante todo o seu ciclo de vida, permitindo que as gerações atuais e futuras possam ter acesso aos dados (BEAGRIE, 2008). Por fim, a análise abrange o uso de tecnologias que ofereçam capacidade computacional para procedimentos analíticos de cientistas em diferentes localidades.

2.3 Workflows

Workflows podem ser descritos por meio de um grafo acíclico direcionado (DAG), onde cada nó é considerado uma tarefa e as arestas representam as suas dependências (YU; BUYYA, 2005; LIU *et al.*, 2014). Eles permitem modelar facilmente aplicações compostas por múltiplos passos como, por exemplo, recuperar dados de um instrumento/base de dados, reformatá-los e analisá-los (BARGA; GANNON, 2007). Uma das principais propriedades de um *workflow* é o processamento intensivo e, principalmente, voltado à manipulação de grande volume de dados.

Um *workflow* pode ser composto de um conjunto de tarefas totalmente sequenciais a tarefas paralelas, que estejam conectadas por outras sequenciais para o pré e pós-processamento de dados (DEELMAN *et al.*, 2009). Para exemplificar a fase de pré-processamento, pode-se considerar uma tarefa principal que necessite de dados formatados em um determinado padrão. Para isso, pode-se associar uma tarefa que precede a execução da tarefa principal para que tal formatação seja executada. Ao final da execução, o resultado da tarefa principal é utilizado como

entrada à tarefa de pós-processamento que analisa os dados e armazena os resultados em uma base de dados.

Um *workflow* pode ser classificado em científico ou de negócio (BARGA; GANNON, 2007). O primeiro usa um sistema gerenciador para compor e executar uma série de passos computacionais ou de manipulação de dados voltados às aplicações científicas. Esse tipo de *workflow* é modelado por cientistas ou adquirido por meio de repositórios compartilhados e modificados, conforme a necessidade do pesquisador. Geralmente, *workflows* científicos manipulam um grande volume de dados e realizam computação intensiva. Por outro lado, *workflow* de negócio é caracterizado por um padrão repetitivo de atividades de negócio para processar informações como, por exemplo, uma transação bancária. A modelagem desse tipo de *workflow* é realizada por especialistas para otimizar os requisitos definidos pelo usuário.

Tanto a execução de *workflows* de negócio quanto a de científicos em nuvem está sujeita a riscos de segurança. Por exemplo, um *workflow* de uma transação bancária deve considerar aspectos como autenticidade da transação, integridade dos dados e a confidencialidade da informação durante o processamento e a comunicação entre as tarefas. Caso um desses aspectos não seja cumprido, a instituição bancária e o cliente estarão sujeitos à ataques maliciosos. Aplicações científicas eventualmente manipulam dados sensíveis[1]. Para exemplificar esse cenário, pode-se considerar pesquisas de uma indústria farmacêutica que executa um *workflow* para simular a eficiência de novas drogas. O vazamento de informações sigilosas seria desastroso a uma instituição que investe milhões de dólares por ano. Portanto, tratar os riscos de segurança de acordo com a natureza da aplicação é primordial quando *workflows* são executados em plataformas como a computação em nuvem.

Deelman *et al.* (2009) define o ciclo de vida de um *workflow* em fases de composição, mapeamento, execução e proveniência. A Figura 2 apresenta a interação entre essas fases que estão explicadas a seguir.

1. **Composição**: associação de tarefas e dados por meio de ferramentas, utilizando serviços web ou software local;

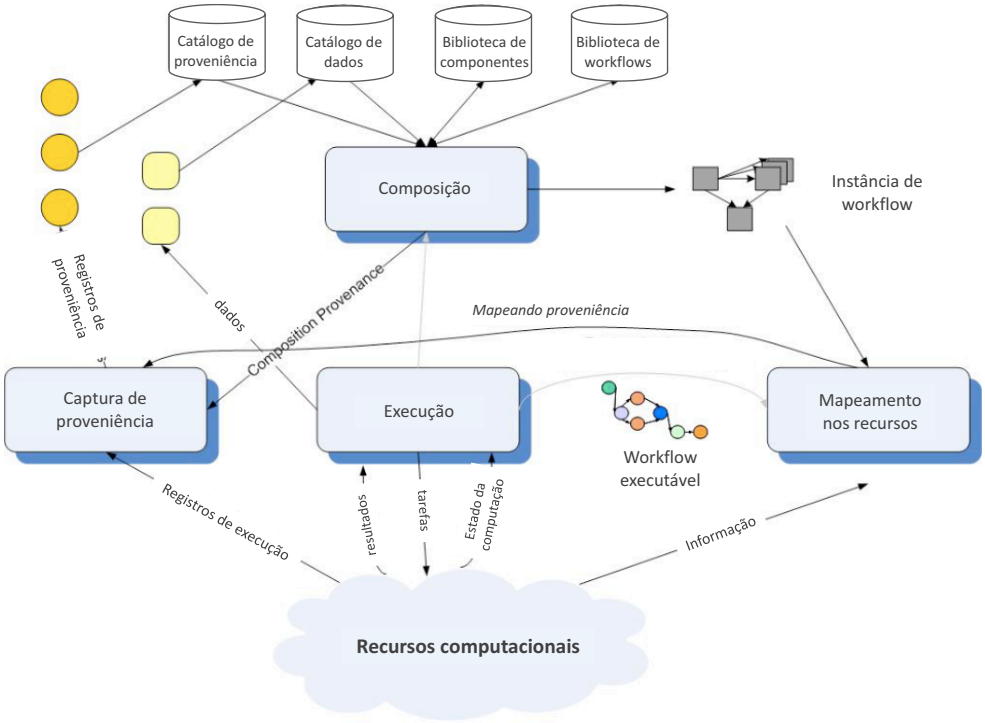
2. **Mapeamento**: vinculação do modelo de *workflow* de acordo com os recursos computacionais disponíveis;

3. **Execução**: realização das atividades do *workflow* nos recursos previamente mapeados;

4. **Proveniência**: registro de metadados e informações de proveniência do ciclo de vida do *workflow*. Tais informações permitem a reprodução de um experimento científico sob as mesmas condições anteriores (FREIRE *et al.*, 2008).

[1] Dados cruciais para uma instituição e que não podem ser acessados por pessoas não autorizadas.

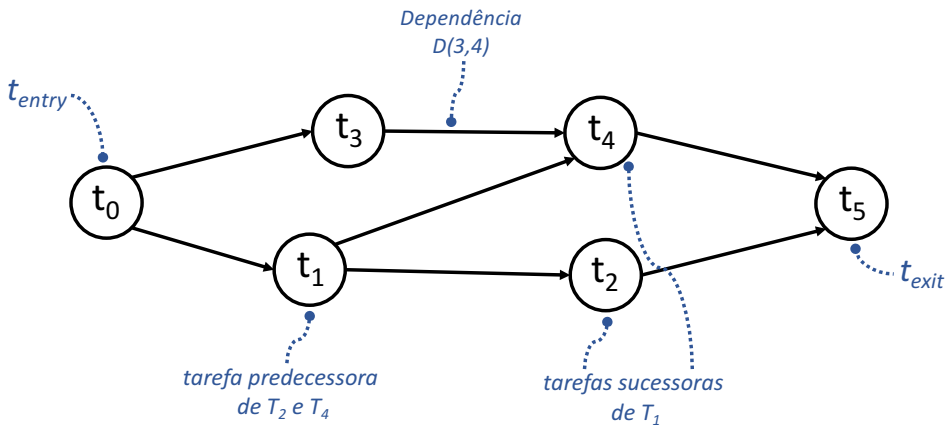
Figura 2 – Ciclo de vida de um *workflow*.



Fonte: Adaptada de Deelman *et al.* (2009).

A composição de um *workflow* é dada pela associação de tarefas e suas dependências (Figura 3). Cada tarefa é definida por uma ação que recebe dados de entrada e produz dados de saída que geralmente servirão como entrada para uma tarefa conectada posteriormente (LUDäSCHER *et al.*, 2006). As conexões que interligam atividades podem ser consideradas como fluxo de dados ou fluxo de controle. O fluxo de dados representa a dependência entre as atividades e os dados, bem como o modo que estes são usados no *workflow*. O fluxo de controle é definido pelas conexões entre tarefas representando a transferência de controle através de estruturas condicionais e de repetição (SHIELDS, 2007).

Figura 3 – Exemplo de um *workflow*.



Fonte: Elaborada pelo autor.

Na fase de mapeamento é realizado o escalonamento das tarefas aos recursos computacionais disponíveis, o qual afeta diretamente no desempenho da execução de um *workflow*. Nessa fase podem ocorrer otimizações para atender restrições definidas pelo usuário como, por exemplo, minimizar o tempo de execução ou limitar a execução a um orçamento. Depois do início da execução, as informações de proveniência e metadados são registrados, assim como os dados coletados da execução.

Um *workflow* pode ser modelado por meio de uma ferramenta gráfica ou uma linguagem para definição do fluxo de controle ou fluxo de dados. Há ainda linguagens híbridas que oferecem os dois tipos de fluxos e possibilitam a representação de *workflows* mais elaborados. Embora haja uma variedade de instruções para fluxo de controle e de dados, é possível modelar *workflows* elaborados com um conjunto reduzido de instruções. A Tabela 1 apresenta as instruções básicas para controle de fluxo segundo a definição de Aalst *et al.* (2003), WFMC (1999):

Tabela 1 – Instruções básicas para definição de um *workflow*.

Instrução	Definição
Sequência	uma tarefa é iniciada após a conclusão da anterior.
Divisão Paralela (*AND-Split*)	um fluxo de execução se divide em dois ou mais fluxos que são executados paralelamente.
Sincronização (*AND-Join*)	ponto de sincronização em que dois ou mais fluxos convergem em um único fluxo.
Junção Simples (*XOR-Join*)	ponto em que dois ou mais fluxos convergem sem a necessidade de sincronização.
Iteração	instrução para repetição de um bloco de tarefas até que uma determinada condição seja atendida.
Composição	um *workflow* pode ser iniciado por outro *workflow* como parte do seu fluxo de execução.

Fonte: WFMC (1999).

Juve *et al.* (2013a), Russell *et al.* (2005) estabelecem as instruções que influenciam na composição e execução de *workflows*, as quais seguem:

- **Pipeline**: é composto por tarefas sequencialmente combinadas, em que cada tarefa recebe como entrada a saída da tarefa anterior;
- **Distribuição de dados**: os dados de saída de uma tarefa servem como entrada para múltiplas tarefas. Ou, um conjunto de dados é particionado em sub-conjuntos para serem processados por outras tarefas;
- **Agregação de dados**: tarefas executadas em múltiplos conjuntos de dados de entrada que ao final geram uma saída de dados combinada;
- **Redistribuição de dados**: tarefas que atuam como ponto de sincronização, recebendo múltiplos conjuntos de dados como entrada e geram várias saídas como resultado.

Diante destas definições, sistemas gerenciadores de *workflows* apoiam o processo de modelagem, execução e monitoramento de recursos e são apresentados na próxima seção.

Sistemas Gerenciadores de workflows

Um Sistema Gerenciador de *workflows* (SGWf) é um sistema para definir, modificar, gerenciar, monitorar e executar *workflows* seguindo um fluxo lógico (LIN *et al.*, 2009). *Workflows* possuem particularidades como a necessidade do acompanhamento de sua execução, suporte à reprodução de experimentos, integração de serviços heterogêneos, suporte à computação de alto desempenho, monitoramento e tolerância a falhas e interoperabilidade.

Os SGWf geralmente são organizados em subsistemas, os quais são brevemente apresentados a seguir:

- **Subsistema de Projeto de *workflow***: responsável pela criação e modificação de *workflows*. Este subsistema produz as especificações de um *workflow*, utilizando uma linguagem para modelagem ou através de interface gráfica;
- **Subsistema de Apresentação e Visualização**: permite visualizar o *workflow* em diferentes formatos, assim como os dados produzidos e as informações de proveniência;
- **Subsistema de Execução**: é a principal parte de um SGWf que provê um ambiente de gerenciamento e execução de acordo com o modelo do *workflow*;
- **Subsistema de Monitoramento**: fornece informações do estado de execução de um *workflow* e é capaz de tratar eventuais falhas durante a execução;
- **Subsistema de Gerenciamento de Tarefas**: trata dos problemas de heterogeneidade de serviços e distribuição das tarefas;
- **Subsistema de Gerenciamento de Proveniência**: responsável pelo gerenciamento de metadados oriundos da proveniência, incluindo a sua representação, armazenamento, busca e consulta;
- **Subsistema de Gerenciamento de Dados**: realiza o tratamento de dados heterogêneos em ambientes distribuídos.

Ambientes de Execução

O processamento de *workflows* pode contemplar várias tarefas que levam considerável tempo, pois envolvem uma grande quantidade de dados. Devido a essa característica, ambientes de alto desempenho como aglomerados, grades e, mais recentemente, nuvens podem ser utilizados para viabilizar a execução de *workflows* em tempo razoável.

Um aglomerado ou *cluster* é uma coleção de estações de trabalhos (nós) conectadas por meio de uma rede de alta velocidade e baixa latência (MARK; RAJKUMAR, 1999). Os nós de um aglomerado comunicam-se entre si por meio de bibliotecas de modo que sejam vistos como um único sistema. Cada nó pode ser uma arquitetura monoprocessada ou multiprocessada e cada processador pode apresentar um ou mais núcleos.

Após a arquitetura baseada em aglomerado, as grades surgiram como um serviço de compartilhamento de processamento e armazenamento. A ideia de ambientes em grade é semelhante às redes de distribuição de energia elétrica, em que diferentes usinas geradoras estão geograficamente interconectadas fornecendo energia aos usuários, sem que estes saibam a origem e o modo como ela é gerada. Analogamente, a arquitetura em grade é caracterizada como um sistema que gerencia recursos que não estão sujeitos a um controle centralizado (KLAUS; RAJKUMAR; MUTHUCUMARU, 2002). Os nós utilizam protocolos e interfaces padronizadas e abertas com o intuito de atender diferentes requisitos de qualidade de serviço (QoS) (FOSTER *et al.*, 2008). Os recursos disponíveis podem ser unidades de processamento, armazenamento, conexões de rede, software e dados que são compartilhados por instituições ou usuários independentes em diferentes esferas administrativas.

Sistemas em grade possuem peculiaridades que envolvem recursos heterogêneos, dinâmicos, voláteis, amplamente distribuídos e controlados de forma descentralizada (KLAUS; RAJKUMAR; MUTHUCUMARU, 2002). A seguir, são abordadas algumas das características que precisam ser consideradas:

- **Compartilhamento de recursos**: múltiplos usuários podem competir pelo mesmo recurso;
- **Gerenciamento distribuído**: diferentemente dos aglomerados, os recursos podem estar distribuídos e podem estar associados a diferentes controladores de domínio com restrições próprias;
- **Recursos heterogêneos**: recursos como processadores, armazenamento e rede geralmente são heterogêneos, englobando diferentes tecnologias e poder de processamento;
- **Dinamicidade**: a disponibilidade de recursos pode variar de acordo com a carga submetida pelos usuários. Assim, os controladores devem se ajustar conforme a disponibilidade dos recursos;
- **Comunicação**: *workflows* científicos costumam envolver uma grande quantidade de dados, sendo necessário considerar o custo de transferência de dados entre os recursos.

Apesar de aglomerados e grades executarem tarefas com expressivo número de instruções e geralmente paralelas, há outras arquiteturas computacionais com características favoráveis a este tipo de aplicação. Uma delas consiste na computação em nuvem que será descrita no

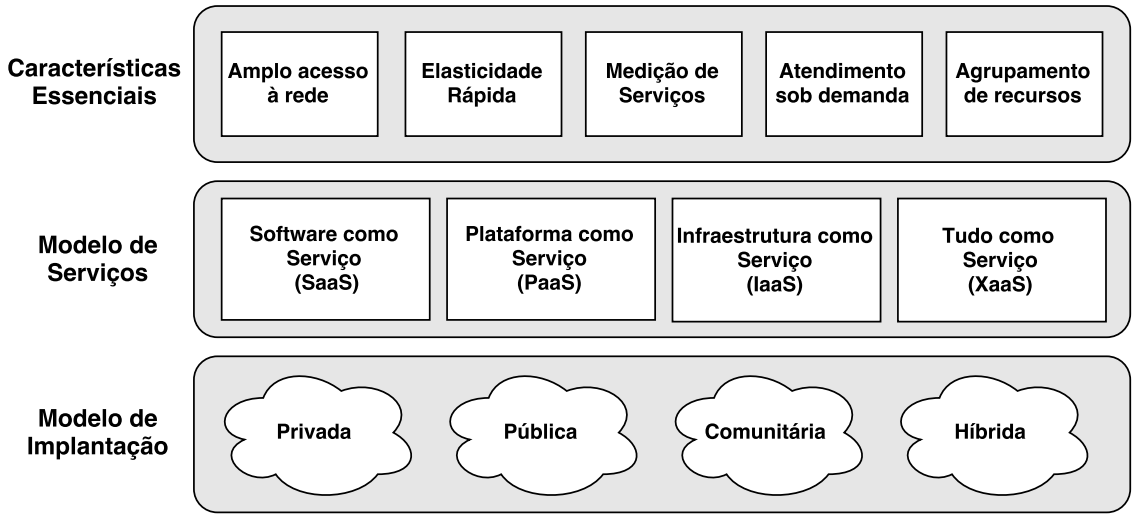
próximo capítulo com mais detalhes; visto que ela permite provisionar recursos sob demanda para escalonar e executar *workflows*.

2.4 Computação em nuvem

Definição

O modelo de computação em nuvem permite o acesso a recursos de modo ubíquo, sob demanda e de maneira compartilhada. De modo geral, é organizada em modelos de implantação, modelo de serviços e contempla características essenciais, conforme representadas na Figura 4 e discutido no decorrer desse capítulo. Os recursos oferecidos envolvem servidores para processamento, armazenamento e implantação de aplicações e serviços que podem ser rapidamente provisionados e liberados com o mínimo de esforço (MELL; GRANCE, 2011). Os provedores de serviços podem contar com recursos heterogêneos e distribuídos em diferentes localizações geográficas com o intuito de prover melhor desempenho e tolerância a falhas (ARMBRUST *et al.*, 2010).

Figura 4 – Modelo organizacional de Nuvem de Computadores.



Fonte: Adaptada de Mell e Grance (2011).

A computação em nuvem pode ser classificada de acordo com três modelos de distribuição de serviços:

- **Software como Serviço (SaaS)**: este tipo de serviço é caracterizado por aplicações implantadas em uma infraestrutura na nuvem e acessíveis por meio de uma interface cliente (ex.: um navegador, cliente de e-mail). Ele apresenta a vantagem da isenção de manutenção de infraestrutura, sendo de total responsabilidade do provedor de serviços (RIMAL; CHOI; LUMB, 2009);

- **Plataforma como Serviço (PaaS)**: oferece um ambiente que inclui recursos para o desenvolvimento, testes e implantação de aplicações (HURWITZ *et al.*, 2012). Nesse ambiente o usuário pode usar serviços usados no ambiente de desenvolvimento como, por exemplo, servidor web, banco de dados, gerenciador de controle de versão, sem a necessidade de adquirir e gerenciar hardware, e licenças de software;

- **Infraestrutura como Serviço (IaaS)**: pode-se obter a infraestrutura computacional necessária por meio deste tipo de serviço, que é baseado em contratos de forma que o usuário paga somente pelos recursos que usar. Uma de suas principais vantagens é a elasticidade da contratação de recursos sob demanda que, eventualmente, pode crescer ou reduzir em determinadas situações (BUYYA; BROBERG; GOSCINSKI, 2011).

Apesar da popularidade dos três modelos de distribuição de serviços mencionados anteriormente, a computação em nuvem pode oferecer outros modelos para qualquer tipo de serviço, denominada de *Everything as a Service* (XaaS) (DUAN *et al.*, 2015). Este modelo segue a política "pague apenas o que utilizar" e busca atender especificamente as necessidades do cliente. O caracter "X" representa a característica essencial do serviço oferecido. Para exemplificar um XaaS, pode-se citar Banco de Dados como um Serviço (DBaaS), Armazenamento como um Serviço (STaaS) e Gerenciamento como um Serviço (MaaS):

- **Banco de Dados como Serviço (DBaaS)**: é um serviço que oferece acesso a uma base de dados remota cuja finalidade é gerenciar dados como se a base de dados estivesse fisicamente na máquina local. O principal objetivo deste modelo é a redução do custo monetário e operacional para manter muitos softwares (licenças) e recursos para o armazenamento dos dados;

- **Armazenamento como Serviço (STaaS)**: o armazenamento de arquivos pode ser oferecido como um serviço, permitindo ao cliente expandir ou reduzir a sua capacidade conforme a demanda. Esse é um serviço frequentemente contratado com outros serviços que têm como base o armazenamento de arquivos;

- **Segurança como Serviço (SecaaS)**: este é um dos principais requisitos para quem vai utilizar a computação em nuvem, pois o armazenamento e a manipulação de dados remotos necessita de recursos de segurança como a autenticação, a verificação de integridade e a criptografia dos dados. Sem eles, pode haver a interceptação e a manipulação de dados privados e sensíveis.

Tipos de nuvens

No início, a Computação em Nuvem surgiu para dar suporte a problemas de aplicações de uso público. Contudo, foram propostos outros modelos de localização e distribuição física.

Desse modo, nuvens podem ser organizadas conforme as suas características de distribuição de serviços e de seus proprietários. Tradicionalmente, nuvens podem ser classificadas em quatro modelos de implantação: privadas, públicas, comunitárias e híbridas. Mell e Grance (2011) definem tais modelos em:

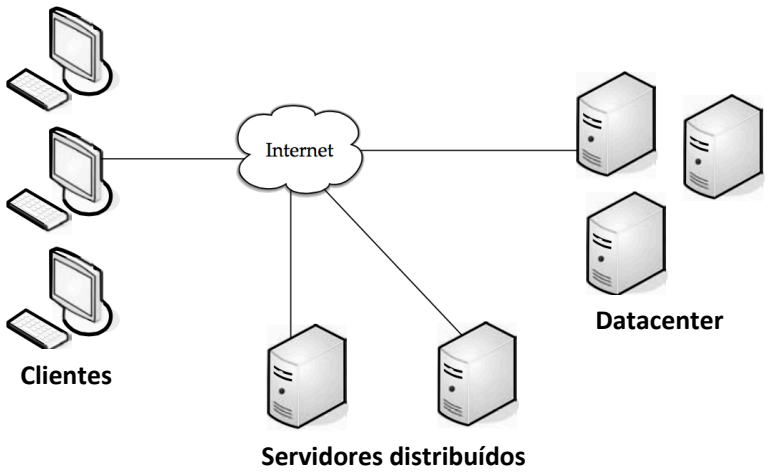
- **Nuvem Privada**: a infraestrutura desta nuvem é direcionada para uso exclusivo de uma única organização e utilizada por múltiplos consumidores. Esse tipo de nuvem pode ser gerenciado e operado por uma organização, por terceiros ou por alguma outra combinação de proprietários. Além disso, a infraestrutura pode estar presente ou não no próprio local da organização;
- **Nuvem Comunitária**: a infraestrutura nesta nuvem é provisionada exclusivamente para um conjunto de organizações que compartilham o mesmo interesse de negócio. Essa nuvem pode ser de propriedade de uma ou mais organizações que podem gerenciar e operar os recursos oferecidos;
- **Nuvem Pública**: os recursos são provisionados para um público generalizado. Esse tipo de nuvem pode ser gerenciado e utilizado por instituições acadêmicas, governamentais ou privadas. Os serviços ficam localizados nas instalações do provedor de nuvem. Como exemplo de provedores de nuvens públicas, pode-se citar empresas como Amazon, Google e Microsoft que oferecem serviços em diferentes modelos (IaaS, SaaS e PaaS);
- **Nuvem híbrida**: nuvens híbridas são compostas por duas ou mais nuvens distintas (pública, comunitária e privada), que são representadas como uma única entidade, mas estão interligadas por meio de uma tecnologia que permite a integração e a compatibilidade de dados e aplicações.

Componentes de uma Nuvem

De modo simplificado, uma solução baseada em computação em nuvem é composta por clientes, *datacenter* e servidores distribuídos, conforme mostrado na Figura 5. Cada elemento tem um propósito específico no funcionamento de uma aplicação em nuvem.

- **Clientes**: em uma arquitetura em nuvem geralmente são os usuários finais que interagem com computadores como desktop, smartphones, tablets e outros;
- **Datacenter**: é um conjunto de servidores localizados em grandes espaços e serve para hospedar a aplicação do cliente. Usualmente, os servidores são virtualizados, permitindo maximizar o uso dos recursos de um único servidor;
- **Servidores dedicados**: são considerados servidores dedicados e dispersos geograficamente, onde a aplicação é hospedada de forma transparente ao usuário.

Figura 5 – Visão simplificada de uma solução baseada na Computação em Nuvem.



Fonte: Adaptada de Velte, Velte e Elsenpeter (2009).

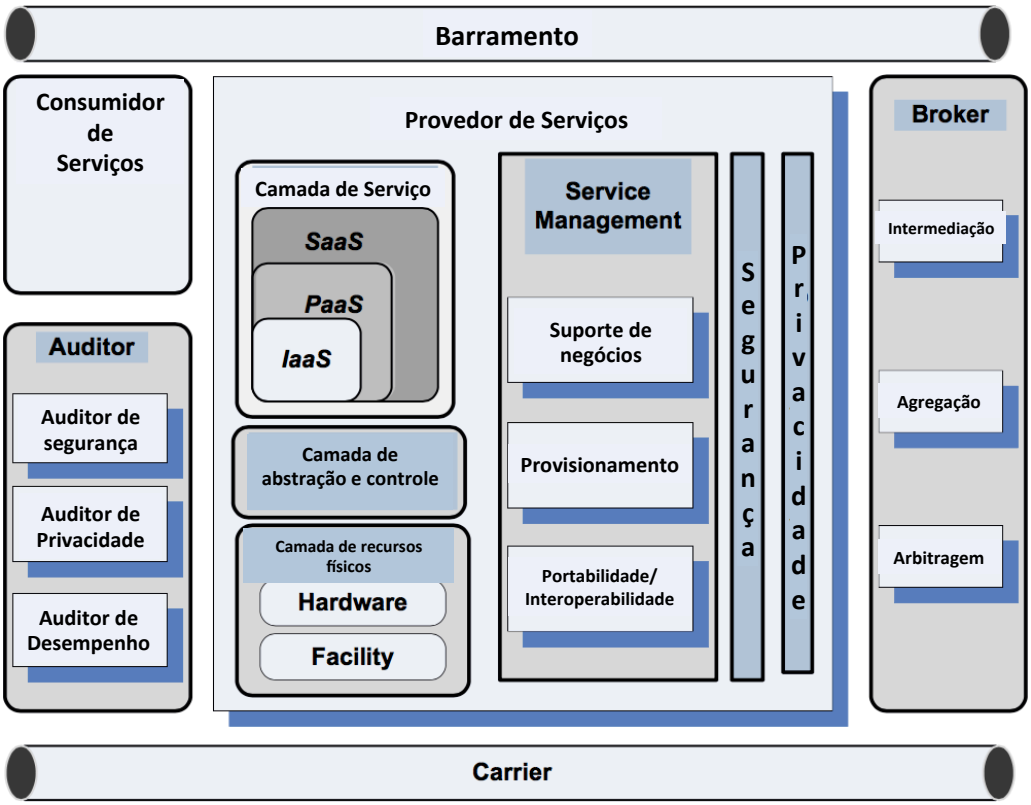
Além dos três componentes citados, uma arquitetura em nuvem é formada por outras entidades para estruturar o uso e o gerenciamento de seus recursos, conforme apresentados na Figura 6. O Consumidor de Serviços é responsável por manter a relação de negócios com o provedor de serviços que, por sua vez, oferta seus serviços ao consumidor. A comunicação entre o consumidor e o provedor de serviço é realizada por meio de um barramento comum. Para gerenciar o desempenho e uso de serviços, um *Broker* é responsável pela disponibilização e negociação dos contratos entre os provedores de serviços e os consumidores. O Auditor é responsável por avaliar o desempenho e a segurança das aplicações e compará-los com um conjunto de critérios previamente estabelecidos.

Características de uma nuvem

Embora a computação em nuvem seja organizada de acordo com sua estrutura e modelo de serviços, há algumas características que a distinguem de outras plataformas de alto desempenho (ZHANG; CHENG; BOUTABA, 2010), tais como:

- **Serviço sob demanda**: o usuário pode realizar o provisionamento de recursos (processamento, armazenamento ou comunicação) sem a intervenção humana para a alocação de recursos;
- **Largo acesso à rede**: os recursos ficam disponíveis na nuvem e podem ser acessados por meio de dispositivos como smartphones, tablets, laptops e estações de trabalho conectados à rede da nuvem;
- ***Pool* de recursos**: os recursos do provedor podem ser compartilhados entre vários usuários que são logicamente isolados por meio da virtualização. A virtualização é realizada por

Figura 6 – Visão simplificada de uma solução baseada na Computação em Nuvem.



Fonte: Adaptada de Marinescu (2018).

meio da configuração de máquinas virtuais que permitem a criação de ambientes isolados e independentes, mas que podem estar em execução em uma mesma máquina física;

- **Elasticidade**: os recursos podem ser rapidamente provisionados e liberados permitindo a escalabilidade do sistema de acordo com a demanda. Essa característica é importante para a execução de workflows, pois a variação do conjunto de recursos afeta diretamente no desempenho das aplicações;

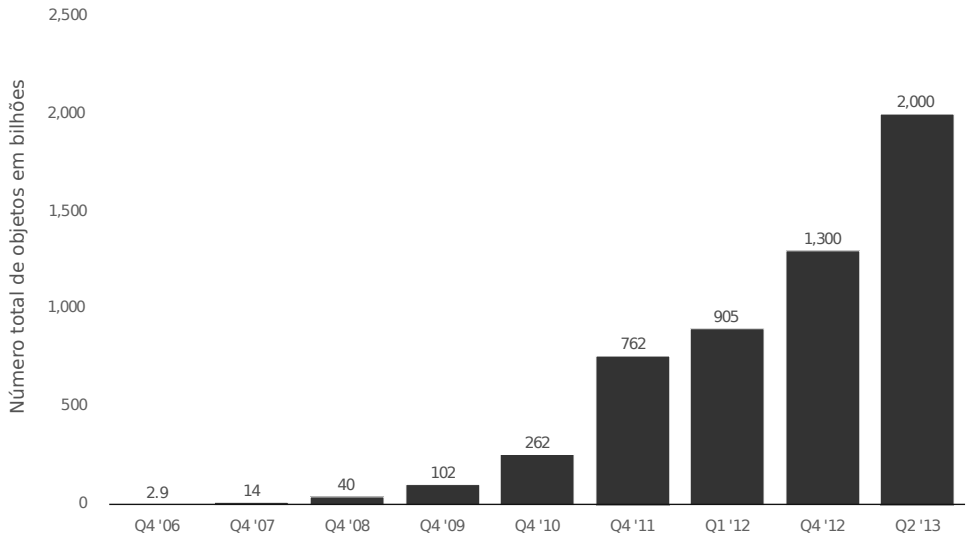
- **Mensuração**: a nuvem é baseada em métricas para gerenciar os recursos utilizados (processamento, armazenamento, rede ou contas). Os recursos podem ser monitorados, controlados e reportados, provendo transparência tanto para o provedor quanto para o consumidor dos serviços.

Segurança e a Nuvem

Desde a sua concepção, a computação em nuvem tem se popularizado na indústria e na academia (SADIKU; MUSA; MOMOH, 2014). O modelo econômico, a escalabilidade, a ubiquidade e o acesso sob demanda são características que têm levado a migração de diversos processos de negócio à nuvem (ABBAS *et al.*, 2015). Para representar a crescente adoção da

Computação em Nuvem, a Figura 7 representa o número de objetos armazenados no Amazon S3 entre 2006 e 2013.

Figura 7 – Número de objetos armazenados na Amazon S3 entre o quarto trimestre de 2006 ao segundo trimestre de 2013.



Fonte: Adaptada de Statista (2013).

No entanto, embora a nuvem possa oferecer benefícios aos usuários, há algumas peculiaridades que devem ser levadas em consideração. De acordo com pesquisa conduzida pela International Data Corporation (IDC), diversos fatores influenciam na adoção da nuvem como: a disponibilidade, o custo, a falta de padronização, a difícil integração com soluções locais, entre outros. Nesse contexto, a falta de segurança é o principal fator que impede o uso da Nuvem pelas instituições (VIEGA, 2009; HASHIZUME *et al.*, 2013).

Proteger dados privados e sigilosos, como registros médicos, de invasores ou de pessoas maliciosas de uma instituição é de fundamental importância (HUI *et al.*, 2009). Além disso, a migração de bases de dados para a nuvem envolve desafios de segurança como a vulnerabilidade de virtualização e de acesso, problemas de integridade e confidencialidade, bem como a perda ou roubo de informações (WANG *et al.*, 2009).

A responsabilidade com a segurança entre usuários e provedores é diferente em cada modelo de serviço na nuvem (TAKABI; JOSHI; AHN, 2010). Em SaaS, os provedores são mais responsáveis pela segurança e privacidade da aplicação do que os usuários. Essa responsabilidade é mais relevante em nuvens públicas do que em nuvens privadas, devido a nuvem pública estar visível a outros usuários. Em PaaS, os usuários são responsáveis por cuidar de suas aplicações desenvolvidas e executadas na plataforma, enquanto os provedores de nuvem são responsáveis por proteger uma aplicação de um usuário dos demais. Por fim, em IaaS o usuário é responsável por proteger todo o sistema operacional e suas aplicações, enquanto o provedor deve proteger o isolamento lógico dos dados de cada usuário.

Os principais problemas contidos implicitamente no paradigma em nuvem são o tráfego de dados sensíveis e a execução de processos críticos. Ao utilizar a nuvem, o usuário deve estar ciente de que os seus dados não estarão em um ambiente sob seu total controle e proteção. Além disso, ao processar os seus dados na nuvem usando IaaS ou PaaS, o provedor terá total controle nesses processos. Assim, é importante haver uma relação de confiança entre o provedor de serviços e o usuário, sendo este um pré-requisito geral na computação em nuvem (JENSEN *et al.*, 2011).

Mesmo assumindo que na maioria dos casos o provedor é confiável, ainda há o risco de ataques de usuários mal intencionados. Jensen *et al.* (2009) apresenta uma visão geral de diferentes tipos de ataques em serviços na nuvem e como estes são realizados:

- **Ataque na virtualização**: Ristenpart *et al.* (2009) discute sobre um ataque ocorrido no serviço Amazon EC2 em que o atacante instanciou máquinas virtuais até que uma delas fosse alocada na mesma máquina física da máquina virtual da vítima. Logo em seguida, o atacante executou um ataque *cross-VM side-channel* para obter e modificar dados da vítima. Os autores apresentaram estratégias para aumentar a probabilidade de atingir a máquina da vítima desejada e propõem técnicas para afastar ataques *cross-VM side-channel*;

- **Ataque de integridade e autenticação**: em (GRUSCHKA; IACONO, 2009), uma falha na interface de gerenciamento da Amazon EC2 foi encontrada. Os autores descobriram que a implementação da assinatura do EC2 era vulnerável ao ataque de *Signature Wrapping* (MCINTOSH; AUSTEL, 2005). Neste caso, o atacante que interceptou uma requisição legítima de uma vítima, pode adicionar uma segunda operação na mensagem mantendo a assinatura original. Devido a esta falha, a modificação da mensagem não é detectada e a operação maliciosa é executada pela vítima e dela também cobrada;

- **Acesso sem autorização**: um dos maiores incidentes em SaaS ocorreu com o Google Docs (KINCAID, 2009). Quando um documento era compartilhado com um ou mais usuários, este ainda era acessível por qualquer outro usuário sem permissão de acesso concedido anteriormente pelo proprietário. Assim, qualquer usuário apenas com o *link* do documento conseguia acessá-lo sem a necessidade de quebrar o sistema de autenticação e permissão do Google Docs.

2.5 Escalonadores

Workflows permitem a modelagem de tarefas e dados sem o conhecimento dos recursos computacionais disponíveis. Desta forma, o cientista se preocupa estritamente com os detalhes da aplicação, deixando de lado o controle da execução. Em um *workflow* é definida a ordem de execução das tarefas e a movimentação dos dados, mas não indicam os recursos computacionais que precisam ser alocados para a execução. Assim, um sistema gerenciador de *workflow* científico

é responsável por mapear e atribuir tarefas e dados nos recursos computacionais e de armazenamento disponíveis. No contexto de *workflow*, tal processo é definido como escalonamento, o qual está apresentado nesta seção.

Definição

No contexto de *workflow*, o escalonamento é um processo que realiza o mapeamento das tarefas de um *workflow* para calcular os recursos necessários para sua execução, de modo a preservar a interdependência das tarefas (LIU *et al.*, 2014). Os objetivos do escalonamento podem variar de acordo com a natureza da aplicação e as necessidades do usuário. Alguns algoritmos focam na otimização do custo e tempo de execução, outros se atentam à confiabilidade, disponibilidade ou eficiência energética. Outros objetivos podem ser derivados a partir da combinação dos critérios mencionados.

Formalmente, um *workflow* é modelado como um grafo acíclico direcionado (DAG). Um *workflow* é definido como $w = (T, E)$, onde T é um conjunto de tarefas e serviços representados por vértices em um DAG e E é um conjunto de dependências representadas por arestas no grafo (SMANCHAT; VIRIYAPANT, 2015). Uma dependência é definida como $e = (t_i, t_j)$ onde t_i e $t_j \in T$, sendo t_j uma tarefa predecessora de t_i, e $t_i \neq t_j$. Uma tarefa t pode iniciar a sua execução se e somente se todas as tarefas predecessoras tiverem sua execução completada e a transferência dos dados de entrada estiverem sido completamente finalizadas com sucesso.

O escalonamento de *workflow* é caracterizado pela associação $T \rightarrow R$, onde cada tarefa $t \in T$ seja atribuída para um recurso $r \in R$ para satisfazer um determinado objetivo. Recursos na nuvem geralmente são caracterizados por máquinas virtuais (VM), que são instanciadas por meio de imagens previamente configuradas na nuvem. No entanto, para a execução de uma tarefa, é preciso incluí-la na imagem que será implantada em uma máquina virtual (WANG *et al.*, 2014).

Uma tarefa t pode ser definida como $t = (id, MI)$, onde id é a identificação de t e $mi \in MI$ é um conjunto de imagens de máquina virtual que pode ser usado para a execução de uma tarefa t. Um recurso computacional r pode ser definido como $r = (mi, mt)$, onde $mt \in MT$ é um tipo de instância em MT oferecida por um provedor na nuvem.

(GRAHAM *et al.*, 1979) adota uma notação de três campos $\alpha|\beta|\gamma$ para representar o problema de escalonamento e que pode ser estendida ao contexto de escalonamento de *workflows*. O campo α representa o conjunto de recursos computacionais disponíveis para execução, o campo β abrange o conjunto de tarefas que devem ser executadas e o campo γ descreve a função objetivo a ser otimizada.

Os recursos utilizados para a execução de um *workflow* são denotados pelo campo α e podem ser representados por P, Q ou R. Os recursos contidos em P são máquinas idênticas em paralelo e uma tarefa t_i que pode ser executada em qualquer máquina j no tempo p_i. Os recursos representados por Q são caracterizados por m máquinas paralelas uniformes com diferentes

velocidades. Cada processamento p_{ij} de uma tarefa t_i é igual a p_i/v_j, onde v_j é a velocidade de uma máquina j. A representação de R é denotada por m máquinas não-uniforme (velocidades diferentes) que podem processar cada tarefa com uma velocidade diferente. Assim, uma máquina j processa uma tarefa t_i em velocidade v_{ij}, logo o tempo p_{ij} que a tarefa t_i gasta para ser executada é p_i/v_j.

O campo β indica características de uma tarefa que podem ser definidas. Uma das informações é a permissão de preempção ou não da tarefa. Caso positivo, uma tarefa pode ter sua execução interrompida e posteriormente continuada. Outra característica é a quantidade de recursos necessários para uma tarefa, em que pode-se definir restrições ou não das máquinas que irão ser usadas na execução. No campo β também é levado em consideração o tempo de disponibilidade de r_i que indica a partir de qual momento uma tarefa t_i pode ser iniciada. Semelhante a essa característica, também é considerado no campo β o prazo d_i. Caso d_i for definido, a tarefa t_i deve ser concluída até um determinado momento, caso contrário, não há nenhuma restrição. Ainda no campo β, considera-se o tempo de comunicação entre máquinas c_{ik}. Esse tempo corresponde ao tempo que uma tarefa t_k precisa aguardar para ser executada após a conclusão de uma tarefa t_i. O motivo dessa latência pode ser a movimentação de grande volume de dados entre as máquinas disponíveis. Geralmente, o campo β é representado por *prec*, o qual indica que há relações entre as tarefas baseadas em um Grafo Acíclico Direcionado.

O campo γ é utilizado para indicar a função objetivo do escalonamento e expressa os critérios escolhidos e as restrições relacionadas à execução do *workflow*. Geralmente, o objetivo a ser otimizado é o tempo de conclusão das tarefas do *workflow*. O instante de término de uma tarefa t_i é representado como C_i e o custo da tarefa t_i que finaliza sua execução no tempo C_i é $f_i(C_i)$. Uma das funções objetivo mais comum é a de *makespan* C_{max}, onde $C_{max} = max\{C_1, C_2, ..., C_n\}$ representa o momento do término da última tarefa. Outra função comumente utilizada é a do custo total para a execução do *workflow*, a qual é definida por $\sum_{i=1}^{n} C_i$.

Há diversas configurações $\alpha|\beta|\gamma$ possíveis com diferentes critérios de classificação dessas abordagens como descrito em (FAKHFAKH; KACEM; KACEM, 2014; SMANCHAT; VIRIYAPANT, 2015). Para exemplificar uma definição de escalonamento utilizando os campos $\alpha|\beta|\gamma$, pode-se considerar R | *prec* | $min(C_{max})$ que descreve a necessidade de escalonamento de tarefas de um DAG ($\beta = prec$) em m máquinas heterogêneas ($\alpha =$ R) de modo a minimizar o *makespan* ($\gamma = C_{max}$).

O escalonamento das tarefas de um *workflow* pode ser realizado estaticamente ou dinamicamente. No escalonamento estático as tarefas são distribuídas aos recursos computacionais disponíveis antes do início da execução e não sofre qualquer mudança até o término do *workflow*. Por outro lado, o escalonamento dinâmico considera a reavaliação das atribuições de tarefas de acordo com a disponibilidade de recursos (WANG *et al.*, 2014; SMANCHAT; VIRIYAPANT, 2015).

Objetivos

O escalonamento de *workflows* é realizado de acordo com um ou mais objetivos γ definidos pelo usuário que variam conforme o ambiente de execução. Em grades computacionais, o escalonamento pode considerar o *makespan*, a disponibilidade, a confiabilidade, o *deadline* ou a segurança. Porém, quando executado em nuvens públicas, os objetivos do escalonamento estendem-se a minimizar o custo da execução ou limitar-se a um orçamento estabelecido pelo usuário. Para melhor compreensão, cada objetivo é definido a seguir.

O tempo é uma métrica básica de desempenho. Em sistemas baseados em *workflows*, o *makespan* é o momento do término da execução da última tarefa de um *workflow* (LIU *et al.*, 2010). Em sistemas críticos pode-se levar em consideração o *deadline*, definido pelo tempo limite para o término de sua execução (RODRIGUEZ; BUYYA, 2014).

A elasticidade da computação em nuvem permite alocar recursos para a execução de *workflows* em menor tempo. Porém, a alocação de recursos em nuvens públicas impacta diretamente no aumento do custo ao usuário. O custo representa o valor monetário da execução de um *workflow*, e é um dos principais objetivos de escalonamento, que pode ser minimizado ou ser limitado a um orçamento (*budget*) estipulado pelo usuário (MARY; JAYAPRIYA, 2014).

A disponibilidade e a confiabilidade são dois fatores que também são considerados como objetivos. A disponibilidade é caracterizada pelo escalonamento apropriado a fim de melhorar a disponibilidade dos recursos na nuvem (SAJID; RAZA, 2013). A confiabilidade está relacionada à probabilidade de uma tarefa ser concluída com sucesso. Para esse propósito, podem ser aplicadas técnicas de replicações, *checkpoint* ou reinicialização (ZHAO; REN; SAKURAI, 2013).

Quando trata-se do uso de serviços da nuvem, a segurança é a maior preocupação dos usuários, sobretudo em relação aos dados transmitidos e nela processados. Portanto, a segurança é um objetivo que deve ser considerado em sistemas onde há a manipulação de dados sensíveis, a fim de manter a confidencialidade dos dados e atribuir recursos confiáveis para a execução das tarefas (TAO *et al.*, 2009). Por exemplo, em aplicações bancárias, usuários maliciosos podem abusar de características da nuvem para realizar ataques específicos. Diante desse cenário, um escalonador deve produzir um esquema de atribuições tarefa-recurso que mitigue os efeitos dos ataques na nuvem.

Tipos de escalonadores

Os escalonadores atuam na distribuição de cada tarefa a um recurso de modo a atender a função objetivo de escalonamento. Para cada natureza de *workflow* e ambiente de execução, é preciso considerar diferentes tipos de escalonadores, os quais são apresentados a seguir.

O escalonador pode ser estático ou dinâmico. No escalonamento estático as tarefas chegam simultaneamente e o escalonamento de recursos disponíveis ocorre depois que cada

tarefa é atribuída a um recurso (SMANCHAT; VIRIYAPANT, 2015). Esse escalonamento assume o conhecimento prévio das instruções de cada tarefa e apresenta menor *overhead* de escalonamento. Por outro lado, escalonadores dinâmicos são voltados a tarefas cujas instruções e dados não são conhecidos antes da execução. Esse modo se adapta melhor às mudanças da carga de processamento e dos recursos disponíveis, mas produz maior *overhead* do que o escalonamento estático.

Outra classificação é a localidade do escalonador, que pode ser centralizada ou distribuída. No primeiro caso há uma unidade de processamento mestre que é usada para uma coleção de tarefas que as envia às demais unidades de processamento (RAVICHANDRAN; NAGANATHAN, 2013). Na segunda classificação são utilizados escalonadores locais para gerenciar as requisições e atualizar os estados das tarefas, porém a sua eficiência é menor do que o escalonamento centralizado.

O mapeamento de tarefas a recursos pode ocorrer de duas formas. O primeiro é o escalonamento *offline* em que cada tarefa é escalonada somente uma vez e não pode ser alterado. Esse método é recomendado para casos em que a taxa de chegada é baixa (NAGADEVI; SATYAPRIYA; MALATHY, 2013). O escalonamento *online* não mapeia as tarefas a recursos no momento de sua chegada, mas coletam e examinam periodicamente as tarefas (VIJAYALAKSHMI; VASUDEVAN, 2015).

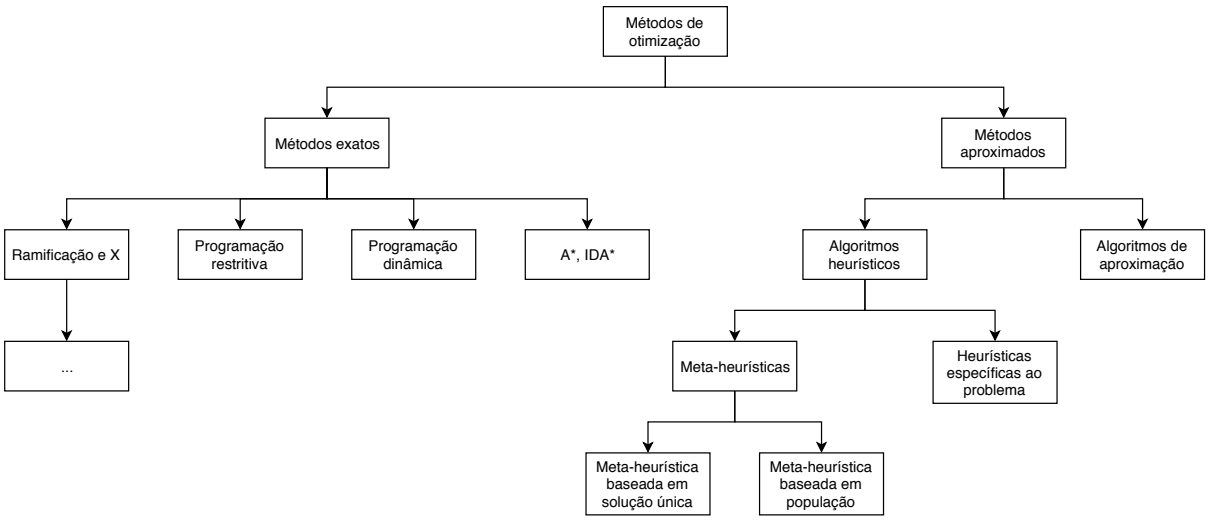
2.6 Computação evolutiva

A garantia da obtenção de soluções ótimas pode ser intratável em diversos problemas de otimização seja na indústria ou no meio científico. No contexto de escalonamento de tarefas de workflows científicos, calcular o escalonamento ótimo considerando a escala de centenas ou até milhares de tarefas e uma diversas instâncias computacionais pode ser inviável em diversas situações. Nesse sentido, um escalonamento razoável obtido por meio de heurísticas ou metaheurísticas pode satisfazer o requisito mínimo de muitas aplicações baseadas em workflows. Esta seção trata da definição dos conceitos fundamentais (Seção 2.6) relacionados à computação evolutiva, bem como descreve os algoritmos evolutivos adotados nesta tese (Seção 2.6). A otimização de objetivos conflitantes também é apresentada na Seção 2.6.

Conceitos fundamentais

As soluções para problemas de otimização podem ser obtidas por meio de métodos exatos ou métodos aproximativos. A complexidade do problema determina se este pode ser resolvido por um método exato ou um método aproximado (Figura 8). Um problema é dito intratável se a complexidade de sua resolução é exponencial como $n!$ ou n^k, onde k representa o número de critérios, sendo $k > 1$, e n representa o número de tarefas do problema (TOSCANI; VELOSO; GARCEZ, 2009).

Figura 8 – Métodos clássicos de otimização.



Fonte: Adaptada de Russell e Norvig (1995).

Métodos exatos

Métodos exatos são capazes de obter soluções ótimas e garantir a solução ótima. Algoritmos exatos apresentam tempo não-polinomial para problemas NP-completo. Há diversos algoritmos que se baseiam em métodos exatos como a programação dinâmica, família branch and *X* (branch and bound, branch and cut, brand and price), programação por restrições e família de algoritmos A* (A*, IDA*) (RUSSELL; NORVIG, 2009). Os métodos enumerativos podem ser classificados como árvores de busca em que o problema é resolvido por meio da sua subdivisão em partes menores.

A programação dinâmica é baseada no princípio Bellman's, o qual afirma que a subpolítica de uma política ótima é ótima por si só (BELLMAN, 1957). Na programação dinâmica, o problema é dividido em subproblemas para tornar a sua resolução mais simples. Esta abordagem evita considerar todo o espaço de busca por meio da poda parcial de soluções que não podem levar a uma solução ótima.

Algoritmos *branch and bound* e A* são baseados na enumeração implícita de todas as soluções de um problema. O espaço de busca é explorado por meio da construção dinâmica de uma árvore em que o nó raiz representa o problema a ser resolvido e os nós folha são potenciais soluções. A poda do espaço de busca é baseada em uma função que remove as sub-árvores que não contém qualquer solução ótima.

A programação por restrições é realizada com uma linguagem desenvolvida em torno dos conceitos de árvores de busca e implicações lógicas. Os problemas são modelos por meio de um conjunto de variáveis associadas a um conjunto de restrições. Na programação de restrições, as variáveis são atribuídas em um domínio finito de valores inteiros. As restrições podem ser

formadas matematicamente ou simbolicamente. De modo geral, métodos exatos podem ser aplicados a pequenas instâncias de problemas complexos.

Métodos aproximados

Considerando os métodos aproximados, há duas subclasses de algoritmos que podem ser definidas como (a) algoritmos de aproximação e (b) algoritmos heurísticos. A diferença básica entre essas duas classes está na qualidade das soluções e o tempo para encontrá-las.

Os algoritmos de aproximação se diferenciam de algoritmos heurísticos ao oferecerem alguma garantia de que a solução encontrada não é pior que uma determinada fração da solução ótima. Tratam-se de algoritmos eficientes, ou seja, polinomiais na obtenção de tais soluções. Por outro lado, Algoritmos heurísticos geralmente encontram boas soluções em tempo razoável, mas sem fornecer garantia da qualidade da solução obtida em relação à solução ótima.

Os algoritmos heurísticos podem ser divididos em duas categorias: heurísticas específicas e meta-heurísticas. As heurísticas se caracterizam por serem dependentes do problema a ser tratado, ou seja, os algoritmos são elaborados a partir de regras e restrições inerentes ao problema em questão. A meta-heurísticas podem ser aplicados na maioria dos problemas de otimização. Eles podem ser visualizados como um método genérico para auxiliar na estratégia de definição de heurísticas para resolver problemas específicos.

Meta-heurísticas

Ao contrário de métodos exatos, as meta-heurísticas permitem lidar com instâncias de problemas complexos e costumam entregar soluções satisfatórias em tempo razoável. Contudo, meta-heurísticas não garantem encontrar as soluções ótimas. A popularização das meta-heurísticas se deu nos últimos 20 anos com sua eficácia e eficiência demonstrada na solução de problemas grandes e complexos. Nota-se que algoritmos de meta-heurísticas têm sido aplicados em diversos problemas de otimização como os descritos em Yang e He (2013), Ali e Hassanien (2016), Wari e Zhu (2016).

A fase de projeto de uma meta-heurística considera dois critérios: (a) exploração do espaço de busca (diversificação) e (b) exploração das melhores soluções encontradas (intensificação). Em intensificação, as regiões promissoras são mais exploradas em busca de soluções ainda melhores. Do outro lado, a diversificação tende a visitar regiões ainda não exploradas para evitar que a busca fique presa a um número reduzido de regiões. Algoritmos de busca aleatória estão na extremidade da diversificação das regiões de busca. Estes geram uma solução aleatória no espaço de busca a cada iteração. Algoritmos de busca local tendem a explorar com maior intensidade uma porção menor de regiões. Sua vantagem é que o histórico de busca é dispensável, ou seja, somente a melhor solução é mantida para compará-la com as novas soluções da próxima iteração.

De acordo com Talbi (2009), as meta-heurísticas também podem ter diferentes classifica-

ções de acordo com características como:

- **Bio-inspirada ou não bio-inspirada**: diversas meta-heurísticas são baseadas em processos naturais como algoritmos evolucionários e sistemas imunológicos artificiais; baseadas em conceitos como colônia de abelhas ou formigas, enxame de partículas ou regras de organização social de diferentes espécies; recozimento simulado da física para aceitação de soluções menos promissoras;
- **Consumo de memória**: algumas meta-heurísticas não necessitam armazenar informações durante a busca de soluções como busca local, GRASP e recozimento simulado. Outros métodos usam a memória para extrair informações durante o processo de busca como, por exemplo, a busca tabu;
- **Determinístico ou estocástico**: meta-heurísticas determinísticas resolvem o problema por meio de decisões exatas como a busca tabu e a busca local. As meta-heurísticas estocásticas passam a utilizar regras aleatórias que são aplicadas durante a busca como o recozimento simulado e algoritmos evolucionários. Em algoritmos estocásticos uma solução inicial pode produzir diferentes soluções finais, enquanto os determinísticos produzem a mesma solução final a partir de uma mesma solução inicial;
- **Baseado em população ou busca de única solução**: algoritmos baseados em solução única (busca local e recozimento simulado) atualizam uma única solução durante todo o processo de busca. Os algoritmos baseados em populações (algoritmos evolucionários e enxame de partículas) consideram a evolução de todos os indivíduos de uma população. Essas duas famílias de algoritmos apresentam características complementares: meta-heurística baseada em solução única tem maior força para intensificar a busca em regiões locais, enquanto as baseadas em população diversificam as buscas em todo o espaço do problema, e;
- **Iterativo ou guloso**: a maioria das meta-heurísticas são baseadas em algoritmos iterativos. Estes algoritmos iniciam com um conjunto de soluções completas e as transformam a cada iteração usando alguns operadores de busca. Os algoritmos gulosos partem de uma solução vazia e a cada iteração uma variável do problema é atribuída até que uma solução completa seja obtida.

Algoritmos Evolutivos

A computação evolutiva (CE) surgiu com base nos princípios da Teoria da Evolução das Espécies proposta por Darwin (1859) em que os sistemas biológicos passam por evolução natural. Logo, os algoritmos em CE se inspiram, por exemplo, no processo de recombinação da informação genética (*crossover*), alteração dessas informações via mutação e processos de seleção onde os mais aptos têm mais chance de passar adiante seu código genético. Do

lado técnico, a CE busca o desenvolvimento de algoritmos capazes de resolver uma variedade de problemas complexos com um adequado balanceamento entre exploração (*exploitation*) e refinamento no espaço de busca.

O Algoritmo 1 apresenta os passos gerais encontrados em diversas classes de algoritmos evolutivos. Um algoritmo evolutivo parte de uma população inicial contendo um conjunto de possíveis soluções P_g e tenta evoluir a população até que um determinado critério de parada seja satisfeito.

Algoritmo 1 – Passos gerais de um algoritmo evolutivo.
Fonte: Toscani, Veloso e Garcez (2009).

```
1:  procedimento AE(parâmetros do problema)
2:      g ← 1                                          ▷ contador de iterações
3:      IniciaPopulação(P_g)
4:      Avalia(P_g)          ▷ calcula o fitness para cada indivíduo i da população
5:      repita
6:          g ← g + 1                                  ▷ incrementa contador de iterações
7:          P_g ← Seleção(P_{g-1})                     ▷ seleção de indivíduo para crossover
8:          Cruzamento(P_g)                            ▷ Cruzamento dos indivíduos
9:          Mutação(P_g)                               ▷ Mutação em indivíduos da nova população
10:         Avalia(P_g)                                ▷ calcula fitness dos indivíduos da população
11:     até Critério de parada ser satisfeito          ▷ checa critério de parada
12:     retorna P_g
13: fim procedimento
```

Terminologia

A computação evolutiva utiliza uma série de conceitos biológicos para fundamentar suas técnicas computacionais (BALLARD, 1997; TEUSCHER *et al.*, 2003). Esses conceitos se apoiam nos princípios de adaptação e evolução genética, em que indivíduos de uma população se reproduzem e competem pela sobrevivência. Um conjunto de regras precisa ser respeitado e algumas estruturas devem ser seguidas para que, de fato, os algoritmos evolutivos tenham semelhança com os fundamentos da evolução dos seres vivos. Os autores em (RAFIQ; MATHEWS; BULLOCK, 2003) definem os componentes essenciais da computação evolutiva:

- **População**: é formada por um ou mais conjuntos de indivíduos que disputam entre si pela chance de reproduzir e permanecer na população;
- **Variabilidade**: a variabilidade dos indivíduos deve ser respeitada para que haja a evolução dos indíviduos. Populações com indivíduos idênticos não beneficiam a evolução de uma população;
- **Hereditariedade**: os indíviduos precisam possuir traços ou características de seus indivíduos progenitores;

- **Aptidão**: retrata a capacidade de um indivíduo sobreviver e proliferar suas características pela população.

Essas definições facilitam a compreensão do processo evolutivo caracterizado pela reprodução, mutação e seleção natural dos indíviduos de uma população. Os autores em (SAIT; YOUSSEF, 1999) descrevem as terminologias frequentemente utilizadas em CE:

- **Cromossomo**: consiste em uma sequência de genes que determinam as características que compõem cada indivíduo. Um cromossomo na CE codifica as principais características do problema a ser solucionado. Logo, a codificação utilizada pelo cromossomo em CE é crucial para obter êxito na exploração do espaço de busca. Vários problemas na literatura são representados por meio de codificação binária, ou seja, cada gene pode assumir o valor 0 ou 1. A solução candidata de um problema é codificada em um cromossomo que determina um indivíduo da população. Desta forma, tanto um indivíduo quanto um cromossomo podem apresentar o mesmo significado no contexto de algoritmos evolutivos;

- **Operadores de Reprodução**: a reprodução é definida como a combinação genética dos cromossomos de dois ou mais indivíduos. O resultado dessa operação é um novo indivíduo, denominado filho, o qual será uma nova e possível solução do problema. A reprodução também pode ser identificada pelos termos *crossover* ou recombinação genética;

- **Mutação**: é uma mudança em uma ou mais partes (genes) de um cromossomo. A mutação provê a diversidade da população e pode levar à formação de indivíduos mais aptos do que os existentes na população;

- **Seleção**: similar ao processo descrito na Teoria de Darwin, a seleção controla a qualidade do processo de reprodução dos indivíduos. Os indivíduos com melhor aptidão de uma população apresentam maior chance de serem selecionados para a reprodução e, consequentemente, terem suas características repassadas para a próxima geração. Além disso, a seleção também pode ser empregada para decidir quais indivíduos irão compor a população na próxima geração.

Operadores genéticos

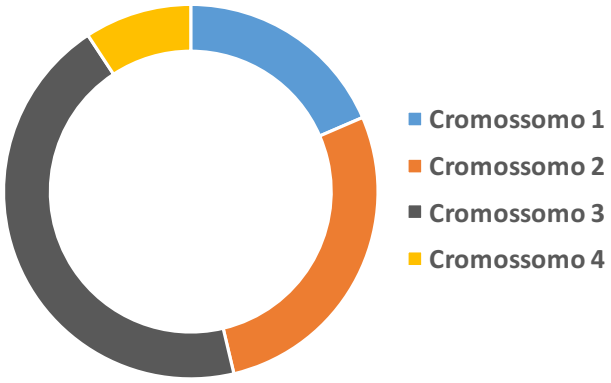
Os operadores genéticos fazem parte do conjunto de regras que permitem aos algoritmos em CE apresentarem um comportamento semelhante ao processo de evolução biológica. Nesse contexto, há os operadores de seleção, recombinação e mutação. Os operadores de seleção por roleta, *ranking* e torneio serão descritos a seguir.

O método da roleta é um dos mais populares devido a sua facilidade de implementação. Nesse método, cada cromossomo é distribuído em áreas proporcionais à sua aptidão. Logo, os indivíduos com maior aptidão têm mais chances de serem selecionados. Uma abstração do

método da roleta é apresentado na Figura 9, em que se deve executar os seguintes passos para selecionar um indivíduo:

1. Uma variável X recebe a somatória da aptidão de todos os indivíduos da população;
2. Uma variável aleatória r recebe um valor no intervalo $[0..X]$;
3. Cada indivíduo é percorrido, acumulando o valor da aptidão de cada indivíduo em uma variável S
4. Se $S \leq r$, então o indivíduo corrente é selecionado, senão continua a percorrer o próximo indivíduo.

Figura 9 – Método da roleta.



Fonte: Elaborada pelo autor.

O método de seleção por *ranking* foi introduzido por Mitchell (1998), o qual tenta impedir a convergência prematura e a dominância do indivíduo com maior aptidão. O primeiro passo deste método é a ordenação dos indivíduos pelo valor numérico que representa sua aptidão. Em seguida, um mapeamento com base nessa ordenação é realizado por um método de avaliação, como o método linear representado na Equação 2.1

$$E(i,t) = Min + (Max - Min) * \frac{rank(i,t) - 1}{N - 1}, \tag{2.1}$$

onde $rank(i,t)$ é a colocação em que um indivíduo i obteve na iteração t, N é o número de indivíduos da população, Min e Max representam o indivíduo com a pior e a melhor colocação no ranking, respectivamente.

A seleção por torneio escolhe aleatoriamente m indivíduos da população para competirem pela chance de reproduzir ou fazer parte de uma população intermediária. O indivíduo i com melhor valor de aptidão entre os m será selecionado. No caso de formação da população intermediária, esse processo é repetido até que sejam selecionados, por exemplo, o mesmo

Algoritmo 2 – Seleção por torneio.
Fonte: Adaptada de Mitchell (1998).

1: $n \leftarrow 0$ ▷ contador para o laço
2: **repita** ▷ executa até que uma população intermediária seja preenchida
3: Seleciona 2 indivíduos aleatoriamente
4: **se** $i_1.fitness > i_2.fitness$ **então** ▷ Competição entre os indivíduos
5: i_1 é escolhido para população intermediária
6: **senão**
7: i_2 é escolhido para população intermediária
8: **fim se**
9: $n \leftarrow n+1$
10: **até** n = tamanho da população original

número de indivíduos da população original. O Algoritmo 2 ilustra esse método, executando um torneio onde $m = 2$.

A operação de cruzamento ou *crossover* se baseia na seleção de dois indivíduos pai p_1 e p_2 para a geração de um novo indivíduo filho i. Como produto do cruzamento, um novo indivíduo será gerado herdando as características de seus pais. Dentre os possíveis operadores de cruzamento, pode-se descrever:

- **Um-ponto**: este operador define um ponto de corte aleatoriamente dos indivíduos pai p_1 e p_2. Um segmento do cromossomo do indivíduo filho é preenchido com os genes do início do cromossomo p_1 até o ponto de corte. A outra parte do filho i é preenchida com os genes do ponto de corte escolhido até o final do cromossomo de p_2.

- **Multi-ponto**: este cruzamento considera a escolha de dois ou mais pontos de corte. Considerando dois pontos de corte, a primeira parte de p_1 é copiado para o filho, em seguida a segunda parte de p_2, e a terceira parte de p_1 são copiadas para o indivíduo filho.

- **Uniforme**: cada gene do indivíduo filho é preenchido aleatoriamente copiando o gene de mesma posição de um dos indivíduos pai. As chances do indivídiuo pai p_1 repassar seu gene são as mesmas de p_2.

Conforme mencionado anteriormente, operadores de mutação conferem maior diversidade à população. Algoritmos evolutivos costumam utilizar uma taxa de probabilidade de mutação que determina as chances de um indivíduo sofrer uma alteração em seus genes, propiciando maior diversidade da população. Uma taxa de mutação pequena pode fazer o algoritmo cair em ótimos locais, levando a uma convergência precoce. Por outro lado, taxas de mutação mais altas aumentam a diversidade da população, mas tornam a convergência do algoritmo mais lenta. Portanto, são necessários ajustes na taxa de mutação e na escolha de operadores de mutação apropriados.

A Figura 10 apresenta o efeito de uma mutação de 100% e 10% em um cromossomo com 10 genes e codificação binária. De acordo com esse exemplo, ao definir um percentual de mutação de 10%, significa que apenas um gene sofrerá mutação, ao posso que a taxa de 100% de mutação produzirá a mutação de todos os 10 genes do cromossomo.

Figura 10 – Dois cenários de mutação de um indíviduo filho.

Fonte: Elaborada pelo autor.

Representação real

A representação binária é a mais simples para representar um cromossomo em algoritmos evolutivos e frequentemente utilizada nos trabalhos de Holland (1975). Entretanto, diversos problemas do mundo real são abstraídos em valores contínuos, tornando inviável o uso da representação binária (DEB *et al.*, 2000). Por exemplo, a codificação de uma representação real em binária pode exigir o uso de uma longa sequência de valores binários e, consequentemente, tornar o processo evolutivo mais lento.

A representação real é uma alternativa na qual os genes podem ser definidos por números reais, distribuídos em um intervalo contínuo de valores. Formalmente, a representação pode ser dada por cromossomos pais $p_1 = (p_1^1, p_1^2, p_1^3, ..., p_1^n,)$ e $p_2 = (p_2^1, p_2^2, p_2^3, ..., p_2^n,)$ e o cromossomo filho $f = (f_1, f_2, f_3, ..., f_n)$, onde $p_1 \in \Re^n$, $p_2 \in \Re^n$ e $f \in \Re^n$.

Cromossomos codificados com valores reais podem usufruir de diversos operadores, como:

- **Crossover média**: considerando cromossomos pais p_i^1 e p_i^2, o cromossomo filho f é calculado por $f = (p_1 + p_2)/2$;
- **Crossover média geométrica**: considerando cromossomos pais p_i^1 e p_i^2, o cromossomo filho f_i é resultante de $f_i = \sqrt{p_i^1 \cdot p_i^2}$, onde $i = 1, 2, ..., n$;
- **Crossover BLX-α**: considerando cromossomos pais p_i^1 e p_i^2, o cromossomo filho f_i é produzido por $f_i = p_i^1 + \beta(p_i^2 - P_i^1)$;
- **Crossover linear**: este operador gera três filhos f_1, f_2, f_3, calculados por $f_1 = 0,5 \cdot p_1 + 0,5 \cdot p_2$, $f_2 = 1,5 \cdot p_1 - 0,5 \cdot p_2$ e $f_3 = -0,5 \cdot p_1 + 1,5 \cdot p_2$. Somente o filho com melhor aptidão será escolhido.

A representação real também possibilita o desenvolvimento de diversos operadores de mutação, como:

- **Mutação uniforme**: substitui aleatoriamente um gene desde que o valor esteja dentro do domínio do problema;
- **Mutação gaussiana**: é a alteração de um gene escolhido por um valor aleatório dentro de uma distribuição normal;
- **Mutação *creep***: adiciona ao gene mutado um valor aleatório seguindo uma distribuição uniforme ou normal. Uma variação desse operador seria multiplicar o gene a ser mutado por um valor próximo de 1, produzindo uma pequena perturbação;
- **Mutação por limite**: operador substitui o gene mutado por um dos limites do intervalo do problema;
- **Mutação não-uniforme**: substitui um gene por um valor extraído de uma distribuição não uniforme, e;
- **Mutação não-uniforme múltipla**: todos os genes sofrem mutação seguindo as regras do operador de mutação não-uniforme.

Otimização Mutiobjetivo

Em diversos problemas de otimização há a necessidade de otimizar objetivos que podem ser conflitantes. Isso implica em melhorar um objetivo em detrimento de outro. A otimização multi-objetivo é definida como o processo de otimização de uma coleção de objetivos sistematicamente e simultaneamente (MARLER; ARORA, 2004). Em problemas mono-objetivo, utiliza-se a melhor solução encontrada durante o processo de otimização. Em problemas multi-objetivo não há garantia de que uma solução é melhor do que as demais. Desta forma, um problema de otimização multi-objetivo pode ser definido como:

$$\begin{array}{ll} \textit{Minimizar} & f(x) = [f_1(x), f_2(x), ..., f_k(x)]^T \\ \textit{Sujeito a} & g_j(x) \leq 0,\ j = 1,2,...,m, \\ & h_l(x) = 0,\ l = 1,2,...,e, \end{array} \tag{2.2}$$

onde k é o número de funções objetivo, m é o número de restrições de desigualdade e e representa o número de restrições de igualdade.

A melhor solução em problemas mono-objetivo é dada pela comparação de valores durante o processo de otimização. Se o objetivo é a minimização de uma função, logo a aptidão da solução candidata deve ser menor do que a melhor solução até então registrada. Com essa lógica é possível obter várias soluções, contudo somente a menor será considerada $f_{min} = min\{f_i(x)|x \in \Re^n\}$.

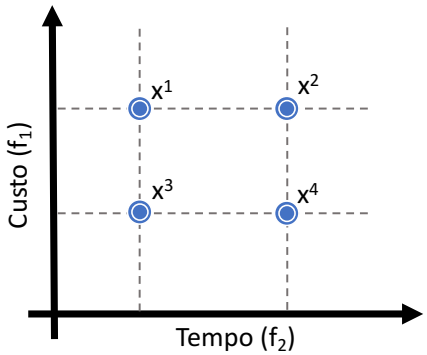
Entretanto, problemas de otimização multi-objetivo não permitem ordenar as soluções por qualidade. A escolha de uma solução é realizada a partir de um conjunto de soluções e critérios de avaliação, aplicando a relação de dominância de Pareto proposta inicialmente por Edgeworth (1881) e generalizada por Pareto (1896).

Definição 1. Segundo Bui e Alam (2008), a relação de dominância de Pareto estabelece que uma solução x_1 é dominante sobre x_2 (notado como $x_1 \prec_{pareto} x_2$), se e somente se

$$\begin{aligned} &\forall i \in \{1,...,k\} : f(x_1) \leq f(x_2) \\ e& \\ &\exists i \in \{1,...,k\} : f(x_1) < f(x_2) \end{aligned} \tag{2.3}$$

Um exemplo da dominância de Pareto é apresentado na Figura 11 com quatro pontos, levando em consideração dois fatores (tempo e custo). A solução x^3 tem valor menor que x^2 em ambos os fatores, logo é dito que $x^3 \prec_{pareto} x^2$. Além disso, x^3 também domina as soluções x^1 e x^4, embora possuam um dos fatores de valor igual ao de x^3. Contudo, x^1 e x^4 não são comparáveis entre si, ou seja, $x^1 \not\prec_{pareto} x^4$ e $x^4 \not\prec_{pareto} x^1$. Pode-se dizer que x^3 é uma solução não dominada. Considerando um cenário onde x^3 não exista, x^1 e x^4 passariam a ser as soluções não-dominadas.

Figura 11 – Representação do conceito de Dominância de Pareto.



Fonte: Elaborada pelo autor.

Definição 2. Segundo (BUI; ALAM, 2008), uma solução $x* \in \mathbb{X}$ é determinada como Pareto ótimo se não existir nenhuma outra solução $x \in \mathbb{X}$ que seja $f(x) \prec_{pareto} f(x*)$.

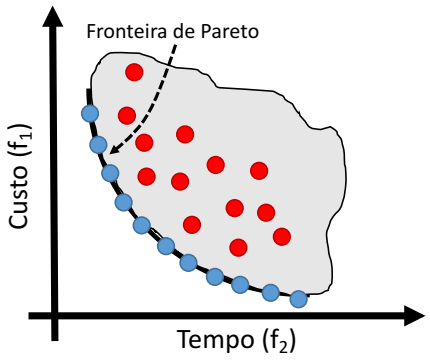
A ideia da Fronteira de Pareto é ilustrada na Figura 12 onde cada ponto representa uma solução. Cada ponto vermelho é uma solução dominada por alguma não-dominada (ponto azul) da Fronteira de Pareto.

Definição 3. Para um conjunto Pareto ótimo $\mathbb{P}$, a Fronteira de Pareto ($\mathbb{PF}$) é definido por

$$\mathbb{PF}^* = \{f(x) = (f_1(x),...,f_k(x)) | x \in \mathbb{P}^*\} \tag{2.4}$$

Há na literatura diferentes abordagens para a resolução de problemas multi-objetivo como a Multi-Criteria Decision Making (MCDM) (HO; XU; DEY, 2010) e a Evolutionary Multi-

Figura 12 – Conjunto de soluções e a Fronteira de Pareto.



Fonte: Elaborada pelo autor.

objective Optimization (EMO) (DEB; DEB, 2014). A primeira faz uso de métodos matemáticos, enquanto a EMO emprega a computação evolutiva para retornar uma fronteira próxima a Fronteira de Pareto ótima. Essa abordagem é utilizada nos problemas em que é impraticável obter o Pareto Ótimo em tempo viável.

Considerações finais

O uso de *workflows* mostra ser uma necessidade à execução de uma sequência de tarefas. Para atender algumas restrições estabelecidas pelo usuário, é preciso utilizar ambientes computacionais capazes de oferecer a elasticidade de recursos computacionais como as nuvens. A computação em nuvem é uma opção atrativa para executar *workflows* limitados pelo tempo de execução ou para instituições sem recursos para adquirir e manter equipamentos. Entretanto, antes da adoção da nuvem deve-se considerar aspectos de segurança de modo a evitar que dados privados e sensíveis sejam alvo de ataques maliciosos. Assim, a fase de escalonamento pode colaborar com a distribuição das tarefas e dos dados para recursos na nuvem que possam atender os requisitos de segurança de cada tarefa. O próximo capítulo aborda os conceitos da fase de escalonamento de *workflows* a fim de esclarecer as peculiaridades envolvidas neste processo.

CAPÍTULO

3

REVISÃO DA LITERATURA

3.1 Considerações iniciais

Para escalonar workflows no ambiente de computação em nuvem, os escalonadores podem considerar diferentes objetivos, incluindo a minimização do tempo total de execução, a minimização do custo monetário, o balanceamento de carga entre diferentes recursos ou outras restrições que vão de encontro com a necessidade do usuário. Nesse contexto, pesquisas têm apresentado abordagens que atendem diferentes configurações de ambientes de execução, critérios e restrições de escalonamento. Este capítulo apresenta um mapeamento sistemático dos algoritmos de escalonamento de workflows voltados para grades e nuvens. Além disso, algoritmos de escalonamento com atenção ao critério de segurança são discutidos para justificar a realização desta tese de doutorado.

3.2 Mapeamento sistemático

O mapeamento sistemático é um método voltado para a construção de esquemas de classificação de uma determinada área de interesse. Os resultados de um mapeamento incluem a apresentação da frequência e quantidade de publicações por categoria. Diferentes perspectivas podem ser consideradas para responder as questões de pesquisa. Esse é um método para realizar uma revisão dos estudos existentes em um tópico de pesquisa que pode ajudar na identificação de lacunas e sugerir pesquisas futuras. A seguir são apresentados os critérios de seleção de base de dados, o método de busca e os critérios de inclusão e exclusão de estudos relacionados ao escalonamento de workflow. Os resultados deste mapeamento contemplam a categorização dos algoritmos de escalonamento, a frequência de publicações de algoritmos de escalonamento de acordo com cada categoria, a identificação de pesquisadores, periódicos e conferências mais relevantes na área. Por fim, os principais artigos de cada categoria são brevemente descritos.

Seleção de base de dados

Pesquisas voltadas a algoritmos de escalonamento de workflows podem ser encontradas em periódicos e anais de conferências. O conteúdo de artigos é indexado em base de dados disponíveis para a comunidade científica. Esse mapeamento sistemático abrange quatro bases de dados comumente utilizados na ciência da computação: Elsevier Scopus (SCOPUS, 2018), IEEE Xplore (IEEE, 2018), ACM Digital Library (ACM, 2018) e Springer (SPRINGER, 2018). Embora haja outras bases de dados, essas quatro oferecem uma amostra razoável para uma revisão da literatura. Esse mapeamento considera trabalhos publicados até 25 de maio de 2018 e, desse modo, artigos publicados após essa data não foram cobertos.

Método de busca

O foco desse mapeamento é a busca de algoritmos de escalonamento de workflows em grades ou nuvens. Primeiramente, uma busca com data aberta foi realizada para identificar as primeiras publicações de escalonamento de workflows em grades. Após essa etapa, a busca foi restringida ao período de 1 de janeiro de 1999 a 25 de maio de 2018. Somente os artigos publicados em periódicos ou anais de conferências, e escritos na língua inglesa foram considerados. A *string* de busca adotada foi - "(workflow* OR dag) AND (schedul*) AND (distributed system* OR grid* or cloud*)" - onde o asterisco é um termo coringa que representa qualquer caracter ou grupo de caracteres. A busca considerou os campos título e palavras-chave.

Critérios de inclusão e exclusão

A atenção desse mapeamento é para algoritmos de escalonamento de workflows para grades ou nuvens. Os artigos devem apresentar abordagens de escalonamento ou melhorias em algoritmos existentes, incluindo a avaliação de desempenho. Registros duplicados, artigos de revisão, comparação de algoritmos ou sem avaliação foram excluídos. Além disso, monografias e teses também foram desconsideradas.

Critérios avaliados

Após o processo de filtragem de inclusão e exclusão de artigos, estes foram classificados em quatro categorias: (a) critério de escalonamento e evolução histórica, (b) citações de autores e periódicos/anais de conferências, (c) estudo geográfico e (d) colaboração de pesquisas.

Critérios de escalonamento e evolução histórica. A listagem e quantificação de publicações baseadas em cada critério de escalonamento pode auxiliar na busca de novas pesquisas na área. Uma análise da frequência de publicações de algoritmos de escalonamento de workflow foi realizada. Cada algoritmo foi classificado de acordo com um ou mais critérios de escalonamento (*makespan, deadline*, custo, orçamento, tolerância a falhas, consumo de energia e segurança).

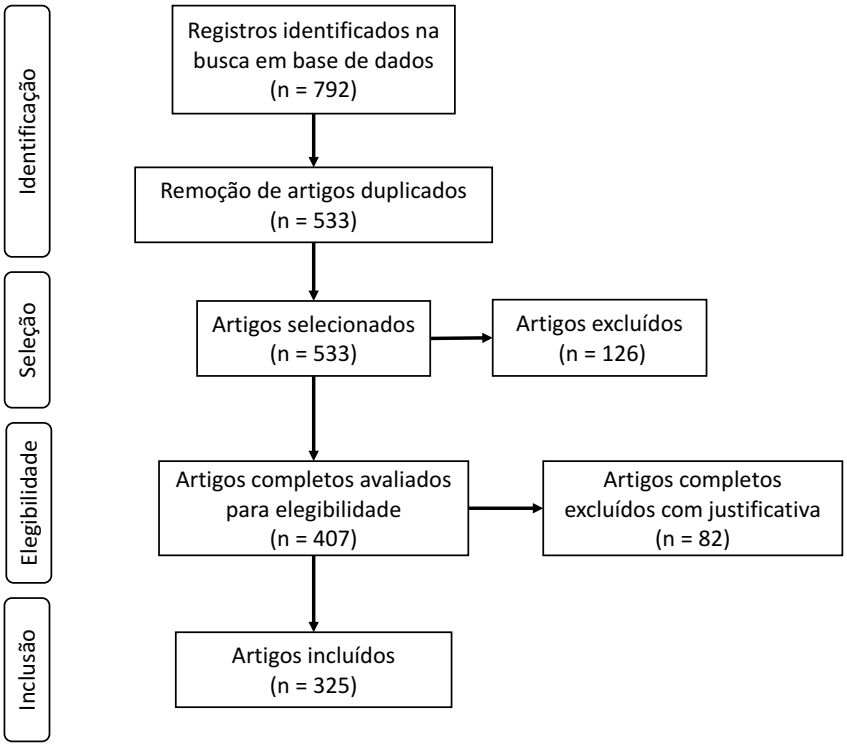
Autores e periódicos/conferências. O número de citações de cada artigo até 25 de maio de 2018 foi contabilizado. O número de publicações ($N1$), o número de citações ($N2$) de cada artigo, e a razão entre citações e pelo número de artigos publicados em conferências/periódicos foram calculados. Adicionalmente, um ranking de publicações entre os principais autores nessa área é apresentado.

Redes de colaboração. A organização entre grupos de pesquisa que estudam escalonamento de workflows, bem como os pesquisadores mais influentes também foram identificados.

Resultados

O protocolo PRISMA definido por Liberati *et al.* (2009) foi adaptado para esse mapeamento. O fluxo das atividades é sintetizado na Figura 13. A consulta nas bases de dados retornou 792 registros, dos quais 259 duplicados foram excluídos. Após a exclusão, restaram 533 que passaram pelos critérios de inclusão e exclusão, onde 126 foram descartados. Oitenta e dois artigos completos foram excluídos por apenas estenderem a análise de resultados ou por proporem um método sem avaliação de desempenho. Por fim, restaram 325 artigos para a extração de dados.

Figura 13 – Diagrama da revisão do mapeamento sistemático.

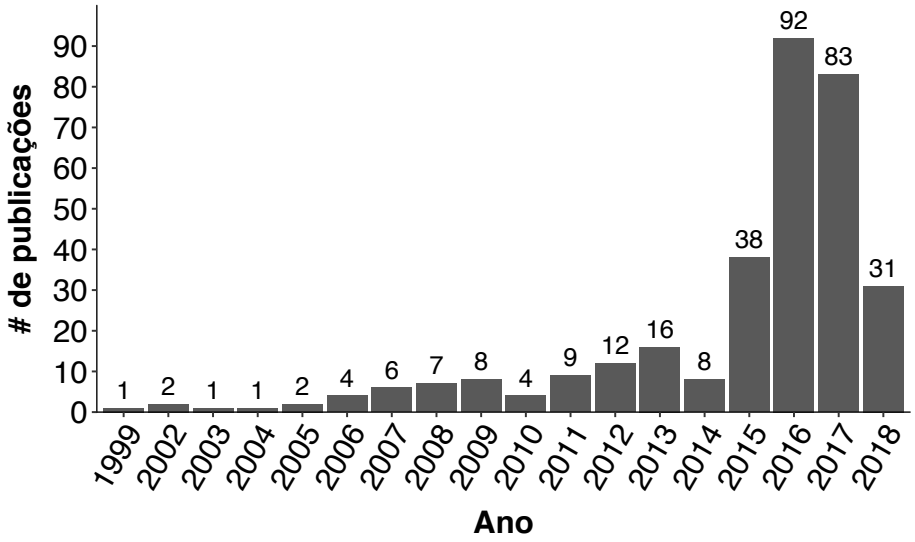


Fonte: Elaborada pelo autor.

Publicações de algoritmos de escalonamento

Uma análise cronológica das publicações de escalonamento de workflow é apresentada na Figura 14. Entre 1999 e 2008 foram publicados em média 3 artigos por ano. De modo geral, as publicações abrangiam o escalonamento de tarefas em *clusters* e grades computacionais. Possivelmente, o interesse nesse assunto cresceu em função do desenvolvimento de novos ambientes computacionais. As grades e, posteriormente, as nuvens têm imposto novos critérios e restrições de escalonamento que vão além da redução do tempo total de execução de um workflow. Isso fica evidente pelo número de publicações nos últimos anos que apresentam algoritmos que otimizam o escalonamento sob diferentes cenários e requisitos de QoS.

Figura 14 – Publicações de algoritmos de escalonamento de workflows. O rótulo sobre cada barra representa as publicações por ano.

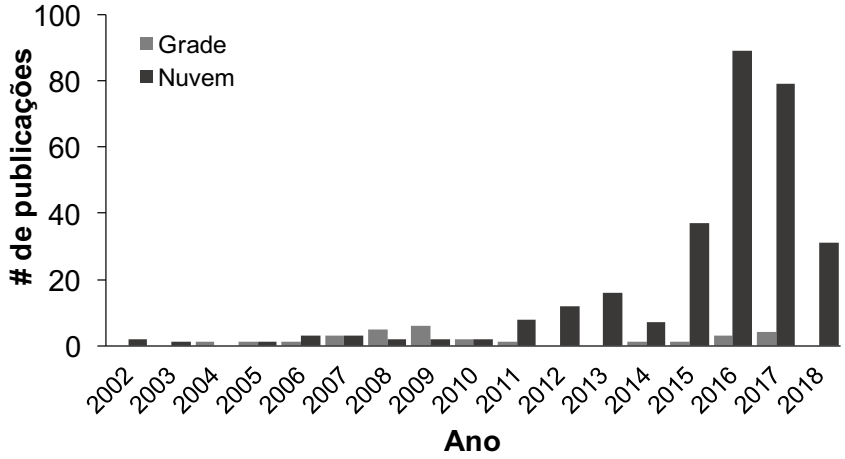


Fonte: Elaborada pelo autor.

As grades e nuvens são duas arquiteturas tipicamente utilizadas na execução de workflows. Uma análise das publicações frente a essas duas arquiteturas é apresentada na Figura 15. O crescimento de algoritmos baseados em grade seguiu o interesse de pesquisas nessa plataforma que teve significante atenção até o ano de 2010. A partir de 2011, as nuvens passaram a ser uma alternativa para a execução de aplicações devido às características como alta escalabilidade, provisionamento dinâmico e recursos computacionais heterogêneos. Contudo, as diferentes categorias de nuvens (públicas, privadas, comunitárias e híbridas) impuseram maior número e combinações de critérios e restrições aos algoritmos de escalonamento. Diante desse cenário, a otimização de critérios como o custo e satisfação de restrições como o *deadline* precisaram ser investigados.

O problema de escalonamento de workflows pode envolver um ou múltiplos objetivos, e ainda exigir que restrições sejam atendidas. Em sistemas em grade, a minimização do *makespan* é o critério mais comum. Outros estudos buscam a minimização do custo e atendem a um

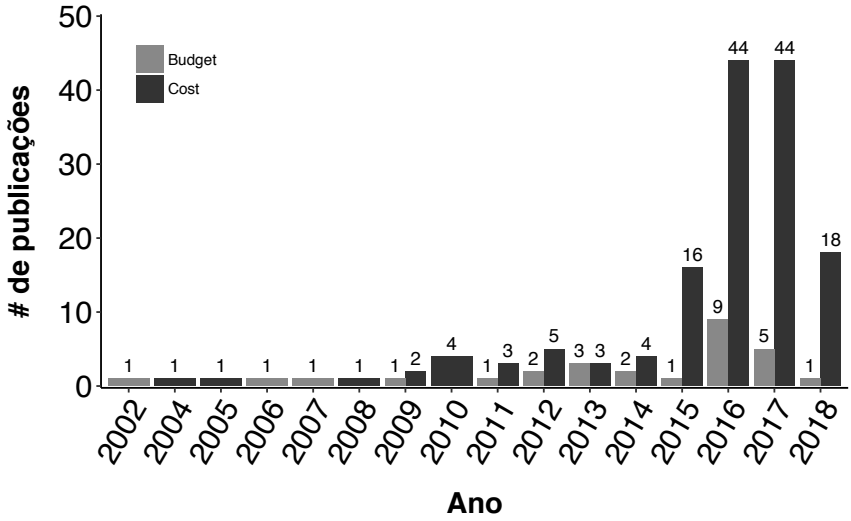
Figura 15 – Publicações de algoritmos de escalonamento de workflows voltados para grades ou nuvens.



Fonte: Elaborada pelo autor.

determinado *deadline*. A produção de algoritmos que envolvem tempo e custo é representada pelas Figuras 16 e 17. Pesquisas que buscam minimizar o *makespan* têm sido realizadas desde 1999, atingindo 176 artigos publicados até 2018. O *makespan* é um critério comumente otimizado devido à necessidade do usuário.

Figura 16 – Publicações de algoritmos de escalonamento com foco na minimização de *makespan* ou satisfação de prazo. O rótulo sobre cada barra representa o número de publicações por ano.

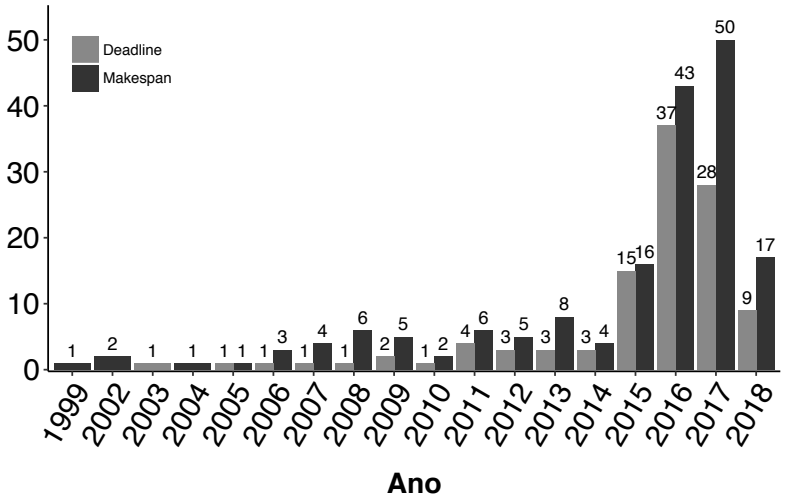


Fonte: Elaborada pelo autor.

Algoritmos de escalonamento que satisfazem *deadline* totalizam 111 publicações como mostrado na Figura 16. Apesar de ser uma restrição de tempo, o *deadline* é frequentemente considerado. A atenção no *deadline* está associada a possibilidade de minimizar o custo de execução utilizando VMs mais baratas, mas ainda satisfazendo o prazo estabelecido.

Algoritmos que consideram a minimização do custo monetário ou a execução de um workflow sem exceder o orçamento também são investigados. De acordo com a Figura 17, foram

Figura 17 – Publicações de algoritmos de escalonamento com foco na minimização de *custo* ou satisfação de orçamento. O rótulo sobre cada barra representa o número de publicações por ano.



Fonte: Elaborada pelo autor.

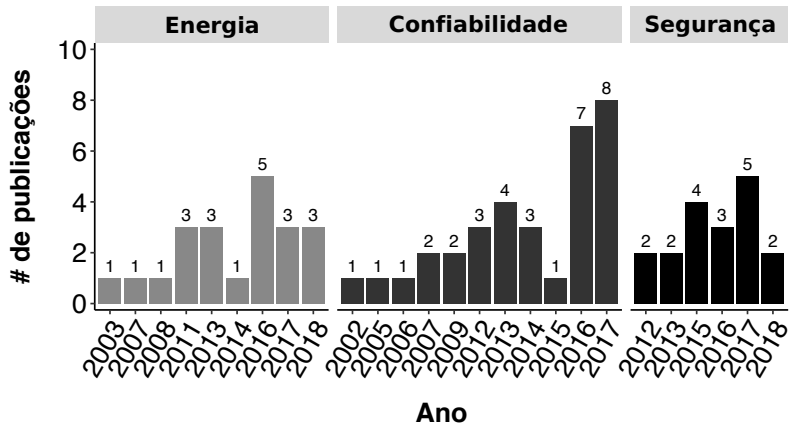
identificados 147 publicações relacionadas à minimização de custo e 29 que focam satisfazer o prazo de execução. Antes da computação em nuvem, pouca atenção era dada ao custo monetário, pois máquinas locais, *clusters* e grades privadas eram usadas. Entretanto, quando as nuvens passaram a ser usadas para execução de workflows, tanto a minimização do custo satisfazendo um prazo, quanto a minimização de *makespan* combinado com um orçamento tiveram que ser considerados no escalonamento de workflows em nuvens.

Outros critérios como o consumo de energia, tolerância a falhas e segurança também têm recebido atenção. O primeiro passou a ser uma preocupação por causa do crescimento de gasto de energia e o impacto ambiental para produzi-la. Dentre os 24 artigos com foco neste critério, o gerenciamento de voltagem dinâmica das máquinas é a abordagem mais explorada. Porém, nuvens públicas compartilham seus recursos entre diferentes usuários, o que torna essa solução pouco razoável. Diante dessa circunstância, os estudos têm se voltado para otimizar o uso de energia para a transmissão de dados na rede ao invés de implementá-lo nos nós de processamento. Atualmente, os trabalhos existentes ainda não consideram a energia gasta durante a transmissão de dados, utilizando apenas modelos simplistas para o cálculo do tempo de transmissão (PEDRAM, 2012).

A tolerância a falhas ou confiabilidade é a capacidade de lidar com potenciais consequências em caso de erros durante a execução de um workflow. Por exemplo, embora máquinas de baixa-confiabilidade sejam mais baratas, elas podem resultar em falhas de execução. Um total de 25 artigos envolvem algoritmos que tentam atribuir instâncias de máquinas virtuais a tarefas críticas e instâncias não-confiáveis (ex. Amazon Spot) para as demais tarefas. Algoritmos de escalonamento tolerantes a falha tipicamente otimizam tempo ou o custo da execução.

Cada tarefa de um workflow geralmente manipula dados que podem ser confidenciais e/ou

Figura 18 – Publicações de algoritmos de escalonamento com foco na otimização do consumo de energia, confiabilidade e segurança. O rótulo sobre cada barra representa o número de publicações por ano.



Fonte: Elaborada pelo autor.

sensíveis. Uma informação sensível compreende um conteúdo que não é legalmente protegido, mas também não deve se tornar pública e pode ser aberta sob circunstâncias limitadas. Dados confidenciais são aqueles que há obrigação legal de não divulgação. Esses dados exigem os mais altos níveis de restrição devido ao risco ou dano que resultará da divulgação, ou de seu uso inadequado. A execução de workflows que envolvem dados confidenciais e/ou sensíveis pode utilizar métodos para a proteção desse dado. Neste mapeamento foram identificados 32 artigos que tratam a segurança na etapa de escalonamento de workflows (Figura 18). Geralmente, há duas abordagens. Uma utiliza serviços de autenticação, criptografia e verificação de integridade de dados, enquanto a outra evita a transferência de dados sensíveis/confidenciais para nuvens não-seguras. Além disso, métodos de particionamento também são usados para classificar quais grupos de tarefas devem ser executados em nuvens privadas e públicas. O número de publicações de algoritmos de escalonamento baseados em segurança é pequeno (6%) em comparação às outras métricas de otimização como tempo (32%) e custo (27%). Diversas instituições têm usado nuvens públicas ou híbridas, entretanto, outras ainda tem rejeitado a sua adoção por causa dos riscos de segurança que permeiam as nuvens. Assim, a tendência é que a segurança seja um requisito essencial na execução de workflows, incluindo a etapa de escalonamento.

Pesquisadores, periódicos e conferências

Este estudo apresenta os dez artigos mais citados sobre algoritmos de escalonamento de workflows (Tabela 2). O número de citações de cada artigo foi obtido por meio do Google Scholar (GOOGLE, 2018). O HEFT (TOPCUOGLU; HARIRI; WU, 2002) é o artigo mais citado (2224 citações), frequentemente usado para a comparação com outros algoritmos de escalonamento baseados em listas. Embora o algoritmo não tenha sido proposto na era da computação em nuvem, este pode ser adaptado para as características da nuvem. Publicado há 16 anos, o HEFT

apresenta uma média de 139 citações por ano, enquanto o segundo artigo mais citado (PANDEY *et al.*, 2010) acumula 83 citações/ano. Rajkumar Buyya é um pesquisador em destaque nessa área e está presente em cinco dos dez artigos mais citados, acumulando 2094 citações. Esse alto número de citações possivelmente está relacionado à variedade de cenários e critérios de escalonamento cobertos em seus estudos.

Tabela 2 – Artigos de algoritmos de escalonamento de workflows mais citados (ordenado pelo número de citações).

#	Título	Periódico/Conferência	Autor(es)	Ano
2224	HEFT: Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing	IEEE Transactions on Parallel and Distributed Systems	Haluk Topcuoglu; Salim Hariri; Min-you Wu	2002
664	A Particle Swarm Optimization-Based Heuristic for Scheduling Workflow applications in Cloud Computing Environments	IEEE International Conference on Advanced Information Networking and Applications	Suraj Pandey; Linlin Wu; Siddeswara Mayura Guru; Rajkumar Buyya	2010
462	Cost-based scheduling of scientific workflow applications on utility grids	International Conference on e-Science and Grid Computing	Jia Yu; Rajkumar Buyya; Chen Khong Tham	2005
416	Auto-Scaling to Minimize Cost and Meet Applications Deadlines in Cloud Workflows	International Conference for High Computing, Networking, Storage and analysis	Ming Mao; Marty Humphrey	2011
364	Scheduling Scientific Workflow applications with Deadline and Budget Constraints Using Genetic algorithms	Scientific Programming	Jia Yu; Rajkumar Buyya	2006
333	Deadline-Constrained Workflow Scheduling algorithms for infrastructure as a Service Clouds	Future Generation Computer Systems	Saeid Abrishami; Mahmoud Naghizadeh; Dick Epema	2013
327	An ant Colony Optimization approach to a Grid Workflow Scheduling Problem with Various QoS Requirements	IEEE Transactions on Systems, Man, and Cybernetics	Wei-Neng Chen; Jun Zhang	2009
323	Power-Aware Scheduling of Bag-of-Tasks applications with Deadline Constraints on DVS Clusters	IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing	Kim Kyong Hoon; Buyya Rajkumar; Kim Jong	2007
293	Algorithms for Cost- and Deadline-Constrained Provisioning for Scientific Workflow Ensembles in Iaas Clouds	Future Generation Computer Systems	Maciej Malawski; Gideon Juve; Ewa Deelman; Jarek Nabrzyski	2015
281	Deadline Based Resource Provisioning and Scheduling algorithms for Scientific Workflows on Clouds	IEEE Transactions on Cloud Computing	Rodriguez, Maria Alejandra; Buyya, Rajkumar	2014

Fonte: Dados da pesquisa.

Foram identificados 633 autores e co-autores nos artigos publicados, em que os 30 mais produtivos estão listados na Tabela 3. Rajkumar Buyya detém quase o dobro de publicações (17) em comparação ao Xiaoping Li (10) presente na segunda colocação. Albert Zomaya, Prasanta Jana e Indrajeet Gupta possuem 9, 8 e 6 publicações, respectivamente. Entretanto, eles aparecem como co-autores na maioria dos artigos. Há ainda outros 52 pesquisadores que estão envolvidos na produção de 3 a 6 artigos; 110 com duas publicações, e; 466 presentes em uma única publicação.

A identificação dos periódicos com maior número de artigos relacionados ao escalonamento de workflows pode auxiliar pesquisadores inexperientes no momento da submissão de seus estudos. Foram contabilizados 88 periódicos com 172 artigos publicados. A Tabela 4 lista os 10 periódicos com maior número de publicações. Os seguintes parâmetros bibliométricos

Tabela 3 – Ranking de autores (ordenado pelo número de publicações).

Autor(es)	# publicações
Rajkumar Buyya	17
Xiaoping Li	10
Albert Zomaya	9
Prasanta Jana	8
Indrajeet Gupta	7
Madhu Kumar; Young Choon Lee; Mikhail Melnik; Mahmoud Naghibzadeh; Radu Prodan; Rizos Sakellariou; Xiaomin Zhu	6
Saied Abrishami; Vahid Arabnejad; Jorge Barbosa; Kris Bubendorfer; Ewa Deelman; Jia Yu; Wei Zheng	5
Hamid Arabnejad; Nikolay Butakov; Huangke Chen; Ritu Garg; Mala Kalra; Edmundo Madeira; Bryan Ng; Kotagiri Romamohanarao; Javid Taheri; Alex Visheratin; Henan Zhao	4

Fonte: Dados da pesquisa.

foram considerados: a) N_1: número de artigos de escalonamento de workflows; b) N_2: total de citações que os N_1 artigos foram citados, e; c) N_2/N_1: média de citações dos artigos.

Tabela 4 – Os dez periódicos com maior número de publicações sobre algoritmos de escalonamento de workflows (ordenada por ordem decrescente do número de publicações). N_1 : quantidade de publicações; N_2 : citações; N_2/N_1 : média de citações por artigo.

Periódico	N1 (documentos)	N2 (citações)	N2/N1 razão
Future Generation Computer Systems	29	1450	50
IEEE Transactions on Parallel and Distributed Systems	13	4100	315
Concurrency and Computation: Practice and Experience	13	412	29
Journal of Parallel and Distributed Computing	7	418	59
Journal of Supercomputing	7	35	5
IEEE Transactions on Cloud Computing	5	409	81
Cluster Computing	4	38	9
Journal of Grid Computing	3	126	42
Computing	3	21	7
Information Sciences	3	133	44

Fonte: Dados da pesquisa.

De acordo com a Tabela 4, os artigos de escalonamento de workflows têm sido publicados em diferentes periódicos, mas o Future Generation Computer Systems (FGCS), o IEEE Transactions on Parallel and Distributed Systems (TPDS) e o Concurrency and Computation (CCPE) publicaram mais de 60% das publicações. Os três periódicos têm interesse no tema de escalonamento de workflows. Atualmente, o fator de impacto (IF) do FGCS e TPDS é 3.99 e 4.18, respectivamente, seguido do CCPE com 1.13. Juntos os artigos desses três periódicos foram citados 5962 vezes. Embora o TPDS seja o periódico com maior número absoluto de citações (4100), somente o artigo HEFT (como visto na Tabela 2) detém 2224 delas. Embora o Journal of Supercomputing esteja na quinta colocação com sete artigos, este possui apenas 5 citações por artigo em média. Na décima colocação, o periódico Information Sciences apresenta 44 citações por artigo. Essa diferença de citações talvez seja explicada devido às publicações do Journal of Supercomputing serem mais recentes (a partir de 2016) do que as da Information Sciences (a partir de 2009).

Várias conferências têm dado atenção ao tópico de escalonamento de workflows. Foram

computadas 94 conferências com 131 artigos publicados. As dez conferências que mais publicaram artigos de escalonamento de workflows são mostradas na Tabela 5. Essas dez conferências apresentam 37 artigos (aprox. 40% das publicações), ao passo que as demais 84 conferências publicaram 94 trabalhos. A IEEE International Conference on Advanced Information Networking and Application (AINA) e a IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID) são as conferências que apresentam maior média de citações por artigo. Isso possivelmente é o reflexo da boa qualidade dos artigos que passam por um rigoroso processo de revisão e também à taxa de aceitação baixa (aprox. 20% em 2017).

Tabela 5 – As dez conferências que mais publicaram artigos sobre algoritmos de escalonamento de workflows (ordenada por ordem decrescente do número de publicações). N_1 : quantidade de publicações; N_2 : citações; N_2/N_1 : média de citações por artigo.

Conferência	N1 (documentos)	N2 (citações)	razão N2/N1
IEEE International Parallel and Distributed Processing Symposium	6	438	73
IEEE International Conference on Advanced Information Networking and Applications	5	799	159
IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing	5	740	148
IEEE International Conference on Cloud Computing	4	74	18
International Conference on Computational Science	4	77	19
International Conference on Parallel Processing	4	194	48
IEEE International Symposium on Network Computing and applications	4	13	3
International Young Scientists Conference on Computational Science	4	13	3
International Conference on Advanced Cloud and Big Data	3	7	2
International Conference on Contemporary Computing	3	8	2

Fonte: Dados da pesquisa.

Estudo geográfico

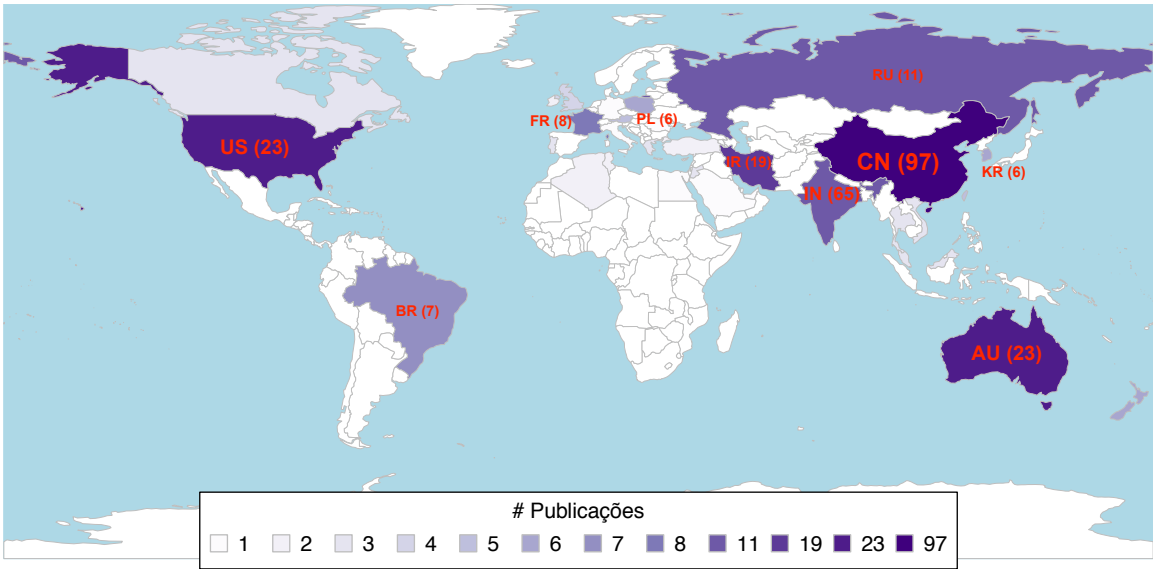
Uma análise dos países onde os estudos de escalonamento de workflow foram conduzidos é apresentada na Figura 19. Foram contabilizados 28 países em que 65% dos artigos foram desenvolvidos em instituições asiáticas, sendo a China responsável por 30% das publicações em todo o mundo. A Europa produziu 11% das publicações, seguida pela América do Norte e Oceania que apresentaram cerca de 9% das publicações cada. O continente africano igualmente a América do Sul participam com 2.1% cada.

3.3 Trabalhos relacionados

Escalonamento *best-effort*

Os primeiros algoritmos de escalonamento focaram na otimização de um objetivo. O *makespan* é um critério frequentemente escolhido no escalonamento. No trabalho de Pandey *et al.* (2010) é proposta uma abordagem que busca minimizar o custo monetário para executar um workflow com grande volume de dados usando a otimização por enxame de partículas. Tanto a transferência de dados entre as máquinas virtuais bem como o custo de execução foram levados

Figura 19 – Países que realizam pesquisas em algoritmos de escalonamento de workflows. Países que apresentaram mais de seis publicações foram destacados em vermelho.



Fonte: Elaborada pelo autor.

em consideração. O custo de cada recurso é calculado separadamente de modo a otimizar o custo total de execução, além de garantir que as tarefas sejam distribuídas entre os recursos alocados. Bilgaiyan, Sagnika e Das (2014) resolveram o mesmo problema, porém, utilizou a meta-heurística de otimização baseada em enxame de gatos (CHU; TSAI; PAN, 2006).

Embora a abordagem *best-effort* minimize um determinado objetivo, aplicações do mundo real podem estar limitadas a algumas restrições. O escalonamento baseado em restrição tenta otimizar algum objetivo, porém, restrito a outros requisitos. No trabalho de Chen e Zhang (2009), foi proposto um algoritmo baseado em múltiplas restrições utilizando a meta-heurística de otimização de colônia de formigas. A abordagem busca encontrar um escalonamento que atenda todas as restrições e otimize o objetivo escolhido pelo usuário. Entre as restrições de escalonamento atendidas, o *deadline* e orçamento são duas categorias que são vastamente estudadas na literatura.

Escalonamento de workflow baseado no *deadline*

Além do tempo de execução e do custo monetário, há outras restrições importantes em aplicações que são executadas na nuvem. Geralmente, recursos mais rápidos são mais caros do que os de menor desempenho. Portanto, o escalonador deve tratar o custo/benefício entre o tempo de execução e o custo monetário ao escolher os recursos computacionais. Para tratar esse problema, é interessante optar por minimizar o custo monetário de acordo com um *deadline*.

A solução mais simples para o escalonamento de workflows restritos a um *deadline* é o uso do método *Minimum Critical Path* (MCP). Os recursos mais baratos são alocados às tarefas não críticas, de modo que a execução destas não seja influenciada. Assim, o custo é reduzido

sem que o tempo de execução viole o *deadline*. Geralmente, o escalonamento com restrição de *deadline* apresenta três fases: particionamento de tarefas, atribuição do *deadline* e seleção de recursos. Abrishami, Naghibzadeh e Epema (2013) propuseram o método IC-PCP e IC-PCPD2 para escalonamento de *workflows* na nuvem. O IC-PCP distribui *deadline* para as partições, enquanto o IC-PCPD2 ainda atribui um *sub-deadline* para cada tarefa.

Utilizar meta-heurísticas é outro modo de otimizar o escalonamento com restrição de *deadline*. Yu e Buyya (2006b) aplicaram um algoritmo genético para minimizar o custo em relação à restrição de *deadline*. Os pesquisadores propuseram um método de representação de escalonamento bidimensional em que a função objetivo considera o menor custo que não viole o *deadline* estabelecido. Wu *et al.* (2010) empregaram a otimização baseada em enxame de partículas (RDPSO) para reduzir o custo total de execução respeitando o *deadline*. No trabalho de Rodriguez e Buyya (2014) é apresentado uma combinação de provisionamento de recursos e escalonamento para a execução de *workflows*. Essa solução também usa o enxame de partículas em que fatores como o tempo de inicialização de VM e o tempo de transmissão de dados na rede são levados em consideração.

Escalonamento de workflow restrito ao orçamento

O escalonamento de *workflows* considerando a restrição de orçamento tem o objetivo de executar o workflow o mais rápido possível com o recurso disponível. Abordagens como heurísticas de único escalonamento, *back-tracking* e meta-heurísticas buscam atender o requisito de orçamento. A heurística de único escalonamento distribui um *workflow* somente uma vez sem realizar operações de *back-tracking*. Algoritmos como GreedyTimeCD (YU; RAMAMOHANARAO; BUYYA, 2009), BHEFT (ZHENG; SAKELLARIOU, 2013a) e HBCS (ARABNEJAD; BARBOSA, 2014) são três heurísticas estendidas do HEFT (TOPCUOUGLU; HARIRI; WU, 2002), as quais se diferem na fase de seleção de recursos. O GreedyTimeCD distribui estaticamente o orçamento para cada tarefa na proporção de seu tempo de execução médio antes do escalonamento. Então, seleciona o recurso que permite finalizar a tarefa o mais rápido possível dentro do orçamento disponível. O BHEFT é semelhante ao GreedyTimeCD, exceto pela distribuição que é realizada dinamicamente.

Heurísticas que usam *back-tracking* partem de atribuições que apresentam desempenho razoável sob um dos dois critérios de otimização (*makespan* ou orçamento) e trocam as tarefas entre os recursos na tentativa de otimizar o outro critério. Há duas abordagens baseadas em *back-tracking*: LOSS e GAIN. A primeira inicia com atribuição ótima para *makespan* com o custo extrapolando o limite do orçamento. As tarefas são trocadas entre diferentes recursos até que o custo de execução atinja um valor menor ou igual ao orçamento disponível. Por outro lado, o GAIN inicia um esquema de menor custo monetário e realiza a troca de tarefas entre os recursos com maior peso até que não haja mais orçamento disponível. De acordo com Sakellariou *et al.* (2007), a abordagem LOSS consegue obter soluções com melhor *makespan* do que o GAIN, pois, o LOSS usa uma heurística que gera um escalonamento inicial mais eficiente. O GAIN

apresenta melhor *makespan* quando o orçamento disponível é próximo ao menor custo possível.

A abordagem com algoritmo genético apresentado por Yu e Buyya (2006a) é capaz de buscar por soluções ótimas com restrição de orçamento. A diferença está na função objetivo que incentiva o algoritmo escolher escalonamentos com menor *makespan* sem violar o orçamento.

Escalonamento baseado em múltiplos critérios

O escalonamento de *workflows* com múltiplos critérios leva em consideração dois ou mais objetivos que eventualmente podem ser conflitantes. Nesse âmbito, são consideradas as abordagens por agregação, $\in$ e Pareto.

A solução mais simples para otimizar um problema multiobjetivo é transformá-lo em uma otimização mono-objetivo. A dificuldade nesta abordagem é a definição dos pesos de cada objetivo. No trabalho de Li *et al.* (2011), foi proposta a heurística CCSH que estende a HEFT em que os seus objetivos são minimizar o *makespan* e o custo. Para isso, foi introduzido um peso entre o tempo de início antecipado e o custo de execução da tarefa para selecionar recursos. Garg, Buyya e Siegel (2010) propuseram três heurísticas para agregar objetivos conflitantes (*makespan* e custo monetário) em um único objetivo.

Fard *et al.* (2012) desenvolveram um algoritmo de múltiplo objetivo baseado em lista. As tarefas são selecionadas usando um $rank_u$ e a seleção de recursos é aproximada aplicando duas estratégias: maximizando a distância para o vetor de restrições para soluções dominantes e minimizá-la se não houver solução dominante.

Dogan e Ozguner (2002) propuseram um escalonamento baseado em nível dinâmico para a otimização de dois critérios (desempenho e confiabilidade). O RDLS adiciona um custo para a métrica de nível dinâmico do DLS. Posteriormente, essa abordagem foi melhorada através da otimização dos níveis de desempenho e confiabilidade usando algoritmo genético. Eles usam a técnica de agregação proposta por Hakem e Butelle (2007) para atingir desempenho e confiabilidade simultaneamente. Lee, Subrata e Zomaya (2009) apresentaram um algoritmos de escalonamento adaptativo (ADOS) que visa reduzir o *makespan* e aumentar o uso dos recursos simultaneamente. O algoritmo gera aleatoriamente uma solução inicial que é submetida ao ADOS repetidamente de modo que a taxa de utilização de recursos seja maximizada e o *makespan* reduzido. Esse algoritmo usa a técnica *branch-and-boud* e dois operadores de mutação.

Diferentemente da abordagem por agregação que resulta em somente um esquema de escalonamento como solução final, a abordagem por Pareto gera um conjunto de escalonamentos não dominados, oferecendo mais flexibilidade para os usuários quando estimarem as suas restrições. Bessai *et al.* (2012) usam a abordagem de Pareto para resolver escalonamento de *workflows* bi-critério (tempo e custo). O algoritmo é implementado com base no escalonamento em lista. Foram propostos três algoritmos de Pareto seguindo três políticas de seleção de recursos baseados em custo, tempo e tempo/custo. Para cada algoritmo, um escalonamento candidato é buscado usando as estratégias *top-down*, *bottom-up*, *mixed*. A estratégia *mixed* seleciona

as tarefas que pertencem ao nível intermediário. As tarefas pertencentes a um nível menor ou maior do que o nível inicial são selecionadas usando as estratégias *bottom-up* e *top-down*, respectivamente. Entre todas as soluções candidatas, os escalonamentos não dominantes são mantidos como escalonamentos de Pareto.

Yu, Kirley e Buyya (2007) usaram um algoritmo evolutivo multiobjetivo para buscar soluções que se encontram na fronteira de Pareto. Esta abordagem é capaz de minimizar simultaneamente os objetivos de tempo de execução e custo monetário, enquanto atende as restrições de *deadline* e orçamento. Para atingir esse objetivo, foram desenvolvidos funções objetivo com penalização quando as restrições não são atendidas. Os algoritmos MOEAs, NSGA-II, DEB2002FAST e SPEA2 (ZITZLER *et al.*, 2001) e MOEA, PAES (KNOWLES; CORNE, 1999) foram implementados e comparados em diferentes níveis de restrições e estruturas de *workflows*. Talukder, Kirley e Buyya (2009) resolveram o mesmo problema usando a abordagem Evolução Diferencial Multiobjetivo (MODE).

Workflow as a Service

Diferentemente do escalonamento de tarefas de um único *workflow*, há a possibilidade de escalonar múltiplas instâncias de workflows, podendo sofrer variações da carga de trabalho ao longo da execução. Esse tipo de escalonamento é denominado por *Workflow as a Service* (WaaS). Nesse contexto, há um provedor de aplicação que é oferecido como um serviço para que usuários possam executar *workflows*. Sistemas WaaS podem adicionar ou remover recursos do provedor de serviço de acordo com a flutuação da carga de trabalho, escalonando as tarefas e os recursos adquiridos.

O principal desafio no escalonamento de workflows na nuvem é determinar a quantidade de recursos necessária para minimizar o custo monetário e maximizar a taxa de utilização dos recursos. Em função disso, Byun *et al.* (2011b) desenvolveram um algoritmo de escalonamento (PBTS) que estima o número mínimo de recursos necessários para a execução de um *workflow* de acordo com um *deadline*. O PBTS é uma extensão da abordagem BTS (BYUN *et al.*, 2011a) que busca reduzir o número de recursos necessários para executar um *workflow*. Isso é realizado atrasando a execução de uma tarefa para que outras possam explorar esse tempo artificialmente ocioso. Por outro lado, o PBTS auxilia na elasticidade ajustando o número de recursos durante a execução das tarefas. Inicialmente, ele determina a distribuição das tarefas a fim de determinar a capacidade dos recursos. Posteriormente, o melhor número de recursos por unidade de tempo cobrado é calculado pelo algoritmo BTS.

Um outro meio para escalonar múltiplos *workflows* é por meio da fusão de suas tarefas, concentrando-as em um único *workflow*. Hönig e Schiffmann (2006) usaram este método para melhorar o paralelismo e utilização de recursos. Os autores propuseram as abordagens de decomposição e fusão de múltiplos *workflows* e dois algoritmos de escalonamento baseados em política justa. Entretanto, é importante notar que a fusão de workflows não é aplicável em ambientes em que a chegada de tarefas seja dinâmica.

Para lidar com diferentes padrões de submissão de *workflows*, o escalonamento dinâmico parece ser razoável. No trabalho de Stavrinides e Karatza (2011) foram avaliadas várias implementações de algoritmos com base nesta ideia. A partir da combinação de políticas e técnicas de seleção, os resultados mostraram que a combinação da política do primeiro *deadline* a vencer e a seleção *best fit* superaram todas as outras heurísticas. Tsai *et al.* (2012) elaboraram uma nova abordagem para melhorar o desempenho do escalonamento proposto por Stavrinides utilizando duas técnicas. Em Bittencourt e Madeira (2007), uma abordagem de agrupamento de tarefas foi proposta para reduzir o tempo de comunicação entre as tarefas e balancear o início da tarefa e a lacuna de tempo ocioso ao invés da alocação *best fit* tradicional.

O modelo hierárquico colabora no escalonamento estático e dinâmico para a geração de soluções. Um estudo de Iverson e Özgüner (1999) apresenta uma estratégia de dois níveis descentralizada. No nível alto, esse mapeia a tarefa a um grupo de recursos usando escalonamento o DLS (SIH; LEE, 1993). Já no nível mais baixo, seleciona um recurso que seja capaz de terminar a tarefa o mais rápido possível. No trabalho de Yu e Shi (2008) foi desenvolvida uma estratégia de escalonamento composta por um Planejador de DAG, um agrupador de tarefas e um executor. Para cada *workflow* recebido, uma instância do planejador prioriza as tarefas usando a abordagem HEFT (TOPCUOUGLU; HARIRI; WU, 2002). O grupo de tarefas armazena as tarefas prontas de todos os *workflows* submetidos. Em seguida, o executor prioriza todas as tarefas prontas e as executa nos recursos disponíveis, assim reduzindo o tempo total entre a submissão e o término da execução do *workflow*.

Escalonamento de workflow robusto

A maioria dos algoritmos de escalonamento se baseia em modelos de tempo de execução e transmissão de dados precisos. Porém, nuvens estão sujeitas à variação de desempenho e falhas que podem levar a tempos de execução incertos. Dejun, Pierre e Chi (2009) avaliaram o desempenho da Amazon EC2 no provisionamento de recursos de aplicações orientadas a serviço. Os resultados mostraram que o desempenho de CPU e I/O de disco de pequenas instâncias são relativamente estáveis em execuções de longos períodos. Entretanto, o desempenho dos comportamentos de várias instâncias pequenas divergem consideravelmente. Desse modo, a variação de desempenho afeta a execução e pode aumentar o *makespan*. Para lidar com esse problema, abordagens robustas de escalonamento tentam tratar a variação de desempenho do ambiente em aplicações onde o custo e tempo são parâmetros críticos.

Para minimizar o impacto causado pela inconstância da nuvem, sugere-se o escalonamento em tempo de execução. Isso significa que cada tarefa só poderá ser escalonada quando estiver em estado de pronto. Malewicz *et al.* (2007) desenvolveram uma heurística que considera a estrutura do workflow para melhorar o desempenho da aplicação. Contudo, a avaliação de Szepieniec e Bubak (2008) mostrou que a efetividade dessa abordagem é limitada por não considerar o tempo de execução da tarefa.

Outra ideia é adaptar algoritmos de escalonamento estático em ambientes inconstantes.

Neste contexto, superestimar o tempo de execução das tarefas individualmente pode ser uma solução. Isso resultaria em desperdício de recursos se a duração da tarefa for muito menor que o tempo estimado. Outra abordagem seria a realocação dinâmica de tarefas para recursos ociosos durante a execução. Porém, a migração de tarefas entre diferentes máquinas virtuais representa um *overhead* extra de comunicação e sincronização.

Um estudo de Sakellariou e Zhao (2004) aponta a importância de usar um bom algoritmo estático para gerar um escalonamento razoável antes de iniciar a execução de escalonamento dinâmico, pois, uma boa atribuição inicial reduz a necessidade de reescalonamentos posteriores (CANON *et al.*, 2008). Zheng e Sakellariou (2013b) introduziram uma abordagem baseada em Monte Carlo chamada de MCS. Cada escalonamento consiste em duas fases, uma de produção e a outra de seleção. A primeira fase gera o número de escalonamentos usando uma heurística determinística a partir de amostras de *workflows*. A segunda fase seleciona o melhor escalonamento para amostras pré-configuradas de um *workflow*. Os resultados apresentaram que a segunda fase sempre produz escalonamentos melhores que os produzidos pela primeira fase.

A tolerância a falhas é outro aspecto de robustez que pode tratar falhas em nível de tarefa e *workflow*. O primeiro nível pode ser dividido em quatro tipos como tentar novamente, alternar recursos, uso de checkpoint/reinicialização (LEON; FISHER; STEENKISTE, 1993), (STELLNER, 1996), (PLANK *et al.*, 1995), e técnicas de replicação (ABAWAJY, 2004) (HWANG; KESSELMAN, 2003). Em Wang, Ramamohanarao e Chen (2012), a técnica de *checkpoint* e reescalonamento foi adotada para tratar falhas de tarefas para *workflows* com grande volume de dados gerados durante a sua execução usando algoritmos genéticos.

As instâncias spot do Amazon EC2 são recursos computacionais extra na Nuvem da Amazon que podem ser alocados com grandes descontos em relação ao preço das instâncias sob demanda. A diferença entre instâncias spot e sob demanda é que as instâncias spot podem ser interrompidas com aviso de dois minutos quando a EC2 precisar de recursos para novas instâncias sob demanda. Apesar do término prematuro de uma *spot instance*, esta pode reduzir o custo em até 70% comparada às instâncias *on-demand*. Esse cenário incentiva pesquisas de algoritmos de escalonamento capazes de tolerar falhas. Poola *et al.* (2014) desenvolveram um algoritmo de escalonamento que mapeia as tarefas prontas às *spot instances* ou *on-demand* instances de acordo com o tempo resultante da diferença entre o *deadline* e o tempo do caminho crítico a partir da atual tarefa em execução. No trabalho de Yi, Kondo e Andrzejak (2010) é explorada uma estratégia para minimizar o custo usando *spot instances* e *checkpoints* para tratar eventuais falhas.

Ambientes híbridos

Em um ambiente de computação em nuvem híbrida, as nuvens atuam de modo complementar. Os recursos de uma nuvem pública podem crescer elasticamente e serem agregados a uma nuvem privada quando necessário. A proposta de Bittencourt e Madeira (2011) busca otimizar o custo de execução em nuvem híbrida (HCOC) que decide como os recursos de uma

nuvem pública serão agregados a nuvens privadas para executar um *workflow* com menor custo atendendo o seu *deadline*. Ostermann considerou o mesmo cenário de Bittencourt e Madeira (2011), mas usando *spot instances*.

Outro trabalho que otimiza o custo é o DCP-C que implementa um mecanismo de seleção de recursos que permite alugar *spot instances*. Ramakrishnan *et al.* (2009) projetaram um arcabouço que permite gerenciar recursos heterogêneos, escalonar e movimentar dados, tolerar falhas e ajustar o desempenho. Esse arcabouço é dividido em um sistema de execução virtual em grade para oferecer uma abstração dos recursos (vgES) e um escalonador de *workflows* que escalona as tarefas a recursos confiáveis providos pelo vgES.

Escalonamento de workflows com grande volume de dados

Um *workflow* é considerado uma aplicação paralela fracamente acoplada que consiste em um conjunto de tarefas ligadas via dependência de dados e fluxo de controle. Usualmente, as tarefas de *workflows* se comunicam por meio de arquivos. Considerando esse cenário, pesquisas foram desenvolvidas de modo a tratar questões relacionadas à movimentação de dados. Juve *et al.* (2010) examinaram o desempenho dos sistemas de armazenamento da Amazon S3, NFS, GlusterFS e PVFS. A conclusão dessa pesquisa é que a penalidade de escrita em discos é o fator que mais influencia no desempenho no EC2, sugerindo que a escolha de um sistema de armazenamento tem impacto significante no tempo de execução de *workflow*.

Diversas pesquisas exploram o problema de escalonamento de *workflows* dirigidos a dados. Park e Humphrey (2008) focaram na melhoria do desempenho da transferência de dados. Bharathi e Chervenak (2009) desenvolveram uma técnica que define vários estágios para a transferência de dados, permitindo reduzir o tempo de execução significativamente. Ramakrishnan *et al.* (2007) avaliaram um método dinâmico que minimiza o uso de espaço por meio da remoção de arquivos que não são mais necessários durante a execução.

Escalonamento de workflows considerando energia

A energia é reconhecida como uma das restrições mais importantes no projeto de arquiteturas computacionais. Devido a este fato, propostas de escalonamento para reduzir o consumo energético vem sendo estudadas. Neste cenário, há duas soluções. A primeira é reduzir o consumo de energia melhorando a utilização dos recursos (BELOGLAZOV; ABAWAJY; BUYYA, 2012). Zhu, Zhu e Agrawal (2010) observaram que um *workflow* aloca diferentes recursos e ainda o uso destes pode variar no decorrer da execução. Assim, a abordagem desenvolvida escalona *workflows* em recursos heterogêneos, oferecendo uma boa relação entre eficiência energética e o *makespan*. A segunda solução é variar a voltagem dinamicamente (DVS) (WU; CHANG; CHAN, 2014). O método mais objetivo é gerar um escalonamento usando um algoritmo *best-effort* com foco em *makespan* e adotar o DVS nos tempos ociosos de uma tarefa. No mesmo caminho, Lee e Zomaya (2011) idealizaram uma função objetivo para balancear o *makespan* e o consumo de energia. Mezmaz *et al.* (2011) e Yassa *et al.* (2013) modelaram tal

problema como uma otimização multi-objetivo, resolvendo-a por meio de algoritmos genéticos e enxame de partículas, respectivamente. Xiao, Hu e Zhang (2013) investigaram o problema do consumo de energia em *workflows* com grande volume de dados intermediários para serem armazenados e recuperados. No trabalho de Mishra *et al.* (2003), *workflows* com restrição de *deadline* e de consumo de energia foram considerados. Neste trabalho foi proposto um esquema de gerenciamento de energia estático baseado no grau de paralelismo para um escalonamento já gerado.

Escalonamento baseado em restrições de segurança

A segurança é o principal aspecto impeditivo para que empresas utilizem serviços em nuvem (IDC, 2015). Tanto que a confidencialidade e auditabilidade dos dados são citadas como dois dos dez obstáculos para adoção da computação em nuvem. De acordo com o relatório emitido pela Gartner, até 2018 a prevenção de vazamento de dados em nuvens públicas levará 20% das organizações a desenvolver programas de segurança de dados (DUNCAN; BUYTENDIJK, 2015). Parte dessa preocupação surge do fato de que recursos computacionais em nuvens públicas podem ser uma grande superfície para ataques maliciosos. Nesse sentido, a execução de *workflows* utilizando nuvens públicas de modo seguro é um cenário a ser tratado por pesquisas.

Em Watson (2011) são introduzidas transformações para quando há comunicação entre nuvens federadas. Esse trabalho descreve um método para determinar automaticamente as opções válidas de particionamento de tarefas e dados de um *workflow* conforme os requisitos de segurança. Quando há mais de um particionamento válido, este método escolhe a melhor opção considerando um modelo de custo.

Marcon *et al.* (2013) apresentam um método de como melhorar o uso de recursos locais para atender os requisitos de desempenho e segurança, e minimizar os custo da execução de *workflows*. Nessa proposta, o escalonador decide onde cada tarefa de ser executada (nuvem pública ou privada) com o objetivo de minimizar o custo dos usuários, atendendo o *deadline* do *workflow*. Depois de determinar os recursos computacionais e de comunicação, o escalonador considera os critérios de desempenho e segurança. A principal contribuição desse estudo é conciliar um escalonamento eficiente em nuvens híbridas de modo a atender totalmente os requisitos de segurança.

Sharif *et al.* (2013) desenvolveram um escalonador de *workflows* multi-critério em um ambiente de nuvens híbridas para minimizar o custo diante das restrições de privacidade e *deadline*, denominado MPHC. O escalonador é baseado no algoritmo *multiterminal cut* que particiona o *workflow* respeitando diferentes níveis de privacidade. As contribuições deste trabalho são (a) preservar a privacidade no escalonamento de *workflows* sob os requisitos definidos em SLA; (b) melhoria do modelo de custo para nuvens híbridas, e; (c) avaliação do algoritmo proposto usando um modelo de custo para vários padrões experimentais de *workflows*. Os resultados mostraram que o MPHC superou os algoritmos IC-PCP e o HT Condor reduzindo o custo da execução em 7 a 31%, mas também executou todas as tarefas antes de seus *deadline*.

Posteriormente, em Sharif *et al.* (2014), é proposto um algoritmo de escalonamento de múltiplos workflows em nuvens híbridas que considera o prazo de execução e atende as restrições de segurança. O algoritmo se baseia em uma heurística que busca executar os workflows em nuvem privada, porém, caso haja o descumprimento do deadline, tarefas que não manipulam dados sensíveis são executadas em nuvem pública.

Liu *et al.* (2012) apresenta um algoritmo para lidar com escalonamento de workflow com restrições de segurança. É utilizado um modelo de segurança que é otimizado usando enxame de partículas com vizinha variável com o objetivo de obter um bom balanço entre a exploração global e local de soluções. No estudo de Zhu *et al.* (2016) e Chen *et al.* (2017), as vantagens de inicializar uma tarefa o mais cedo possível com o intuito de minimizar o makespan foram investigadas. Com base nesta análise, os autores propuseram um algoritmo de escalonamento que utiliza as lacunas de tempo ocioso das máquinas virtuais para duplicar tarefas predecessoras seletivamente de modo a reduzir o makespan, bem como aumentar a utilização dos recursos.

No trabalho de Jianfang, Junjie e Qingshan (2014) foi proposto um algoritmo de escalonamento para nuvem baseado na otimização por enxame de partículas (CWDPSO) considerando segurança, *deadline* e custo. Os resultados mostraram que comparado com o algoritmo LGPPSO (GUO *et al.*, 2012), o CWDPSO obteve um desempenho melhor em relação ao *makespan*, e o custo.

A movimentação de dados entre diferentes nuvens é uma tarefa que merece atenção, devido à possibilidade da interceptação dos dados durante sua transmissão. Além disso, a transmissão de dados entre nuvens pode elevar o custo da execução de *workflows*. Portanto, é razoável definir a localidade de dados sensíveis. Contudo, em casos em que essa opção não é possível, é preciso considerar métodos de alocação de dados que reduzem o custo de transmissão inter-nuvem. A partir desse cenário, Zeng, Veeravalli e Li (2015) propuseram um algoritmo de escalonamento para minimizar o custo e atender as restrições de segurança e orçamento da aplicação, denominado SABA. Para isso introduziram o conceito de conjunto de dados imóveis que agrupa tarefas que utilizam os mesmos dados, reduzindo o custo de transferência entre datacenter. Os resultados revelaram uma melhora considerável sob várias situações especialmente em *workflows* de comunicação intensiva com alto grau de paralelismo. Além disso, o algoritmo foi capaz de lidar com diferentes conjuntos de dados atendendo os requisitos de segurança e orçamento, além de melhorar a utilização dos recursos.

Li *et al.* (2016) propuseram um algoritmo de escalonamento que considera a segurança e o custo utilizando otimização por enxame de partículas (SCAS). O objetivo deste estudo é minimizar o custo da execução total do workflow atendendo as restrições de risco e *deadline*. Para isso, desenvolveram uma estratégia para resolver o problema de escalonamento utilizando otimização multi-dimensional com duas restrições. Experimentos realizados com os *workflows* LIGO, SIPHT e CyberShake mostraram que esse método é superior a outras abordagem como o VM_Hom, Max_Level e Min_Level. Outras duas propostas também utilizam o modelo proposto

por Li *et al.* (2016). Em Arunarani, Manjula e Vijayan (2017), a otimização do escalonamento é realizada por meio da combinação das meta-heurísticas de vagalume e morcegos, enquanto em Shishido *et al.* (2017) utiliza algoritmo genético multipopulacional (MPGA).

Naidu e Bhagat (2018) propõem um algoritmo modificado de enxame de partículas com um observador adaptado (MPSO-SA) e um operador de mutação cíclico para minimizar o custo de execução e atender restrições de segurança. Em Wen *et al.* (2018), é modelado um problema de escalonamento de workflow considerando a privacidade de dados e a otimização de múltiplos objetivos (makespan e custo). Neste estudo, as tarefas que manipulam dados sensíveis são escalonadas para *datacenters* considerados confiáveis, enquanto as não sensíveis são escalonadas em qualquer outro datacenter de modo a minimizar o custo e o makespan. Neste trabalho foi aplicado um algoritmo multi-objetivo baseado em algoritmos genéticos para minimizar tanto o makespan quanto o custo de execução do workflow de modo a atender os requisitos de segurança.

As características de cada algoritmo como o objetivo de otimização, os modelos de nuvem, cobrança, *overhead*, execução, os aspectos de segurança, a heurística/meta-heurística utilizada e o modelo de *datacenter* estão classificados na Tabela 6 (página seguinte).

Tabela 6 – Comparativo entre os algoritmos de escalonamento de *workflows* que consideram segurança. r-restrição, c-critério, M-Meta-heurística, H-Heurística, E-Estático, D-Dinâmico.

	Critério					Nuvem		Segurança		Workflow			
Autor	**Makespan**	**Deadline**	**Custo**	**Orçamento**	**Segurança**	**Pública/privada**	**Híbrida**	**Serviços de segurança**	**Localidade**	**Único**	**Múltiplos**	**Otimização**	**Escalonamento**
(LIU *et al.*, 2012)	•				• (r)	•		•		•		M	E
(WATSON, 2011)	•		•		• (r)		•		•	•		H	D
(MARCON *et al.*, 2013)	•				• (r)		•		•	•		H	D
(SHARIF *et al.*, 2013)		•		•	• (c)		•		•		•	H	D
(SHARIF *et al.*, 2014)		•		•	• (c)		•		•		•	H	D
(JIANFANG; JUNJIE; QINGSHAN, 2014)	•		•		• (r)		•	•		•		M	E
(ZENG; VEERAVALLI; LI, 2015)			•		• (r)		•		•	•		H	D
(CHEN *et al.*, 2015)	•		•		• (r)		•		•	•		M	E
(LI *et al.*, 2016)		•	•		• (r)	•		•		•		M	E
(ZHU *et al.*, 2016)	•		•		• (r)	•		•		•		M	E
(NAIDU; BHAGAT, 2018)			•		• (r)	•		•	•	•		M	E
(WEN *et al.*, 2018)	•		•		• (r)		•		•	•		M	E
(ARUNARANI; MANJULA; VIJAYAN, 2017)		•	•		• (r)	•		•		•		M	E
(CHEN *et al.*, 2017)	•		•		• (r)		•	•		•		H	D
(SHISHIDO *et al.*, 2017)		•	•		• (r)	•		•		•		M	E

Fonte: Dados da pesquisa.

3.4 Considerações finais

Este capítulo apresentou os trabalhos relacionados e serve como base para o direcionamento da proposição do projeto de qualificação. A maioria dos algoritmos de escalonamento de *workflows* em nuvem focam métricas como o tempo de execução e o custo monetário. Porém,

o escalonamento pode ser estendido a outras métricas como a segurança, a confiabilidade e o consumo de energia as quais são alvo de preocupação em pesquisas de computação em nuvem. Conforme este levantamento bibliográfico, as pesquisas que tratam a segurança e o consumo energético estão em evidência e ascensão. Essas métricas podem ser incluídas como restrições de execuções que consideram a minimização do tempo ou do custo.

CAPÍTULO

4

MODELOS DO SISTEMA E FORMULAÇÃO DO PROBLEMA

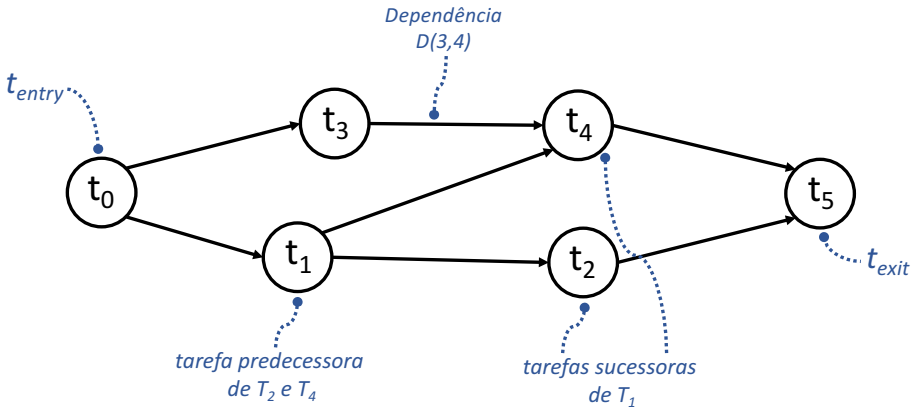
4.1 Considerações iniciais

A implementação de uma nuvem privada pode ser inviável devido aos recursos necessários para aquisição de equipamentos e softwares, além da necessidade de uma equipe qualificada para sua manutenção. A escalabilidade, o rápido provisionamento de recursos e o preço razoável de nuvens públicas são atrativos para a execução de workflows. Porém, o modelo de multi-locação de recursos a diferentes usuários pode produzir brechas de segurança quando há o envolvimento de dados sensíveis. Nesse contexto, o problema desta tese consiste em escalonar as tarefas de workflow a fim de minimizar o custo de execução sob um *deadline* e oferecer segurança a dados sensíveis. Neste capítulo são descritos os modelos de workflow, *cloud datacenter*, overhead de segurança, análise de risco, custo e tempo de execução. Ao final deste, a formulação do problema é apresentada.

4.2 Modelo de workflow

Um workflow pode ser formalmente representado como um grafo acíclico direcionado DAG(T,D), onde $T = \{t_0, t_1, ..., t_i, ..., t_{n-1}\}$ é o conjunto de n tarefas, e; D é o conjunto de arestas entre tarefas para representar suas dependências. Por exemplo, uma aresta $D(i, j)$ representa que uma tarefa t_i deve ser completada antes que t_j seja iniciada. A tarefa t_i é predecessora imediata da tarefa t_j. Uma tarefa que não possui tarefas predecessoras é denominada de tarefa de entrada (t_{entry}), e uma tarefa que não possui tarefas sucessoras é denominada de tarefa de saída (t_{exit}). A Figura 20 apresenta um exemplo de workflow com seis tarefas e suas dependências.

Figura 20 – Exemplo de um workflow.



Fonte: Elaborada pelo autor.

4.3 Modelo de cloud datacenter

Uma série de recursos computacionais (ex.: máquinas virtuais) $CR = \{cr_1, cr_2, ..., cr_m, ..., cr_k\}$ foi considerada, onde m representa uma posição da série CR. Cada série contém um conjunto de máquinas virtuais $cr_m = \{cr_m^1, cr_m^2, ..., cr_m^l, ..., cr_m^h\}$ de diferentes capacidades, onde l é uma máquina virtual, e h é o número total de VMs em uma série cr_m. Cada VM cr_m^l apresenta diferentes características em termos de custo por hora c_m^l, capacidade de processamento p_m^l, memória m_m^l e armazenamento st_m^l. Uma máquina virtual com características apropriadas de acordo com a carga de trabalho de cada tarefa precisa ser selecionada para fins de otimização de tempo de execução e custo. Este modelo assume que todas as máquinas virtuais estão conectadas por uma mesma largura de banda NB.

4.4 Modelo de overhead de segurança

A segurança é uma das principais preocupações para a adoção de nuvens. Entre as diversas categorias de ataques, há três que são comumente praticadas: (1) *spoofing*, (2) *snooping* e (3) alteração de dados.

O ataque baseado em *spoofing* refere-se a enganar sistemas computacionais por meio da falsificação da identidade de uma pessoa ou um sistema. O método mais comum de falsificação de identidade é o e-mail que envolve o envio de mensagens de um endereço de e-mail falso. A maioria dos servidores de e-mail possui recursos de segurança que impedem que usuários não autorizados enviem mensagens. Outra forma de falsificação é a de endereço IP que dificulta a identificação de qual computadores está transmitindo os dados.

O ataque de *snooping* consiste no acesso não-autorizado aos dados de outra pessoa ou companhia. Isso inclui a observação casual de e-mail que aparece na tela de outro computador ou até mesmo ao observar o que outra pessoa está digitando. No contexto de workflows

executados em nuvens, ataques de *snooping* visam o monitoramento das atividades de uma máquina virtual ou até mesmo no canal de comunicação entre VMs. Esse ataque pode levar a perda de privacidade de vários informações que devem ser estritamente privadas a um conjunto de usuários/computadores como: (a) senhas; (b) dados privados, e; (c) informações financeiras.

O ataque de alteração é o ato de modificar dados (destruição, manipulação ou edição) deliberadamente através de meios não-autorizados. Os dados podem ser classificados em estado de trânsito ou repouso. Em ambos os casos, os dados podem ser interceptados e adulterados. Por exemplo, nos casos em que dados são transmitidos sem qualquer proteção, um indivíduo mal intencionado pode interceptar os pacotes e modificar o seu conteúdo e/ou alterar seu endereço de destino. Considerando os dados em repouso, uma aplicação pode sofrer uma violação de segurança em que um invasor implanta um código mal intencionado que corrompa os dados ou o código da aplicação. Em ambos os casos, a invasão é maliciosa e os efeitos podem ser extremamente danosos às entidades envolvidas.

Esta tese considera o uso de três serviços de segurança (autenticação, verificação de integridade e criptografia) para proteger os dados de um workflow contra ataques maliciosos. O conjunto de serviços de segurança de uma tarefa t_i é formado por ss_i que pode ser especificado como uma q-tupla $ss_i = \{ss_i^1, ss_i^2, ..., ss_i^l, ..., ss_i^q\}$, onde ss_i^l representa o enésimo serviço de segurança. No modelo adotado nesta tese, considerou-se três serviços de segurança ($q = 3$). Os termos α, β, γ foram definidos para simplificar a representação dos serviços de autenticação, verificação de integridade e confidencialidade, respectivamente. É assumido que cada tarefa pode ou não usar concomitantemente os três serviços de segurança de acordo a sensibilidade dos dados. Por exemplo, assumindo que a tarefa t_i manipule dados sensíveis, é esperado que os três serviços de segurança sejam aplicados; caso contrário, não há necessidade de proteger dados não sensíveis, evitando o *overhead* produzido pelos serviços de segurança.

Serviços de segurança produzem *overhead* adicional em sistemas computacionais. A Tabela 7 apresenta o *overhead* produzido por diferentes algoritmos de segurança. Serviços de verificação de integridade e de confidencialidade produzem *overhead* de acordo com a quantidade de dados a ser protegida e a complexidade do algoritmo adotado. Portanto, o *overhead* SO^l desses dois serviços é expressado por

$$SO^l(t_i) = F^l(ss_i^l, a_i^l), \qquad l \in \{\beta, \gamma\} \tag{4.1}$$

onde F^l é uma função para calcular o *overhead* para proteger uma quantidade de dados a_i^l da tarefa t_i usando um serviço de segurança ss_i^l.

O *overhead* do serviço de autenticação é um valor constante que depende majoritariamente da complexidade do algoritmo de autenticação e é computado por

$$SO^l(t_i) = F^l(ss_i^l), \qquad l \in \{\alpha\} \tag{4.2}$$

O *overhead* geral $SO(t_i)$ introduzido em uma tarefa t_i pode ser calculado por:

$$SO(t_i) = \sum_{l \in \{\alpha,\beta,\gamma\}} SO^l(t_i) \tag{4.3}$$

Tabela 7 – Algoritmos de segurança.

Serviço de segurança	***Overhead***
Authentication (CBC_MAC_AES)	163 ms
Integrity (TIGER)	4.36 kB/ms
Confidentiality (IDEA)	13.50 kB/ms

Fonte: Adaptada de Xie e Qin (2006a).

4.5 Modelo de execução de tarefa

A execução segura de uma tarefa requer o uso de serviços de segurança implantados em instâncias de VMs. Assumindo que a tarefa t_i é a sucessora imediata da tarefa t_{i-1}, o acesso a um conjunto de dados somente pode ser realizada por usuários autenticados para prevenir ataques de *spoofing*. Então, os dados de saída da tarefa t_{i-1} podem ser transferidos para a tarefa t_i. Todos os dados de entrada da tarefa t_i são representados por ID_i. O tempo gasto com a transferência para a tarefa t_i é calculado pela Eq. 4.4. Caso a transferência de dados ocorra entre tarefas alocadas em uma mesma máquina virtual, nenhum tempo é computado.

$$CT(t_i) = \begin{cases} ID_i/NB, & se\ vm_{t_{(i-1)}} \neq vm_{t_{(i)}} \\ 0, & caso\ contrario \end{cases} \tag{4.4}$$

O tempo de execução da tarefa t_i em uma VM cr_k^l é dado pela Eq. 4.5, onde L_i é a carga de trabalho da tarefa t_i, e p_k^l é a capacidade de processamento de cr_k^l. Uma vez que o serviço de verificação de integridade garantiu que os dados não sofreram modificação, o serviço de confidencialidade é aplicado nos dados de saída para conter ataques de *snooping*. Esse serviço evita que indivíduos não-autorizados tenham livre acesso aos dados durante a sua transmissão ou repouso.

$$RT(t_i, cr_k^l) = L_i/p_k^l \tag{4.5}$$

Finalmente, o tempo de processamento total P_i de uma tarefa t_i é calculado pela soma do tempo de transferência de dados $CT(t_i)$, tempo de processamento da carga de trabalho da tarefa $RT(t_i, cr_k^l)$ e o *overhead* de segurança $SO(t_i)$:

$$P(t_i, cr_k^l) = CT(t_i) + RT(t_i, cr_k^l) + SO(t_i) \tag{4.6}$$

4.6 Modelo de análise de risco e desperdício de segurança

No quesito segurança, os dados podem ser classificados em sensíveis ou não-sensíveis. A primeira classificação considera a manipulação de dados que podem ser acessados somente por usuários com permissão e não podem se tornar públicos, enquanto a segunda não produz qualquer dano a uma pessoa ou instituição caso seja acessada por terceiros. Os dados sensíveis de aplicações baseadas em workflows não são livre de riscos quando executadas em nuvens. Portanto, é essencial que workflows sejam modelados por usuários que conheçam quais tarefas manipulam dados sensíveis ou não para que métodos de segurança sejam aplicados adequadamente (BALASKO; FARKAS; KACSUK, 2013). Além disso, avaliar a segurança oferecida na execução do workflow é essencial. Para isso, duas métricas para medir a exposição de dados sensíveis a ataques maliciosos foram propostas. O risco $Risk_{(w)}$ representa o risco geral de vazamento de dados sensíveis do workflow e é calculado por

$$Risk_{(w)} = \begin{cases} 1 - \left(\frac{HT}{ST_{(w)}}\right) & if\ ST_{(w)} > 0 \\ 0 & otherwise \end{cases} \tag{4.7}$$

onde $ST_{(w)}$ é o número de tarefas sensíveis de um workflow, e HT representa o número de sensíveis que receberam proteção de serviços de segurança.

Algumas abordagens para a execução segura de workflows aplicam serviços de segurança em tarefas não-sensíveis. Isso acarreta no uso ineficiente de recursos, encarece o custo e prolonga o tempo de execução do workflow. Nesse sentido, é interessante avaliar a alocação adequada de serviços de segurança em tarefas exclusivamente sensíveis. A segunda métrica $NSTS_{(w)}$ consiste em representar o percentual de tarefas não-sensíveis que foram asseguradas, a qual é computada pela Equação 4.8.

$$NSTS_{(w)} = 1 - \left(\frac{ST_{(w)} - HT}{ST_{(w)}}\right) \tag{4.8}$$

4.7 Formulação do problema

A abordagem desta tese é otimizar a busca de escalonamento para a execução de um workflow em nuvem pública, minimizando o custo e o risco de acesso não-autorizado a dados sensíveis. Além disso, o escalonamento deve considerar a restrição de *deadline*. O escalonamento foi definido como:

$$schedule = \{LR, SO, RM, Risk_{(w)}, E^c, E^t\} \tag{4.9}$$

que considera o conjunto de VMs a serem alugadas LR, *overhead* de segurança SO, mapeamento VM-tarefa RM, risco de segurança da execução $Risk_{(w)}$, custo de execução E^c e o makespan E^t. Os recursos alugados para cada tarefa é o conjunto $LR = \{lr_0, lr_1, ..., lr_i, lr_{n-1}\}$, onde lr_i contém três tuplas $lr_i = (cr_k^l(t_i), LST(t_i, cr_k^l), LET(t_i, cr_k^l))$. O tipo de VM alugada para executar uma tarefa t_i é representada por $cr_k^l(t_i)$ que possui o tempo do início da locação $LST(t_i, cr_k^l)$ e o tempo de término da locação $LET(t_i, cr_k^l)$. O conjunto de tarefas asseguradas é definido em $SO = \{so_i | i = 0, 1, ..., n-1\}$. RM é o mapeamento de recursos-tarefas composto por $rm(t_i, cr_k^l) = (t_i, cr_k^l, TS(t_i), TE(t_i))$, onde t_i é a tarefa a ser escalonada para uma VM cr_k^l, $TS(t_i)$ é o tempo de início da tarefa t_i, e $TE(t_i)$ é o tempo de término da tarefa. O tempo de transferência de dados é considerado de modo que o tempo $LST(t_i, cr_k^l)$ inclui o tempo de transferência de dados de entrada da(s) tarefa(s) pai(s). $Risk_{(w)}$ é o risco da execução do workflow que é computado pela Eq. 4.7. O custo total de execução workflow E^c e o tempo total de execução E^t são calculados pelas Equações 4.11 e 4.10, respectivamente.

$$E^c = \sum_{i=0}^{n-1} c_k^l \cdot \lceil LET(t_i, cr_k^l) - LST(t_i, cr_k^l) \rceil \tag{4.10}$$

$$E^t = max\{P(t_i) | t_i \in T\} \tag{4.11}$$

O termo $c_k^l \cdot \lceil LET(t_i, cr_k^l) - LST(t_i, cr_k^l) \rceil$ refere-se ao custo para executar a tarefa t_i. Para se obter o custo total é realizado o somatório do limite superior do tempo de uso da VM por cada tarefa t_i devido ao modelo de cobrança por hora adotado neste estudo. Basendo-se nas definições dos modelos, o problema desta tese pode ser definido como: encontrar escalonamento que minimize o custo de execução considerando o uso de serviços de segurança, de modo que o *deadline* DL seja respeitado. Formalmente, descrito como: *Minimizar*(E^c e $Risk_{(w)}$), sujeito a: $E^t \leq DL$.

4.8 Considerações finais

Esse capítulo apresentou os modelos de workflow, *cloud datacenter*, overhead de segurança, análise de risco, custo e tempo de execução. Esses modelos são considerados no decorrer das simulações desta tese. A formulação do problema de escalonamento também foi definida e considerada nos experimentos ao longo da tese. O próximo capítulo apresenta uma extensão para o simulador WorkflowSim desenvolvida para simular a execução de workflows considerando o uso de serviços de segurança fazendo uso dos modelos propostos nesse capítulo.

CAPÍTULO

5

SIMULADOR PARA AVALIAÇÃO DE ALGORITMOS DE ESCALONAMENTO

5.1 Considerações iniciais

Como já apresentado no Capítulo 3 o escalonamento de *workflows* em nuvens é vastamente estudado considerando diferentes critérios como makespan, custo, consumo de energia e confiabilidade (RODRIGUEZ; BUYYA, 2016). A segurança é um critério que tem sido investigado recentemente em algoritmos de escalonamento de *workflows* quando executados em nuvem (SHISHIDO; ESTRELLA, 2017). A avaliação de algoritmos de escalonamento pode ser realizada por meio de simulações, permitindo que usuários avaliem a sua corretude e eficiência antes de implantá-lo em ambiente real. Outro benefício oferecido por simuladores é o estudo de um problema em diferentes níveis de abstração sem a necessidade de pagar pela locação de uma nuvem real. Contudo, os simuladores de nuvem não contemplam a sobrecarga produzida por serviços de segurança no processo de execução de workflow. Nesse sentido, este capítulo apresenta a proposta e o desenvolvimento de uma extensão do simulador WorkflowSim para considerar a sobrecarga de segurança na execução de *workflows* em nuvens.

5.2 Segurança em execução de *workflows* em nuvens

Esta tese baseia-se no uso de serviços de segurança em tarefas sensíveis como medida de proteção contra ataques maliciosos durante a execução de *workflows* em nuvem. A segurança é aqui definida como uma medida quantitativa do nível de proteção atribuído às tarefas que manipulam dados sensíveis, seja aplicando serviços de segurança em tarefas sensíveis ou executando tarefas sensíveis em nuvem privada (confiável) e as demais em nuvem pública. Uma tarefa sensível é definida como uma tarefa que manipula dados que não podem se tornar públicos e deveriam ser abertos somente sob circunstâncias limitadas. Além disso, é preciso de

autorização para acessá-los, pois, a abertura, alteração ou destruição de tais dados pode causar danos perceptíveis a uma organização.

Há várias formas de proteger dados sensíveis. Este estudo considera a adoção de serviços de segurança como autenticação, verificação de integridade e criptografia. Serviços de autenticação são aplicados para evitar a abertura ou o acesso não-autorizado quando uma tarefa solicita que os dados de outra sejam enviados a ela. Este é o processo de confirmação da identificação de uma entidade (usuário, sistemas, máquinas, etc.) que está requisitando acesso a outros recursos. A autenticação é frequentemente confundida com o termo autorização. Enquanto a autenticação é o processo que certifica uma identidade, a autorização verifica se a entidade tem permissões suficientes para acessar o recurso requisitado. Tecnicamente, a autenticação é um passo que antecede a autorização (JOSANG, 2017).

Nuvens públicas são caracterizadas pela multi-locação de modo que diferentes usuários possam ser alocados fisicamente dentro de um mesmo recurso computacional, porém virtualmente separados (RONG; NGUYEN; JAATUN, 2013). Esse cenário cria condições para ataques de adulteração em que usuários maliciosos poderiam modificar os dados de um outro usuário explorando problemas na rede, por exemplo. O uso de serviços de verificação de integridade é recomendado para garantir a consistência e precisão dos dados transmitidos. Na execução de *workflows*, todo serviço de segurança tem o mesmo objetivo que é certificar que os dados enviados de uma tarefa sejam recebidos pela outra exatamente igual como foram enviados.

O processamento de uma tarefa requer que os dados estejam em um formado legível pela aplicação. Entretanto, a transmissão de dados sensíveis de uma tarefa a outra não pode ser realizado deixando os dados em formato legível para indivíduos não-autorizados. Portanto, criptografar os dados em ambientes multi-locados pode reduzir o risco de sua abertura quando interceptados indevidamente. Dados criptografados também são chamados de texto-cifrado, enquanto os não criptografados são denominados texto-simples. Algoritmos de criptografia geralmente transformam texto-simples em forma cifrada de modo que somente usuários com acesso à chave ou senha podem transformá-los em texto-simples novamente.

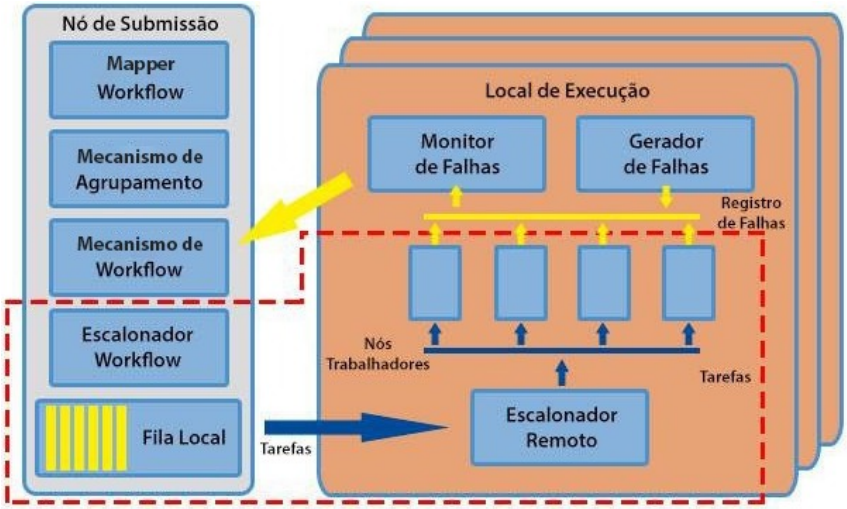
Há dois tipos de algoritmos de criptografia: assimétrica e simétrica. Algoritmos de chave simétrica usam a mesma chave para criptografar e descriptografar dados. Esse método é mais rápido do que a abordagem assimétrica, porém o emissor precisa trocar a chave com o destinatário antes do processo de descriptografia. Por outro lado, a criptografia assimétrica, também chamada de criptografia de chave pública, usa duas chaves (uma pública e outra privada). A chave pública pode ser compartilhada com qualquer um, enquanto a privada precisa ser protegida. Qualquer um pode criptografar uma mensagem usando a chave pública do destinatário. Porém, a mensagem em questão só pode ser descriptografada pela chave privada do destinatário. Essa característica da criptografia assimétrica oferece também uma funcionalidade de autenticação (ZAIDAN *et al.*, 2010).

5.3 Estudo de caso: WorkflowSim

A avaliação de desempenho de algoritmos de escalonamento de *workflows* em infraestruturas reais é complexa e pode levar muito tempo. Como consequência, experimentos baseados em simulações têm se consolidado como uma forma aceitável de avaliar sistemas de workflows. As simulações podem reduzir a complexidade da configuração experimental, salvar esforços na execução de *workflows* e oferecer um ambiente controlado e reprodutível. Outra vantagem dos simuladores é a capacidade de avaliar algoritmos de escalonamento em nuvens públicas sem a necessidade de realmente gastar recursos financeiros para alugar máquinas virtuais. Em contrapartida, as simulações podem esbarrar em limitações como o processamento computacionalmente intensivo exigido por alguns simuladores, e apresentar resultados imprecisos devido ao uso de modelos que não descrevem adequadamente o sistema real (FRENZ, 2007).

As ferramentas de simulação usadas para a execução de *workflows* tais como WorkflowSim (CHEN; DEELMAN, 2012), DynamicCloudSim (BUX; LESER, 2015) e SimpleWorkflow são geralmente baseadas no CloudSim (CALHEIROS. *et al.*, 2011). O WorkflowSim é uma ferramenta desenvolvida na Universidade do Sul da California que estende o CloudSim por meio da adição de uma camada de gerenciamento de workflow. O WorkflowSim segue o modelo de execução central do CloudSim que itera sobre uma lista de eventos futuros. Entretanto, o WorkflowSim extrai o modelo de execução de *workflows* do SGWf Pegasus (Figura 21), permitindo a avaliação de algoritmos de escalonamento. Basicamente, o WorkflowSim considera três componentes básicos no processo de simulação: *Workflow Mapper*, *Workflow Engine* e *Workflow Scheduler*.

Figura 21 – Visão geral do WorkflowSim. A área cercada por linhas tracejadas é suportada pelo CloudSim.



Fonte: Adaptada de Chen e Deelman (2012).

Um *workflow* pode ser representado por meio de um grafo acíclico direcionado, onde cada nó representa uma tarefa a ser computada e as arestas representam os fluxos de dados entre tarefas. O componente *Workflow Mapper* é responsável pela importação de arquivos de descrição

de *workflows* em formato XML que contém o tamanho da tarefa, dependências, tamanho dos arquivos de entrada e saída, e outros metadados. Depois de carregar o workflow, o *Workflow Mapper* constrói a lista de tarefas e associa essas tarefas a um recurso ocioso em tempo de execução (CHEN; DEELMAN, 2012).

O *Workflow Engine* é responsável por controlar a execução das tarefas de acordo com suas dependências para garantir que uma tarefa seja executada somente se a(s) sua(s) tarefa(s) pai(s) tenham sido concluídas com sucesso. O *Workflow Engine* trabalha em conjunto com o *Workflow Scheduler*, liberando somente as tarefas em estado de pronto para o escalonador (CHEN; DEELMAN, 2012).

Conforme as tarefas são marcadas em estado de pronto pelo *Workflow Engine*, o *Workflow Scheduler* associa tarefas aos recursos computacionais (ex.: VMs) baseado no algoritmo de escalonamento selecionado. Em algoritmos de escalonamento estático, tarefas são atribuídas a um recurso computacional no estágio de planejamento. No escalonador remoto, as tarefas aguardam até que o recurso seja liberado. Para algoritmos de escalonamento dinâmico, tarefas são associadas a um recurso computacional ocioso (CHEN; DEELMAN, 2012).

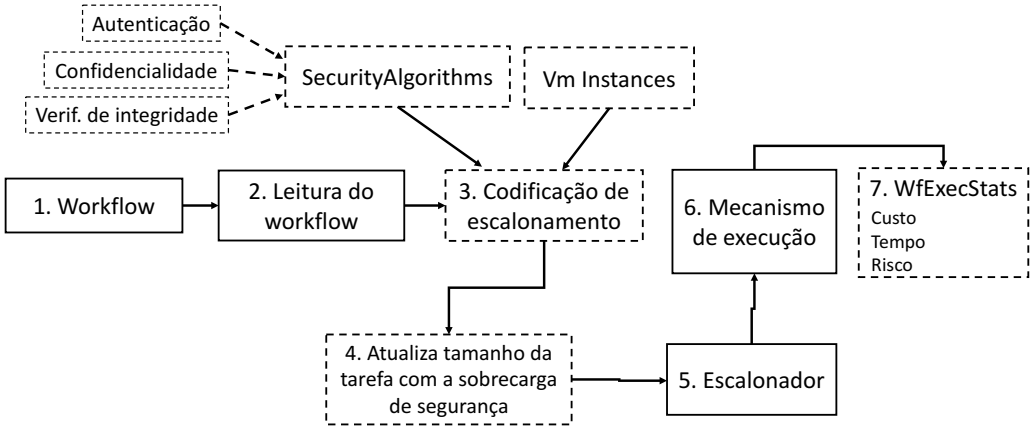
A execução de *workflows* pode levar um tempo razoável, principalmente quando o número e a complexidade das tarefas é alta ou quando há um grande volume de dados a ser processado. Possíveis falhas na execução de *workflows*, bem como o *overhead* de transferência de dados e inicialização de máquinas virtuais podem influenciar na execução de workflows. A partir dessas circunstâncias, diferentes camadas de *overhead* e falhas foram introduzidas no WorkflowSim para melhorar o modelo de simulação. O *overhead* produzido pela aplicação de serviços de segurança em *workflows* também precisa ser considerado, principalmente em simulações de *workflows* que envolvem grande volume de dados. Porém, este aspecto não é considerado em nenhum dos simuladores citados, sendo também parte do estudo desta tese.

5.4 Requisitos para simulação de segurança de workflows

Ao considerar o *overhead* de segurança na simulação de execução de *workflows*, é proposto um fluxo de execução do processo de execução de *workflow* em SGWf conforme representado pela Figura 22. São propostos sete passos para adição do *overhead* de segurança, em que o primeiro considera descrever um *workflow* apropriadamente em um arquivo. Os principais detalhes das tarefas consistem em sua identificação, quantidade de instruções e dependências. Os requisitos para simulação do *overhead* de segurança podem ser realizados através do acréscimo da quantidade de instruções de cada tarefa que irá ter seus dados protegidos por serviços de segurança.

Para identificar as tarefas sensíveis é proposta a adição de um atributo `secRequired` em cada descritor de tarefa `<job>` como mostrado na Arquivo 1. Neste estudo, o atributo `secRequired` é representado por valor booleano para representar se a tarefa requer ou não

Figura 22 – Fluxo da simulação de execução de *workflows* adicionando *overhead* de segurança.



Fonte: Elaborada pelo autor.

segurança. Contudo, a representação de segurança pode ser estendida a múltiplos atributos como autenticação, verificação de integridade e criptografia, em que cada um pode assumir diferentes níveis de proteção.

Arquivo 1 – Exemplo do uso do atributo secRequired na definição de tarefa sensível.

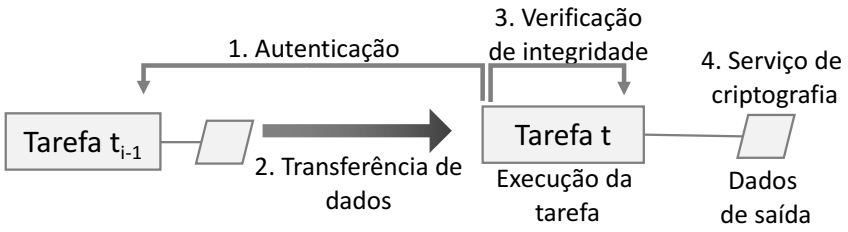
```
1  <job id="ID00001" namespace="CyberShake" name="ZipSeis" version="1.0"
2  runtime="0.19" secRequired="true">
3    <uses file="Cybershake_Seismograms.zip" link="output" register="true"
4    transfer="true" optional="false" type="data" size="43659"/>
5  </job>
```

Fonte: Elaborada pelo autor.

O componente *Workflow Parser* necessita carregar as tarefas do *workflow* incluindo os seus atributos de segurança para uma entidade `Task` com o objetivo de calcular o *overhead* de segurança. A sequência da aplicação dos serviços de segurança nos dados de uma tarefa é apresentado na Figura 23. Primeiro, a tarefa t_i precisa ser autenticada para a transferência dos dados de entrada a partir de sua(s) tarefa(s) predecessora(s). Assim que a transferência de todo os dados de entrada é concluída, ocorre a verificação de integridade dos dados. Se os dados não tiverem sido adulterados ou corrompidos, a execução das instruções da tarefa pode prosseguir. No final da execução da tarefa, os dados de saída são submetidos ao serviço de criptografia antes de serem enviados para a(s) tarefa(s) sucessora(s) de t_i.

A codificação do escalonamento é proposta para otimizar o escalonamento usando meta-heurísticas. Uma interface `Encoding` oferece um modo extensível para a implementar diferentes codificações de escalonamento. Se usada, a codificação substitui todos os atributos de segurança definidos no arquivo de descrição do workflow. Por exemplo, uma classe `Encoding` poderia representar o escalonamento de diferentes instâncias de VMs e os níveis de serviços de segurança para cada tarefa como mostrado na Figura 24. Esse exemplo mostra N tarefas onde cada uma é configurada para usar um tipo de instância de VM e diferentes níveis de serviço de autentiçacão,

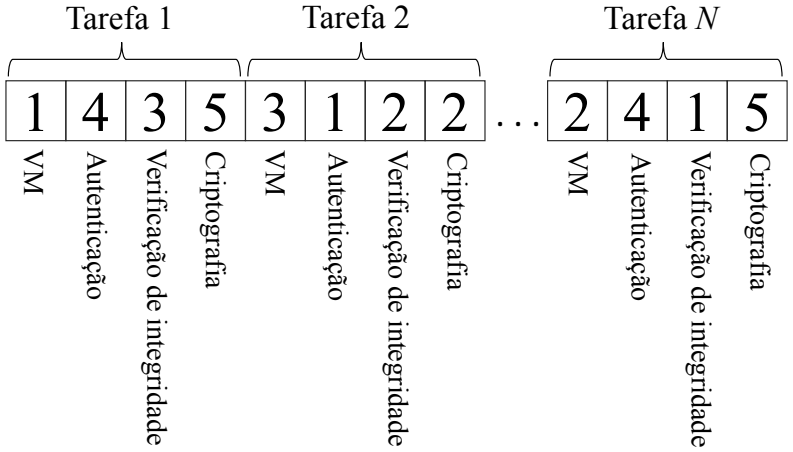
Figura 23 – Processo de aplicação dos serviços de segurança em uma tarefa.



Fonte: Elaborada pelo autor.

verificação de integridade e criptografia.

Figura 24 – Exemplo de codificação de escalonamento para uso de otimização baseada em meta-heurísticas.



Fonte: Elaborada pelo autor.

Há diversos algoritmos para cada serviço de segurança que oferecem diferentes níveis de proteção e *overhead*. Foi proposta uma entidade denominada `SecurityAlgorithms` contendo vários algoritmos para se adaptar aos requisitos de segurança de cada tarefa. Algoritmos com maior nível de proteção demandam mais tempo para processar a mesma quantidade de dados. Desta forma, a associação de diferentes níveis de segurança de acordo com os requisitos das tarefas pode produzir menor *overhead*, e consequentemente, reduzir o tempo e o custo de execução de um workflow.

O ambiente em nuvem geralmente oferece recursos computacionais heterogêneos, em que cada um é caracterizado por diferentes capacidades de CPU, memória, armazenamento, rede, custo, etc. A entidade `VmInstances` foi criada para o propósito de otimização uma vez que o tempo de execução dos algoritmos de segurança pode variar de acordo com a instância de VMs e a complexidade do algoritmo.

Depois que a codificação do escalonamento é definida, o tamanho das tarefas deve ser atualizado. Os simuladores de execução de *workflows* costumam utilizar a métrica de milhões de instruções (MI) para representar o tamanho de uma tarefa. O *overhead* de segurança foi

representado da mesma maneira de modo que pudesse ser somado ao tamanho de instruções da tarefa. Cada serivço de segurança é medido usando diferentes métricas. Por exemplo, a autenticação é medida em milisegundos (ms), enquanto a verificação de integridade e criptografia são baseados em kilobytes por segundo (kb/s). As Equações 5.1 e 5.2 apresentam a função para obter o *overhead* de segurança SL^{β} quando serviços de segurança são aplicados em uma tarefa t_i, onde β consiste no serviço de segurança tal que a, g, c representam autenticação, verificação de integridade e criptografia, respectivamente. Nota-se que o serviço de autenticação leva um tempo constante, enquanto a verificação de integridade e criptografia são dependentes do tamanho dos dados.

$$SL^{\beta}(t_i) = \frac{secAlg_L}{cpu_capacity}, \qquad \beta \in \{a\} \tag{5.1}$$

$$SL^{\beta}(t_i) = \frac{(data_size/secAlg_L)}{cpu_capacity}, \qquad \beta \in \{g, c\} \tag{5.2}$$

Na fase de escalonamento, cada tarefa é atribuída a um recurso computacional. Se o escalonamento for previamente definido no início da simulação através da entidade `Encoding`, o escalonador não é invocado durante a execução. Assim, todas as tarefas são submetidas ao *workflow engine* que controla o fluxo de execução das tarefas, liberando as tarefas em estado de pronto aos recursos e serviços de segurança definidos na codificação. Ao término da execução do workflow, o simulador apresenta ao usuário as métricas de custo, tempo e segurança.

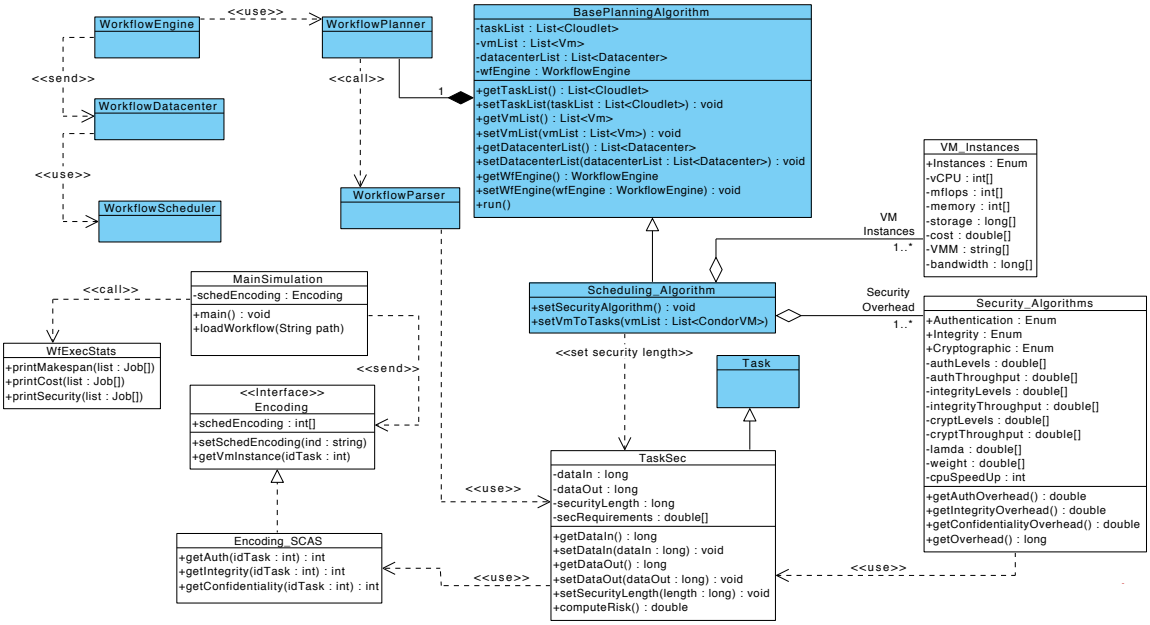
5.5 Extensão WfSecurity

A extensão proposta para a simulação de *workflows* atentando-se ao *overhead* de segurança baseou-se na ferramenta WorkflowSim. A Figura 25 retrata o diagrama de classe da extensão proposta. A nova classe `TaskSec` estende a classe original `Task` usando ainda três outras classes: `VM_Instances`, `Security_Algorithms` e `Encoding` que são necessárias para calcular o *overhead* de segurança.

A classe `TaskSec` pode receber os parâmetros de segurança pela leitura de um arquivo de *workflow* DAX ou definindo uma codificação na inicialização da simulação. O escalonamento é realizado através da implementação da interface `Encoding` que recebe o mapa de escalonamento representado pelo vetor `schedEncoding` como mostrado na Figura 24. O uso de um vetor para representar o escalonamento é útil para fins de otimização onde algoritmos de meta-heurísticas podem auxiliar no ajuste do tipo de instância de VM e parâmetros de segurança para cada tarefa.

A classe `TaskSec` tem o atributo `security_length` que representa a quantidade de instruções de segurança. Após calculado, o atributo `security_length` é incorporado ao atributo `length` herdado da classe `Task`. Desta forma, o *overhead* de segurança fica implícito nas instruções da tarefa. Os dados de entrada `dataIn` e de saída `dataOut` são submetidos aos serviços

Figura 25 – Diagrama de classes da extensão WfSecurity. As classes na cor azul representam o núcleo do WorkflowSim, enquanto as brancas representam a extensão do WfSec.



Fonte: Elaborada pelo autor.

de verificação de integridade e criptografia, respectivamente, como mostrado na Figura 23. Os requisitos de segurança também são definidos na classe `TaskSec` para o cálculo de cumprimento de QoS. Cada tarefa computa o risco de segurança usando a implementação do método `computeRisk()`. O cálculo baseia-se nos requisitos de segurança e serviços concedidos em cada tarefa do workflow. Essa função utiliza a classe `Security_Algorithms` para obtenção dos níveis de segurança dos algoritmos de autenticação, verificação de integridade e criptografia.

O *overhead* de segurança de cada tarefa é calculado através das Equações 5.1 e 5.2 e, em seguida, somado no atributo `length` da tarefa. O método `printSecurity()` é responsável por calcular o risco da execução do *workflow* baseando-se em um modelo definido pelo usuário.

5.6 Exemplo de uso

Esta extensão está disponível no GitHub. Inicialmente, a interface `Encoding` foi implementada seguindo a codificação apresentada na Figura 24. A classe `Security_Algorithms` foi definida pela adição de uma lista de algoritmos de segurança e os seus respectivos *throughput* e nível de segurança como mostrado na Tabela 8. Além disso, a classe `VM_Instances` foi implementada baseada em instâncias da Amazon EC2 e em seu modelo de preço.

Para fins de validação do simulador, uma métrica para avaliação do risco de sucesso de ataques maliciosos em uma tarefa t_i em um determinado critério de segurança foi considerada. O método `computeRisk()` da classe `TaskSec` foi implementado considerando três tipos de serviços

Tabela 8 – Lista de algoritmos de segurança adicionados à extensão WfSecurity.

Serviço de segurança	Algoritmo	Nível de segurança	*Overhead* (kB/ms)
Criptografia	SEAL	0.08	168.75
	RC4	0.14	96.43
	Blowfish	0.36	37.50
	Knufu/Khafre	0.40	33.75
	RC5	0.46	29.35
	Rijndael	0.64	21.09
	DES	0.90	15.00
	IDEA	1.00	13.50
Verificação de integridade	MD4	0.18	23.90
	MD5	0.26	17.09
	RIPEMD	0.36	12.00
	RIPEMD128	0.45	9.73
	SHA1	0.63	6.88
	RIPEMD160	0.77	5.69
	TIGER	1.00	4.36
Autenticação	HMAC_MD5	0.55	90 (ms)
	HMAC_SHA_1	0.91	148 (ms)
	CBC_MAC_AES	1.00	163 (ms)

Fonte: Elaborada pelo autor.

de segurança conforme apresentado na Eq. 5.3.

$$Risk(t_i, sl_i^l) = 1 - exp(-\lambda^l (sr_i^l - sr_i^l)), \quad l \in \{a, g, c\} \tag{5.3}$$

O coeficiente de risco λ^l pode variar de um serviço de segurança a outro. Por exemplo, uma VM pode sofrer três ataques de interceptação, dois ataques de adulteração e quatro ataques de identidade falsa em dado período de tempo. A probabilidade de uma falha de segurança é representada pelo expoente negativo e cresce conforme a diferença entre $sr_i^l - sl_i^l$, onde sr_i^l é o nível de segurança requerido para uma tarefa t_i a um serviço de segurança l, e sl_i^l representa o nível de segurança concedido a tarefa t_i para um serviço de segurança l. Assim, o risco geral de uma tarefa t_i é calculado pela Equação 5.4.

$$Risk(t_i) = 1 - \prod_{l \in \{a,g,c\}} (1 - Risk(t_i, sl_i^l)) \tag{5.4}$$

O cálculo do risco geral da execução de um *workflow* é dado pela Eq. 5.5.

$$Risk(T) = 1 - \prod_{t_i \in T} (1 - Risk(t_i)) \tag{5.5}$$

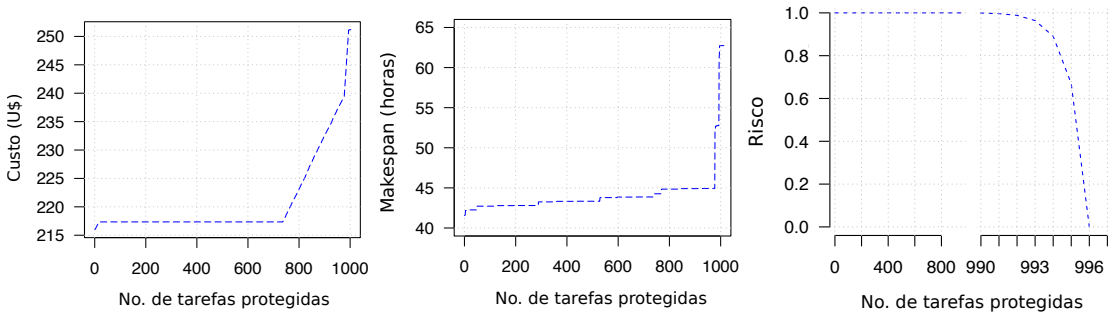
Os métodos de exibição das métricas de execução foram implementados na classe `WfExecStats` e são chamados ao término da simulação. Para fim de validação, foram executadas simulações usando o *workflow* Epigenomics com 997 tarefas com 1TB de dados por tarefa. A avaliação do *overhead* produzido pela aplicação de serviços de segurança foi realizada através da variação do número de tarefas protegidas (1 a 997 tarefas). O nível de segurança requisitado por cada tarefa foi definido em 1.0 (máximo). O objetivo desses experimentos foi validar a extensão do simulador proposta, então considerou-se apenas um tipo de instância de VM para a

execução do workflow. Contudo, a atribuição de diferentes tipos de VM poderia ser otimizada conforme a complexidade computacional de cada tarefa. Atribuir VMs mais rápidas em tarefas que extrapolam a hora/uso em poucos minutos pode reduzir o tempo e o custo de processamento.

A Figura 26 apresenta o risco de segurança presente na execução do workflow. Como pode ser visto, o risco é de aproximadamente 100% quando menos de 993 tarefas são asseguradas. O risco somente está abaixo dos 50% e próximo de 0% quando 995 e 997 tarefas são submetidas ao uso de serviços de segurança, respectivamente. Esse é um comportamento comum para a métrica adotada na avaliação desse simulador, onde qualquer falha pode resultar em perda significativa para um usuário ou organização. No entanto, outras métricas podem ser consideradas para a avaliação do risco de segurança na execução de workflows.

A Figura 26 apresenta a influência do *overhead* de segurança no tempo de execução do workflow. Como esperado, o tempo de execução cresceu conforme o número de tarefas submetidas a serviços de segurança aumentou. O tempo de execução foi maior quando as últimas tarefas do *workflow* foram asseguradas, pois manipulam maior quantidade de dados. Além do tempo, o custo total de execução também foi avaliado e é apresentado na Figura 26. Como o modelo de cobrança das VMs é baseado em horas de uso, pode ser notado que o custo de execução foi de aproximadamente U$217 para 720 tarefas asseguradas, e posteriormente cresceu significativamente até U$250 quando todas as tarefas foram asseguradas. Isso ocorreu devido ao aumento do tempo de execução pelos serviços de segurança e consequentemente resultou em uma nova hora de aluguel em algumas máquinas virtuais.

Figura 26 – Métricas de execução de workflow. (a) Custo monetário aplicando serviços de segurança nas tarefas do workflow; (b) Tempo de execução aplicando serviços de segurança nas tarefas do workflow; (c) Risco de aplicação de serviços de segurança nas tarefas do workflow.



Fonte: Elaborada pelo autor.

5.7 Considerações finais

Neste capítulo foi proposta uma extensão para simulação de *overhead* de segurança e análise de risco em execuções de *workflows* para fornecer uma estrutura extensível que permita adicionar parâmetros de serviços de segurança e tipos de VM para simulação de workflows. A adaptabilidade desta extensão permite a implementação de diferentes modelos de análise de

risco. Embora esta extensão trate o *overhead* de serviços de segurança, há outra abordagem para proteger a execução de *workflows* que é baseada na alocação de tarefas confidenciais em nuvens privadas e as não sensíveis em nuvens públicas. Portanto, mais pesquisas precisam ser realizadas para calcular o custo, o mapeamento e as implicações de segurança na alocação de tarefas para nuvens híbridas.

CAPÍTULO

6

ESTUDO EXPLORATÓRIO DE METAHEURÍSTICAS

6.1 Considerações iniciais

Diversos métodos de otimização foram introduzidos na literatura, os quais podem ser classificados como métodos do tipo exato, aproximativo, heurístico e metaheurístico. Os métodos exatos garantem a melhor solução, permitindo afirmar que estes encontram a solução ótima. Contudo, a adoção de métodos exatos em problemas caracterizados como NP-completo pode demandar tempo exponencial para encontrar a solução ótima. Esse é o o caso do problema estudado nesta tese de doutorado. Assim, há na literatura várias aplicações de meta-heurísticas no escalonamento de workflow em nuvem. Algoritmos genéticos (AGs) e Enxame de Partículas (PSO) são métodos comumente utilizados para escalonar workflows. Este capítulo examina o efeito de AGs e PSO no problema de escalonamento de workflows para apoiar a escolha da meta-heurística a ser usada no decorrer desta tese.

6.2 Estratégia de codificação

Este capítulo baseia-se no estudo conduzido em Li *et al.* (2016) que tenta encontrar uma solução de escalonamento para reduzir o custo, respeitando um *deadline* e uma restrição de segurança. Para isso, otimiza a busca de instância de VM e do nível de cada serviço de segurança para cada tarefa do workflow. A Eq. 6.1 apresentada a estratégia de codificação (representação da solução) adotada na otimização do problema, onde Γ é a função de custo que combina os tipos de VMs (*compute-resources*) $cr_k^l \in CR_k$ e o nível de segurança $ss_i^l \in ss^l$ para cada tarefa $t_i \in T$. O termo ss^l representa o conjunto de níveis de um serviço de segurança l que pode variar entre 0 e 1. Por exemplo, a Tabela 9 apresenta um conjunto de oito algoritmos de criptografia

representados em $ss^c = \{0, 0.08, 0.14, 0.36, 0.40.0.46, 0.64, 0.90, 1.00\}$.

$$\begin{aligned} Minimizar : TEC = \Gamma(&cr_s^k(t_0), ss_0^a, ss_0^g, ss_0^c, ..., \\ &cr_s^k(t_i), ss_i^a, ss_i^g, ss_i^c, ..., \\ &cr_s^k(t_{n-1}), ss_{n-1}^a, ss_{n-1}^g, ss_{n-1}^c) \end{aligned} \tag{6.1}$$

Tabela 9 – Algoritmos de criptografia.

Algorithm	Security level	Overhead (kB/ms)
SEAL	0.08	168.75
RC4	0.14	96.43
Blowfish	0.36	37.50
Knufu/Khafre	0.40	33.75
RC5	0.46	29.35
Rijndael	0.64	21.09
DES	0.90	15.00
IDEA	1.00	13.50

Fonte: Li *et al.* (2016).

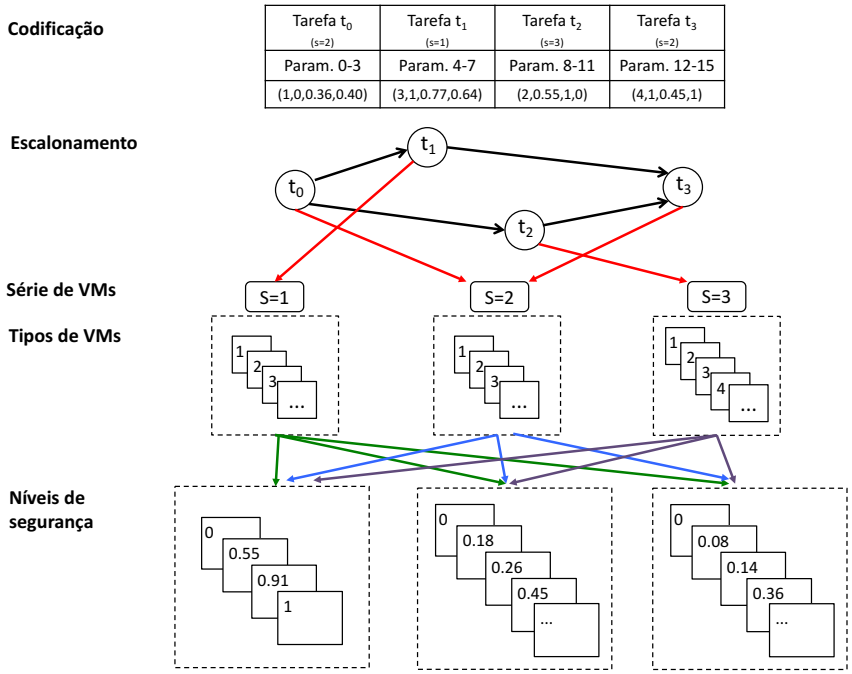
A dimensão da codificação é denotada por $D = 4 \cdot n$, onde n é o número de tarefas do workflow. Considerando o escalonamento de um workflow com quatro tarefas, a função de minimização deve incluir 16 parâmetros (como mostrado na Figura 27). Em algoritmos evolutivos, cada parâmetro é considerado um gene. A cada quatro genes é definida a configuração de uma tarefa, que inclui o tipo de VM e os três níveis de serviços de segurança. Por exemplo, para a tarefa t_0, o parâmetro na posição 0 representa o tipo de VM atribuído à tarefa, e os parâmetros nas posições 1, 2 e 3 determinam os serviços de autenticação, verificação de integridade e confidencialidade, respectivamente. Em seguida, o parâmetro na posição 4 define o tipo de VM da tarefa t_1, enquanto as posições 5, 6 e 7 representam os três serviços de segurança para esta tarefa. Segue-se a mesma lógica para o restante dos parâmetros de codificação.

6.3 Geração do escalonamento

Uma solução candidata é decodificada em um mapa de escalonamento pelo Algoritmo 3. Primeiro, é preciso inicializar o conjunto de máquinas virtuais CR, o conjunto de níveis de segurança ss e o conjunto de mapeamento tarefa-recurso M como vazio (linha 1). O custo total de execução E^c, o tempo de execução E^t e o risco $Risk_{(w)}$ são inicializados com 0. Então, o escalonamento itera sobre cada quatro posições do vetor pos. Isso é realizado através da estratégia de codificação definida na Seção 6.2, onde as coordenadas $4i$, $4i+1$, $4i+2$, $4i+3$ correspondem a tarefa t_i. Assim, $pos[4i]$, $pos[4i+1]$ e $pos[4i+2]$, $pos[4i+3]$ determinam o tipo de VM, bem como o serviço de autenticação, verificação de integridade e confidencialidade, respectivamente.

O tempo de início $ST(t_i)$ de cada tarefa envolve dois possíveis cenários. O primeiro, considera a tarefa t_i sendo uma tarefa de entrada, ou seja, uma tarefa que não possui tarefas

Figura 27 – Exemplo de codificação de escalonamento de um workflow de quatro tarefas.



Fonte: Adaptada de Li *et al.* (2016).

predecessoras. Nesse caso, $ST(t_i)$ é inicializado com 0. No segundo cenário, se a tarefa possui um ou mais pais, $ST(t_i)$ recebe o tempo da última tarefa pai a ser completada.

Os serviços IaaS de nuvens pagas geralmente são cobrados por hora de uso. Logo, VMs podem ficar ociosas após o término da execução de uma tarefa. Por exemplo, supondo que uma tarefa t_{i-1} é a predecessora da tarefa t_i, se o processamento da tarefa $P(t_{i-1}) = 80$ min, ainda restam 40 min de ociosidade da máquina alugada por 120 min. Independentemente das VMs serem utilizadas ou não, o usuário pagará por duas horas de uso. Assim, é razoável reutilizar a mesma VM para a tarefa t_i para evitar tanto a ociosidade da VM quanto a transferência de dados da tarefa t_{i-1} para a tarefa t_i. Essa medida pode auxiliar na redução do custo e tempo de execução do workflow. O algoritmo assume que se a tarefa predecessora t_{i-1} é mapeada para o mesmo tipo de máquina virtual da tarefa t_i ($cr_s^{pos[4i]}(t_i) = cr_s^{pos[4(i-1)]}(t_{i-1})$), a VM pode ser reaproveitada, como mostrado na linha 9 do Algoritmo 3.

Quando a VM é reutilizada, o tempo de processamento $PT(t_i, cr_s^{pos[4i]})$ é calculado considerando o tempo de execução $ET(t_i, cr_s^{pos[4i]})$, tempo de transferência de dados e o *overhead* de segurança como apresentado na Eq. 4.6 (linha 11). Ao final da execução da tarefa, $ET(t_i)$ é calculado pela adição de $PT(t_i, cr_s^{pos[4i]})$ ao $ST(t_i)$ (linha 12). Então, o tempo de fim de uso da VM pela tarefa t_{i-1} é atualizado para $ET(t_i)$ (linha 13). O tempo de início $LST(t_i, cr_s^{pos[4i}$) e fim do aluguel $LET(t_i, cr_s^{pos[4i]})$ para tarefa t_i são definidos como 0 para não haver duplicidade na cobrança de VM para a tarefa t_i que reaproveitou a VM da tarefa predecessora (linha 14).

Entretanto, se a tarefa t_i não reutilizar a VM da tarefa t_{i-1}, o tempo de processamento

$PT(t_i, cr_s^{pos[4i]})$ e tempo do término $ET(t_i)$ da tarefa t_i são calculados pela adição do tempo de transferência (linhas 16 e 17). Em seguida, o tempo de início $LST(t_i, cr_s^{pos[4i]})$ e término $LET(t_i, cr_s^{pos[4i]})$ do uso da VM são calculados.

Os conjuntos M, R e SS são atualizados (linha 21) a cada tarefa. Uma vez que o algoritmo percorreu todas as tarefas, R irá conter todos recursos a serem alugados na nuvem, incluindo o início e término das VMs. O mapeamento de cada tarefa a um recurso estará contido em M e os níveis de segurança a serem atribuídos em cada tarefa estarão em SS. Essas informações permitem calcular o risco de execução do workflow $Risk_{(w)}$, o custo E^c e o tempo E^t total de execução do workflow.

O Algoritmo 3 é combinado com uma meta-heurística para encontrar um escalonamento próximo do ótimo.

Algoritmo 3 – Decodificação de escalonamento.
Fonte: Li *et al.* (2016).

1: **função** SCHEDULINGDECODING(pos)
2: $R = M = \phi$ and $E^t = E^c = P(T) = 0$
3: **para todo** *task* $t_i \in T$ **faça**
4: **se** t_i has no parents **então**
5: $ST(t_i) \leftarrow 0$
6: **senão**
7: $ST_i \leftarrow max\{ET(t_j)|t_j \in pre(t_i)\}$
8: **fim se**
9: **se** $cr_s^{pos[4i]} = cr_s^{pos[4(i-1)]}$ **então**
10: $I_{i-1,i} \leftarrow 0$
11: $PT(t_i, cr_s^{pos[4i]}) \leftarrow TT(t_i) + ET(t_i, cr_s^{pos[4i]}) + SC(t_i)$
12: $ET(t_i) \leftarrow PT(t_i, cr_s pos[4i]) + ST(t_i)$
13: $LET(t_{i-1}, cr_s^{pos[4(i-1)]}) \leftarrow ET(t_i)$
14: $LST(t_i, cr_s^{pos[4i}) \leftarrow LET(t_i, cr_s^{pos[4i]}) \leftarrow 0$
15: **senão**
16: $PT(t_i, cr_s^{pos[4i]}) \leftarrow TT(t_i) + ET(t_i, cr_s^{pos[4i]}) + SC(t_i)$
17: $ET(t_i) \leftarrow PT(t_i, cr_s^{pos[4i]}) + ST(t_i)$
18: $LST(t_i, cr_s^{pos[4i]}) \leftarrow ST(t_i)$
19: $LET(t_i, cr_s^{pos[4i]}) \leftarrow PT(t_i, cr_s^{pos[4i]}) + LST(t_i, cr_s^{pos[4i]})$
20: **fim se**
21: Add $m(t_i, cr_s^{pos[4i]})$ *to the set* M, r_i *to* R *and* ss_i *to* SS
22: **fim para**
23: $TEC \leftarrow \sum_{i=0}^{n-1} c_s^{pos[i]} \cdot \lceil LET(t_i, cr_s^{pos[i]}) - LST(t_i, cr_s^{pos[i]}) \rceil$
24: $TET \leftarrow max\{ET(t_i)|t_i \in T\}$
25: **fim função**

6.4 Função *fitness*

O objetivo da otimização desse estudo é minimizar o custo de execução E^c calculado pelo Algoritmo 3. O tempo total de execução E^t, o *deadline* DL e o risco de execução $Risk_{(w)}$ são utilizados para calcular o coeficiente $\Phi(x)$. Para avaliar a qualidade superior das possíveis soluções viáveis, considera-se: (1) ao decidir entre duas opções viáveis, deve-se escolher a melhor entre elas; (2) uma solução viável é sempre melhor que uma solução inviável, e; (3) ao decidir entre duas soluções inviáveis, deve-se escolher aquela que tem a menor soma de restrições violadas conforme representado na Eq. 6.2.

$$\Phi(x) = max(0, E^t - DL) + max(0, Risk_{(w)}) \tag{6.2}$$

6.5 Enxame de partículas

O algoritmo de Enxame de Partículas (PSO) foi introduzido por Eberhart e Kennedy (1995). Partículas podem se deslocar sobre o espaço de busca com a velocidade regulada de acordo com a sua experiência de vôo e de seus vizinhos. Uma partícula é um ponto representado por $X_i = (x_1, x_2, ..., x_{iD})$, onde D é a dimensão do espaço de busca. Cada partícula tem a sua melhor posição obtida pelo melhor *fitness* que é representada como $PB_i = (pb_{i1}, pb_{i2}, ..., pb_{iD})$. A melhor partícula em uma população é representada pela variável g. A velocidade de uma partícula i é representada por $v_i = (v_{i1}, v_{i2}, ..., v_{iD})$. A velocidade e posição de cada partícula é calculada por Eq. 6.3a e 6.3b, respectivamente, onde ω é um valor entre [0,1], φ_1 e φ_2 são duas constantes positivas para representar a aceleração da partícula, e $rand_1()$ e $rand_2()$ são duas funções que geram valores aleatórios entre 0 e 1.

$$\begin{aligned} v_{id} = \omega \cdot v_{id} + \varphi_1 \cdot rand_1() \cdot (pb_{id} - x_{id}) + \\ \varphi_2 \cdot rand_2() \cdot (pb_{gd} - x_{id}) \end{aligned} \tag{6.3a}$$

$$x_{id} = x_{id} + v_{id} \tag{6.3b}$$

O algoritmo de Enxame de Partículas usado neste experimento está descrito no Algoritmo 4. A fórmula de velocidade é composta por vários parâmetros que afetam a convergência do algoritmo. O peso ω pode ser interpretado como a fluidez do meio em que cada partícula se move e isso determina até que ponto as velocidades anteriores afetam a velocidade atual. O uso de pesos maiores aumentará a velocidade das partículas e ajudará na exploração global. Por outro lado, um peso menor beneficia a otimização da busca local.

A magnitude das forças aleatórias na direção da partícula p_i e da melhor vizinhança p_g é determinada pelas constantes φ_1 e φ_2, respectivamente. Valores maiores fazem as partículas divergirem, enquanto valores menores limitam seus movimentos. A recomendação para a soma

Algoritmo 4 – Algoritmo de Otimização por Enxame de Partículas.
Fonte: Adaptada de Eberhart e Kennedy (1995).

```
1:  função ENXAMEDEPARTICULAS(partículas N, dimensão D)
2:      gbest ← 0; p_g ← 0
3:      Inicialização aleatória das partículas e velocidades
4:      para todo particle i ate N faça
5:          Inicialização aleatória x_i and v_i
6:          p_i ← x_i e calcula pbest_i
7:      fim para
8:      enquanto condição de parada não atingida faça
9:          para todo particle i ate N faça
10:             Avalia cada partícula
11:             se fitness(p_i) ≥ fitness(pbest_i) então
12:                 pbest_i ← fitness(p_i)
13:                 p_i ← x_i
14:             fim se
15:         fim para
16:         Armazena o índice da melhor aptidão entre todas as partículas
17:         para todo particula i ate N faça
18:             Atualiza a velocidade e posição da partícula
19:             Mantém a partícula dentro do espaço de busca
20:         fim para
21:     fim enquanto
22: fim função
```

dessas duas constantes é $\varphi_{sum} = \varphi_1 + \varphi_2 > 4$. Outros parâmetros também afetam a eficiência do PSO. O tamanho da população é empiricamente definido com base na dimensionalidade e dificuldade percebida de um problema. Valores variando entre 20 e 50 são comuns. A dimensão das partículas e os limites de movimentação são dois parâmetros que são definidos de acordo com o tipo de problema a ser resolvido.

6.6 Algoritmo genético

O conceito chave de algoritmos genéticos (AGs) é encontrar soluções, dado um espaço de busca, através da evolução de um conjunto de soluções como descrito pelo Algoritmo 5. O processo de inicialização (linhas 2-5) usa o operador *bestSec* (recomendado neste estudo), onde todos os três serviços de segurança $pos[4i+1]$, $pos[4i+2]$ e $pos[4i+3]$ são inicializados pelo uso do melhor nível de segurança possível (quanto maior, melhor), enquanto o gene de VMs $pos[4i]$ é inicializado com valor aleatório. Desse modo, o operador *bestSec* inicializa um esquema de escalonamento que produz o menor risco possível para a execução do workflow. Caso a execução não dependa do menor risco possível, o processo de otimização pode evoluir para um esquema considerando maior risco, porém produzindo menor tempo e custo de execução.

O processo evolutivo (linhas 6-15) continua até que o tempo limite definido pelo usuário

Algoritmo 5 – Algoritmo Genético.
Fonte: Adaptada de Goldberg (1989).

```
1:  função ALGORITMOGENÉTICO
2:      para todo ind na populacao faça
3:          inicializa(ind)
4:          avalia(ind)
5:      fim para
6:      enquanto prazo não atingido faça
7:          seleciona(parents)
8:          filho ← cruzamento(pais)
9:          λ ← valor aleatorio ∈ [0.0,1.0];
10:         se λ ≤ taxa de mutacao então
11:             mutacao(filho)
12:         fim se
13:         avalia(filho)
14:         adiciona(filho na populacao)
15:     fim enquanto
16: fim função
```

seja atingido. Três operadores de cruzamento (*crossUniform*, *crossAverage* e *crossBLX*) foram desenvolvidos para gerar um indivíduo filho de modo que um desses operadores é aleatoriamente escolhido quando a função de $cruzamento()$ é chamada. No operador $crossUniform$, o valor da posição g do indivíduo filho $Child.pos[g]$ é herdado do pai $Parent1.pos[g]$ ou $Parent2.pos[g]$. No *crossAverage*, cada valor em $pos[g]$ é definido de acordo com a Eq. 6.4.

$$Child.pos[g] = 0.5(Parent1.pos[g] + Parent2.pos[g]) \tag{6.4}$$

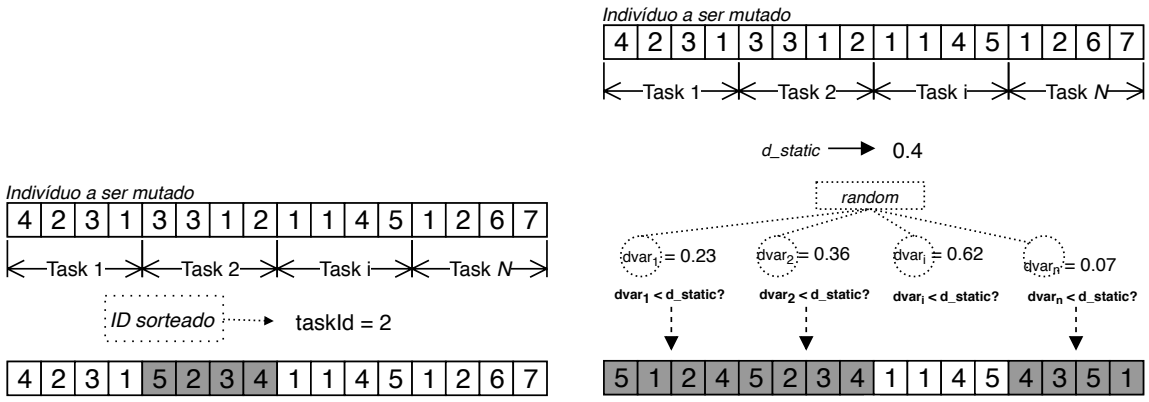
Por fim, o operador *crossBLX* proposto em Takahashi e Kita (2001) foi implementado de modo que cada valor de $pos[g]$ é determinado a partir de um valor aleatório $\beta \in [-\alpha; 1+\alpha]$ como expressado na Eq. 6.5.

$$Child.pos[g] = \beta * Parent1.pos[g] + (1-\beta) * Parent2.pos[g] \tag{6.5}$$

Com base em experimentos empíricos, foram desenvolvidos dois operadores de mutação, os quais são selecionados aleatoriamente sempre que a taxa de mutação λ é satisfeita (linhas 10-12). O primeiro deles é chamado de *mutLimit* que modifica os parâmetros de uma tarefa, configurando todos os três serviços de segurança para o nível mais baixo, enquanto o tipo de VM é definido aleatoriamente. Este operador produz uma mutação com poucas modificações do indivíduo. Por outro lado, também criamos outro operador chamado *mutStrong* que provoca maiores modificações em um indivíduo. Nesse operador, um número $d_{static} \in [0,1]$ é gerado aleatoriamente. Para cada tarefa t_i, outro número aleatório $d_{var} \in [0,1]$ é gerado e comparado com d_{static}. Se d_{var} for menor que $d_{estatico}$, então os quatro genes da tarefa t_i serão aleatoriamente

alterados de acordo com seu domínio particular; caso contrário, seus valores serão mantidos. A Figura 28a e 28b ilustram *mutLimit* e *mutStrong*, respectivamente. O indivíduo filho é avaliado e adicionado à população (linha 14), caso seja melhor que um de seus pais.

Figura 28 – Operadores de mutação. (a) mutLimit e (b) mutStrong.



Fonte: Elaborada pelo autor.

6.7 Algoritmo genético multi-populacional

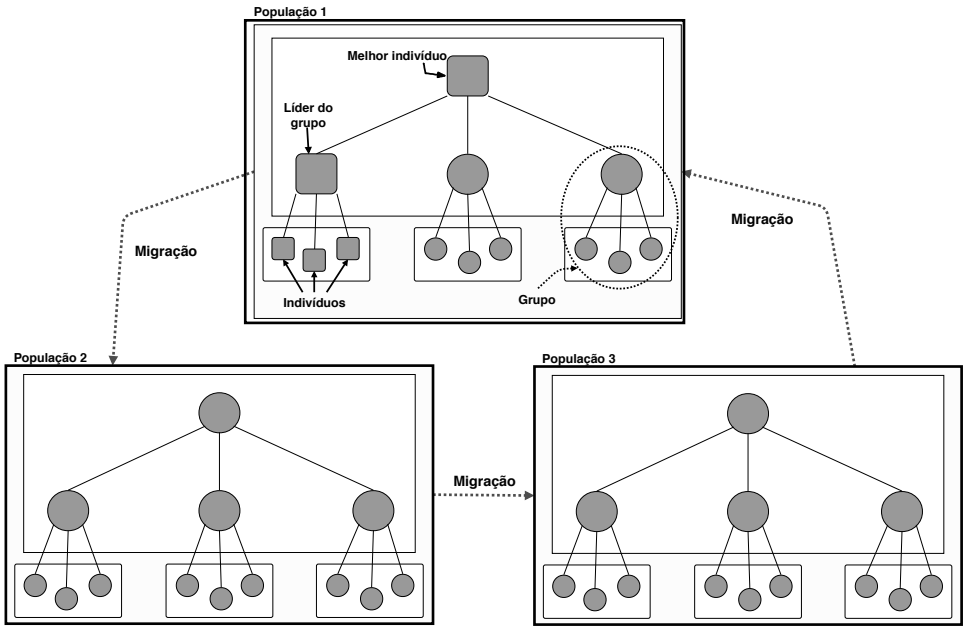
O segundo método baseado em algoritmo genético evolui populações hierarquicamente estruturadas em árvores. Esta abordagem denominada de Multi-population Genetic Algorithm (MPGA) ou algoritmo genético multi-populacional foi proposta pela primeira vez por França, Mendes e Moscato (2001) sendo aplicada para resolver vários problemas da literatura como descrito em Toledo *et al.* (2009), Arantes *et al.* (2017). Na Figura 29 é apresentada a estrutura hierárquica em questão.

A população é organizada em uma árvore onde os indivíduos são representados como nós. Um *cluster* tem indivíduos seguidores e um indivíduo líder, onde os seguidores têm uma "aptidão"pior do que seu líder. Desta forma, o melhor indivíduo estará na raiz, enquanto os piores indivíduos tendem a ser folhas. O MPGA evolui várias populações com os melhores indivíduos migrando de uma população para outra, como mostrado na Figura 29.

Essas populações evoluem independentemente umas das outras por um certo número de gerações (tempo de isolamento). Após o tempo de isolamento, um número de indivíduos é migrado entre as populações. Isso não é bem verdade: O número de indivíduos trocados (taxa de migração), o método de seleção dos indivíduos para migração e o esquema de migração determinam quanta diversidade genética pode ser tranferida entre populações. O processo de evolução do MPGA é descrito no Algoritmo 6.

O operador de inicialização *bestSec* é empregado para gerar indivíduos para cada população (linhas 4-7). Assim que inicializada, a população é organizada de modo que os melhores

Figura 29 – Estrutura e migração da população.



Fonte: Adaptada de França, Mendes e Moscato (2001).

indivíduos de cada população estejam na raiz da árvore, enquanto os piores são alocados nas folhas (linha 8). O processo evolutivo é realizado logo em seguida (linhas 9-21) até que o tempo limite de otimização seja atingido. O parâmetro *nCross* define o número de novos indivíduos gerados em cada evolução, conforme expresso na Eq. 6.6, onde $Pop.Size$ contém o número de indivídiuos da população e $crossRate$ é a taxa de mutação da população.

$$nCross = Pop.Size * crossRate; \tag{6.6}$$

Um indivíduo seguidor e seu líder são selecionados aleatoriamente como pais (linha 11), e um operador de cruzamento é escolhido aleatoriamente (linha 12). Da mesma forma, dois operadores de mutação definidos anteriormente para o AG, também são usados pelo MPGA, com um deles sendo selecionado aleatoriamente a cada vez. A mutação ocorre quando o critério da taxa de mutação λ é satisfeito (linhas 14-16). O novo indivíduo *Child* é avaliado e adicionado na população (linhas 17-18). *Child* somente substitui um dos pais se ele apresentar *fitness* melhor que seu pior pai. Após a aplicação dos operadores de cruzamento e mutação, a estrutura da população é reorganizada (linha 20) para preservar a estrutura hierárquica. O MPGA converge quando nenhum *Child* é incluído em nenhuma das populações. Em seguida, o melhor indivíduo de uma população migra para a próxima (linhas 23-25). A partir de então, o melhor indivíduo da população e o recém migrado serão mantidos em cada população na próxima inicialização.

Algoritmo 6 – Algoritmo Genético Multi-populacional.
Fonte: França, Mendes e Moscato (2001).

```
1: função <MULTIPOPULATIONGENETICALGORITHM
2:     enquanto time limit is not reached faça
3:         para todo pop in populations faça
4:             para todo ind in pop faça
5:                 initialize(ind)
6:                 evaluate(ind)
7:             fim para
8:             organize(pop)
9:             enquanto !converge(pop) faça
10:                para todo j=1 até nCruzamento faça
11:                    select(parents)
12:                    child ← cruzamento(parents)
13:                    λ ← a random value ∈ [0.0,1.0]
14:                    se λ ≤ mutation rate então
15:                        mutation(child)
16:                    fim se
17:                    evaluate(child)
18:                    add(child in pop)
19:                fim para
20:                organize(pop)
21:            fim enquanto
22:        fim para
23:        para todo pop in populations faça
24:            migrate(pop)
25:        fim para
26:    fim enquanto
27: fim função
```

6.8 Comparação PSO vs. AG vs. MPGA

Nesta seção são apresentados os parâmetros de simulação usados nos algoritmos AG, MPGA e PSO. Além disso, três workflows baseados em aplicações do mundo real utilizados para a avaliação das metaheurísticas são descritos. A descrição detalhada de cada um deles, incluindo a sua estrutura, dados e requisitos computacionais está disponível em Juve *et al.* (2013b).

O Cybershake é uma aplicação para estimar os riscos de terremotos através de uma análise probabilística. É caracterizado por ser um workflow de computação intensiva e requer até 1870 MB de memória. Quase todo o tempo de execução (97 %) do Cybershake é gasto na execução das tarefas *SeismogramSynthesis*. Em média, cada tarefa lê aproximadamente 547 MB. No entanto, o tamanho total de todos os arquivos de entrada é de cerca de 150-250MB. Isso sugere que alguns dos dados de entrada são lidos várias vezes por uma tarefa. Neste workflow, o processador é utilizado durante 92 % de todo o tempo de execução.

O SIPHT é um workflow pequeno que busca por RNA não-traduzido. Ele é formado por

treze tarefas diferentes e únicas, exceto a quantidade de tarefas *Patser* que depende do número de entradas para a descrição das regiões de transcrição. Quase todas as tarefas fazem uso intensivo de CPU, sendo a tarefa *Blast* a de tempo de execução mais longo (aprox. 74 %). *Finderm* e *Blast_QRNA* também ocupam um tempo de execução significativo para operações de leitura e escrita em disco. Como no Cybershake, a maioria das tarefas do SIPHT apresentam uma alta taxa de uso de CPU. Contudo, a tarefa *PatserConcate*, responsável por concatenar os dados de saída de cada tarefa *Patser* consome apenas 18 % da CPU.

O Epigenomics é um workflow que realiza o mapeamento do estado epigenético de células humanas. É caracterizado por vários pipelines que calculam fragmentos independentes de dados em paralelo. O Epigenomics é formado por oito categorias de tarefas. *fastQSplit* é a tarefa de entrada que divide a sequência de DNA em vários trechos, durante o processo de análise genética. Posteriormente, *filterContams* remove dados ruidosos e contaminados de cada fragmento para que possam ser convertidos em um formato reconhecível pelo utilitário *sol2sanger*. Os dados são convertidos para o formato *fastQ* pela tarefa *fast2bfq* para otimizar o tempo de processamento e reduzir o uso de disco. Então, as sequências restantes são alinhadas pela tarefa *map*, a qual passa maior parte do tempo fazendo uso de CPU (aprox. 96 %). Os resultados dos mapeamentos individuais são combinados por meio das tarefas *mapMerge* e *mapIndex*. No final do workflow, a tarefa *pileup* organiza os dados de saída a serem exibidos por uma GUI. Tanto a tarefa *pileup* quanto a *mapIndex* também fazem uso intensivo do processador.

Todas as simulações desta seção foram executadas no *cluster* Halley alocado no Laboratório de Sistemas Distribuídos e Programação Concorrente (LaSDPC) da USP - São Carlos. O *cluster* contém 12 nós e é equipado com processadores Intel Core i7 U4790 3.6 GHz, 32 GB de RAM e sistema operacional Linux. A adaptação do WorkflowSim apresentada na Seção 5 foi usada para simular o ambiente de nuvem no qual as VMs são alocadas no mesmo *datacenter* e conectadas por meio de uma rede de 1 Gbit. As características das máquinas virtuais e o preço cobrado pela Amazon EC2 foram levados em consideração. Além disso, o Workflow Generator foi usado para gerar os workflows utilizando os parâmetros descritos na Tabela 10.

Tabela 10 – Parâmetros de geração da carga de trabalho de workflows.

Cybershake		**SIPHT**		**Epigenomics**	
Jobs	30	Jobs	30	Jobs	46
Max_Ruptures	30	Mean_Patsers	18	Max_Sequences	160
Max_Variations	30	Partner_Factor	936	Lane_Factor	100
Bias	0.05	Runtime_Factor	2	Runtime_Factor	1.5
Min_inputs	1				
Extract_factor	0.0081				
Runtime_factor	1				

Fonte: Elaborada pelo autor.

A restrição de *deadline DL* foi arbitrariamente escolhida como a média dos tempos de execução mais lentos e mais rápidos. O primeiro é calculado a partir do tempo de execução de um workflow que inclui a VM mais barata e mais lenta, enquanto o segundo faz uso do tipo

de VM mais cara (provavelmente a mais rápida). Desse modo sempre haverá uma solução que possa satisfazer o *deadline*. Em ambas as execuções, os serviços de segurança são definidos no nível mais alto (1.0) para todas as tarefas do workflow. Os parâmetros de meta-heurísticas foram definidos na Tabela 11. Os parâmetros do AG e MPGA foram definidos empiricamente com base em ensaios anteriores. No entanto, a maioria dos parâmetros foram obtidos a partir de descrições na literatura como em Toledo *et al.* (2009) e Arantes *et al.* (2017). Os parâmetros do PSO foram definidos usando os valores aplicados por Li *et al.* (2016).

Tabela 11 – Parâmetros usados nas meta-heurísticas.

AG		MPGA		PSO	
Population size	100	**Number of populations**	3	**Particles**	50
Selection method	Tournament	**Number of individuals**	7	**Iterations**	100
Tournament size	3	**Crossover rate**	5.0	φ_1	2.05
Mutation rate	0.1	**Mutation rate**	0.7	φ_2	2.05
		Structure	Tree	ω	0.5
		Ramification	2		
Initialization operator	bestSec				
Crossover operators	crossUniform, crossAverage, crossBLX				
Mutation operators	mutLimit, mutStrong				

Fonte: Dados da pesquisa.

Os métodos HMAC-MD5, HMAC-SHA-1 e CBC-MAC-AES foram assumidos como estando disponíveis para o serviço de autenticação. Também foram considerados sete métodos de verificação de integridade (MD4, MD5, RIPEMD, RIPEMD-128, SHA-1, RIPEMD-160 e Tiger) e oito algoritmos de criptografia (SEAL, RC4, Blowfish, Knufu / Khafre, RC5, Rijndael, DES e IDEA) para a verificação de integridade e criptografia. A função de *overhead* $SC^l(t_i)$ dos serviços de integridade g e criptografia c é expressada pela Eq. 6.7, onde β representa a quantidade de ataques realizados em cada tipo de serviço de segurança $l \in \{g,c\}$. Para esse experimento, supõe-se que $\beta^a = 3$, $\beta^g = 4$ e $\beta^c = 1$.

$$SC^l(t_i) = F^l(sl_i^l) = \beta^l \cdot sl_i^l \cdot d_i^l, \qquad l \in \{g,c\} \tag{6.7}$$

O serviço de autenticação é independente da quantidade de dados. Assim, o *overhead* de segurança inclui apenas o custo do método de autenticação, conforme representado pela Eq. 6.8.

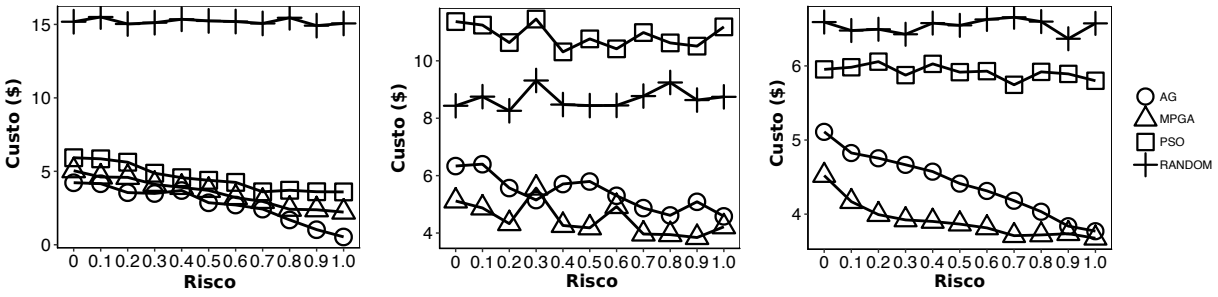
$$SC^l(t_i) = F^l(sl_i^l) = \beta^l \cdot sl_i^l, \qquad l \in \{a\} \tag{6.8}$$

Os algoritmos de Enxame de Partículas, Algoritmo Genético e Algoritmo Genético Multi-populacional foram avaliados comparando o custo monetário enquanto a taxa de risco é alterada. Testes estatísticos foram realizados para verificar se há diferença significativa entre as três meta-heurísticas. Finalmente, o tempo de resposta desses métodos foi analisado.

6.9 Comparação do custo de execução

Experimentos foram realizados abrangendo três workflows. Para fins de comparação, os métodos AG, MPGA, PSO e Geração Aleatória (*baseline*) foram analisados em termos de custo e considerando diferentes restrições de risco. A Geração Aleatória consiste na produção de um mapa de escalonamento onde cada gene é gerado aleatoriamente respeitando os valores válidos para cada posição g. A taxa de risco (P_c) variou entre 0 a 1 com incrementos de 0.1, onde 0 representa baixo risco de sofrer ataques enquanto 1 denota um alto risco de ser violado. Os resultados da otimização são mostrados na Figura 30.

Figura 30 – Custo total de execução para diferentes restrições de risco.



(a) Cybershake (b) SIPHT (c) Epigenomics

Fonte: Elaborada pelo autor.

Na maioria dos casos, como esperado, o custo foi menor quanto maior o risco P_c. Os algoritmos evolutivos conseguiram encontrar soluções que demandam menor custo do que o Enxame de Partículas nos workflows Cybershake, especialmente quando o risco $P_c \geq 0.7$ (Figura 30a). Possivelmente, os operadores genéticos foram mais eficientes que o PSO para encontrar soluções quando a segurança é mais relaxada. A abordagem aleatória não encontrou soluções razoáveis para qualquer taxa de risco.

Os esquemas de escalonamento encontrados pelo AG e MPGA tornam o custo de execução mais barato no workflow SIPHT do que os algoritmos Random e PSO, conforme mostrado na Figura 30b. O algoritmo Random supera PSO no workflow SIPHT. Esse resultado pode ser explicado pelas características da carga de trabalho desse workflow. O SIPHT tem uma única tarefa CPU-bound, *Blast*, responsável pela maior parte do seu tempo de execução (74%). Possivelmente, o PSO não produziu nenhuma movimentação das partículas capaz de encontrar uma solução em que uma VM rápida é atribuída à tarefa *Blast*, enquanto VMs mais lentas (isto é, mais baratas) foram atribuídas às tarefas que não as utilizam por mais de uma hora. Assim, supõe-se que o PSO fica preso em um ótimo local durante o processo de otimização. Por outro lado, os dois operadores de mutação (*mutLimit* e *mutStrong*) podem ter auxiliado para que os algoritmos evolutivos escapassem de solução ótima local.

O AG e o MPGA também superaram o PSO e Random na otimização do escalonamento

do workflow Epigenomics. Embora o PSO tenha obtido melhores soluções do que o Random, os algoritmos evolutivos encontraram soluções que eram pelo menos 17% ($P_c = 0$) a 84% ($P_c = 1.0$) mais baratas que o PSO. Ao contrário do custo obtido pela otimização do escalonamento dos workflows Cybershake e SIPHT, o MPGA obteve soluções significativamente melhores do que o AG para $P_c < 0.9$. O PSO só encontrou soluções mais baratas quando a taxa de risco foi reduzida no workflow Cybershake. No caso dos outros dois workflows, os resultados mostram que o PSO se comporta de maneira semelhante ao Random. Aparentemente, o algoritmo PSO não foi capaz de evitar o ótimo local, como investigado anteriormente em (SCHMITT; WANKA, 2015).

6.10 Validação estatística

Os critérios de normalidade, independência e homocedasticidade devem ser satisfeitos ao se estabelecer os testes paramétricos. O critério de normalidade é satisfeito se as amostras obtidas tiverem uma distribuição que seja pelo menos próxima de uma distribuição normal. O teste de Anderson e Darling (1954) pode ser conduzido para verificar o grau de normalidade. A amostra é independente se os valores em uma amostra não fornecerem informações sobre os valores em outra amostra. Este critério pode ser verificado por meio do coeficiente de correlação de Pearson descrito em Benesty *et al.* (2009). O critério de homoscedasticidade significa que os grupos de amostras têm o mesmo gráfico de dispersão. A homocedasticidade pode ser determinada pelo teste de Levene (1961). Se pelo menos um critério não for satisfeito, testes não paramétricos devem ser usados.

Um total de 1093 execuções foram realizadas para cada meta-heurística, envolvendo 33 repetições para cada taxa de risco. A amostra dos resultados deste estudo não atendeu completamente os três critérios para a aplicação de teste paramétrico e, portanto, testes não paramétricos foram aplicados. Os testes de Nemenyi (1962) e Friedman (1937) foram utilizados para verificar se há diferenças significativas no custo total de execução. Um *p-value* $p = 0.0001 < 0.05$ é retornado pelo teste de Friedman rejeitando a hipótese nula. Assim, a hipótese alternativa é que ao menos um dos métodos é significativamente diferente dos demais.

As Tabelas 12 e 13 fazem uma comparação baseada nos testes de Friedman e Nemenyi. Os métodos de otimização foram avaliados aplicando o teste de Nemenyi para cada workflow individualmente. Os métodos foram avaliados para todos os workflows juntos para obter uma classificação geral. Os resultados mostram que os algoritmos evolutivos superaram as abordagens PSO e Random em todos os testes no quesito custo monetário. O MPGA superou todos os outros métodos, exceto no workflow Cybershake, no qual o AG teve um desempenho melhor. Finalmente, a Random superou a abordagem PSO no workflow SIPHT. Esse fato é discutido em mais detalhes na seção 6.11.

Tabela 12 – Teste de Friedman para avaliação da otimização do custo obtido por cada método.

WF	Método	Média de ranking	Soma do ranking
Cybershake	MPGA	389.5	137525
	AG	511.7	182664
	PSO	658.8	210155
	Random	1205.3	426692
SIPHT	MPGA	311.1	112601.5
	AG	466.7	168930.5
	PSO	948.3	341378
	Random	1170.0	424718
Epigenomics	MPGA	254.7	91168
	AG	479.4	173554.5
	PSO	962.2	349271.5
	Random	1188.5	427852
All workflows	MPGA	1139.4	1227107.5
	AG	1395.8	1503227
	PSO	2624.6	2742716.5
	Random	3406.3	3658350

Fonte: Dados da pesquisa.

Tabela 13 – Teste estatístico de Nemeneyi para prover um ranking geral entre os métodos de otimização. A nota *A* representa a melhor posição no ranking, enquanto D a pior.

Método	Cybershake	SIPHT	Epigenomics	All workflows
MPGA	B	A	A	A
AG	A	B	B	B
PSO	C	D	C	C
Random	D	C	D	D

Fonte: Dados da pesquisa.

6.11 Análise do tempo de resposta

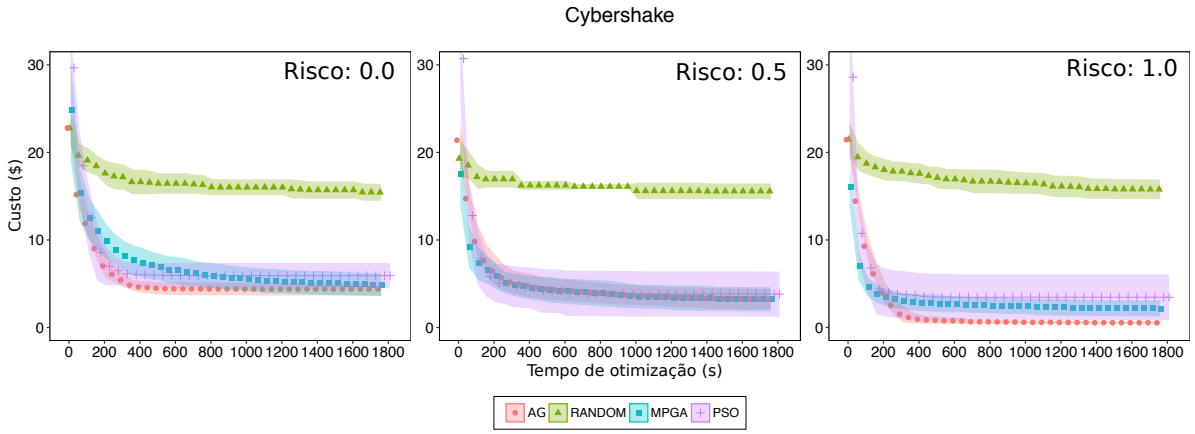
As Figuras 31, 32 e 33 apresentam os melhores custos de execução encontrados durante o processo de otimização (0 a 1800s), com intervalo de medição de 50s para os workflows Cybershake, Epigenomics e SIPHT, respectivamente, a taxa de risco $P_c = \{0.0, 0.5, 1.0\}$. O AG obteve o melhor *fitness* (custo) para todas as taxas de risco no workflow Cybershake. No entanto, o MPGA obteve melhores soluções para $P_c = \{0.5, 1.0\}$ antes dos 200s. O AG tem uma taxa de convergência mais rápida do que o MPGA e o PSO. Esse comportamento pode ser explicado pelo fato de que o AG apenas evolui uma população, enquanto o MPGA possui três populações que evoluíram separadamente. O tempo gasto pelo MPGA, durante o processo de evolução e migração entre essas três populações, pode afetar sua convergência geral dentro de um período mais curto. Embora o PSO não tenha alcançado melhores resultados do que os algoritmos evolutivos, ele apresentou convergência similar ao AG antes dos 200s. Depois disso, o PSO provavelmente ficou preso em um ótimo local.

O MPGA obteve os custos mais baratos seguidos pelo AG, enquanto o Random foi mais eficiente do que o PSO na maioria das configurações de taxa de risco para o workflow do SIPHT (ver Seção 6.9). Todos os métodos otimizaram seu custo total de execução durante um período de tempo, como apresentado na Figura 33. Embora o PSO tenha mantido a otimização de custo

por mais tempo que os outros métodos, ele não é realmente melhor que o Random. Ao observar a Figura 33, nota-se que há uma variação de custo similar para todos os métodos.

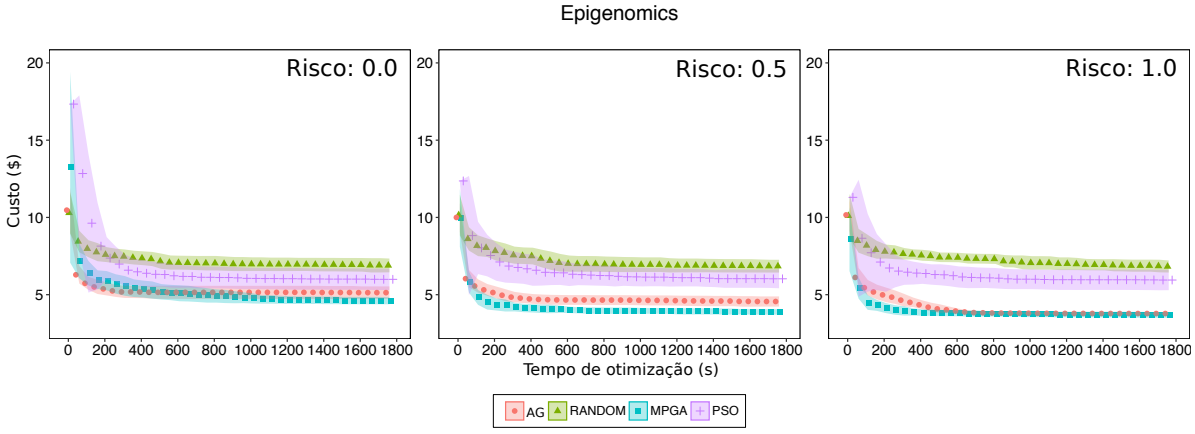
Por fim, houve uma clara diferença entre os métodos no workflow Epigenomics. O MPGA superou os outros métodos após 500s e 150s para $P_c = \{0.0, 1.0\}$ e $P_c = \{0.5\}$, respectivamente. O PSO não produziu um esquema de escalonamento melhor que os algoritmos evolutivos. Como já mencionado, provavelmente foi incapaz de escapar de uma solução ótima local. Atribuímos o melhor desempenho de AG e MPGA aos operadores de cruzamento e mutação propostos, uma vez que eles fornecem uma ampla gama de soluções razoáveis durante o processo de busca.

Figura 31 – Tempo de otimização das meta-heurísticas AG, MPGA e PSO na otimização do custo de execução do workflow Cybershake.



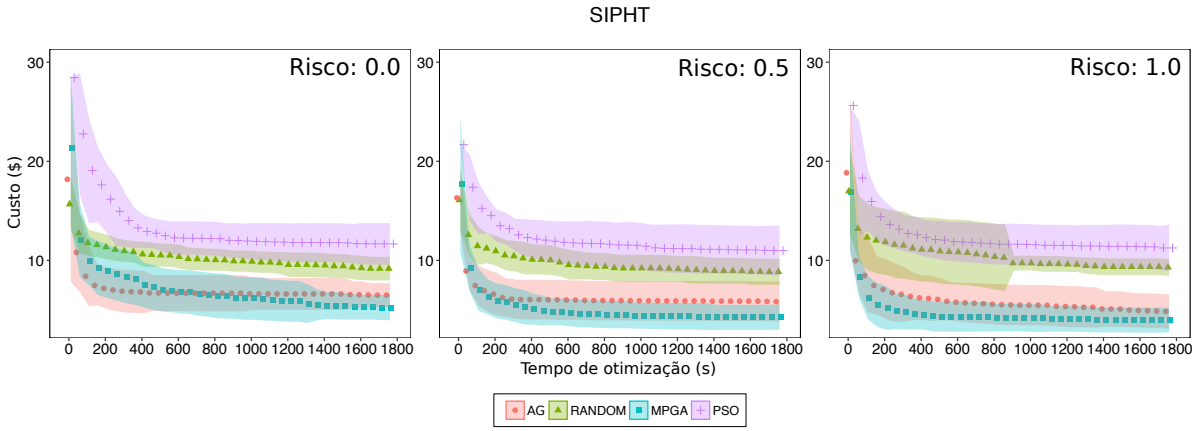
Fonte: Elaborada pelo autor.

Figura 32 – Tempo de resposta das meta-heurísticas AG, MPGA e PSO na otimização do custo de execução do workflow Epigenomics.



Fonte: Elaborada pelo autor.

Figura 33 – Tempo de resposta das meta-heurísticas AG, MPGA e PSO na otimização do custo de execução do workflow SIPHT.



Fonte: Elaborada pelo autor.

6.12 Considerações finais

Neste capítulo foram comparados as meta-heurísticas PSO e algoritmos evolutivos (AG e MPGA) na otimização do escalonamento de workflows em nuvens com restrição de segurança. Os algoritmos evolutivos reduziram o custo de execução dos workflows, atendendo o *deadline* e as restrições de segurança. Além disso, foram introduzidos um operador de inicialização (*bestSec*), dois de mutação (*mutStrong* e *mutLimit*), e três de cruzamento (*crossUniform*, *crossAverage* e *crossBLX*). Os resultados sugerem que tanto AG quanto MPGA foram consistentemente mais eficientes que PSO. Os algoritmos evolutivos retornaram melhores esquemas de escalonamento que reduzem o custo para os três workflows avaliados. Os resultados deste estudo servem como apoio para a adoção de algoritmos evolutivos nas abordagens propostas nesta tese.

CAPÍTULO

7

MÉTODOS DE ESCALONAMENTO PARA A PROTEÇÃO DE DADOS SENSÍVEIS

7.1 Considerações iniciais

Escalonamento de workflow na computação em nuvem é um tópico amplamente investigado. Ao contrário do escalonamento em grades computacionais cujo foco é minimizar o makespan, há outros critérios a serem otimizados em nuvens. As soluções consideram, por exemplo, o makespan, o custo, a energia e a confiabilidade. Recentemente, os algoritmos de escalonamento de workflow também se concentram na segurança de dados sensíveis, como em Watson (2011), Sharif *et al.* (2013), Liu *et al.* (2012), Zeng, Veeravalli e Li (2015), Li *et al.* (2016), Zhu *et al.* (2016). Os algoritmos de escalonamento de workflow com restrição de segurança consideram escalonar tarefas que manipulam dados sensíveis para nuvens privadas. Outra abordagem, é o uso de serviços de segurança como autenticação, verificação de integridade e criptografia em todas as tarefas do workflow quando executadas em nuvens públicas.

Li *et al.* (2016) e Arunarani, Manjula e Vijayan (2017) apresentam algoritmos para encontrar escalonamentos que combinam diferentes níveis de serviços de segurança e diferentes tipos de VM para cada tarefa de modo a minimizar o custo de execução, atendendo ao requisito mínimo de segurança e *deadline*. No entanto, esses algoritmos não garantem que todas as tarefas que lidam com dados sensíveis sejam protegidas adequadamente. Por exemplo, considere um workflow composto por quatro tarefas A - B - C - D, em que somente a tarefa D manipula dados sensíveis. Vamos supor uma solução s_1 que aplica serviços de segurança nas tarefas A - B - C e deixa D sem a proteção apropriada, e s_2 que assegura A - C - D, enquanto a tarefa B não é protegida. Os dois trabalhos citados consideram que ambas as soluções s_1 e s_2 protegeram três tarefas e apresentam o mesmo risco de segurança. No entanto, a solução s_1 sofre uma violação de segurança, deixando a tarefa sensível D sem proteção e protege desnecessariamente A-C, enquanto a solução s_2 protege a tarefa crítica D, mas desperdiça tempo de processamento

protegendo as tarefas *B* e *C*.

Diante desse contexto, esta tese aborda a necessidade de proteger apropriadamente as tarefas sensíveis de um workflow sem o desperdício de tempo de processamento e custo. Duas abordagens para a seleção de tarefas são apresentadas: (1) políticas de seleção de tarefas e (2) anotação de tarefas. A primeira abordagem concebe cinco políticas que podem ser selecionadas pelo usuário na fase de modelagem do workflow, a saber: $entry/end$, $gather$, $scatter$, $gatherescatter$ e all. Cada política determina quais tarefas serão protegidas por serviços de segurança. A segunda abordagem propõe a anotação de tarefas para identificar individualmente quais tarefas são sensíveis ou não, servindo de apoio para a etapa de escalonamento.

Neste capítulo é introduzido o ciclo de vida de um workflow que abrange desde a etapa de modelagem à execução das tarefas de um workflow. As abordagens de políticas de seleção e anotação de tarefas sensíveis são introduzidas, bem como a concepção da codificação e decodificação do escalonamento de tarefas, a fim de minimizar o custo monetário, atendendo o prazo e os requisitos de segurança para a execução de workflows em nuvens públicas. Além disso, a função objetivo e os operadores genéticos para apoiar a otimização das soluções de escalonamento são apresentados.

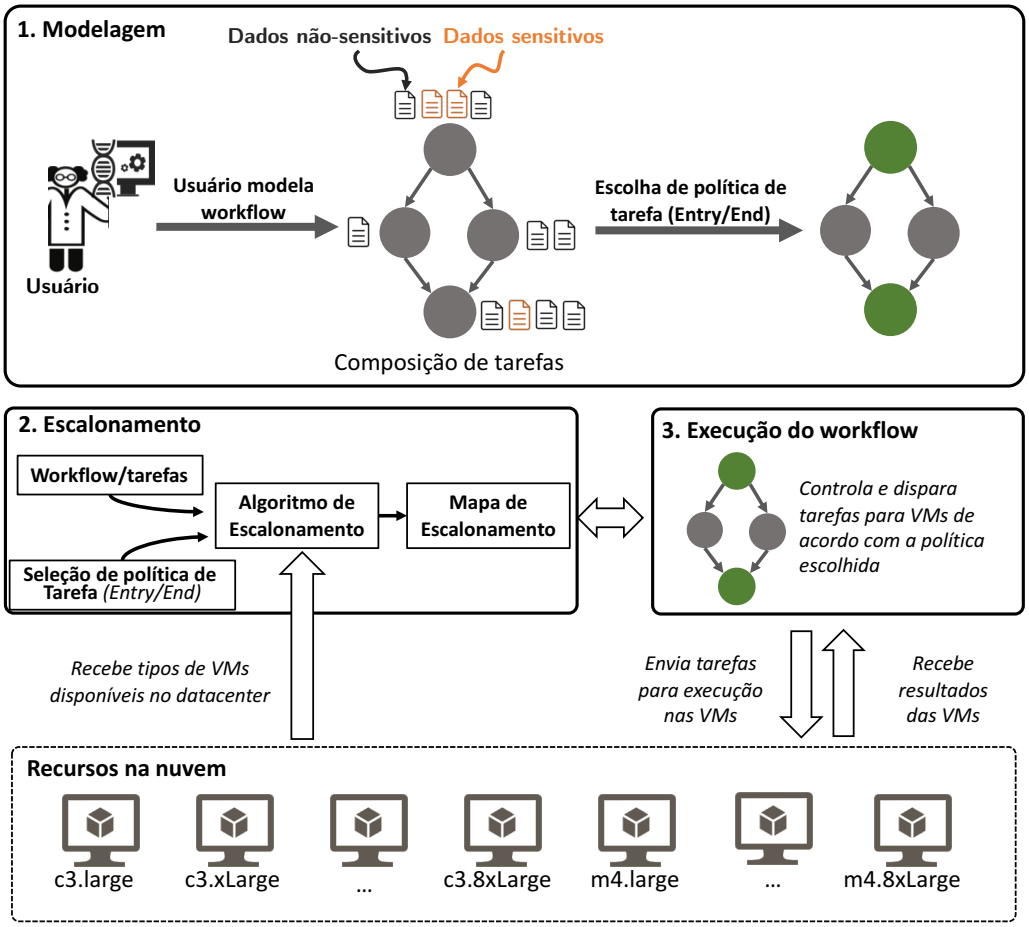
Os métodos propostos pressupoem que (1) o usuário tem acesso a uma estação de trabalho onde um sistema gerenciador de workflow esteja devidamente instalado; (2) os dados de entrada do workflow estejam em sistemas de armazenamento na nuvem; (3) a execução do workflow seja realizada em nuvens públicas com diferentes tipos de instâncias de máquinas virtuais, e; (4) o usuário seja capaz de identificar quais tarefas do workflow envolvem dados sensíveis.

7.2 Ciclo de vida

O ciclo de vida de um workflow pode ser organizado em três fases (modelagem, escalonamento e execução) conforme representado na Figura 34. A fase de modelagem consiste na representação de tarefas e suas dependências para o controle do fluxo de execução do workflow. A descrição de um workflow pode ser realizada por meio de um grafo acíclico direcionado em um arquivo XML. Essa tese considera o formato DAX (Directed Acyclic Graph in XML) que permite representar todas as tarefas computacionais e sua ordem de execução representada por arestas em um grafo, e cada entrada necessária e saída esperada, bem como os argumentos para a chamada de cada tarefa.

Os conjuntos de dados e tarefas são referenciados por identificadores lógicos. A Listagem 2 apresenta um workflow que executa duas tarefas em ordem sequencial: *hello* e *world*. Nesse exemplo, a primeira tarefa com ID0001 refere-se a um executável chamado *hello* identificado pelo atributo *name*. Esta tarefa considera um arquivo de entrada (AE) *f.a* e gera um arquivo de saída (AS) *f.b*. De forma semelhante, a segunda tarefa *world* usa o AE *f.b* e gera a saída *f.c*.

Figura 34 – Ciclo de vida da execução de um workflow.



Fonte: Elaborada pelo autor.

Arquivo 2 – Exemplo de *Hello World* em um arquivo DAX

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <adag name="hello_world">
3      <job id="ID0001" namespace="hello_world" name="hello">
4          <uses name="f.b" link="output"/>
5          <uses name="f.a" link="input"/>
6      </job>
7      <job id="ID0002" namespace="hello_world" name="world">
8          <uses name="f.b" link="input"/>
9          <uses name="f.c" link="output"/>
10     </job>
11     <child ref="ID0002">
12         <parent ref="ID0001"/>
13     </child>
14 </adag>
```

Fonte: Elaborada pelo autor.

A fase de escalonamento adota um algoritmo de escalonamento que estabelece o mapeamento tarefa-VM considerando todas as tarefas do workflow e as diferentes instâncias de VMs de modo a minimizar o custo e atender o prazo e os requisitos de segurança do workflow. Na fase seguinte, um mecanismo de controle gerencia a execução das tarefas em uma ou mais

máquinas virtuais.

As duas abordagens propostas para proteger os dados sensíveis atuam nas fases de modelagem e escalonamento do workflow e serão discutidas em mais detalhes nas próximas seções.

7.3 Políticas de seleção de tarefas

Um workflow pode apresentar diferentes tipos de tarefas como *entrada*, *saída*, *agrupamento*, *dispersão*, e *agrupamento e dispersão* de dados. Cada tipo executa atividades específicas que sofrem consequências caso seus dados sejam vítimas de ataques maliciosos.

Uma tarefa de entrada é definida como a(s) primeira(s) tarefa(s) de um workflow que não possui nenhuma tarefa predecessora (pai) e é responsável pela leitura de todos os dados de entrada (se existirem). Esse tipo de tarefa precisa de atenção adequada, pois geralmente manipula dados brutos, ou seja, sem qualquer tratamento. Para fim de exemplificação, pode-se considerar um workflow que possui uma tarefa de entrada que lê imagens médicas, incluindo os dados pessoais do paciente que serão submetidas a tarefas de diagnóstico de uma doença. O vazamento desses dados pode afetar o indivíduo na sociedade.

Uma tarefa de saída é a última tarefa a ser executada em um workflow que não tem nenhuma sucessora (filha). Esse tipo de tarefa realiza o processamento e armazenamento dos dados de saída do workflow e pode gerar danos irreparáveis caso ataques maliciosos tenham sucesso. Por exemplo, um workflow na área farmacêutica que simula o efeito de diferentes substâncias químicas para a produção de medicamentos. O resultado da execução deste workflow pode conter a lista de substâncias e concentrações testadas, bem como a sua eficácia no tratamento de uma doença.

Uma tarefa de dispersão de dados é responsável por dividir e enviar dados para duas ou mais tarefas sucessoras. Por outro lado, uma tarefa de agrupamento tem duas ou mais tarefas predecessoras (pais) e pode ser aplicada para unir os dados das tarefas pai. Ambos os tipos de tarefas podem ser usados para dispersão e agrupamento de dados parciais e estão frequentemente presentes em aplicações paralelas.

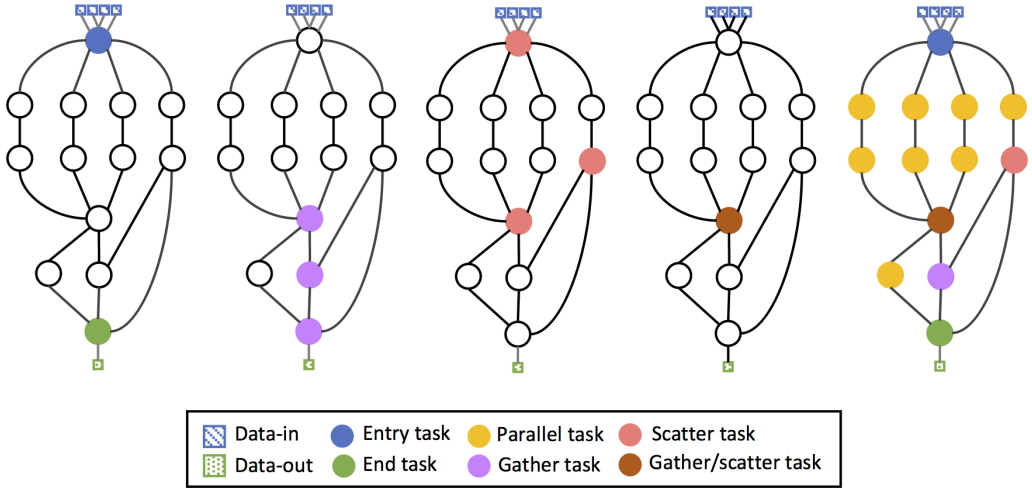
Para exemplificar uma tarefa de dispersão, pode-se considerar um workflow para simulação de diferentes soluções aerodinâmicas na área automobilística. Considerando que o workflow é composto pelas tarefas A, B, $C_1, C_2, C_3, ..., C_n$ e D, onde A é uma tarefa de entrada que lê os parâmetros iniciais, B seleciona as melhores estruturas aerodinâmicas para que sua eficiência seja avaliada em uma tarefa $C = \{C_1, C_2, C_3, ..., C_n\}$, e a tarefa D reúne o resultado de toda a simulação. O vazamento da tarefa de dispersão B contendo as melhores estruturas dinâmicas poderia acarretar em perdas à instituição detentora da informação. Não menos importante, o sucesso de ataques maliciosos na tarefa de agrupamento D também implicaria em efeitos tão

danosos quanto os da tarefa *B*.

Por fim, tarefas *scatter*/*gather* realizam tanto o agrupamento de dados de outras tarefas predecessoras quanto a dispersão a múltiplas tarefas imediatamente sucessoras.

Considerando esses cinco tipos de tarefas de workflow, propõem-se cinco políticas de seleção de tarefas que podem ser escolhidas de acordo com o requisito de segurança de cada tarefa denominado de *Workflow Scheduling with Task Security Policies* (WS-TSP), conforme mostrado na Figura 35. A primeira política é a política *Entry* e *End* que aplica serviços de segurança em todas as tarefas de entrada e saída. Essa política pode ser adotada em workflows cujos dados iniciais e os resultados finais devem ser protegidos. As políticas *Gather* e *Scatter* buscam proteger tarefas que agrupam dados de múltiplas tarefas predecessoras e tarefas que distribuem dados sensíveis para várias tarefas sucessoras, respectivamente. A política *GathereScatter* é uma combinação das polícias *Gather* e *Scatter* para proteger tanto a transferência de dados de múltiplas tarefas pai, quanto as tarefas filhas. Por fim, a política *All* é usada em casos em que os dados de todas as tarefas precisam de um tratamento seguro.

Figura 35 – Políticas de seleção de tarefas: (a) Entry e End; (b) Gather; (c) Scatter; (d) Gather e Scatter; (e) All.



Fonte: Elaborada pelo autor.

As cinco políticas aplicam métodos de autenticação, verificação de integridade e criptografia para proteger as tarefas. As políticas podem ser oferecidas pelo sistema gerenciador de workflows (SGWf) como opções na fase de modelagem do workflow. Para proteger a execução do workflow, o usuário pode escolher a política que melhor se adapta às tarefas sensíveis. Para este propósito, é essencial que o usuário tenha o conhecimento da semântica das tarefas para escolher a política apropriada. Assim, o workflow e a política de segurança são submetidos ao algoritmo de escalonamento que recebe a lista de tipos de VMs e, em seguida, realiza a otimização do escalonamento que atenda às restrições do usuário.

7.4 Otimização multi-objetivo

Alguns algoritmos de escalonamento de workflows otimizam múltiplos critérios. ε-fuzzy DPSO (GARG; SINGH, 2011) e NSPSO (GARG; SINGH, 2014) são dois algoritmos que aplicam enxame de partículas para produzir *trade-off* de Pareto entre custo e makespan. Uma evolução diferencial multi-objetivo foi descrita em Talukder, Kirley e Buyya (2009) para produzir um conjunto de soluções com diferentes *trade-off* de makespan e de custo. Um conjunto de soluções de escalonamento considerando diferentes *trade-off* de makespan e custo são introduzidos em Durillo e Prodan (2014) por meio do escalonamento baseado em Pareto. Yassa *et al.* (2013) propuseram a abordagem DVFS-MODPSO que busca encontrar soluções otimizadas de custo, makespan e consumo de energia. No entanto, nenhum desses estudos considera a segurança no processo de escalonamento de workflow.

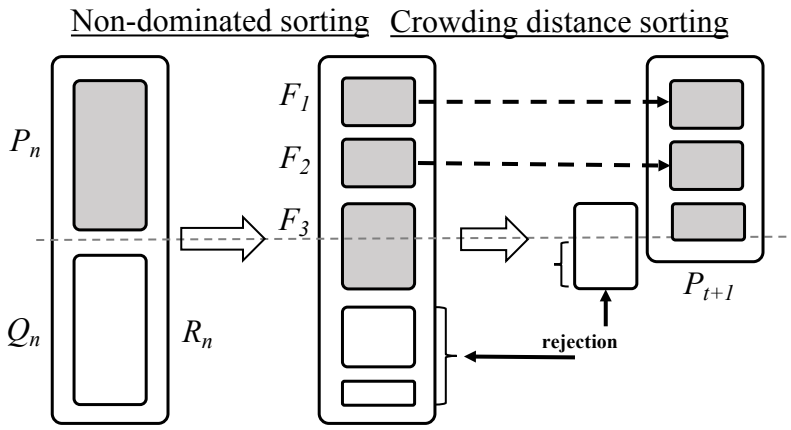
Nesse contexto, é proposta a otimização multi-objetivo do escalonamento de workflow em nuvens que considera o *trade-off* de Pareto entre o custo e o makespan, aplicando diferentes políticas de seleção de tarefas para proteger a execução do workflow. Esta seção apresenta o uso da meta-heurística NSGA-II aplicada na busca de soluções com diferentes *trade-off* entre o custo e o makespan em cada uma das políticas. Dessa forma, usuários podem escolher soluções aceitáveis com base em suas preferências.

O NSGA-II é um dos algoritmos evolutivos multi-objetivo (MOEAs) amplamente utilizados na literatura (TAN; MA; MEI, 2017; UYDURAN *et al.*, 2016; BEIRIGO; SANTOS, 2016). O NSGA-II envolve dois aspectos: ordenação rápida não-dominada (*non-dominated sorting*) de indivíduos e a estratégia de seleção elitista (*crowding distance sorting*). A ordenação não-dominada é formada por índices de classificação não-dominados e pela distância de aglomeração. A relação de dominância de Pareto determina a classificação das soluções não-dominadas. Considerando um problema multi-objetivo com duas funções, a dominação de Pareto é definida a seguir. Para vetores de soluções x_1 e x_2, x_1 é chamada de solução dominante de Pareto quando as duas condições seguintes forem atendidas:

$$[f_i(x_1) \leq f_i(x_2), i = 1,2] \wedge [f_i(x_1) < f_i(x_2), \exists i \in \{1,2\}] \tag{7.1}$$

O processo de evolução no NSGA-II é ilustrado na Figura 36. Os indivíduos partem de dois grupos na n-ésima geração: indivíduos pai (P_n) e indivíduos descendentes (Q_n). A classificação não-dominada D e o índice de distância de aglomeração para cada indivíduo são calculados para a nova população de indivíduos $R_n = (P_n, Q_n)$. O índice de distância de superposição representa a uniformidade da distribuição individual através da população atual. Os melhores são selecionados com base na classificação não-dominada. Para indivíduos com o mesmo valor de classificação, o índice de distância de aglomeração é usado para desempate, em que se opta por aqueles com o índice mais alto. Esse processo na n-ésima geração termina quando a nova população P_{n+1} atinge o número total de indivíduos.

Figura 36 – Processo de evolução do algoritmo NSGA-II.



Fonte: Adaptada de Deb *et al.* (2000).

7.5 Anotação de tarefas sensíveis

Workflows científicos geralmente manipulam grande volume de dados na ordem de gigabytes e terabytes, e geram quantidade semelhante durante o processamento (DEELMAN; CHERVENAK, 2008). O uso de serviços de segurança em todos os dados pode se tornar inviável, incluindo os não sensíveis. Além disso, mesmo que os serviços de segurança usem algoritmos de complexidade computacional reduzida, isso não parece ser uma opção razoável. Assim, é essencial a investigação de abordagens que possam minimizar o risco e o custo dentro do prazo requerido para a execução de workflows.

No melhor de nosso conhecimento, há duas categorias de usuário no contexto do uso de sistemas gerenciadores de workflows: o desenvolvedor de workflows e o usuário final (BALASKO; FARKAS; KACSUK, 2013; PLANKENSTEINER *et al.*, 2013). O primeiro usa o conhecimento de cada tarefa para criar e compartilhar uma aplicação baseada em workflow. Além disso, o usuário desenvolvedor atua tanto como o administrador quanto como o cientista com habilidades de desenvolvimento. Por outro lado, o usuário final é aquele que exclusivamente importa e executa o workflow.

Semelhante às políticas de segurança introduzidas na Seção 7.3, a abordagem de anotação de tarefas ou *Scheduling Annotation of Sensitive Tasks*(SAST) também usa o conhecimento prévio sobre as tarefas para a minimização do risco de segurança. A anotação de tarefas sensíveis deve ser realizada na fase de modelagem do workflow e é representada nesta tese por um atributo booleano chamado *sensitive*. O formato DAX foi escolhido para descrever um workflow, considerando a mesma entrada utilizada pelo *Pegasus Workflow Management System* (DEELMAN *et al.*, 2015). A Listagem 3 representa uma descrição simplificada de um workflow que faz uso do atributo *sensitive* para identificar quais tarefas são sensíveis ou não. A descrição das tarefas nas linhas 4, 8 e 12 inclui o atributo *sensitive*, onde apenas a $tarefa$ 3 é definida como sensível.

Arquivo 3 – Exemplo de uma descrição de um workflow no formato DAX usando o atributo *sensitive* para anotação de tarefa.

```
1  <?xml encoding="UTF-8"?>
2  <adag xmlns="..." xsi:schemaLocation="..."
3        name="Example" jobCount="3" fileCount="4">
4   <job id="ID01" name="task1" sensitive="false" >
5     <uses file="load.db" link="input" size="4158867"/>
6     <uses file="ff.db" link="output" size="1558867"/>
7   </job>
8   <job id="ID02" name="task2" sensitive="false">
9     <uses file="ff.db" link="input" size="1558867"/>
10    <uses file="s1.txt" link="output" size="52837210"/>
11  </job>
12  <job id="ID03" name="task3" sensitive="true" >
13    <uses file="s1.txt" link="input" size="1558867"/>
14    <uses file="out.fl" link="output" size="391847"/>
15  </job>
16  <child ref="ID02">
17    <parent ref="ID01"/>
18  </child>
19  <child ref="ID03">
20    <parent ref="ID02"/>
21  </child>
22 </adag>
```

Fonte: Elaborada pelo autor.

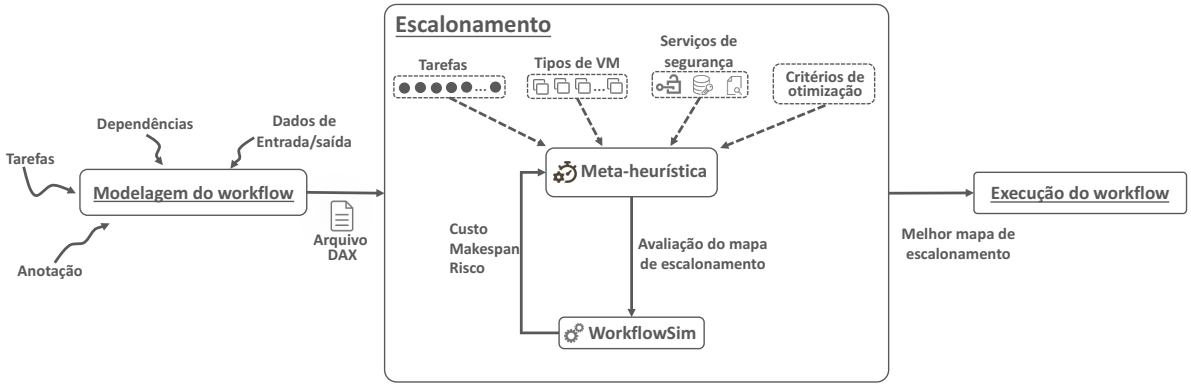
O escalonamento baseado em anotação de tarefas envolve três etapas: (1) modelagem do workflow; (2) otimização do escalonamento, e; (3) execução do workflow conforme apresentadas na Figura 37. Na etapa de modelagem são definidas as tarefas e suas dependências, os arquivos de entrada e saída de cada tarefa e as anotações de segurança de cada tarefa. Em seguida, a etapa de escalonamento inicialmente recebe parâmetros de entrada para o processo de otimização (meta-heurística) como o número de tarefas do workflow, os tipos de VMs disponíveis na nuvem, os serviços de segurança e os critérios e restrições de otimização.

Baseado nos parâmetros de entrada e na codificação de escalonamento apresentados a seguir na Seção 7.6, a meta-heurística produz iterativamente soluções que são submetidas ao WorkflowSim responsável pela sua decodificação, simulando a execução do mapa de escalonamento considerando os modelos do sistema (Capítulo 4) para avaliar a aptidão da solução. A avaliação da solução contempla o makespan, o custo e o risco de execução que são retornados à meta-heurística. Esse processo iterativo ocorre até que uma condição de parada (ex.: tempo) seja satisfeita. Ao fim da etapa de otimização de escalonamento, a solução contendo o melhor mapa de escalonamento é enviada ao módulo de execução de workflow.

7.6 Codificação do escalonamento

A codificação da solução é fundamental no processo de otimização baseado em meta-heurística. Arunarani, Manjula e Vijayan (2017) Li *et al.* (2016) consideram quatro parâmetros na codificação de cada tarefa: (a) tipo de VM; (b) autenticação; (c) verificação de integridade

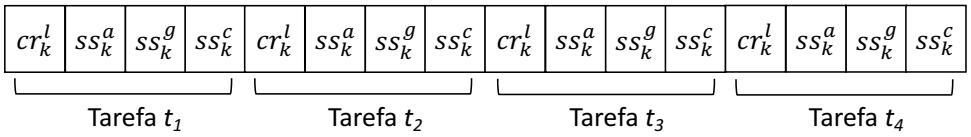
Figura 37 – Processo de otimização do escalonamento de tarefas do workflow apoiado pela anotação de tarefas.



Fonte: Elaborada pelo autor.

e; (d) confidencialidade. Logo, conforme explicado na Seção 6.2, o tamanho da codificação de um workflow com n tarefas é dado por $D = 4 \cdot n$. Por exemplo, um workflow com 4 tarefas tem 16 parâmetros, como mostra a Figura 38. Os parâmetros de 0 a 3 correspondem à tarefa 1, os parâmetros de 4 a 7 correspondem à tarefa 2 e assim por diante. Entretanto, essa codificação dá abertura à atribuição de serviços de segurança para tarefas com dados não sensíveis, enquanto nenhum serviço pode ser atribuído a tarefas que lidam com dados confidenciais.

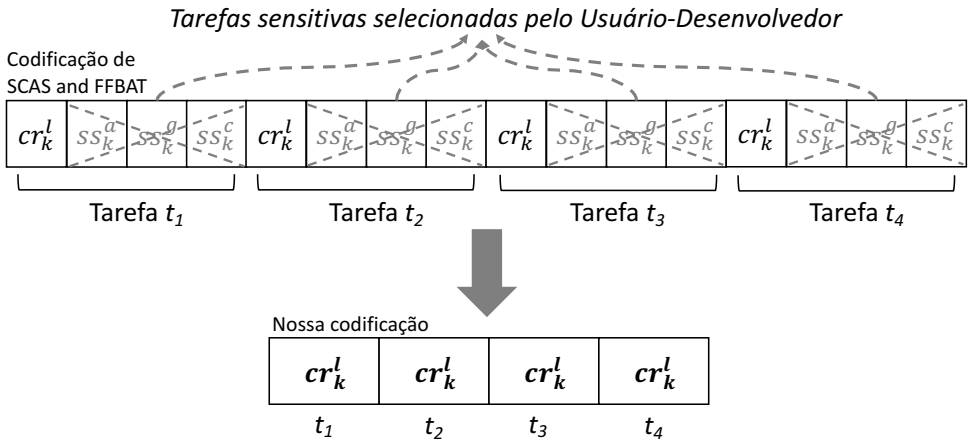
Figura 38 – Codificação adotada pela abordagem SCAS (LI *et al.*, 2016) e FFBAT (ARUNARANI; MANJULA; VIJAYAN, 2017).



Fonte: Elaborada pelo autor.

Para contornar tais problemas, a WS-TSP e a SAST utilizam o conhecimento prévio do Usuário-Desenvolvedor para definir quais tarefas são sensíveis ou não. Desta forma, a codificação otimiza exclusivamente o tipo de VM de cada tarefa, reduzindo a dimensão de codificação para $D = n$. A Figura 39 ilustra a codificação proposta. Por exemplo, a codificação de um workflow de 4 tarefas apresenta 4 parâmetros de otimização. O parâmetro 0 determina o tipo de VM à tarefa 1, o parâmetro 1 tem o tipo de VM atribuído à tarefa 2 e assim por diante. As tarefas sensíveis são protegidas de acordo com as anotações realizadas pelo Usuário-Desenvolvedor. Assim, as abordagens WS-TSP e a SAST evitam atribuir serviços de segurança a tarefas que manipulam dados não sensíveis, o que se traduz em melhor proteção às tarefas sensíveis e redução do makespan e custo de execução.

Figura 39 – Codificação utilizada na abordagem baseada em anotações.



Fonte: Elaborada pelo autor.

7.7 Decodificação

A decodificação de uma solução produz um mapa de escalonamento que é avaliado seguindo o Algoritmo 7. Primeiro, os tipos de VMs disponíveis $CR = \{cr_1, cr_2, ..., cr_m, ..., CR_k\}$ e o conjunto de mapeamento tarefa-recurso M são inicializados como vazio. O tempo de execução total E^t e o custo de execução total E^c são inicializados com valor 0. Em seguida, o horário de início de cada tarefa é calculado; se a tarefa t_i não possui nenhuma tarefa pai, então esta é considerada uma tarefa de entrada e o tempo de início é 0 (linha 4); senão, t_i recebe o tempo de execução mais longo de suas tarefas predecessoras $pred(t_i)$ (linha 6).

O tempo de transferência de dados entre as tarefas t_i e t_{i-1} também é considerado (linha 8). Em seguida, o tempo de execução da tarefa $P(t_i, cr_k^l)$ e o tempo de término da execução são calculados (linhas 9 e 10). O elemento *pos* consiste no vetor de parâmetros que contém os tipos de VMs endereçados para cada tarefa. O momento de início da locação da VM $LST(t_i, cr_k^{pos[i]})$ é dado pela atribuição do tempo de início da tarefa t_i (linha 11) e o momento da liberação da VM $LET(t_i, cr_k^{pos[i]})$ é calculado pela soma de $PT(t_i, cr_k^{pos[i]})$ e $LST(t_i, cr_k^{pos[i]})$.

No final da iteração de cada tarefa, o mapeamento tarefa-recurso $m(t_i, cr_k^{pos[i]})$ e o recurso r_i de cada tarefa t_i são respectivamente adicionados ao conjunto M e R. Após iterar sobre todas as tarefas do workflow, o custo total de execução E^c, o makespan E^t e o risco de segurança são calculados como mostrado nas linhas 16 e 17. Esse algoritmo é usado como base para calcular a função de aptidão da meta-heurística adotada.

7.8 Métricas de avaliação

Duas métricas foram estabelecidas para medir o risco de execução e o processamento desnecessário para proteger tarefas não sensíveis. O primeiro considera o risco percentual $Risk(t_i)$ de cada tarefa sensível t_i de todo o workflow, conforme definido pela Equação 7.2. Os

Algoritmo 7 – Algoritmo para decodificação de escalonamento.

função <SCHEDULINGDECODING
Init $R = M = \phi$ and $E^c = E^t = Risk(w) = 0$
para todo task t_i *in* T **faça**
se t_i não tem tarefas predecessoras **então**
$ST(t_i) \leftarrow 0$
senão
$ST_i \leftarrow max\{ET(t_j)|t_j \in pre(t_i)\}$
fim se
$TT_{i-1,i} \leftarrow I_i/B$
$P(t_i, cr_k^{pos[i]}) \leftarrow TT(t_i) + ET(t_i, cr_k^{pos[i]}) + SC(t_i)$
$ET(t_i) \leftarrow PT(t_i, cr_k^{pos[i]}) + ST(t_i)$
$LST(t_i, cr_k^{pos[i]}) \leftarrow ST(t_i)$
$LET(t_i, cr_k^{pos[i]}) \leftarrow PT(t_i, cr_k^{pos[i]}) + LST(t_i, cr_k^{pos[i]})$
$M \leftarrow m(t_i, cr_k^{pos[i]})\ R \leftarrow r_i$
fim para
$E^c \leftarrow \sum_{i=0}^{n-1} c_s^{pos[i]} \cdot \lceil LET(t_i, cr_k^{pos[i]}) - LST(t_i, cr_k^{pos[i]}) \rceil$
$E^t \leftarrow max\{ET(t_i)|t_i \in T\}$
$Risk(w) \leftarrow calcWorkflowRisk()$
fim função

Fonte: Elaborada pelo autor.

termos $pred(t_i)$ e $succ(t_i)$ representam o total de tarefas pai e filho de uma tarefa sensível t_i, e $sensitive_edges(t_w)$ é a soma das arestas de todas as tarefas que lidam com dados sensíveis. O risco geral de execução $Risk(w)$ é obtido pela soma do percentual de tarefas em risco $Risk(t_i)$ como mostrado na Eq. 7.3. A soma considera apenas as tarefas em que o atributo de ativação de segurança é igual a 1.

$$Risk(t_i) = \frac{pred(t_i) + succ(t_i)}{sensitive_edges(w)} \tag{7.2}$$

$$Risk(w) = 1 - \sum_{i=1}^{nSec} Risk(t_i) * sec(t_i) \tag{7.3}$$

A segunda métrica representa o percentual de tarefas não sensíveis protegidas por serviços de segurança (NSTS), desperdiçando tempo e custo com este tipo de processamento. O NSTS é calculado pela Equação 7.4. O termo $WfSecTasks$ representa o número de tarefas do workflow que precisam ser protegidas e $SecTasks$ consiste no número de tarefas sensíveis que receberam serviços de segurança.

$$NSTS = \frac{WfSecTasks - SecTasks}{WfSecTasks} \tag{7.4}$$

Assim, foi estabelecida uma função de *fitness* com o objetivo de minimizar o custo monetário E^c, o desperdício de processamento assegurando tarefas não-sensíveis $NSTS_{(w)}$ e o risco de segurança $Risk_{(w)}$ como mostrado na Equação 7.5. Os três pesos w_1, w_2 e w_3 são utilizados para a priorização dos critérios E^c, $Risk_{(w)}$ e $NSTS_{(w)}$, respectivamente.

$$Fitness = w_1\ E^c + w_2\ (E^c\ Risk_{(w)}) + w_3\ (E^c\ NSTS_{(w)}) \tag{7.5}$$

Além disso, o tempo de execução total E^t não pode exceder o *deadline DL*. Assim, esta estratégia pode ser formalizada como:

$$otimizacao = \begin{cases} Minimizar : Fitness \\ Sujeito\ a : E^t \leq DL \end{cases}$$

7.9 Considerações finais

Este capítulo apresentou as abordagens WS-TSP e SAST que envolvem a proteção de tarefas sensíveis e a otimização do escalonamento considerando o *overhead* adicional no tempo de execução do workflow. Além disso, são propostas duas métricas para a avaliação do critério de segurança que levam em conta o risco percentual de execução e o desperdício de processamento para assegurar tarefas não-sensíveis do workflow. No próximo capítulo, as abordagens WS-TSP e SAST são avaliadas e comparadas à abordagem comumente referenciada na literatura.

CAPÍTULO

8

AVALIAÇÃO EXPERIMENTAL

8.1 Considerações iniciais

As abordagens WS-TSP e SCAS são avaliadas em um ambiente de simulação com base no *WorkflowSim* e a extensão *WfSecurity* introduzida no Capítulo 5. Um cluster foi utilizado nos experimentos para reduzir o tempo de execução dos experimentos que contaram com diferentes parâmetros de execução e múltiplas repetições para validação estatística. Essas abordagens também são avaliadas considerando otimização mono e multiobjetivo.

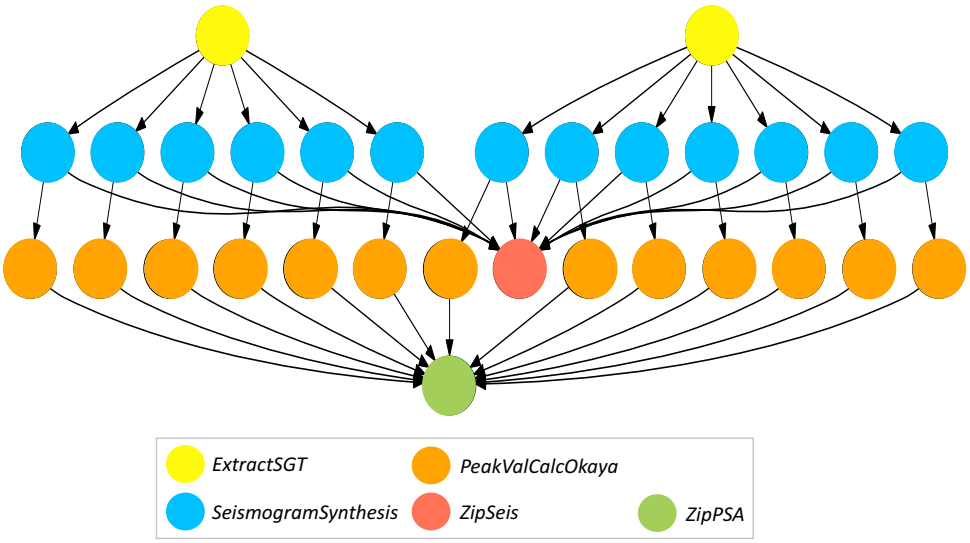
8.2 Caracterização da carga de trabalho

As simulações de execução de workflows adotaram o simulador *WorkflowSim* utilizando a extensão proposta nessa tese denominada de *WfSecurity* (Capítulo 5). Workflows do mundo real de diferentes áreas do conhecimento como CyberShake, Montage, Epigenomics e SIPHT foram usados.

O Cybershake é uma aplicação para avaliar o risco de terremotos por meio de uma análise probabilística, cuja topologia é representada na Figura 40. Esse workflow executa uma grande quantidade de dados. Os dados *Strain Green Tensor* (SGT) gerados a partir de simulações são mantidos na forma de grandes arquivos SGT que quantificam a relação entre o movimento de uma pequena região em relação às demais localidades. Além dos dados do SGT, há uma coleção de futuras falhas de rupturas que são combinados para estimar o risco sísmico no local. As tarefas *ExtractSGT* extraem os SGTs pertencentes a uma determinada ruptura de arquivos SGT maiores, de modo que tal processamento possa ser realizado em paralelo. Em seguida, sismogramas são gerados para cada variação de ruptura pelas tarefas *SismogramaSynthesis*, caracterizadas pelo processamento intensivo. Os valores de intensidade de pico são calculados pelas tarefas *PeakValueCalcOkaya* para cada sismograma. Os sismogramas e intensidades de pico resultantes

são coletados e compactados pelas tarefas *ZipSeismograms* e *ZipPeakSA* (tarefas de agregação).

Figura 40 – Workflow Cybershake. Círculos representam as diferentes tarefas do workflow. As setas representam a dependência de dados entre as tarefas.

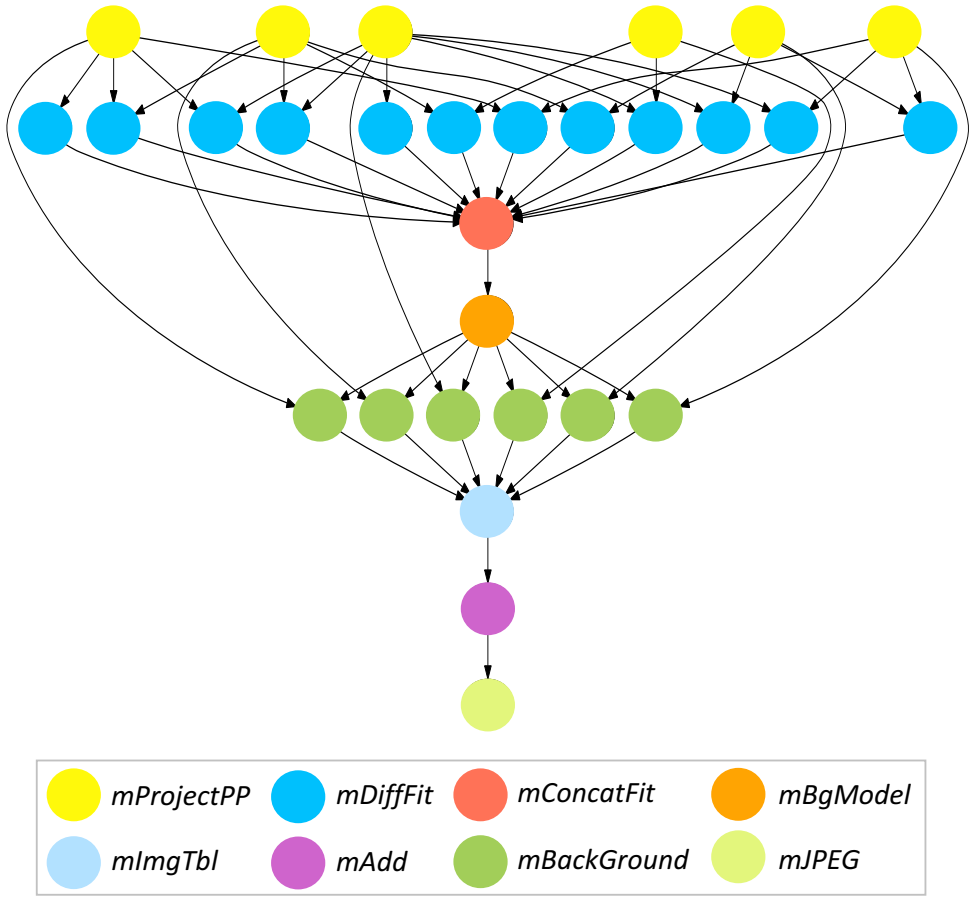


Fonte: Deelman *et al.* (2018).

O Montage é um workflow para a construção de um mosaico de imagens espaciais e é qualificado como uma aplicação I/O bound devido às tarefas de leitura de imagens de entrada. A topologia desse workflow é apresentada na Figura 41. O número de tarefas desse workflow depende da quantidade de imagens para a construção do mosaico. No nível superior do workflow Montage estão as tarefas *mProjectPP* que realizam o pré-processamento das imagens de entrada. O número de tarefas *mProjectPP* é igual ao número de imagens de entrada. A diferença para cada par de imagens sobrepostas é calculada pelas tarefas *mDiffFit*, cujo número varia de acordo com as imagens de entrada que se sobrepõem. Em seguida, as imagens são agregadas pela tarefa A *mConcatFit* que as submete para uma correção (*mBgModel*) que busca obter um bom ajuste global dos pares. Uma correção do segundo plano é aplicada em cada imagem pelas tarefas *mBackground*. A tarefa *mImgTbl* agrega metadados de todas as imagens e cria uma tabela que poderá ser usada futuramente por outros workflows. Múltiplos níveis de tarefas *mImgTbl* e *mAdd* podem ser usados em grandes workflows. Finalmente, a tarefa *mShrink* reduz a imagem e a submete para a tarefa *mJPEG* para compatá-la.

O Epigenomics realiza o mapeamento do estado epigenético das células humanas. É caracterizado por vários *pipelines* que calculam pedaços independentes de dados em paralelo. O Epigenomics é formado por oito categorias de tarefas distintas conforme representados na Figura 42. *fastQSplit* é a tarefa de entrada, que divide a sequência de DNA em vários trechos durante o processo de análise genética. Posteriormente, *filterContams* remove os dados ruidosos e contaminados de cada fragmento para que possa ser convertido em um formato reconhecível pelo utilitário *sol2sanger*. Os dados são convertidos para o formato *fastQ* pela tarefa *fast2bfq* para otimizar o tempo de processamento e reduzir o uso de disco. Então, as sequências restantes

Figura 41 – Workflow Montage. Círculos representam as diferentes tarefas do workflow. As setas representam a dependência de dados entre as tarefas.

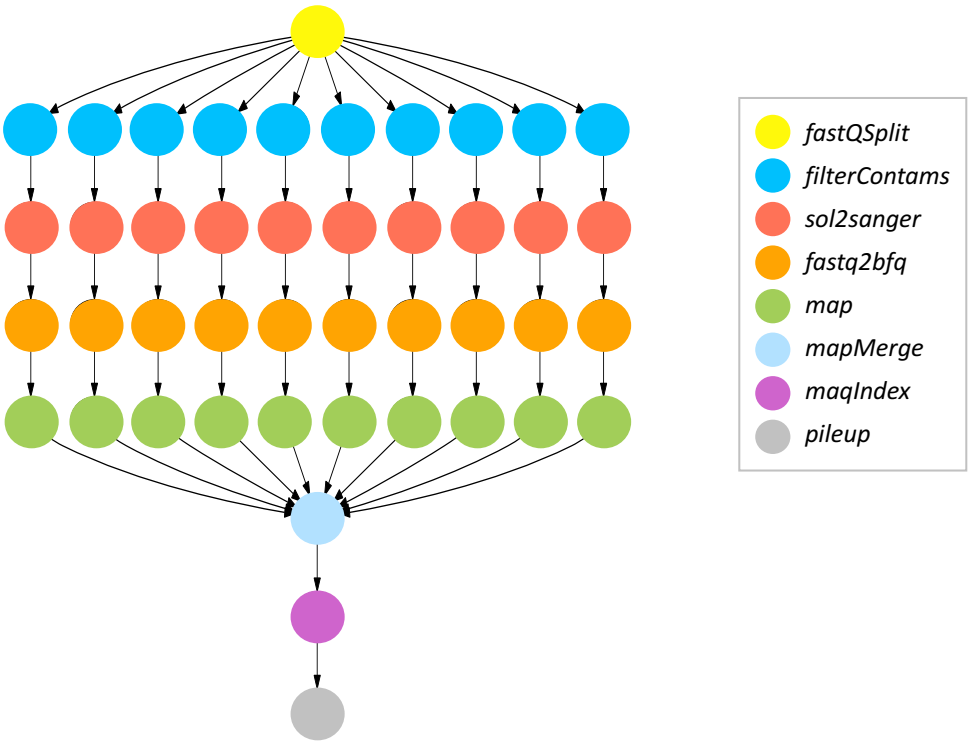


Fonte: Deelman *et al.* (2018).

são alinhadas pela tarefa *map* que apresenta o maior uso de CPU (aprox. 96%). Os resultados de mapas individuais são combinados por meio das tarefas *mapMerge* e *mapIndex*. Finalmente, a tarefa *pileup* estrutura os dados de saída de *mapMerge* e *mapIndex*.

O SIPHT é uma aplicação que pesquisa por pequenos RNAs não traduzidos (sRNAs) (Figura 43). Todos os workflows SIPHT têm estrutura quase idêntica à da Figura 43. A única diferença entre duas instâncias desse workflow é o número de tarefas *Patser* que examinam sequências com matrizes de pontuação. O número de tarefas *Patser* depende de entradas que descrevem os locais de ligação do fator de transcrição (TFBSs). Os resultados dessas tarefas *Patser* são concatenados pela tarefa *Patser_Concate*. Há várias tarefas BLAST no workflow que comparam diferentes combinações de sequências. A tarefa *Blast* e *Blast_QRNA* operam em todas as possíveis regiões inter-genéricas (IGRs) de outros "replicons". Mesmo que não aparentes na Figura 43, as tarefas *BLAST* operam em centenas de arquivos de dados. Há três tarefas no workflow que pesquisam os terminadores de transcrição - *FindTerm*, *RNAMotif* e *Transterm*. A previsão do sRNA é executada pela tarefa *SRNA* que opera nas saídas das tarefas acima, bem como na saída da tarefa *Blast*. A saída deste trabalho é usada pelas outras tarefas *BLAST*. A

Figura 42 – Workflow Epigenomics. Círculos representam as diferentes tarefas do workflow. As setas representam a dependência de dados entre as tarefas.

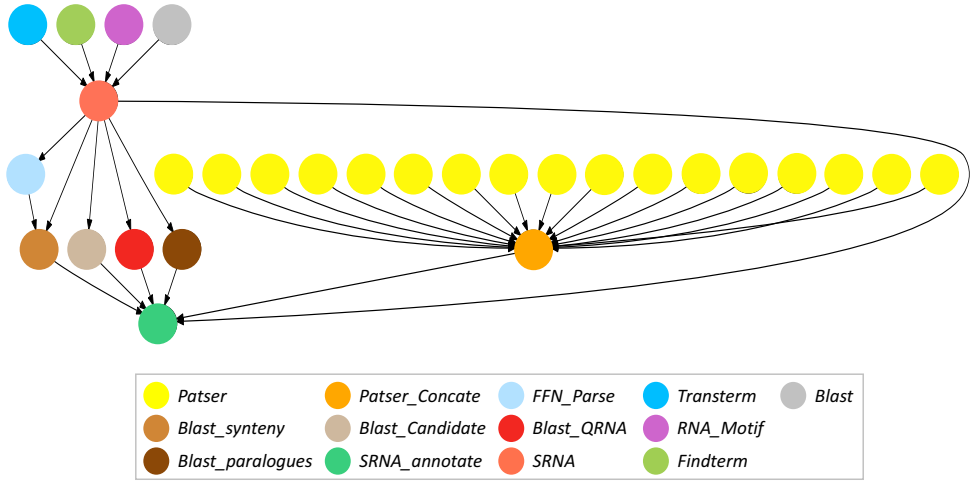


Fonte: Deelman *et al.* (2018).

tarefa *SRNA_Annotate* anota os "locos" de sRNA candidatos que foram encontrados bem como sua conservação em outras cepas bacterianas, sua associação com TFBSs e sua homologia com outros sRNAs previamente identificados. De todas as tarefas do workflow, as tarefas *BLAST* que comparam sequências entre vários pares de réplicas são as mais computacionalmente intensivas. A outra tarefa mais computacionalmente intensiva é a de previsão de sRNA, seguida pelas tarefas que identificam os terminadores de transcrição.

Um resumo das estatísticas do perfil de execução para todos os workflows que adotamos em simulações é apresentado em (JUVE *et al.*, 2013b). A escolha desses workflows para simulação foi realizada com base nas diferentes características de cada workflow. Entre eles estão um workflow pequeno (SIPHT), workflow vinculado a uso intensivo de CPU (Epigenomics), workflow de E/S (Montage), workflow com uso intensivo de dados e recursos (CyberShake). Além disso, esses workflows são largamente utilizados na avaliação de algoritmos de escalonamento de workflows como em (BITTENCOURT; MADEIRA, 2011; ABRISHAMI; NAGHIBZADEH; EPEMA, 2013; LI *et al.*, 2016; ZENG; VEERAVALLI; LI, 2015).

Figura 43 – Workflow SIPHT. Círculos representam as diferentes tarefas do workflow. As setas representam a dependência de dados entre as tarefas.



Fonte: Deelman *et al.* (2018).

Tabela 14 – Visão geral dos perfis de execução dos workflows. Essa tabela resume as informações do perfil em nível de workflow.

Workflow	**Tarefas**	**CPU (h)**	**E/S leitura (GB)**	**E/S escrita (GB)**	**Pico de memória**	**Utilização de CPU (%)**
Cybershake	815,823	9,192.45	217,369.37	920.93	1,870.38	90.89
Montage	10,429	4.93	146.01	49.93	16.77	31.04
Epigenomics	529	7.45	24.14	5.36	197.47	95.91
SIPHT	31	1.23	1.7	1.48	116.38	93.37

Fonte: Adaptada de Juve *et al.* (2013b).

8.3 Ambiente de execução

Todos os experimentos utilizaram o cluster Halley do Laboratório de Sistemas Distribuídos e Programação Concorrente (LaSDPC) do Insituto de Ciências Matemáticas e de Computação (ICMC) da Universidade de São Paulo (USP). O cluster Halley conta com 13 nós de processamento, sendo que cada nó contém as especificações descritas na Tabela 15.

O framework *WorkflowSim* foi usado para simular um ambiente em nuvem onde as VMs estão instanciadas em um mesmo datacenter e conectadas através de uma rede de 1Gbit. As características das instâncias de VMs oferecidas pela Amazon EC2 bem como o seu modelo de cobrança foram considerados na avaliação conforme apresentado na Tabela 16.

A restrição do *deadline* foi arbitrariamente escolhida como a média do tempo de execução mais rápido e mais lento do workflow. O primeiro é calculado a partir do tempo de execução do workflow que inclui o tipo de VM mais barato, ou seja, mais barato para todas as tarefas, enquanto a execução mais rápida atribui o tipo de VM mais rápida para todas as tarefas. Em ambos os casos, todas as tarefas foram asseguradas para que sempre haja uma solução que possa

Tabela 15 – Especificações dos nós do *Cluster* Halley

Cluster Halley	
Número de nós	13
Processador	Intel Core I7 Processor LGA -1150 – 4790 3.60GHZ DMI 5GT/S 8MB
RAM	32 GB RAM DDR3 Corsair Vegeance
Hard Drives	HD 2TB Seagate Sata III 7200RPM
GPU	VGA Nvidia GTX 650 – 1GB
Motherboard	Gigabyte GA-Z97X-SLI ATX
Fonte de Alimentação	ATX 650W Real
Sistema Operacional	Linux Ubuntu 14.04.4 LTS

Fonte: Dados da pesquisa.

Tabela 16 – Configurações de máquinas virtuais consideradas no processo de otimização do escalonamento das tarefas do workflow.

VM	**# vCPU**	**Mflops**	**Memory (GB)**	**Storage (GB)**	**Hour cost (U$)**
Tipo 1	2	8800	4	32	0.105
Tipo 2	4	17600	8	80	0.210
Tipo 3	8	35200	15	160	0.420
Tipo 4	16	70400	30	320	0.840
Tipo 5	32	140800	60	640	1.680

Fonte: Dados da pesquisa.

satisfazer a restrição de *deadline*.

As meta-heurísticas utilizaram os parâmetros apresentados na Tabela 17. Os parâmetros do AG e do MPGA foram definidos empiricamente por meio de testes anteriores. Entretanto, a maioria dos parâmetros dos algoritmos genéticos partem das melhores configurações descritas na literatura, como em (TOLEDO *et al.*, 2009; ARANTES *et al.*, 2017). Os parâmetros do PSO também foram definidos baseando-se no trabalho de (LI *et al.*, 2016).

Tabela 17 – Parâmetros usados nas meta-heurísticas.

AG		**MPGA**		**PSO**	
Tamanho da população	100	**Número de populações**	3	**Partículas**	50
Método de seleção	Torneio	**Número de indivíduos**	7	**Iterações**	100
Tamanho do torneio	3	**Taxa de crossover**	5.0	φ_1	2.05
Taxa de mutação	0.1	**Taxa de mutação**	0.7	φ_2	2.05
		Estrutura	Árvore	ω	0.5
		Ramificação	2		
Operador de inicialização	bestSec				
Operadores de crossover	crossUniform, crossAverage, crossBLX				
Operadores de mutação	mutLimit, mutStrong				

Fonte: Dados da pesquisa.

Foi assumido que os métodos de segurança listados na Tabela 18 estão disponíveis para o serviço de autenticação, verificação de integridade e confidencialidade. A função de sobrecarga

dos serviços de integridade e segurança é expressa pela Equação 6.7 no pressuposto de que $\beta^a = 3$, $\beta^g = 4$ e $\beta^c = 1$. Como um serviço de autenticação independe da quantidade de dados, o *overhead* de um serviço de autenticação pode ser representado conforme a Equação 6.8.

Tabela 18 – Lista de algoritmos de segurança disponíveis para a simulação do overhead de serviços de segurança.

Security Service	Algorithm	Security level	Overhead (kB/ms)
Encryption	SEAL	0.08	168.75
	RC4	0.14	96.43
	Blowfish	0.36	37.50
	Knufu/Khafre	0.40	33.75
	RC5	0.46	29.35
	Rijndael	0.64	21.09
	DES	0.90	15.00
	IDEA	1.00	13.50
Integrity verification	MD4	0.18	23.90
	MD5	0.26	17.09
	RIPEMD	0.36	12.00
	RIPEMD128	0.45	9.73
	SHA1	0.63	6.88
	RIPEMD160	0.77	5.69
	TIGER	1.00	4.36
Authentication	HMAC_MD5	0.55	90 (ms)
	HMAC_SHA_1	0.91	148 (ms)
	CBC_MAC_AES	1.00	163 (ms)

Fonte: Adaptada de Xie e Qin (2006b).

8.4 Desenho de experimentos - WS-TSP

Uma das práticas de simulação computacional é estimar as variações nos parâmetros do sistema que representem a aleatoriedade do mundo real para evitar o enviesamento dos resultados. Nesse sentido, diferentes fatores e níveis aderentes ao problema dessa tese foram definidos para a avaliação da abordagem WS-TSP. A carga de trabalho é um fator a ser considerado na avaliação da abordagem WS-TSP e é realizada considerando a variação dos workflows (níveis). Quatro workflows (Cybershake, Montage, Epigenomics e SIPHT) foram considerados na tentativa de representar diferentes classes de problemas como mostrado na Tabela 19.

Tabela 19 – Descrição da carga de trabalho adotada na avaliação da abordagem WS-TSP. Fator: "Workflow" Níveis: Cybershake, Montage, Epigenomics e SIPHT.

Workflow	Classe
Cybershake	Uso intensivo de dados e recursos
Montage	Predominância de operações de E/S
Epigenomics	Uso intensivo de CPU
SIPHT	Pequeno com aproximadamente 30 tarefas.

Fonte: Juve *et al.* (2013b).

Workflows podem apresentar diferentes tipos de tarefas que manipulam dados sensíveis. Desse modo, o fator "tipo de tarefa" foi considerado de modo a avaliar seis tipos de tarefas

(níveis) descritas na Tabela 20.

Tabela 20 – Descrição dos tipos de tarefas em que há dados sensíveis na avaliação da abordagem WS-TSP. Fator: "Tipo de tarefa". Níveis: No, Entry/End, Gather, Scatter, Gather/Scatter e All.

Tipo de tarefa	**Descrição**
No	Nenhuma tarefa do workflow é sensível.
Entry/End	As tarefas de entrada e saída contém dados sensíveis.
Gather	As tarefas de agrupamento de dados manipulam dados sensíveis.
Scatter	As tarefas de dispersão de dados manipulam dados sensíveis.
Gather/Scatter	Tarefas que realizam agrupamento e dispersão de dados sensíveis
All	Todas as tarefas do workflow processam dados sensíveis.

Fonte: Dados da pesquisa.

Para fim de comparação de desempenho do WS-TSP, o fator "abordagem" que considera quatro níveis: WS-TSP, SCAS, Slowest e Fastest descritas na Tabela 21.

Tabela 21 – Descrição das abordagens de escalonamento. Fator: "Abordagem". Níveis: WS-TSP, SCAS, Slowest e Fastest.

Abordagem	**Descrição**
WS-TSP	Proposta nessa tese. Fundamenta-se na ideia do conhecimento prévio dos requisitos de segurança das tarefas em que escolhe-se a política adequada para proteção das tarefas sensíveis. Além disso, usa Algoritmo Genético Multi-População para otimizar os tipos de VMs para cada tarefa.
SCAS	Proposta em (LI *et al.*, 2016) em que serviços de segurança são aplicados em tarefas escolhidas com base na otimização de enxame de partículas.
Slowest	Considera o tipo de VM mais lenta e barata. Emprega serviços de segurança em todas as tarefas do workflow.
Fastest	Considera o tipo de VM mais rápida e cara. Emprega serviços de segurança em todas as tarefas do workflow.

Fonte: Dados da pesquisa.

O desenho de experimentos para a avaliação dos algoritmos desenvolvidos segue os padrões de avaliação de desempenho de sistemas computacionais (JAIN, 1991). Três fatores com seus respectivos conjunto de níveis foram definidos como mostrado na Tabela 22.

Tabela 22 – Desenho de experimentos para avaliação da abordagem WS-TSP.

Fator	**Nível 1**	**Nível 2**	**Nível 3**	**Nível 4**	**Nível 5**	**Nível 6**
Workflow	Cybershake	Montage	Epigenomics	SIPHT		
Tipo de tarefa	No	Entry/End	Gather	Scatter	Gather/Scatter	All
Abordagem	WS-TSP	SCAS	Slowest	All		

Fonte: Dados da pesquisa.

As abordagens WS-TSP e SCAS foram comparadas entre si e com as abordagens Slowest e Fastest que servem como parâmetro balizador. Todos essas abordagens foram implementadas no framework *WorkflowSim*. A carga de trabalho contempla workflows do mundo real: Cybershake, Montage, Epigenomics e SIPHT. Os quatro workflows foram gerados por meio da ferramenta WorkflowGenerator (BHARATHI *et al.*, 2008). Esses quatro workflows foram definidos por apresentarem diferentes características de carga como descrito na Seção 8.2. Para avaliar o desempenho das abordagens, seis configurações foram definidas para estabelecer quais tarefas manipulam dados sensíveis: *No*, *Gather*, *Scatter*, *Gather*/*Scatter*, *Entry*/*End* e *All*. A combinação desses fatores para os níveis estabelecidos é apresentada na Tabela 23. As variáveis de saída são o makespan, custo monetário, o risco de segurança e o desperdício de processamento assegurando tarefas não sensíveis. As abordagens WS-TSP e SCAS foram executadas 33 vezes para cada configuração para que haja representatividade estatística.

Tabela 23 – Cenários de avaliação.

Experimento	**Abordagem**	**Workflow**	**Tipo de tarefa**
1	Todas	Cybershake	No
2	Todas	Cybershake	Entry/End
3	Todas	Cybershake	Gather
4	Todas	Cybershake	Scatter
5	Todas	Cybershake	Gather/Scatter
6	Todas	Cybershake	All
7	Todas	Montage	No
8	Todas	Montage	Entry/End
9	Todas	Montage	Gather
10	Todas	Montage	Scatter
11	Todas	Montage	Gather/Scatter
12	Todas	Montage	All
13	Todas	Epigenomics	No
14	Todas	Epigenomics	Entry/End
15	Todas	Epigenomics	Gather
16	Todas	Epigenomics	Scatter
17	Todas	Epigenomics	Gather/Scatter
18	Todas	Epigenomics	All
19	Todas	SIPHT	No
20	Todas	SIPHT	Entry/End
21	Todas	SIPHT	Gather
22	Todas	SIPHT	Scatter
23	Todas	SIPHT	Gather/Scatter
24	Todas	SIPHT	All

Fonte: Dados da pesquisa.

8.5 WS-TSP vs. SCAS

8.5.1 Avaliação Mono-Objetivo

Os resultados obtidos utilizando otimização mono-objetivo serão apresentados a seguir. Nossa abordagem utiliza o algoritmo genético com múltiplas populações (MPGA) como método de otimização. Comparamos o algoritmo WS-TSP ao SCAS introduzido em (LI *et al.*, 2016) em relação ao makespan, custo, risco de segurança e desperdício com a proteção de dados não sensíveis.

Comparação do makespan e custo

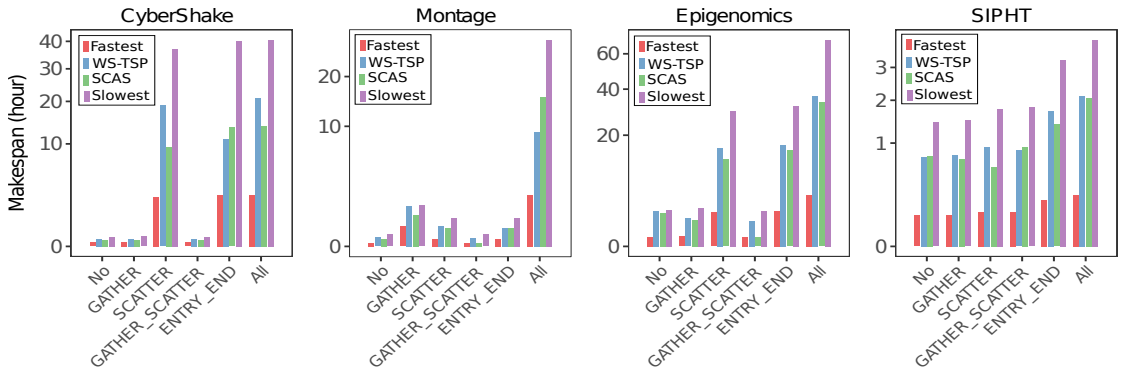
As Figuras 44 e 45 mostram os resultados da otimização. O WS-TSP encontrou melhores mapeamentos tarefa-tipoVM do que o SCAS. O WS-TSP minimizou o custo de execução próximo à configuração *Slowest* e também reduziu o makespan em pelo menos metade da configuração *Slowest*. Em alguns casos, o SCAS encontrou escalonamentos que fazem o makespan um pouco menor que o WS-TSP, mas com custos mais altos. Os escalonamentos encontrados pelo SCAS resultaram em processamento desnecessário para assegurar tarefas não-sensíveis e exigiram tipos de VMs com maior capacidade de processamento, resultando em maior custo de execução.

Os resultados levam em consideração execuções usando o tipo mais rápido de VM (Fastest) e o tipo mais lento (Slowest), WS-TSP e SCAS. Como esperado, as VMs mais rápidas apresentaram o menor makespan para todos os workflows, independentemente das políticas de seleção de tarefas (Figura 44). No entanto, o custo de execução excede $1500 (Figura 45). Por outro lado, as execuções com VMs mais lentas exigiram um orçamento menor, mas demandaram um longo período de execução.

A redução do makespan nos workflows CyberShake e Montage não é significativa para execuções usando a política *No* em instâncias de VM mais rápidas. Isso é explicado, pois os dois workflows concentram a maior parte da utilização de CPU em poucas tarefas. Assim, instâncias mais rápidas em todas as tarefas não garantem makespan melhores. O Epigenomics apresenta o maior makespan em VMs mais lentas devido a presença de mais tarefas que fazem uso intensivo de CPU do que Cybershake e Montage. A política *All* aplica serviços de segurança a todas as tarefas do workflow e, consequentemente, produz o maior makespan em todos os cenários. Essa política apresenta a maior utilização de VMs e é a mais beneficiada pelas VMs mais rápidas para processar os serviços de segurança.

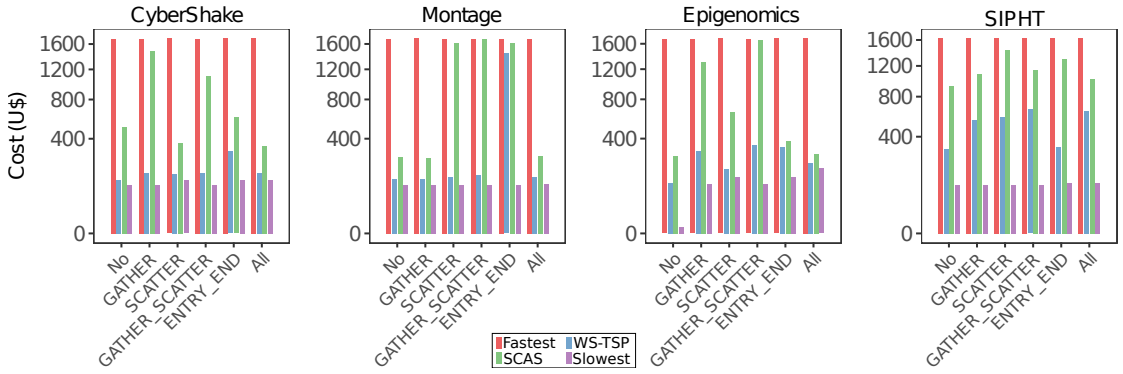
A política *Gather* aumentou o *overhead* nos workflows Montage, Epigenomics e SIPHT, mas não no CyberShake. O motivo é que o CyberShake tem duas tarefas de coleta, mas ambas envolvem poucos dados a serem protegidos, resultando no baixo *overhead*. A política *Scatter* influenciou substancialmente no CyberShake, pois cada tarefa no segundo nível manipula uma grande quantidade de dados (aprox. 547MB). O Epigenomics tem apenas uma tarefa de dispersão

Figura 44 – Makespan de diferentes políticas de seleção de tarefas em workflows usando as abordagens Fastest, Slowest, SCAS e WS-TSP.



Fonte: Elaborada pelo autor.

Figura 45 – Custo para diferentes políticas de seleção de tarefas e workflows usando as abordagens Fastest, Slowest, SCAS e WS-TSP.



Fonte: Elaborada pelo autor.

que lê 242MB de entrada e os divide em vários arquivos de saída. Os serviços de segurança aumentaram ligeiramente o *overhead* devido essa única tarefa de dispersão que processa 48MB de dados de entrada.

Considerando os quatro workflows, somente o SIPHT tem uma tarefa de coleta/dispersão que estendeu levemente o makespan. As políticas *Entry_End* e *All* aumentaram o makespan no CyberShake, possivelmente porque as tarefas *Entry_End* lidam com a maioria dos dados do workflow. *Entry_End* também prolongaram o makespan da execução do SIPHT. E, de modo geral, a política *All* apresentou o maior makespan usando VMs mais lentas conforme o esperado.

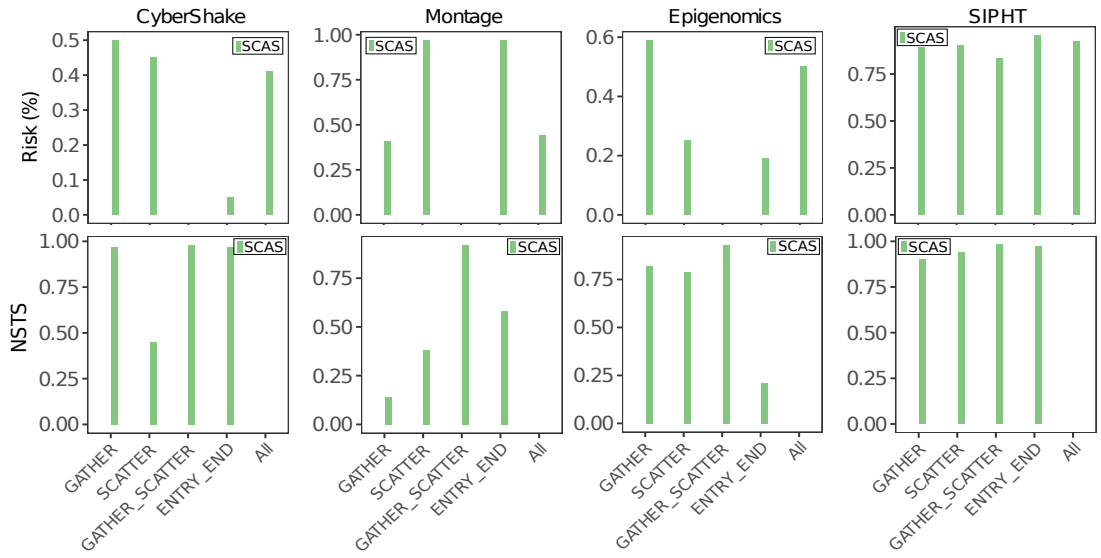
Risco de ataques

A abordagem WS-TSP consiste em cinco políticas que podem ser selecionadas no processo de modelagem do workflow considerando os requisitos de segurança das tarefas. Assumimos que o usuário conhece a semântica das tarefas do workflow para selecionar a política que possa abranger as tarefas sensíveis. O SCAS codifica quatro atributos por tarefa: tipo de VM, autenticação, verificação de integridade e confidencialidade. Para comparar o WS-TSP ao

SCAS, simplificamos a codificação do SCAS para dois atributos (instância da VM e ativação de segurança) para cada tarefa. O atributo de ativação de segurança corresponde aos três serviços de segurança (autenticação, verificação de integridade e criptografia). Ele pode assumir os valores 0 e 1, em que 0 não define nenhum serviço de segurança para a tarefa e 1 aplica os três serviços de segurança à tarefa.

Os resultados mostram que o WS-TSP é capaz de lidar com todas as tarefas que envolvem dados sensíveis conforme mostrado na Figura 46. Isso ocorre porque o conhecimento prévio de tarefas sensíveis permite a seleção da política adequada. De outro modo, o SCAS deixou tarefas sensíveis expostas a riscos durante sua execução. Pode-se observar que os dados de tarefas *gather*, *scatter*, *entry*/*end*, *all* ficaram parcialmente vulneráveis a ataques maliciosos. Os workflows CyberShake, Montage e Epigenomics não têm tarefas *gather*/*scatter*; então o risco de segurança é 0.

Figura 46 – Percentual de risco de execução para diferentes políticas de seleção de tarefas e workflows, considerando o algoritmo SCAS.



Fonte: Elaborada pelo autor.

A Figura 46 mostra o processamento desnecessário assegurando tarefas não sensíveis. Considerando que a política segurança foi escolhida corretamente conforme os requisitos de cada tarefa, o WS-TSP não apresentou risco de deixar tarefas sensíveis desprotegidas e também evitou o desperdício de processamento (NSTS) aplicando serviços de segurança em tarefas não sensíveis. Em contraste, o SCAS otimizou a segurança com base em um índice geral de segurança, mas por não considerar os requisitos de segurança de cada tarefa deixou tarefas sensíveis expostas, aumentando o risco de vazamento de dados. Além disso, o SCAS desperdiçou processamento aplicando serviços de segurança a tarefas com dados não sensíveis, exceto quando todas as tarefas do workflow eram sensíveis conforme a métrica NSTS apresentada na Figura 46. No cenário *All* em que todas as tarefas manipulam dados sensíveis, qualquer esquema de alocação de segurança não resultará em desperdício de processamento com serviços de segurança.

Tempo de resposta da otimização

O tempo de resposta do WS-TSP também foi avaliado. O algoritmo foi executado 33 vezes para cada configuração de avaliação por 15 minutos. Os valores médios de *fitness* foram capturados em intervalos de 5 segundos durante a execução. A Figura 47 apresenta o tempo de resposta dos workflows sob diferentes políticas de seleção. O WS-TSP conseguiu encontrar esquemas razoáveis de tarefa-VMs para todas as políticas dos workflows CyberShake, Montage e SIPHT. Porém, a política $Entry/End$ não apresentou desempenho semelhante às outras no workflow Montage. Esse resultado pode estar associado ao fato de que apenas uma tarefa ($mAdd$) é executada por aproximadamente 28 horas, enquanto as demais ($\sim$1000 tarefas) levaram apenas alguns minutos de execução. Possivelmente, os operadores propostos não foram capazes de produzir indivíduos em que a tarefa *mAdd* fosse associada a uma VM rápida, enquanto as demais ficariam atribuídas a um tipo de VM mais lenta e barata.

Por outro lado, no workflow CyberShake, a política $Entry/End$ poderia obter uma otimização melhor, pois há várias tarefas $entry$. O Epigenomics apresentou a maior variabilidade de fitness durante o processo de otimização. Isso sugere que a grande quantidade de tarefas que exigem uso intensivo de CPU e o alto nível de paralelismo contribuíram para encontrar melhores escalonamentos tanto para a política $Entry/End$ quanto para All usando os operadores propostos.

Figura 47 – O tempo de resposta de diferentes políticas de seleção de tarefas.

Fonte: Elaborada pelo autor.

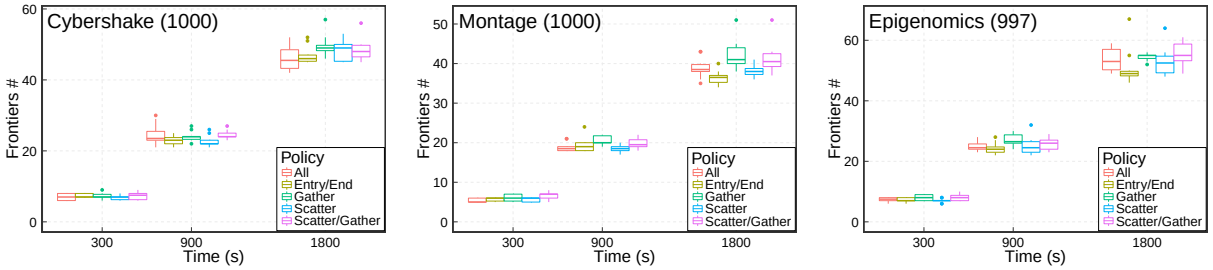
8.5.2 Otimização multi-objetivo

Nesta seção, são apresentados os parâmetros para a avaliação do algoritmo multi-objetivo NSGA-II na abordagem WS-TSP. Consideramos três workflows: Cybershake, Montage e Epigenomics. A abordagem WS-TSP multi-objetivo foi executada para os três workflows e as seis políticas de seleção de tarefas. O tamanho da população, o tamanho do torneio e a taxa de mutação foram definidos em 30, 3 e 0.1, respectivamente. A condição de parada foi ajustada para 300, 600, 900, 1200, 1500 e 1800s para avaliar as fronteiras de Pareto encontradas ao longo do

tempo. A população foi inicializada usando o operador *bestWorstMachines* e *randMachines*. Os experimentos foram repetidos 30 vezes para cada configuração - workflow, política de seleção de tarefas e condição de parada.

Os resultados são apresentados de duas maneiras. A primeira analisa o número de fronteiras retornadas durante o processo de otimização para cada configuração. O segundo apresenta soluções de Pareto ótimas a partir de diferentes tempos de otimização.

Figura 48 – Fronteiras de Pareto encontradas em diferentes condições de parada.



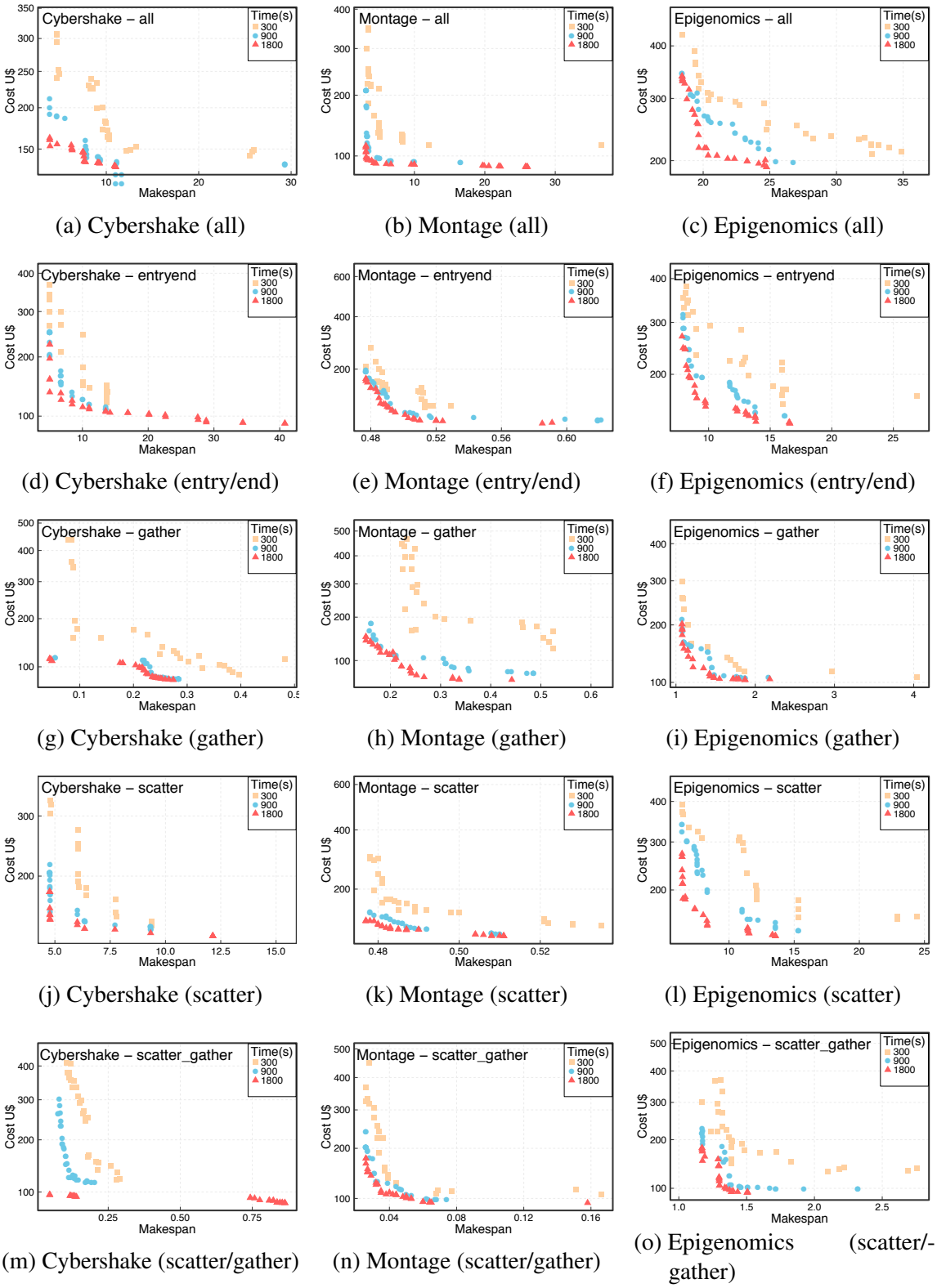
Fonte: Elaborada pelo autor.

O número de fronteiras foi avaliado para identificar a capacidade do NSGA-II encontrar soluções em diferentes tempos de execução. A Figura 48 indica oito fronteiras encontradas em aproximadamente 5 minutos para os workflows. Após 15 minutos, 20-25 fronteiras foram identificadas com uma grande variação no número de soluções. Essa variação aumenta para um tempo de execução maior (30 minutos). O número de fronteiras é semelhante às diferentes políticas de seleção de tarefas nos três workflows. Isso é explicado pelo mesmo nível de complexidade ao procurar por solução, independentemente da política de seleção de tarefa. Os *outliers* aparentemente crescem conforme a condição de parada é estendida.

A Figura 49 apresenta as Fronteiras de Pareto das diferentes políticas de seleção de tarefas referentes ao makespan e ao custo com as diferentes condições de parada. Todas as soluções encontradas atendem o compromisso (*deadline*) de execução conforme estabelecido na Seção 8.3, pois a função objetivo penaliza as soluções que não satisfazem o *deadline*. Nesse sentido, qualquer uma das soluções escolhidas satisfazem o *deadline*, cabendo ao usuário optar por uma solução que demande menor tempo ou menor custo de execução. Em geral, todos os workflows e políticas de seleção de tarefas melhoraram as fronteiras de Pareto durante todo o tempo de execução. No entanto, cada política de seleção apresentou comportamentos específicos.

Os workflows Cybershake e Montage usando a política *all* (Figuras 49a e 49b) tiveram uma melhoria de custo para makespan mais curtos. Esse resultado é explicado pela concentração de dados em apenas algumas tarefas para os dois workflows, uma vez que a política *all* aplica serviços de segurança a todas as tarefas. O operador de inicialização *bestWorstMachines* atribui o melhor e o pior tipo de VM de maneira intercalada para cada tarefa e, portanto, estima-se que as instâncias mais lentas foram inicialmente atribuídas a tarefas com maior carga de trabalho, enquanto as mais rápidas foram associadas àquelas com menor carga. Por outro lado, o

Figura 49 – Fronteiras de Pareto considerando custo e makespan.



(a) Cybershake (all) (b) Montage (all) (c) Epigenomics (all)

(d) Cybershake (entry/end) (e) Montage (entry/end) (f) Epigenomics (entry/end)

(g) Cybershake (gather) (h) Montage (gather) (i) Epigenomics (gather)

(j) Cybershake (scatter) (k) Montage (scatter) (l) Epigenomics (scatter)

(m) Cybershake (scatter/gather) (n) Montage (scatter/gather) (o) Epigenomics (scatter/-gather)

Fonte: Elaborada pelo autor.

Epigenomics apresentou soluções dispersas (Figura 49c), principalmente no início do processo de otimização. No entanto, foi capaz de oferecer soluções de makespan menor que 25 horas.

As soluções que produzem menor makespan não tiveram custos baixos quando comparadas ao Cybershake e Montage, o que é explicado pela quantidade homogênea de dados manipulados por cada tarefa no Epigenomics. Algumas dessas tarefas já receberam instâncias de VM mais rápidas durante o estágio de inicialização da solução.

Na política *entry/end*, as fronteiras originalmente ofereceram soluções com makespan inferior a 13 e 0,54 para Cybershake e Montage (Figuras 49d e 49e), respectivamente. Ao final da otimização, foram encontradas soluções com maior makespan e menor custo de execução. Isso pode estar associado ao uso dos operadores de mutação para sair de ótimos locais. Na política *gather*, há uma melhoria significativa para o makespan nos três workflows (Figuras 49g, 49h e 49i). Isso acontece porque as VMs mais rápidas são alocadas para as tarefas *gather* onde os serviços de segurança são aplicados, ao passo que os recursos mais baratos e lentos são alocados às demais tarefas.

A mesma análise anterior se adequa às políticas *scatter* e *scatter/gather*. Considerando a fronteira de Pareto encontrada para o workflow Cybershake aos 1800s. Usando a política *scatter*, há soluções que levam aproximadamente 5 horas de execução com custo de $185 e soluções que levam maior tempo (12 horas), porém com custo de $100. Por outro lado, as soluções para a política *scatter/gather* têm custo semelhante, já que este workflow não tem tarefas de agrupamento e dispersão de dados e, consequentemente, não produz nenhum *overhead* de segurança. Independentemente da configuração das VMs, o makespan permaneceu abaixo de 1 hora com um custo em torno de $100.

No workflow Montage, aplicando *scatter* nos 1800s, é razoável pagar $90 para executar em 31 minutos do que pagar $130 por uma execução 5 minutos mais rápida. Analisando o uso da política *scatter/gather*, as soluções apresentam makespan baixo pois não há tarefa que realize operações de agrupamento e dispersão de dados. As soluções encontradas na fronteira de 1800s mostram que o maior tempo de execução não ultrapassa 9 minutos, enquanto o mais curto leva cerca de 2 minutos. Portanto, parece razoável esperar mais sete minutos para a execução do workflow do que pagar aproximadamente o dobro do preço ($180) para executá-lo.

Ao otimizar o escalonamento do workflow Epigenomics, considerando a política *scatter* em 1800s, o NSGA-II retornou soluções que exigem 14 horas de execução pagando $110 e 6 horas gastando $280. A política *scatter/gather* não produziu nenhum *overhead* substancial no workflow Epigenomics, uma vez que não há tarefa que se encaixe nessa política. Além disso, as soluções encontradas ao longo dos 1800s variaram de 1.2 a 2 horas de makespan e de $90 a aproximadamente $180.

Em resumo, o Cybershake usando as políticas *all*, *entry/end* e *scatter* apresentou uma diversidade razoável de soluções que é explicada pelo significativo volume de dados que correspondem a essas políticas. Usando a política *gather*, o Cybershake não aproveita tão bem a otimização multiobjetivo, devido ao baixo volume de dados que são protegidos aplicando essa política. A política *scatter/gather* tem um comportamento semelhante ao *gather*, pois

o workflow Cybershake não tem tarefas de agrupamento e dispersão de dados. O Montage apresentou uma boa relação custo/benefício na política *all*. É possível executar o workflow em 2.5 horas pagando $115, ou pagando $95 e esperando por 26 horas de execução. Embora as outras quatro políticas apresentem uma diferença significativa entre o menor e o maior custo da solução, o makespan de soluções mais caras não é proporcional ao seu preço.

Para o Epigenomics, o NSGA-II obteve o maior custo e variação de makespan. Isso pode ser explicado pela homogeneidade do tamanho dos dados de cada tarefa do workflow. A tarefa de entrada englobada pelas políticas *all*, *entry*/*end* e *scatter* influi significativamente no makespan por manipular maior volume de dados, o que requer mais tempo para o processamento dos serviços de segurança. No final da execução do Epigenomics, os dados resultantes da execução dos pipelines paralelos são reunidos por uma tarefa. Ao aplicar a política *gather*, o makespan não excedeu 2 horas. Isso indica que a tarefa de agrupamento recebeu uma pequena quantidade de dados dos pipelines. Como esperado, a otimização do escalonamento usando a política *scatter*/*gather* apresentou soluções com menor makespan em comparação as outras políticas, uma vez que o Epigenomics não possui tarefas de agrupamento e dispersão de dados.

8.6 SAST vs. SCAS

Os resultados computacionais também foram obtidos a partir dos três workflows: Cybershake, Montage e Epigenomics. O desempenho da abordagem SAST é avaliado com base na taxa de tarefas sensíveis (RST) definida como o número total de tarefas sensíveis sobre o número total de tarefas do workflow. Assim, foram definidos arbitrariamente cinco valores de RST: 0, 0.25, 0.50, 0.75 e 1.0. As tarefas sensíveis foram escolhidas aleatoriamente de acordo com cada RST. Três magnitudes de dados foram considerados para cada tarefa do workflow: 1MB, 1GB e 1TB, onde cada valor foi atribuído a todas as tarefas do workflow.

As especificações e os preços das instâncias de VM seguem o esquema da Amazon EC2 (AMAZON..., 2018) considrando instâncias *On-demand*. Também foi assumido que as máquinas virtuais estão localizadas no mesmo datacenter e a taxa de transmissão entre VMs é de 1 Gbit.

O *deadline* para cada workflow foi estimado usando o tempo de execução mais lento e mais rápido. O primeiro considerou o tempo de execução com base na atribuição de tipos de VM mais baratas para cada tarefa, enquanto o segundo considerou um escalonamento, em cada tarefa foi associada ao tipo de VM mais rápida. Em ambos os casos, todas as tarefas do workflow são definidas para serem protegidas de modo que sempre haja uma solução viável. Assim, o valor médio do tempo de execução mais lento e mais rápido foi definido como *deadline* de cada workflow.

O processo de otimização adotou o MPGA que foi configurado para a evolução de 3 populações com 30 indivíduos cada, 0.1 de taxa de mutação e critério de parada de 1800s. Em

suma, avaliamos cada combinação dos 3 workflows, 5 taxas de RST, 3 magnitudes de dados e 3 abordagens de escalonamento. O MPGA foi executado 33 vezes em cada configuração, totalizando 4455 avaliações.

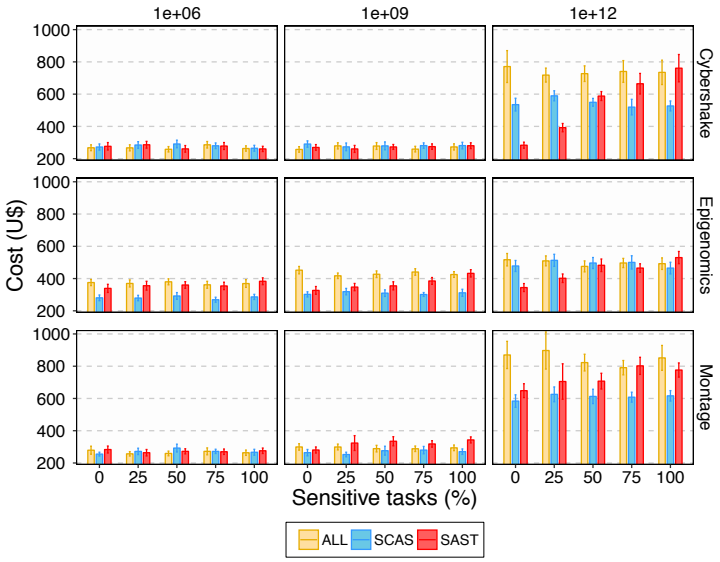
Comparação de custo e risco

Nós comparamos as seguintes abordagens de escalonamento:

- *All*: os serviços de segurança são aplicados a todos os dados do fluxo de trabalho e otimizam os tipos de VMs para cada tarefa;
- *SCAS*: essa abordagem foi introduzida por (LI *et al.*, 2016). Ele otimiza os tipos de VM e diferentes níveis para cada serviço de segurança (autenticação, verificação de integridade e confidencialidade);
- *SAST*: nossa abordagem proposta baseada em anotações anteriores para proteger tarefas sensíveis.

Cada abordagem foi avaliada em termos de custo, tempo de execução, risco, *overhead* de segurança e desperdício de processamento protegendo tarefas não-sensíveis.

Figura 50 – Custo monetário considerando diferentes quantidades de dados (por painel - bytes) e proporção de tarefas sensíveis (eixo X - %) usando as políticas *SAST*, *SCAS* e *ALL*.



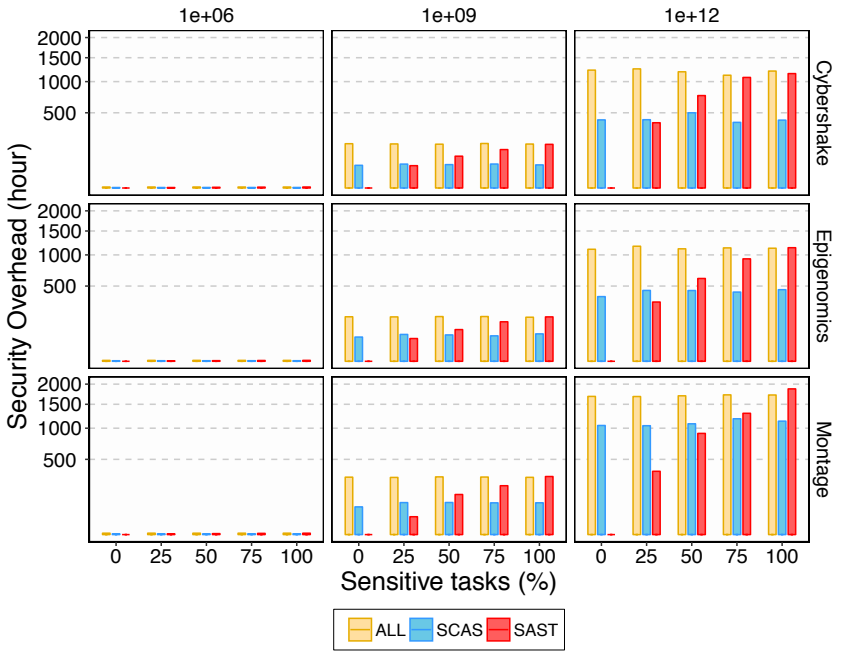
Fonte: Elaborada pelo autor.

A Figura 50 apresenta os custos de execução de cada abordagem de escalonamento executando workflows com diferentes tamanhos de dados (1MB, 1GB e 1TB) em cada tarefa. O custo de execução é semelhante nos três workflows com tarefas que lidam com dados pequenos (1 MB) e médios (1 GB), independentemente da taxa de tarefas sensíveis RST. Porém, quando o volume de dados dos workflows foi na ordem de 1 TB, foi possível notar maior diferença de

custo, makespan e overhead entre as abordagens de escalonamento. A abordagem *All* encontrou soluções de escalonamento caras principalmente em cenários em que há grandes volume de dados com baixa proporção de tarefas sensíveis, pois protege os dados de todo o workflow. O *SCAS* obteve custos inferiores ao *SAST* para quantias maiores de tarefas sensíveis ($RST \geq 0.5$). No entanto, o *SCAS* deixou tarefas sensíveis sem proteção, enquanto o *SAST* aplicou serviços de segurança a todas as tarefas sensíveis. O *SAST* produziu soluções com custo mais alto que o *SCAS* em todos os casos ao executar o workflow Montage. Isso ocorre porque esse workflow tem o maior volume de dados transferidos, como mostrado na Figura 52.

A Figura 51 mostra o *overhead* acumulado para proteger os workflows. É possível ver que o Montage gastou tempo expressivo protegendo dados. Embora a adição de serviços de segurança possa reduzir o vazamento de dados sensíveis, ela demanda mais processamento. Além disso, as tarefas exigem tipos de VM mais rápidas para cumprir o *deadline*, o que naturalmente resulta em custos monetários mais altos.

Figura 51 – Tempo de sobrecarga acumulada das tarefas protegidas pelas diretivas *SAST*, *SCAS* e *ALL* considerando diferentes quantidades de dados (por painel - bytes) e proporção de tarefas sensíveis (eixo X - %).

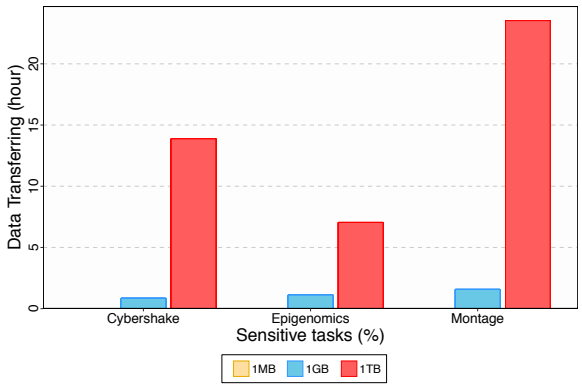


Fonte: Elaborada pelo autor.

A Figura 52 representa a somatória do tempo gasto para transferir os dados entre as tarefas do workflow e considera que a transferência de diferentes volume de dados (1MB, 1GB e 1TB) para cada tarefa. Como visto nessa figura, a execução dos três workflows não consumiu tempo considerável em transferência para dados pequenos (1MB por tarefa). Por outro lado, ao executar os workflows com volume de dados maiores (1GB e 1TB), a transferência demandou razoavelmente mais tempo para ser realizada, sobretudo em tarefas que manipularam 1TB.

Assim, ao analisar o tempo de transferência de dados e o *overhead* de serviços de segurança, notou-se que o tamanho dos dados influenciam significativamente no tempo de execução do workflow conforme o percentual de tarefas sensíveis aumenta utilizando o *SAST*. Por outro lado, o *SCAS* manteve um *overhead* mais homogêneo independentemente do percentual de tarefas sensíveis. Tal comportamento é explicado pela ausência da identificação do requisito de segurança de cada tarefa no *SCAS*.

Figura 52 – Tempo gasto na transferência de dados entre máquinas virtuais considerando três quantidades de dados.



Fonte: Elaborada pelo autor.

Comparamos a diferença entre a solução SAST e o SCAS em termos de custo e risco de segurança na Tabela 24. Para workflows sem tarefas sensíveis (RST=0%), o SAST e o SCAS não têm diferença em termos de risco. O SCAS otimizou a segurança das tarefas, mas pode ter gasto tempo protegendo dados não-sensíveis. O custo de execução usando o SAST é até 47% menor para tarefas de dados maiores (1TB) do que usando o SCAS. A proteção de tarefas não-sensíveis pelo SCAS aumenta o tempo de aluguel de VMs e, consequentemente, eleva o custo de execução em workflows sem tarefas sensíveis. Em tamanhos de dados pequenos (1 MB) e médios (1 GB), a diferença de custo não foi tão significativa quanto em tarefas com arquivos grandes. A relação de baixo custo (-7% - 21%) sugere baixo *overhead* para garantir a pequena e média quantidade de dados pelo SCAS.

Tabela 24 – Trade-off entre o risco de vazamento de dados sensíveis e o custo monetário.

	Cybershake						**Montage**						**Epigenomics**					
	(1MB)		**(1GB)**		**(1TB)**		**(1MB)**		**(1GB)**		**(1TB)**		**(1MB)**		**(1GB)**		**(1TB)**	
Sens.	Risk	Cost	Risk	Cost	Risk	Cost	Risk	Cost	Risk	Cost	Risk	Cost	Risk	Cost	Risk	Cost	Risk	Cost
0	0.00	0.02	0.00	-0.07	0.00	-0.47	0.00	0.11	0.00	0.06	0.00	0.11	0.00	0.21	0.00	0.08	0.00	-0.28
25	-0.66	0.01	-0.61	-0.05	-0.52	-0.33	-0.65	-0.03	-0.64	0.28	-0.47	0.13	-0.60	0.27	-0.55	0.09	-0.45	-0.22
50	-0.64	-0.10	-0.66	-0.02	-0.55	0.07	-0.70	-0.07	-0.67	0.21	-0.54	0.15	-0.66	0.23	-0.61	0.15	-0.50	-0.03
75	-0.69	-0.01	-0.67	-0.02	-0.61	0.28	-0.67	-0.01	-0.68	0.14	-0.54	0.32	-0.71	0.32	-0.66	0.28	-0.54	-0.07
100	-0.71	-0.02	-0.70	0.00	-0.59	0.44	-0.67	0.03	-0.71	0.05	-0.57	0.26	-0.68	0.34	-0.65	0.39	-0.55	0.14

Fonte: Dados da pesquisa.

O tempo total gasto na proteção de dados é de até 1156, 1140 e 1875 horas para os

workflows CyberShake, Epigenomics e Montage, respectivamente. Como esperado para o SAST, o alto *overhead* aparece no cenário com maior volume de dados por tarefa (1 TB) e com maior porcentagem de tarefas sensíveis. Portanto, o custo de execução no SAST está relacionado ao volume de dados a ser protegido. Por outro lado, os mapas de escalonamento produzidos pelo *SCAS* excedem o *SAST* em termos de custo para $RST \leq 50\%$, devido ao uso desnecessário de serviços de segurança em tarefas não sensíveis. Quando $RST \geq 75\%$, o *SCAS* apresentou custo mais baixo que o SAST, porém deixou as tarefas sensíveis sem proteção.

Comparação do makespan

A abordagem *SAST* busca a minimização do custo de execução e o risco da execução de workflows. A atribuição de VM mais barata (capacidades mais lentas) poderia minimizar o custo, no entanto, isso resultaria em um maior período de tempo. Portanto, *SAST* considera um *deadline* aceitável para o usuário. As três abordagens apresentaram tempo de execução semelhante para dados pequenos (1MB). Como mostrado na Figura 51 e 52, pequenos dados geram baixo *overhead* de segurança e consomem menos tempo com transferência de dados em comparação ao tempo total de execução (Figura 53).

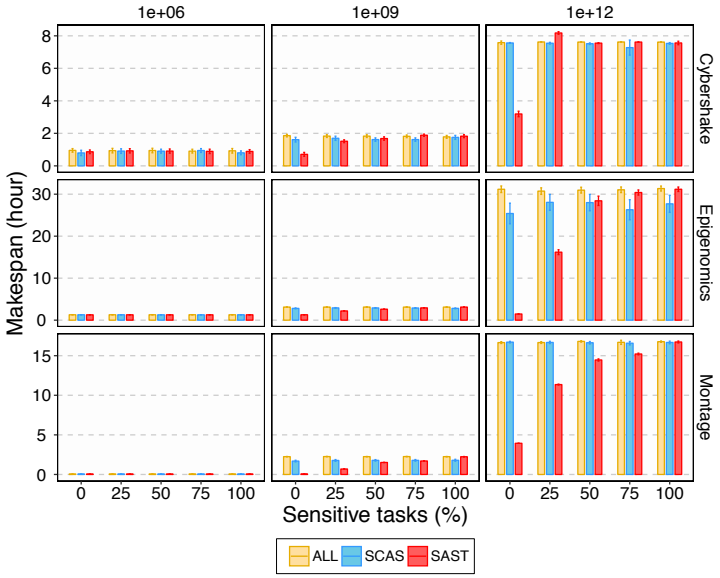
Em simulações com dados médios (1GB), *All* e *SCAS* também apresentaram makespan similar. *SAST* tem diferença relevante no makespan para tarefas que lidam com dados médios (1 GB). *SAST* encontrou soluções cujo makespan são menores que o *SCAS* e o *All*, indicando que workflows com baixo número de tarefas sensíveis pode ser privilegiado por uma abordagem seletiva, conforme introduzido no *SAST*. O *overhead* produzido pelo *SAST* em workflows que manipulam dados de tamanho médio cresce seguindo a proporção de tarefas sensíveis. O *overhead* gerado no *SAST* foi maior que o *SCAS* quando $RST \geq 0.50$. Por outro lado, o *SAST* ofereceu execuções até 66% e 71% mais seguras do que o *SCAS* para tarefas sensíveis em que $RST \leq 25\%$ e $RST \geq 50\%$.

Dados grandes (1 TB) afetam drasticamente o tempo total de execução. As abordagens *All* e *SCAS* produziram longos makespan nas cinco proporções de tarefas sensíveis. Por outro lado, *SAST* exigiu menos da metade do tempo para a execução dos três workflows sem tarefas sensíveis ($RST = 0\%$). O *overhead* de transferência de dados foi o único fator que causou o aumento do makespan devido a grande quantidade de dados. A execução do Montage e Cybershake usando *SAST* levou mais tempo em dados médios e $RST = 0\%$. Isso é uma consequência do aumento de comunicação gasto entre as VMs nesses dois workflows, conforme mostrado na Figura 52. Em contrapartida, o makespan usando *SAST* foi menor ou próximo do *SCAS* ($RST = \{25\%, 50\%\}$) por um custo similar.

Desperdício de processamento assegurando tarefas não sensíveis

A aplicação de serviços de segurança implica em *overhead* durante a execução do workflow. Portanto, proteger tarefas não sensíveis não parece ser uma medida razoável. Nesse sentido, analisamos o desperdício dos algoritmos *All*, *SCAS* e *SAST* para proteger tarefas não

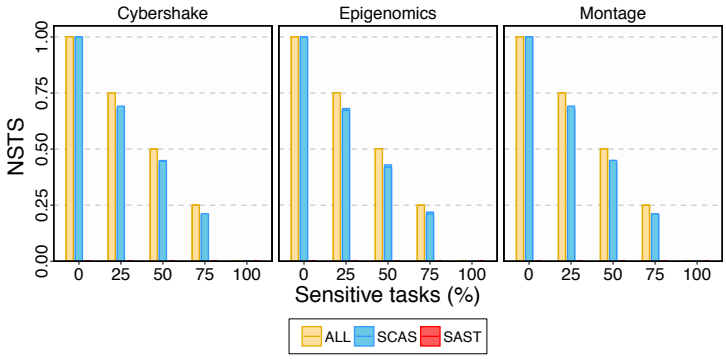
Figura 53 – Makespan considerando diferentes quantidades de dados (por painel - bytes) e proporção de tarefas sensíveis (eixo X - %) usando as políticas *SAST*, *SCAS* e *ALL*.



Fonte: Elaborada pelo autor.

sensíveis usando a métrica definida na Equação 7.4.

Figura 54 – Desperdício para proteger tarefas não sensíveis.



Fonte: Elaborada pelo autor.

A abordagem *All* considera a execução de todas as tarefas do workflow e, como esperado, quanto menor o percentual de tarefas sensíveis, a *All* produziu mais desperdício de processamento, como mostrado na Figura 54. Em workflows cuja porcentagem de tarefas sensíveis é 0%, o *All* produziu 100% de desperdício, protegendo desnecessariamente todas as tarefas do workflow. O desperdício gerado pelo *All* é reduzido com o aumento do número de tarefas sensíveis. O *SCAS* tentou otimizar o custo e o risco de escalonamento. Embora tenha minimizado o custo mais do que o *All*, ele desperdiçou processamento protegendo tarefas não sensíveis. Assim, notou-se eficiência do *SCAS* foi discretamente melhor do que o *All*. Finalmente, o *SAST*

não produziu nenhum desperdício, porque a proteção foi atribuída exclusivamente às tarefas anotadas como sensíveis.

CAPÍTULO

9

CONCLUSÃO

As abordagens de escalonamento WS-TSP e SAST apresentadas nesta tese possibilitam executar workflows em nuvens públicas de forma protegida e buscam minimizar o custo de execução satisfazendo um *deadline* estipulado pelo usuário. Este capítulo apresenta as conclusões, as contribuições, os artigos produzidos, os trabalhos futuros e as dificuldades enfrentadas durante o desenvolvimento desta tese.

9.1 Argumentação geral

O avanço dos modelos para a representação de problemas do mundo real tem refletido em experimentos *in-silico* mais complexos. Assim, experimentos *in-silico* tem adotado o modelo de workflow para auxiliar na colaboração entre diferentes áreas do conhecimento, facilitando o processo de orquestração, reprodução e disseminação de experimentos. Contudo, a execução de um workflow em uma estação de trabalho pode levar tempos inaceitáveis. Assim, o escalonamento de workflow em arquiteturas distribuídas como grades e nuvens tem sido alvo de investigação nos últimos anos para obter tempos razoáveis de execução e ainda atender outros critérios como o custo e a segurança de dados.

Esta tese apresentou o WS-TSP e a SAST, abordagens de escalonamento que apoiam a segurança de dados sensíveis e ainda otimizam o custo de execução satisfazendo um *deadline* estipulado pelo usuário. Ambas as abordagens se baseiam na possibilidade de identificação de tarefas do workflow em que serviços de segurança como autenticação, verificação de integridade e criptografia são aplicados para proteger dados sensíveis. Tanto o WS-TSP quanto a SAST representam o escalonamento por meio de uma codificação que pode ser otimizada por meta-heurísticas para a escolha de máquina virtual para cada tarefa do workflow.

A reprodução de estudos da literatura foi realizada com a finalidade de compreender o impacto dos modelos de execução, transferência de dados e, principalmente, *overhead* de

segurança. Ao final desse processo, pôde-se identificar que os algoritmos de escalonamento tentam proteger as tarefas do workflow aplicando serviços de segurança. Entretanto, esses algoritmos acabam protegendo tarefas desnecessariamente e deixando tarefas sensíveis sem qualquer proteção. Nesse contexto, um dos objetivos da abordagem WS-TSP e da SAST é utilizar o conhecimento que o Desenvolvedor de Aplicações tem de todo o workflow para a identificação das tarefas sensíveis em que serviços de segurança são devidamente utilizados. A identificação das tarefas é realizada pela seleção de uma política que se adeque às tarefas sensíveis (WS-TSP) ou pela anotação das tarefas sensíveis (SAST). A otimização do custo de execução satisfazendo um *deadline* é desempenhada por uma meta-heurística baseada em Algoritmo Genético Multi-populacional que evolui iterativamente com base nas avaliações providas pelo simulador WorkflowSim.

O WS-TSP e o SAST foram avaliados em um ambiente de nuvem simulado (a) para permitir a análise da combinação de diferentes fatores; (b) pela falta de recurso financeiro para a execução dos workflows em nuvem pública real, e; (c) ambiente controlado que permite a reprodutibilidade dos experimentos. Ambas as abordagens delineiam um modelo conceitual para que possam ser adaptadas em sistemas gerenciadores de workflows científicos. Comparados aos algoritmos da literatura que consideram a segurança no processo de escalonamento, o WS-TSP e o SAST apresentaram menor risco de segurança a um tempo e custo de execução razoável, o que vai em direção à hipótese levantada no início desta tese. A propósito, ambas as abordagens não gastam processamento aplicando desnecessariamente serviços de segurança em tarefas que não envolvem dados sensíveis.

As duas abordagens expressam um conceito que pode ser adotado em sistemas gerenciadores de workflows. Inclusive, um dos motivos da adoção do simulador WorkflowSim foi justamente por este abstrair a estrutura do SGWf Pegasus, oferecendo um cenário real de implantação das abordagens propostas.

9.2 Contribuições

As principais contribuições desta tese são:

- Estudo bibliométrico para auxiliar novatos na área de escalonamento de workflows que estabelece um levantamento quantitativo das publicações de algoritmos de escalonamento de workflows; uma análise de quais são os principais trabalhos, autores, conferências, periódicos e países que atuam na investigação de algoritmos de escalonamento de workflows;

- Uma extensão para o simulador WorkflowSim que considera o *overhead* de segurança e a análise do risco em execuções de workflows. Essa extensão é essencial para a elaboração de novas abordagens de escalonamento;

- Uma avaliação comparativa entre as meta-heurísticas de Enxame de Partículas, Algoritmos Genéticos e Algoritmos Genéticos Multi-populacional aplicados no escalonamento de workflows;

- Proposição, desenvolvimento e avaliação de duas abordagens de escalonamento de workflows (WS-TSP e SAST) para proteção de tarefas sensíveis e que buscam minimizar o custo de execução satisfazendo um *deadline* em nuvem;

- Otimização multi-objetivo para a busca de soluções de escalonamento com maior ponderação à minimização do custo ou tempo de execução considerando diferentes políticas de segurança.

9.3 Publicações

Durante o período do doutorado foram publicados os seguintes artigos em conferências e periódicos:

- SHISHIDO, H. Y.; ESTRELLA, J. C.; TOLEDO, C. F. M.; ARANTES, M. S. **Genetic-based algorithms applied to a workflow scheduling algorithm with security and deadline constraints in clouds.** Computers & Electrical Engineering, 69(1), 378-394. DOI: <https://doi.org/10.1016/j.compeleceng.2017.12.004>
  Fator de impacto (JCR): 1.57 - 2 citações

- SHISHIDO, H. Y.; ESTRELLA, J. C. **Bibliometric Analysis of Workflow Scheduling in Grids and Clouds**. Em: 35th International Conference of the Chilean Computer Science Society, 2017, Arica - Chile.
  Qualis: B3

- SHISHIDO, H. Y.; ESTRELLA, J. C.; TOLEDO, C. F. M.; REIFF-MARGANIEC, S. **Tasks Selection Policies for Securing Sensitive Data on Workflow Scheduling in Clouds**. Em: IEEE International Conference on Services Computing, 2018, San Francisco - USA.
  Qualis: A2

- SHISHIDO, H. Y.; ESTRELLA, J. C.; TOLEDO, C. F. M. **Multi-objective optimization for workflow scheduling under task selection policies in clouds**. Em: IEEE Congress on Evolutionary Computation, 2018, Rio de Janeiro - Brazil.
  Qualis: A1

- SHISHIDO, H. Y.; ESTRELLA, J. C.; TOLEDO, C. F. M.; REIFF-MARGANIEC, S. **A CloudSim extension for evaluating security overhead in workflow execution in clouds**. Em: The Sixth International Symposium on Computing and Networking. Hida

Takayama - Japan.
Qualis: B2

Há ainda outro artigo submetido aguardando revisão:

- SHISHIDO, H. Y.; ESTRELLA, J. C.; TOLEDO, C. F. M. **Optimizing security and cost of workflow execution with task annotation and genetic-based algorithm.** Future Generation Computer Systems.
Fator de impacto (JCR): 4.63

9.4 Dificuldades

O desenvolvimento de algoritmos de escalonamento de workflows envolve uma série de componentes para sua realização. Entre os diversos requisitos, é necessário compreender os conceitos, as ferramentas, as metodologias de avaliação e também o estado da arte de uma linha de pesquisa que ainda não havia sido estudada no laboratório onde esta tese foi desenvolvida. O mapeamento do estado da arte demandou um longo período devido ao amplo volume de artigos publicados. Por falta de conhecimento prévio, a compreensão dos modelos do sistema envolvidos no processo de escalonamento de workflows também demandou esforços e tempo para que pudessem ser implementados corretamente.

A experimentação simulada foi adotada como meio de avaliação das abordagens propostas e das utilizadas para comparação. Porém, poucos dos trabalhos publicados oferecem os parâmetros e modelos suficientes para a reprodução dos experimentos. Além disso, os manuscritos não indicam nenhum repositório onde algoritmos e/ou ambientes de simulação utilizados para download. Diante dessas circunstâncias, foi necessário despender esforços e tempo prolongado para adaptar um simulador de nuvem para avaliar e comparar as abordagens propostas e os algoritmos existentes.

A otimização do custo de execução respeitando um *deadline* foi realizada através de algoritmos genéticos. O desenvolvimento de operadores genéticos bem como a calibração do número de indivíduos, número e populações, taxa de mutação e cruzamento também demandaram tempo e experimentos extensivos. As inúmeras repetições de experimentos em função do comportamento estocástico das meta-heurísticas produziu uma quantia de dados razoavelmente significativa para tratar. *Scripts* foram desenvolvidos na tentativa de auxiliar no processamento das inúmeras avaliações.

Por envolver diferentes linhas de pesquisa, a colaboração com profissionais de outros laboratórios foi substancial para o desenvolvimento desta tese. O relacionamento entre pesquisadores de diferentes áreas da computação é fundamental para o avanço na área. E não menos importante, a falta de financiamento por parte de órgãos de fomento para a execução e a

validação das abordagens propostas em nuvens públicas (pagas) também limitou a abrangência dos resultados.

9.5 Trabalhos futuros

A execução de workflows em nuvens precisa lidar com diversos aspectos que envolvem desde a minimização do tempo de execução dos experimentos, mas também com a otimização do custo, tolerância a falhas, segurança, consumo de energia e outras possibilidades. Desta forma, para complementar as abordagens introduzidas nesta tese, ainda sugere-se:

- **Parâmetros de segurança mais realistas**: os valores utilizados para o cálculo de *overhead* são estáticos e não consideram as diferentes capacidades computacionais de VMs. Logo, desenvolver modelos que possam estimar o *throughput* de cada algoritmo de segurança com base nas características do recurso computacional alvo pode contribuir com a precisão do cálculo de *overhead*;

- **Operadores genéticos**: um dos alicerces da computação evolutiva é o conjunto de operadores genéticos (mutação e cruzamento). A pesquisa de operadores mais eficientes pode reduzir o tempo de convergência da otimização encontrando soluções tão boas quanto às encontradas nesta tese;

- **Implementação em um SGWf**: embora esta tese tenha concebido abordagens para proteger a execução de workflows, a avaliação limitou-se em um ambiente simulado e controlado. No entanto, ainda é necessário implantar o modelo proposto em um SWGf para a avaliação deste modelo em ambiente real;

- **Abordagem multi-objetivo**: a abordagem WS-TSP apresenta uma análise multi-objetivo baseado no NSGA-II. Sugere-se implementar também o NSGA-II no SAST. Além do mais, a comparação de outros métodos multi-objetivos como MOEA-D, NSGA-III, PESA-II, SM-PSO e VEGA pode ser realizada;

- **Spot-instances**: esta tese considerou instâncias Amazon EC2 *on-demand* cujo modelo de cobrança baseia-se em hora/uso. Sugere-se a implementação da capacidade de tolerar a interrupção de *spot-instances* de modo que seja possível minimizar o custo de execução.

REFERÊNCIAS

AALST, W. van der; HOFSTEDE, A. ter; KIEPUSZEWSKI, B.; BARROS, A. Workflow patterns. **Distributed and Parallel Databases**, v. 14, n. 1, p. 5–51, Jul 2003. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1023/A:1022883727209>. Citado na página 41.

ABAWAJY, J. H. Fault-tolerant scheduling policy for grid computing systems. In: **18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.** [s.n.], 2004. p. 238–. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/IPDPS.2004.1303290>. Citado na página 82.

ABBAS, A.; BILAL, K.; ZHANG, L.; KHAN, S. U. A cloud based health insurance plan recommendation system: A user centered approach. **Future Generation Computer Systems**, v. 43-44, p. 99 – 109, 2015. Acessado em 19 de jun. 2018. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167739X14001587>. Citado na página 48.

ABRISHAMI, S.; NAGHIBZADEH, M.; EPEMA, D. H. Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds. **Future Generation Computer Systems**, v. 29, n. 1, p. 158 – 169, 2013. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1016/j.future.2012.05.004>. Citado nas páginas 78 e 140.

ACM. 2018. Acessado em 19 de jun. 2018. Disponível em: <https://dl.acm.org/>. Citado na página 68.

ALI, A. F.; HASSANIEN, A.-E. **A Survey of Metaheuristics Methods for Bioinformatics Applications**. Springer International Publishing, 2016. 23–46 p. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1007/978-3-319-21212-8_2>. Citado na página 56.

ALI, M.; KHAN, S. U.; VASILAKOS, A. V. Security in cloud computing: Opportunities and challenges. **Information Sciences**, v. 305, p. 357 – 383, 2015. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1016/j.ins.2015.01.025>. Citado na página 30.

AMAZON EC2 Pricing. 2018. Acessado em 19 de jun. 2018. Disponível em: <https://aws.amazon.com/pt/ec2/pricing/>. Citado na página 153.

ANDERSON, T. W.; DARLING, D. A. A test of goodness of fit. **Journal of the American statistical association**, Taylor & Francis, v. 49, n. 268, p. 765–769, 1954. Acessado em 19 de jun. 2018. Disponível em: <https://www.jstor.org/stable/pdf/2281537.pdf>. Citado na página 120.

ARABNEJAD, H.; BARBOSA, J. G. A budget constrained scheduling algorithm for workflow applications. **Journal of Grid Computing**, v. 12, n. 4, p. 665–679, 2014. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1007/s10723-014-9294-7>. Citado na página 78.

ARANTES, J. d. S.; ARANTES, M. d. S.; TOLEDO, C. F. M.; JúNIOR, O. T.; WILLIAMS, B. C. Heuristic and genetic algorithm approaches for uav path planning under critical situation. **International Journal on Artificial Intelligence Tools**, v. 26, n. 01, p. 1760008, 2017.

Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1142/S0218213017600089>. Citado nas páginas 114, 118 e 142.

ARMBRUST, M.; FOX, A.; GRIFFITH, R.; JOSEPH, A. D.; KATZ, R.; KONWINSKI, A.; LEE, G.; PATTERSON, D.; RABKIN, A.; STOICA, I.; ZAHARIA, M. A view of cloud computing. **Commun. ACM**, ACM, New York, NY, USA, v. 53, n. 4, p. 50–58, abr. 2010. Acessado em 19 de jun. 2018. Disponível em: <http://doi.org/10.1145/1721654.1721672>. Citado na página 44.

ARUNARANI, A. R.; MANJULA, D.; VIJAYAN, S. Ffbat: A security and cost-aware workflow scheduling approach combining firefly and bat algorithms. **Concurrency and Computation: Practice and Experience**, v. 29, n. 24, p. e4295, 2017. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1002/cpe.4295>. Citado nas páginas 15, 16, 31, 32, 86, 125, 132 e 133.

BALASKO, A.; FARKAS, Z.; KACSUK, P. Building science gateways by utilizing the generic ws-pgrade/guse workflow system. **Computer Science**, v. 14, n. 2, p. 307, 2013. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.7494/csci.2013.14.2.307>. Citado nas páginas 93 e 131.

BALLARD, D. H. **An Introduction to Natural Computation**. [S.l.]: MIT Press, 1997. (A Bradford Book: Complex Adaptive Systems). Acessado em 19 de jun. 2018. Citado na página 58.

BARGA, R.; GANNON, D. **Scientific versus Business Workflows**. London: Springer London, 2007. 9–16 p. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1007/978-1-84628-757-2_2>. Citado nas páginas 29, 38 e 39.

BEAGRIE, N. Digital curation for science, digital libraries, and individuals. **International Journal of Digital Curation**, v. 1, n. 1, p. 3–16, 2008. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.2218/ijdc.v1i1.2>. Citado na página 38.

BEIRIGO, B. A.; SANTOS, A. G. dos. Application of nsga-ii framework to the travel planning problem using real-world travel data. In: **2016 IEEE Congr. on Evol. Comput.** [s.n.], 2016. p. 746–753. Acessado em 19 de jun. 2018. Disponível em: <http://dx.doi.org/10.1109/CEC.2016.7743866>. Citado na página 130.

BELLMAN, R. E. **Dynamic programming**. [S.l.]: Princeton University Press, 1957. Acessado em 19 de jun. 2018. Citado na página 55.

BELOGLAZOV, A.; ABAWAJY, J.; BUYYA, R. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. **Future Generation Computer Systems**, v. 28, n. 5, p. 755 – 768, 2012. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1016/j.future.2011.04.017>. Citado na página 83.

BENESTY, J.; CHEN, J.; HUANG, Y.; COHEN, I. Pearson correlation coefficient. In: ____. **Noise Reduction in Speech Processing**. Springer Berlin Heidelberg, 2009. p. 1–4. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1007/978-3-642-00296-0_5>. Citado na página 120.

BESSAI, K.; YOUCEF, S.; OULAMARA, A.; GODART, C.; NURCAN, S. Bi-criteria workflow tasks allocation and scheduling in cloud computing environments. In: **2012 IEEE Fifth International Conference on Cloud Computing**. [s.n.], 2012. p. 638–645. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/CLOUD.2012.83>. Citado na página 79.

BHARATHI, S.; CHERVENAK, A. Data staging strategies and their impact on the execution of scientific workflows. In: **Proceedings of the Second International Workshop on Data-aware Distributed Computing**. ACM, 2009. (DADC '09). Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1145/1552280.1592459>. Citado na página 83.

BHARATHI, S.; CHERVENAK, A.; DEELMAN, E.; MEHTA, G.; SU, M. H.; VAHI, K. Characterization of scientific workflows. In: **2008 Third Workshop on Workflows in Support of Large-Scale Science**. [s.n.], 2008. p. 1–10. Acessado em 19 de jun. 2018. Disponível em: <http://dx.doi.org/10.1109/WORKS.2008.4723958>. Citado na página 145.

BILGAIYAN, S.; SAGNIKA, S.; DAS, M. Workflow scheduling in cloud computing environment using cat swarm optimization. In: **2014 IEEE International Advance Computing Conference (IACC)**. [s.n.], 2014. p. 680–685. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/IAdCC.2014.6779406>. Citado na página 77.

BITTENCOURT, L. F.; MADEIRA, E. R. M. A performance-oriented adaptive scheduler for dependent tasks on grids. **Concurrency and Computation: Practice and Experience**, v. 20, n. 9, p. 1029–1049, 2007. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1002/cpe.1282>. Citado na página 81.

_____. Hcoc: a cost optimization algorithm for workflow scheduling in hybrid clouds. **Journal of Internet Services and Applications**, v. 2, n. 3, p. 207–227, Dec 2011. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1007/s13174-011-0032-0>. Citado nas páginas 82, 83 e 140.

BUI, L. T.; ALAM, S. An introduction to multi-objective optimization. **Multi-Objective Optimization in Computational Intelligence: Theory and Practice**, IGI Global Snippet, p. 1–19, 2008. Acessado em 19 de jun. 2018. Citado na página 64.

BUX, M.; LESER, U. Dynamiccloudsim: Simulating heterogeneity in computational clouds. **Future Generation Computer Systems**, v. 46, p. 85 – 99, 2015. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1016/j.future.2014.09.007>. Citado na página 97.

BUYYA, R.; BROBERG, J.; GOSCINSKI, A. M. **Cloud Computing Principles and Paradigms**. [S.l.]: Wiley Publishing, 2011. Acessado em 19 de jun. 2018. Citado na página 45.

BYUN, E.-K.; KEE, Y.-S.; KIM, J.-S.; DEELMAN, E.; MAENG, S. Bts: Resource capacity estimate for time-targeted science workflows. **Journal of Parallel and Distributed Computing**, v. 71, n. 6, p. 848 – 862, 2011. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1016/j.jpdc.2011.01.008>. Citado na página 80.

BYUN, E.-K.; KEE, Y.-S.; KIM, J.-S.; MAENG, S. Cost optimized provisioning of elastic resources for application workflows. **Future Generation Computer Systems**, v. 27, n. 8, p. 1011 – 1026, 2011. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1016/j.future.2011.05.001>. Citado na página 80.

CALHEIROS., R.; RANJAN, R.; BELOGLAZOV, A.; ROSE, C. A. F. D.; BUYYA, R. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. **Software: Practice and Experience**, v. 41, n. 1, p. 23–50, 2011. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1002/spe.995>. Citado na página 97.

CANON, L.-C.; JEANNOT, E.; SAKELLARIOU, R.; ZHENG, W. **Comparative Evaluation Of The Robustness Of DAG Scheduling Heuristics**. Springer US, 2008. 73–84 p. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1007/978-0-387-09457-1_7>. Citado na página 82.

CHEN, C.; LIU, J.; WEN, Y.; CHEN, J.; ZHOU, D. A hybrid genetic algorithm for privacy and cost aware scheduling of data intensive workflow in cloud. In: WANG, G.; ZOMAYA, A.; MARTINEZ, G.; LI, K. (Ed.). **Algorithms and Architectures for Parallel Processing**. Cham: Springer International Publishing, 2015. p. 578–591. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1007/978-3-319-27119-4_40>. Citado na página 86.

CHEN, H.; ZHU, X.; QIU, D.; LIU, L.; DU, Z. Scheduling for workflows with security-sensitive intermediate data by selective tasks duplication in clouds. **IEEE Transactions on Parallel and Distributed Systems**, v. 28, n. 9, p. 2674–2688, 2017. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/TPDS.2017.2678507>. Citado nas páginas 31, 85 e 86.

CHEN, W.; DEELMAN, E. Workflowsim: A toolkit for simulating scientific workflows in distributed environments. In: **2012 IEEE 8th International Conference on E-Science**. [s.n.], 2012. p. 1–8. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/eScience.2012.6404430>. Citado nas páginas 97 e 98.

CHEN, W. N.; ZHANG, J. An ant colony optimization approach to a grid workflow scheduling problem with various qos requirements. **IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)**, v. 39, n. 1, p. 29–43, 2009. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/TSMCC.2008.2001722>. Citado na página 77.

CHU, S.-C.; TSAI, P.-w.; PAN, J.-S. Cat swarm optimization. In: YANG, Q.; WEBB, G. (Ed.). **PRICAI 2006: Trends in Artificial Intelligence**. [s.n.], 2006. p. 854–858. Acessado em 10 de out. 2018. Disponível em: <https://link.springer.com/chapter/10.1007/978-3-540-36668-3_94>. Citado na página 77.

DARWIN, C. **On the origin of species**. [S.l.]: Routledge, 1859. Acessado em 19 de jun. 2018. Citado na página 57.

DEB, K.; AGRAWAL, S.; PRATAP, A.; MEYARIVAN, T. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. In: SCHOENAUER, M.; DEB, K.; RUDOLPH, G.; YAO, X.; LUTTON, E.; MERELO, J. J.; SCHWEFEL, H.-P. (Ed.). **Parallel Problem Solving from Nature PPSN VI**. Springer Berlin Heidelberg, 2000. p. 849–858. Acessado em 19 de jun. 2018. Disponível em: <https://dl.acm.org/citation.cfm?id=668937>. Citado nas páginas 62 e 131.

DEB, K.; DEB, K. Multi-objective optimization. In: ____. **Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques**. [s.n.], 2014. p. 403–449. ISBN 978-1-4614-6940-7. Disponível em: <https://doi.org/10.1007/978-1-4614-6940-7_15>. Citado na página 65.

DEELMAN, E.; CHERVENAK, A. Data management challenges of data-intensive scientific workflows. In: **Proceedings of the 2008 Eighth IEEE International Symposium on Cluster Computing and the Grid**. [s.n.], 2008. p. 687–692. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/CCGRID.2008.24>. Citado na página 131.

DEELMAN, E.; FILGUEIRA, R.; MAYANI, R.; DO, T. M. A.; RYNGE, M.; VAHI, K.; PAPADIMITRIOU, G. **Pegasus Workflow Management System**. 2018. Disponível em: <https://pegasus.isi.edu/>. Citado nas páginas 138, 139, 140 e 141.

DEELMAN, E.; GANNON, D.; SHIELDS, M.; TAYLOR, I. Workflows and e-science: An overview of workflow system features and capabilities. **Future Generation Computer Systems**, v. 25, n. 5, p. 528 – 540, 2009. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1016/j.future.2008.06.012>. Citado nas páginas 29, 38, 39 e 40.

DEELMAN, E.; JUVE, G.; BERRIMAN, G. B. Using clouds for science, is it just kicking the can down the road? In: **Proceedings of the 2nd International Conference on Computing and Services Science**. [s.n.], 2012. p. 127–134. Acessado em 19 de jun. 2018. Disponível em: <http://doi.org/10.1.1.308.7864>. Citado na página 30.

DEELMAN, E.; VAHI, K.; JUVE, G.; RYNGE, M.; CALLAGHAN, S.; MAECHLING, P. J.; MAYANI, R.; CHEN, W.; SILVA, R. F. da; LIVNY, M.; WENGER, K. Pegasus, a workflow management system for science automation. **Future Generation Computer Systems**, v. 46, p. 17 – 35, 2015. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1016/j.future.2014.10.008>. Citado na página 131.

DEJUN, J.; PIERRE, G.; CHI, C.-H. Ec2 performance analysis for resource provisioning of service-oriented applications. In: **Proceedings of the 2009 International Conference on Service-oriented Computing**. Springer-Verlag, 2009. (ICSOC/ServiceWave'09), p. 197–207. Acessado em 19 de jun. 2018. Disponível em: <http://dl.acm.org/citation.cfm?id=1926618.1926641>. Citado na página 81.

DOGAN, A.; OZGUNER, F. Matching and scheduling algorithms for minimizing execution time and failure probability of applications in heterogeneous computing. **IEEE Transactions on Parallel and Distributed Systems**, v. 13, n. 3, p. 308–323, Mar 2002. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/71.993209>. Citado na página 79.

DUAN, Y.; FU, G.; ZHOU, N.; SUN, X.; NARENDRA, N. C.; HU, B. Everything as a service (xaas) on the cloud: Origins, current and future trends. In: **2015 IEEE 8th International Conference on Cloud Computing**. [s.n.], 2015. p. 621–628. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/CLOUD.2015.88>. Citado na página 45.

DUNCAN, A. D.; BUYTENDIJK, F. **How to Establish a Data-Driven Culture in the Digital Workplace**. [S.l.], 2015. Acessado em 19 de jun. 2018. Disponível em: <https://www.gartner.com/doc/3072117?refval=&pcp=mpe>. Citado na página 84.

DURILLO, J. J.; PRODAN, R. Multi-objective workflow scheduling in amazon ec2. **Cluster Computing**, v. 17, n. 2, p. 169–189, 2014. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1007/s10586-013-0325-0>. Citado na página 130.

EBERHART, R.; KENNEDY, J. A new optimizer using particle swarm theory. In: **Micro Machine and Human Science, 1995. MHS '95., Proceedings of the Sixth International Symposium on**. [s.n.], 1995. p. 39–43. Acessado em 19 de jun. 2018. Disponível em: <http://doi.org/10.1109/MHS.1995.494215>. Citado nas páginas 111 e 112.

EDGEWORTH, F. Y. **Mathematical psychics : an essay on the application of mathematics to the moral sciences**. [S.l.]: Lond. Sch. of Econ. and Polit. Sci London, 1881. Acessado em 19 de jun. 2018. Citado na página 64.

FAKHFAKH, F.; KACEM, H. H.; KACEM, A. H. Workflow scheduling in cloud computing: A survey. In: **IEEE 18th International Enterprise Distributed Object Computing Conference Workshops and Demonstrations**. [s.n.], 2014. p. 372–378. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/EDOCW.2014.61>. Citado na página 52.

FARD, H. M.; PRODAN, R.; BARRIONUEVO, J. J. D.; FAHRINGER, T. A multi-objective approach for workflow scheduling in heterogeneous environments. In: **2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)**. [s.n.], 2012. p. 300–309. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/CCGrid.2012.114>. Citado na página 79.

FOSTER, I.; KESSELMAN, C. (Ed.). **The Grid: Blueprint for a New Computing Infrastructure**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999. Acessado em 19 de jun. 2018. Citado na página 37.

Editors. In: FOSTER, I.; KESSELMAN, C. (Ed.). **The Grid 2 (2)**. 2. ed. Burlington: Morgan Kaufmann, 2004, (The Morgan Kaufmann Series in Computer Architecture and Design). p. vii – viii. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1016/B978-155860933-4/50000-6>. Citado na página 37.

FOSTER, I.; KESSELMAN, C.; TUECKE, S. The anatomy of the grid: Enabling scalable virtual organizations. **The International Journal of High Performance Computing Applications**, v. 15, n. 3, p. 200–222, 2001. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1177/109434200101500302>. Citado na página 37.

FOSTER, I.; ZHAO, Y.; RAICU, I.; LU, S. Cloud computing and grid computing 360-degree compared. In: **Proceedings of Grid Computing Environments Workshop**. [s.n.], 2008. p. 1–10. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/GCE.2008.4738445>. Citado na página 43.

FRANçA, P. M.; MENDES, A.; MOSCATO, P. A memetic algorithm for the total tardiness single machine scheduling problem. **European Journal of Operational Research**, v. 132, n. 1, p. 224 – 242, 2001. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1016/S0377-2217(00)00140-5>. Citado nas páginas 114, 115 e 116.

FREIRE, J.; KOOP, D.; SANTOS, E.; SILVA, C. T. Provenance for computational tasks: A survey. **Computing in Science Engineering**, v. 10, n. 3, p. 11–21, 2008. Acessado em 19 de jun. 2018. Citado na página 39.

FRENZ, C. M. Possibilities and limitations of computer simulation. **IEEE Potentials**, v. 26, n. 2, p. 30–33, 2007. Acessado em 14 set. 2018. Disponível em: <http://doi.org/10.1109/MP.2007.343025>. Citado na página 97.

FRIEDMAN, M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. **Journal of the american statistical association**, Taylor & Francis, v. 32, n. 200, p. 675–701, 1937. Acessado em 19 de jun. 2018. Citado na página 120.

GARG, R.; SINGH, A. K. Multi-objective workflow grid scheduling based on discrete particle swarm optimization. In: PANIGRAHI, B. K.; SUGANTHAN, P. N.; DAS, S.; SATAPATHY, S. C. (Ed.). **Swarm, Evolutionary, and Memetic Computing**. Springer Berlin Heidelberg, 2011. p. 183–190. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1007/978-3-642-27172-4_23>. Citado na página 130.

____. Multi-objective workflow grid scheduling using ε-fuzzy dominance sort based discrete particle swarm optimization. **J. Supercomput.**, v. 68, n. 2, p. 709–732, 2014. Acessado em 19 de jun. 2018. Disponível em: <http://dx.doi.org/10.1007/s11227-013-1059-8>. Citado na página 130.

GARG, S. K.; BUYYA, R.; SIEGEL, H. J. Time and cost trade-off management for scheduling parallel applications on utility grids. **Future Generation Computer Systems**, v. 26, n. 8, p. 1344 – 1355, 2010. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1016/j.future.2009.07.003>. Citado na página 79.

GOLDBERG, D. E. **Genetic Algorithms in Search, Optimization and Machine Learning**. 1st. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989. Acessado em 19 de jun. 2018. Citado na página 113.

GOOGLE. **Google Acadêmico**. 2018. Acessado em 19 de jun. 2018. Disponível em: <http://scholar.google.com>. Citado na página 73.

GRAHAM, R.; LAWLER, E.; LENSTRA, J.; KAN, A. Optimization and approximation in deterministic sequencing and scheduling: a survey. In: HAMMER, P.; JOHNSON, E.; KORTE, B. (Ed.). **Discrete Optimization II**. Elsevier, 1979, (Annals of Discrete Mathematics, v. 5). p. 287 – 326. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1016/S0167-5060(08)70356-X>. Citado na página 51.

GRANDINETTI, L.; PISACANE, O.; SHEIKHALISHAHI, M. An approximate e-constraint method for a multi-objective job scheduling in the cloud. **Future Generation Computer Systems**, v. 29, n. 8, p. 1901 – 1908, 2013. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1016/j.future.2013.04.023>. Citado na página 30.

GRUSCHKA, N.; IACONO, L. L. Vulnerable cloud: Soap message security validation revisited. In: **Proceedings of 2009 IEEE International Conference on Web Services**. [s.n.], 2009. p. 625–631. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/ICWS.2009.70>. Citado na página 50.

GUO, L.; ZHAO, S.; SHEN, S.; JIANG, C. Task scheduling optimization in cloud computing based on heuristic algorithm. **Journal of Networks**, v. 7, n. 3, p. 547–553, 2012. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.4304/jnw.7.3.547-553>. Citado na página 85.

HAKEM, M.; BUTELLE, F. Reliability and scheduling on systems subject to failures. In: **2007 International Conference on Parallel Processing (ICPP 2007)**. [s.n.], 2007. p. 38–38. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/ICPP.2007.72>. Citado na página 79.

HASHIZUME, K.; ROSADO, D. G.; FERNÁNDEZ-MEDINA, E.; FERNANDEZ, E. B. An analysis of security issues for cloud computing. **Journal of Internet Services and Applications**, v. 4, n. 1, p. 5, Feb 2013. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1186/1869-0238-4-5>. Citado nas páginas 30 e 49.

HEY, A. J. G.; TANSLEY, S.; TOLLE, K. M. Jim gray on escience: a transformed scientific method. In: **The Fourth Paradigm**. [S.l.: s.n.], 2009. Acessado em 19 de jun. 2018. Citado nas páginas 36 e 38.

HEY, T. The fourth paradigm – data-intensive scientific discovery. In: KURBANOĞLU, S.; AL, U.; ERDOĞAN, P. L.; TONTA, Y.; UÇAK, N. (Ed.). **E-Science and Information Management**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 1–1. Acessado em 19 de jun. 2018. Citado na página 36.

HEY, T.; TREFETHEN, A. E. The uk e-science core programme and the grid. **Future Generation Computer Systems**, Elsevier, v. 18, n. 8, p. 1017–1031, 2002. Acessado em 19 de jun. 2018. Citado na página 29.

____. The uk e-science core programme and the grid. **Future Generation Computer Systems**, v. 18, n. 8, p. 1017 – 1031, 2002. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1016/S0167-739X(02)00082-1>. Citado na página 35.

____. Cyberinfrastructure for e-science. **Science**, American Association for the Advancement of Science, v. 308, n. 5723, p. 817–821, 2005. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1126/science.1110410>. Citado na página 37.

HO, W.; XU, X.; DEY, P. K. Multi-criteria decision making approaches for supplier evaluation and selection: A literature review. **European Journal of Operational Research**, v. 202, n. 1, p. 16 – 24, 2010. ISSN 0377-2217. Acessado em 20 de set. 2018. Disponível em: <https://doi.org/10.1016/j.ejor.2009.05.009>. Citado na página 64.

HOLLAND, J. H. **Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence**. [S.l.]: Ann Arbor : University of Michigan Press, 1975. Acessado em 19 de jun. 2018. Citado na página 62.

HÖNIG, U.; SCHIFFMANN, W. A meta-algorithm for scheduling multiple dags in homogeneous system environments. In: **Proceedings of the 18th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS'06)**. [S.l.: s.n.], 2006. Acessado em 19 de jun. 2018. Citado na página 80.

HUI, M.; JIANG, D.; LI, G.; ZHOU, Y. Supporting database applications as a service. In: **Proceedings of 25th IEEE International Conference on Data Engineering**. [s.n.], 2009. p. 832–843. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/ICDE.2009.82>. Citado na página 49.

HURWITZ, J.; KAUFMAN, M.; HALPER, F.; KIRSCH, D. **Hybrid cloud for dummies**. [S.l.]: John Wiley & Sons, 2012. Acessado em 19 de jun. 2018. Citado na página 45.

HWANG, S.; KESSELMAN, C. Grid workflow: a flexible failure handling framework for the grid. In: **High Performance Distributed Computing, 2003. Proceedings. 12th IEEE International Symposium on**. [s.n.], 2003. p. 126–137. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/HPDC.2003.1210023>. Citado na página 82.

IDC. **Public Cloud Computing to Reach Nearly $70 billion in 2015 Worldwide**. [S.l.], 2015. Acessado em 19 de jun. 2018. Disponível em: <https://www.idc.com/getdoc.jsp?containerId=prUS25797415>. Citado na página 84.

IEEE. 2018. Acessado em 19 de jun. 2018. Disponível em: <https://ieeexplore.ieee.org>. Citado na página 68.

IVERSON, M. A.; ÖZGüNER, F. Hierarchical, competitive scheduling of multiple dags in a dynamic heterogeneous environment. **Distributed Systems Engineering**, v. 6, n. 3, p. 112, 1999. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/HCW.1998.666546>. Citado na página 81.

JAIN, R. **The art of computer systems performance analysis - techniques for experimental design, measurement, simulation, and modeling.** [S.l.]: Wiley, 1991. I-XXVII, 1-685 p. (Wiley professional computing). Acessado em 19 de jun. 2018. Citado na página 144.

JENSEN, M.; SCHWENK, J.; BOHLI, J. M.; GRUSCHKA, N.; IACONO, L. L. Security prospects through cloud computing by adopting multiple clouds. In: **Proceedings of the 2011 International Conference on Cloud Computing**. [s.n.], 2011. p. 565–572. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/CLOUD.2011.85>. Citado na página 50.

JENSEN, M.; SCHWENK, J.; GRUSCHKA, N.; IACONO, L. L. On technical security issues in cloud computing. In: **Proceedings of the 2009 IEEE International Conference on Cloud Computing**. [s.n.], 2009. p. 109–116. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/CLOUD.2009.60>. Citado na página 50.

JIANFANG, C.; JUNJIE, C.; QINGSHAN, Z. An optimized scheduling algorithm on a cloud workflow using a discrete particle swarm. **Cybern. Inf. Technol.**, Walter de Gruyter GmbH, v. 14, n. 1, p. 25–39, mar. 2014. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.2478/cait-2014-0003>. Citado nas páginas 85 e 86.

JOSANG, A. A consistent definition of authorization. In: LIVRAGA, G.; MITCHELL, C. (Ed.). **Security and Trust Management**. Springer International Publishing, 2017. p. 134–144. Acessado em 14 set. 2018. Disponível em: <http://doi.org/10.1007/978-3-319-68063-7_9>. Citado na página 96.

JUVE, G.; CHERVENAK, A.; DEELMAN, E.; BHARATHI, S.; MEHTA, G.; VAHI, K. Characterizing and profiling scientific workflows. **Future Generation Computer Systems**, v. 29, n. 3, p. 682 – 692, 2013. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1016/j.future.2012.08.015>. Citado na página 41.

____. Characterizing and profiling scientific workflows. **Future Generation Computer Systems**, v. 29, n. 3, p. 682 – 692, 2013. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1016/j.future.2012.08.015>. Citado nas páginas 116, 140, 141 e 143.

JUVE, G.; DEELMAN, E.; VAHI, K.; MEHTA, G.; BERRIMAN, B.; BERMAN, B. P.; MAECHLING, P. Data sharing options for scientific workflows on amazon ec2. In: **2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis**. [s.n.], 2010. p. 1–9. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/SC.2010.17>. Citado na página 83.

KINCAID, J. Google privacy blunder shares your docs without permission. **TechCrunch, March**, 2009. Acessado em 19 de jun. 2018. Disponível em: <https://techcrunch.com/2009/03/07/huge-google-privacy-blunder-shares-your-docs-without-permission/>. Citado na página 50.

KLAUS, K.; RAJKUMAR, B.; MUTHUCUMARU, M. A taxonomy and survey of grid resource management systems for distributed computing. **Software: Practice and Experience**, v. 32, n. 2, p. 135–164, 2002. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1002/spe.432>. Citado na página 43.

KNOWLES, J.; CORNE, D. The pareto archived evolution strategy: a new baseline algorithm for pareto multiobjective optimisation. In: **Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)**. [s.n.], 1999. v. 1, p. 105 Vol. 1. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/CEC.1999.781913>. Citado na página 80.

LATIF, R.; ABBAS, H.; ASSAR, S.; ALI, Q. Cloud computing risk assessment: A systematic literature review. In: PARK, J. J. J. H.; STOJMENOVIC, I.; CHOI, M.; XHAFA, F. (Ed.). **Future Information Technology**. [s.n.], 2014. p. 285–295. Acessado em 19 de jun. 2018. Disponível em: <http://dx.doi.org/10.1007/978-3-642-40861-8_42>. Citado na página 30.

LEE, Y. C.; SUBRATA, R.; ZOMAYA, A. Y. On the performance of a dual-objective optimization model for workflow applications on grid platforms. **IEEE Transactions on Parallel and Distributed Systems**, v. 20, n. 9, p. 1273–1284, 2009. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/TPDS.2008.225>. Citado na página 79.

LEE, Y. C.; ZOMAYA, A. Y. Energy conscious scheduling for distributed computing systems under different operating conditions. **IEEE Transactions on Parallel and Distributed Systems**, v. 22, n. 8, p. 1374–1381, 2011. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/TPDS.2010.208>. Citado na página 83.

LEON, J.; FISHER, A. L.; STEENKISTE, P. **Fail-Safe PVM: A Portable Package for Distributed Programming with Transparent Recovery**. Pittsburgh, PA, USA, 1993. Acessado em 19 de jun. 2018. Citado na página 82.

LEVENE, H. Robust tests for equality of variances. **Contributions to probability and statistics. Essays in honor of Harold Hotelling**, Stanford University Press, p. 279–292, 1961. Acessado em 19 de jun. 2018. Citado na página 120.

LI, J.; SU, S.; CHENG, X.; HUANG, Q.; ZHANG, Z. Cost-conscious scheduling for large graph processing in the cloud. In: **2011 IEEE International Conference on High Performance Computing and Communications**. [s.n.], 2011. p. 808–813. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/HPCC.2011.147>. Citado na página 79.

LI, Z.; GE, J.; YANG, H.; HUANG, L.; HU, H.; HU, H.; LUO, B. A security and cost aware scheduling algorithm for heterogeneous tasks of scientific workflow in clouds. **Future Generation Computer Systems**, v. 65, p. 140 – 152, 2016. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1016/j.future.2015.12.014>. Citado nas páginas 15, 16, 31, 85, 86, 107, 108, 109, 110, 118, 125, 132, 133, 140, 142, 144, 146 e 154.

LIBERATI, A.; ALTMAN, D. G.; TETZLAFF, J.; MULROW, C.; GØTZSCHE, P. C.; IOANNIDIS, J. P.; CLARKE, M.; DEVEREAUX, P. J.; KLEIJNEN, J.; MOHER, D. The prisma statement for reporting systematic reviews and meta-analyses of studies that evaluate health care interventions: explanation and elaboration. **PLoS medicine**, Public Library of Science, v. 6, n. 7, 2009. Acessado em 19 de jun. 2018. Disponível em: <http://dx.doi.org/10.1136/bmj.b2700>. Citado na página 69.

LIN, C.; LU, S.; FEI, X.; CHEBOTKO, A.; PAI, D.; LAI, Z.; FOTOUHI, F.; HUA, J. A reference architecture for scientific workflow management systems and the view soa solution. **IEEE Transactions on Services Computing**, v. 2, n. 1, p. 79–92, 2009. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/TSC.2009.4>. Citado na página 42.

LIU, H. bo; ABRAHAM, A.; SNášEL, V.; MCLOONE, S. Swarm scheduling approaches for work-flow applications with security constraints in distributed data-intensive computing environments. **Information Sciences**, v. 192, p. 228 – 243, 2012. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1016/j.ins.2011.12.032>. Citado nas páginas 85, 86 e 125.

LIU, K.; JIN, H.; CHEN, J.; LIU, X.; YUAN, D.; YANG, Y. A compromised-time-cost scheduling algorithm in swindew-c for instance-intensive cost-constrained workflows on a cloud computing platform. **Int. J. High Perform. Comput. Appl.**, Sage Publications, Inc., v. 24, n. 4, p. 445–456, nov. 2010. Acessado em 19 de jun. 2018. Disponível em: <http://dx.doi.org/10.1177/1094342010369114>. Citado na página 53.

LIU, L.; ZHANG, M.; LIN, Y.; QIN, L. A survey on workflow management and scheduling in cloud computing. In: **Proceedings of 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing**. [s.n.], 2014. p. 837–846. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/CCGrid.2014.83>. Citado nas páginas 38 e 51.

LUDäSCHER, B.; ALTINTAS, I.; BERKLEY, C.; HIGGINS, D.; JAEGER, E.; JONES, M.; LEE, E. A.; TAO, J.; ZHAO, Y. Scientific workflow management and the kepler system: Research articles. **Concurr. Comput. : Pract. Exper.**, v. 18, n. 10, p. 1039–1065, 2006. Acessado em 19 de jun. 2018. Disponível em: <http://dx.doi.org/10.1002/cpe.v18:10>. Citado na página 40.

MALEWICZ, G.; FOSTER, I.; ROSENBERG, A. L.; WILDE, M. A tool for prioritizing dagman jobs and its evaluation. **Journal of Grid Computing**, v. 5, n. 2, p. 197–212, Jun 2007. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1007/s10723-007-9065-9>. Citado na página 81.

MARCON, D. S.; BITTENCOURT, L. F.; DANTAS, R.; NEVES, M. C.; MADEIRA, E. R. M.; FERNANDES, S.; KAMIENSKI, C. A.; BARCELOS, M. P.; GASPARY, L. P.; FONSECA, N. L. S. da. Workflow specification and scheduling with security constraints in hybrid clouds. In: **2nd IEEE Latin American Conference on Cloud Computing and Communications**. [s.n.], 2013. p. 29–34. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/LatinCloud.2013.6842219>. Citado nas páginas 84 e 86.

MARINESCU, D. C. Cloud service providers and the cloud ecosystem. In: MARINESCU, D. C. (Ed.). **Cloud Computing**. Second edition. Morgan Kaufmann, 2018. p. 13 – 49. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1016/B978-0-12-812810-7.00002-9>. Citado na página 48.

MARK, B.; RAJKUMAR, B. Cluster computing: the commodity supercomputer. **Software: Practice and Experience**, v. 29, n. 6, p. 551–576, 1999. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1002/(SICI)1097-024X(199905)29:6<551::AID-SPE248>3.0.CO;2-C>. Citado na página 43.

MARLER, R.; ARORA, J. Survey of multi-objective optimization methods for engineering. **Structural and Multidisciplinary Optimization**, v. 26, n. 6, p. 369–395, Apr 2004. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1007/s00158-003-0368-6>. Citado na página 63.

MARY, N. A. B.; JAYAPRIYA, K. An extensive survey on qos in cloud computing. **International Journal of Computer Science and Information Technologies**, v. 5, p. 1–5, 2014. Acessado em 19 de jun. 2018. Citado na página 53.

MCINTOSH, M.; AUSTEL, P. Xml signature element wrapping attacks and countermeasures. In: **Proceedings of the 2005 Workshop on Secure Web Services**. New York, NY, USA: ACM, 2005. p. 20–27. Acessado em 19 de jun. 2018. Disponível em: <http://doi.org/10.1145/1103022.1103026>. Citado na página 50.

MELL, P.; GRANCE, T. The nist definition of cloud computing. Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology Gaithersburg, 2011. Acessado em 19 de jun. 2018. Disponível em: <https://csrc.nist.gov/publications/detail/sp/800-145/final>. Citado nas páginas 44 e 46.

MEZMAZ, M.; MELAB, N.; KESSACI, Y.; LEE, Y.; TALBI, E.-G.; ZOMAYA, A.; TUYTTENS, D. A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems. **Journal of Parallel and Distributed Computing**, v. 71, n. 11, p. 1497 – 1508, 2011. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1016/j.jpdc.2011.04.007>. Citado na página 83.

MISHRA, R.; RASTOGI, N.; ZHU, D.; MOSSE, D.; MELHEM, R. Energy aware scheduling for distributed real-time systems. In: **Proceedings International Parallel and Distributed Processing Symposium**. [s.n.], 2003. p. 9. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/IPDPS.2003.1213099>. Citado na página 84.

MITCHELL, M. **An Introduction to Genetic Algorithms**. Cambridge, MA, USA: MIT Press, 1998. Acessado em 19 de jun. 2018. Citado nas páginas 60 e 61.

NAGADEVI, S.; SATYAPRIYA, K.; MALATHY, D. A survey on economic cloud schedulers for optimized task scheduling. **International Journal of Advanced Engineering Technology**, v. 5, p. 58–62, 2013. Acessado em 19 de jun. 2018. Citado na página 54.

NAIDU, P. S.; BHAGAT, B. Secure workflow scheduling in cloud environment using modified particle swarm optimization with scout adaptation. **International Journal of Modeling, Simulation, and Scientific Computing**, v. 09, n. 01, p. 1750064, 2018. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1142/S1793962317500647>. Citado na página 86.

NEMENYI, P. Distribution-free multiple comparisons. In: **Biometrics**. [S.l.: s.n.], 1962. v. 18, n. 2, p. 263. Acessado em 19 de jun. 2018. Citado na página 120.

PANDEY, S.; WU, L.; GURU, S. M.; BUYYA, R. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In: **2010 24th IEEE International Conference on Advanced Information Networking and Applications**. [s.n.], 2010. p. 400–407. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/AINA.2010.31>. Citado nas páginas 74 e 76.

PARETO, V. **Cours D'Economie Politique**. [S.l.]: F. Rouge, 1896. Acessado em 19 de jun. 2018. Citado na página 64.

PARK, S. M.; HUMPHREY, M. Data throttling for data-intensive workflows. In: **2008 IEEE International Symposium on Parallel and Distributed Processing**. [s.n.], 2008. p. 1–11. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/IPDPS.2008.4536306>. Citado na página 83.

PEDRAM, M. Energy-efficient datacenters. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 31, n. 10, p. 1465–1484, 2012. Acessado em 19 de jun. 2018. Disponível em: <http://doi.org/10.1109/TCAD.2012.2212898>. Citado na página 72.

PLANK, J. S.; BECK, M.; KINGSLEY, G.; LI, K. Libckpt: Transparent checkpointing under unix. In: **Proceedings of the USENIX 1995 Technical Conference Proceedings**. Berkeley, CA, USA: USENIX Association, 1995. p. 18–18. Acessado em 19 de jun. 2018. Citado na página 82.

PLANKENSTEINER, K.; PRODAN, R.; JANETSCHEK, M.; FAHRINGER, T.; MONTAGNAT, J.; ROGERS, D.; HARVEY, I.; TAYLOR, I.; BALASKÓ, Á.; KACSUK, P. Fine-grain interoperability of scientific workflows in distributed computing infrastructures. **Journal of Grid Computing**, v. 11, n. 3, p. 429–455, 2013. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1007/s10723-013-9261-8>. Acesso em: 23 de julho de 2018. Citado na página 131.

POOLA, D.; GARG, S. K.; BUYYA, R.; YANG, Y.; RAMAMOHANARAO, K. Robust scheduling of scientific workflows with deadline and budget constraints in clouds. In: **2014 IEEE 28th International Conference on Advanced Information Networking and Applications**. [s.n.], 2014. p. 858–865. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/AINA.2014.105>. Citado na página 82.

RAFIQ, M. Y.; MATHEWS, J. D.; BULLOCK, G. N. Conceptual building design: Evolutionary approach. **Journal of Computing in Civil Engineering**, v. 17, n. 3, p. 150–158, 2003. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1061/(ASCE)0887-3801(2003)17:3(150)>. Citado na página 58.

RAMAKRISHNAN, A.; SINGH, G.; ZHAO, H.; DEELMAN, E.; SAKELLARIOU, R.; VAHI, K.; BLACKBURN, K.; MEYERS, D.; SAMIDI, M. Scheduling data-intensiveworkflows onto storage-constrained distributed resources. In: **Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid '07)**. [s.n.], 2007. p. 401–409. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/CCGRID.2007.101>. Citado na página 83.

RAMAKRISHNAN, L.; KOELBEL, C.; KEE, Y. S.; WOLSKI, R.; NURMI, D.; GANNON, D.; OBERTELLI, G.; YARKHAN, A.; MANDAL, A.; HUANG, T. M.; THYAGARAJA, K.; ZAGORODNOV, D. Vgrads: enabling e-science workflows on grids and clouds with fault tolerance. In: **Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis**. [s.n.], 2009. p. 1–12. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1145/1654059.1654107>. Citado na página 83.

RAVICHANDRAN, S.; NAGANATHAN, D. E. Dynamic scheduling of data using genetic algorithm in cloud computing. **International Journal of Computing Algorithm**, Citeseer, v. 2, n. 1, p. 127–133, 2013. Acessado em 19 de jun. 2018. Disponível em: <http://doi.org/10.20894/IJCOA.101.002.001.003>. Citado na página 54.

RIMAL, B. P.; CHOI, E.; LUMB, I. A taxonomy and survey of cloud computing systems. In: **2009 Fifth International Joint Conference on INC, IMS and IDC**. [s.n.], 2009. p. 44–51. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/NCM.2009.218>. Citado na página 44.

RISTENPART, T.; TROMER, E.; SHACHAM, H.; SAVAGE, S. Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds. In: **Proceedings of the 16th ACM Conference on Computer and Communications Security**. New York, NY, USA: ACM, 2009. p. 199–212. Acessado em 19 de jun. 2018. Disponível em: <http://doi.org/10.1145/1653662.1653687>. Citado na página 50.

RODRIGUEZ, M. A.; BUYYA, R. Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds. **IEEE Transactions on Cloud Computing**, v. 2, n. 2, p. 222–235, 2014. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/TCC.2014.2314655>. Citado nas páginas 53 e 78.

____. A taxonomy and survey on scheduling algorithms for scientific workflows in iaas cloud computing environments. **Concurrency and Computation: Practice and Experience**, v. 29, n. 8, p. e4041, 2016. Acessado em 19 de jun. 2018. Disponível em: <http://dx.doi.org/10.1002/cpe.4041>. Citado nas páginas 30 e 95.

RONG, C.; NGUYEN, S. T.; JAATUN, M. G. Beyond lightning: A survey on security challenges in cloud computing. **Computers & Electrical Engineering**, v. 39, n. 1, p. 47 – 54, 2013. Acessado em 14 set. 2018. Disponível em: <https://doi.org/10.1016/j.compeleceng.2012.04.015>. Citado na página 96.

RUSSELL, N.; HOFSTEDE, A. H. M. ter; EDMOND, D.; AALST, W. M. P. van der. Workflow data patterns: Identification, representation and tool support. In: **Proceedings of the 24th International Conference on Conceptual Modeling**. Berlin, Heidelberg: Springer-Verlag, 2005. (ER'05), p. 353–368. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1007/11568322_23>. Citado na página 41.

RUSSELL, S.; NORVIG, P. **Intelligence Artificial: A modern approach**. [S.l.]: Wiley Publishing, 1995. Acessado em 19 de jun. 2018. Citado na página 55.

____. **Artificial Intelligence: A Modern Approach**. 3rd. ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2009. Acessado em 19 de jun. 2018. Citado na página 55.

SADIKU, M. N. O.; MUSA, S. M.; MOMOH, O. D. Cloud computing: Opportunities and challenges. **IEEE Potentials**, v. 33, n. 1, p. 34–36, 2014. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/MPOT.2013.2279684>. Citado na página 48.

SAIT, S. M.; YOUSSEF, H. **Iterative Computer Algorithms with Applications in Engineering: Solving Combinatorial Optimization Problems**. 1st. ed. Los Alamitos, CA, USA: IEEE Computer Society Press, 1999. Acessado em 19 de jun. 2018. Citado na página 59.

SAJID, M.; RAZA, Z. Cloud computing: Issues & challenges. In: **Proceedings of the 2013 International Conference on Cloud, Big Data and Trust**. [S.l.: s.n.], 2013. v. 20, n. 13, p. 13–15. Acessado em 19 de jun. 2018. Citado na página 53.

SAKELLARIOU, R.; ZHAO, H. A low-cost rescheduling policy for efficient mapping of workflows on grid systems. **Sci. Program.**, IOS Press, v. 12, n. 4, p. 253–262, dez. 2004. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1155/2004/930924>. Citado na página 82.

SAKELLARIOU, R.; ZHAO, H.; TSIAKKOURI, E.; DIKAIAKOS, M. D. Scheduling workflows with budget constraints. In: ____. **Integrated Research in GRID Computing: CoreGRID Integration Workshop 2005**. Springer US, 2007. p. 189–202. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1007/978-0-387-47658-2_14>. Citado na página 78.

SCHMITT, M.; WANKA, R. Particle swarm optimization almost surely finds local optima. **Theoretical Computer Science**, v. 561, p. 57 – 72, 2015. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1016/j.tcs.2014.05.017>. Citado na página 120.

SCHROEDER, R. e-sciences as research technologies: reconfiguring disciplines, globalizing knowledge. **Social Science Information**, v. 47, n. 2, p. 131–157, 2008. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1177/0539018408089075>. Citado na página 36.

SCOPUS. 2018. Acessado em 19 de jun. 2018. Disponível em: <http://www.scopus.com>. Citado na página 68.

SHARIF, S.; TAHERI, J.; ZOMAYA, A. Y.; NEPAL, S. Mphc: Preserving privacy for workflow execution in hybrid clouds. In: **2013 International Conference on Parallel and Distributed Computing, Applications and Technologies**. [s.n.], 2013. p. 272–280. Acessado em 19 de jun. 2018. Disponível em: <http://doi.org/10.1109/PDCAT.2013.49>. Citado nas páginas 31, 84, 86 e 125.

____. Online multiple workflow scheduling under privacy and deadline in hybrid cloud environment. In: **2014 IEEE 6th International Conference on Cloud Computing Technology and Science**. [s.n.], 2014. p. 455–462. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/CloudCom.2014.128>. Citado nas páginas 31, 85 e 86.

SHIELDS, M. **Control- Versus Data-Driven Workflows**. London: Springer London, 2007. 167–173 p. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1007/978-1-84628-757-2_11>. Citado na página 40.

SHISHIDO, H. Y.; ESTRELLA, J. C. Bibliometric analysis of workflow scheduling in grids and clouds. In: **36th International Conference of the Chilean Computer Science Society**. [s.n.], 2017. p. 1–9. Acessado em 19 de jun. 2018. Disponível em: <http://dx.doi.org/10.1109/SCCC.2017.8405129>. Citado nas páginas 31 e 95.

SHISHIDO, H. Y.; ESTRELLA, J. C.; TOLEDO, C. F. M.; ARANTES, M. S. Genetic-based algorithms applied to a workflow scheduling algorithm with security and deadline constraints in clouds. **Computers & Electrical Engineering**, 2017. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1016/j.compeleceng.2017.12.004>. Citado nas páginas 15, 31 e 86.

SIH, G. C.; LEE, E. A. A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures. **IEEE Transactions on Parallel and Distributed Systems**, v. 4, n. 2, p. 175–187, 1993. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/71.207593>. Citado na página 81.

SMANCHAT, S.; VIRIYAPANT, K. Taxonomies of workflow scheduling problem and techniques in the cloud. **Future Generation Computer Systems**, v. 52, p. 1 – 12, 2015. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1016/j.future.2015.04.019>. Citado nas páginas 51, 52 e 54.

SPRINGER. 2018. Acessado em 19 de jun. 2018. Disponível em: <https://link.springer.com>. Citado na página 68.

STATISTA. 2013. Acessado em 19 de jun. 2018. Disponível em: <https://www.statista.com/statistics/222309/total-number-of-objects-stored-in-amazons-s3/>. Citado na página 49.

STAVRINIDES, G. L.; KARATZA, H. D. Scheduling multiple task graphs in heterogeneous distributed real-time systems by exploiting schedule holes with bin packing techniques. **Simulation Modelling Practice and Theory**, v. 19, n. 1, p. 540 – 552, 2011. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1016/j.simpat.2010.08.010>. Citado na página 81.

STELLNER, G. Cocheck: checkpointing and process migration for mpi. In: **Proceedings of the 10th International Parallel Processing Symposium**. [s.n.], 1996. p. 526–531. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/IPPS.1996.508106>. Citado na página 82.

SZEPIENIEC, T.; BUBAK, M. Evaluation of eligible jobs maximization algorithm for dag scheduling in grids. In: BUBAK, M.; ALBADA, G. D. van; DONGARRA, J.; SLOOT, P. M. A. (Ed.). **Computational Science – ICCS 2008**. Springer Berlin Heidelberg, 2008. p. 254–263. Acessado em 19 de jun. 2018. Disponível em: <http://doi.org/10.1007/978-3-540-69384-0_31>. Citado na página 81.

TAKABI, H.; JOSHI, J. B. D.; AHN, G. J. Security and privacy challenges in cloud computing environments. **IEEE Security Privacy**, v. 8, n. 6, p. 24–31, 2010. Acessado em 19 de jun. 2018. Disponível em: <http://doi.org/10.1109/MSP.2010.186>. Citado na página 49.

TAKAHASHI, M.; KITA, H. A crossover operator using independent component analysis for real-coded genetic algorithms. In: **Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)**. [s.n.], 2001. v. 1, p. 643–649 vol. 1. Acessado em 19 de jun. 2018. Disponível em: <http://doi.org/10.1109/CEC.2001.934452>. Citado na página 113.

TALBI, E.-G. **Metaheuristics: from design to implementation**. [S.l.]: John Wiley & Sons, 2009. v. 74. Citado na página 56.

TALUKDER, A. K. M. K. A.; KIRLEY, M.; BUYYA, R. Multiobjective differential evolution for scheduling workflow applications on global grids. **Concurr. Comput. : Pract. Exper.**, John Wiley and Sons Ltd., Chichester, UK, v. 21, n. 13, p. 1742–1756, set. 2009. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1002/cpe.v21:13>. Citado nas páginas 80 e 130.

TAN, B.; MA, H.; MEI, Y. A nsga-ii-based approach for service resource allocation in cloud. In: **2017 IEEE Congr. on Evol. Comp.** [s.n.], 2017. p. 2574–2581. Acessado em 19 de jun. 2018. Disponível em: <http://dx.doi.org/10.1109/CEC.2017.7969618>. Citado na página 130.

TAO, Q.; CHANG, H.; YI, Y.; GU, C.; YU, Y. Qos constrained grid workflow scheduling optimization based on a novel pso algorithm. In: **2009 Eighth International Conference on Grid and Cooperative Computing**. [s.n.], 2009. p. 153–159. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/GCC.2009.39>. Citado na página 53.

TEP, K. S.; MARTINI, B.; HUNT, R.; CHOO, K. R. A taxonomy of cloud attack consequences and mitigation strategies: The role of access control and privileged access management. In: **2015 IEEE Trustcom/BigDataSE/ISPA**. [s.n.], 2015. v. 1, p. 1073–1080. Acessado em 19 de jun. 2018. Disponível em: <https://dx.doi.org/10.1109/Trustcom.2015.485>. Citado na página 31.

TEUSCHER, C.; MANGE, D.; STAUFFER, A.; TEMPESTI, G. Bio-inspired computing tissues: towards machines that evolve, grow, and learn. **Biosystems**, v. 68, n. 2, p. 235 – 244, 2003. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1016/S0303-2647(02)00100-4>. Citado na página 58.

TOLEDO, C.; FRANçA, P.; MORABITO, R.; KIMMS, A. Multi-population genetic algorithm to solve the synchronized and integrated two-level lot sizing and scheduling problem. **International Journal of Production Research**, Taylor & Francis, v. 47, n. 11, p. 3097–3119, 2009. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1080/00207540701675833>. Citado nas páginas 114, 118 e 142.

TOPCUOGLU, H.; HARIRI, S.; WU, M.-Y. Performance-effective and low-complexity task scheduling for heterogeneous computing. **IEEE Transactions on Parallel and Distributed Systems**, v. 13, n. 3, p. 260–274, 2002. Acessado em 19 de jun. 2018. Disponível em: <http://doi.org/10.1109/71.993206>. Citado na página 73.

TOPCUOUGLU, H.; HARIRI, S.; WU, M.-y. Performance-effective and low-complexity task scheduling for heterogeneous computing. **IEEE Trans. Parallel Distrib. Syst.**, IEEE Press, v. 13, n. 3, p. 260–274, 2002. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/71.993206>. Citado nas páginas 78 e 81.

TOSCANI, L.; VELOSO, P.; GARCEZ, R. **Complexidade de Algoritmos: Série Livros Didáticos Informática UFRGS - Vol. 13**. [S.l.]: Editora Bookman, 2009. Acessado em 19 de jun. 2018. Citado nas páginas 54 e 58.

TSAI, Y. L.; HUANG, K. C.; CHANG, H. Y.; KO, J.; WANG, E. T.; HSU, C. H. Scheduling multiple scientific and engineering workflows through task clustering and best-fit allocation. In: **2012 IEEE Eighth World Congress on Services**. [s.n.], 2012. p. 1–8. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/SERVICES.2012.15>. Citado na página 81.

UYDURAN, H. G.; İşERI, O. K.; ÜSTüNES, Y.; DURSUN, O. Optimizing wall insulation material parameters in renovation projects using nsga-ii. In: **2016 IEEE Congr. on Evol. Comput.** [s.n.], 2016. p. 4208–4213. Acessado em 19 de jun. 2018. Disponível em: <http://dx.doi.org/10.1109/CEC.2016.7744324>. Citado na página 130.

VELTE, T.; VELTE, A.; ELSENPETER, R. **Cloud computing, a practical approach**. [S.l.]: McGraw-Hill, Inc., 2009. Acessado em 19 de jun. 2018. Citado na página 47.

VIEGA, J. Cloud computing and the common man. **Computer**, v. 42, n. 8, p. 106–108, 2009. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/MC.2009.252>. Citado na página 49.

VIJAYALAKSHMI, R.; VASUDEVAN, V. Static batch mode heuristic algorithm for mapping independent tasks in computational grid. **Journal of Computer Science**, Science Publications, v. 11, n. 1, p. 224, 2015. Acessado em 19 de jun. 2018. Disponível em: <http://doi.org/10.3844/jcssp.2015.224.229>. Citado na página 54.

WALKER, S. J. Big data: A revolution that will transform how we live, work, and think. **International Journal of Advertising**, Routledge, v. 33, n. 1, p. 181–183, 2014. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.2501/IJA-33-1-181-183>. Citado na página 38.

WANG, C.; WANG, Q.; REN, K.; LOU, W. Ensuring data storage security in cloud computing. In: **2009 17th International Workshop on Quality of Service**. [s.n.], 2009. p. 1–9. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/IWQoS.2009.5201385>. Citado na página 49.

WANG, J.; KORAMBATH, P.; ALTINTAS, I.; DAVIS, J.; CRAWL, D. Workflow as a service in the cloud: Architecture and scheduling algorithms. **Procedia Computer Science**, v. 29, p. 546 – 556, 2014. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1016/j.procs.2014.05.049>. Citado nas páginas 51 e 52.

WANG, M.; RAMAMOHANARAO, K.; CHEN, J. Dependency-based risk evaluation for robust workflow scheduling. In: **2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops PhD Forum**. [s.n.], 2012. p. 2328–2335. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/IPDPSW.2012.286>. Citado na página 82.

WARI, E.; ZHU, W. A survey on metaheuristics for optimization in food manufacturing industry. **Appl. Soft Comput.**, Elsevier, v. 46, n. C, p. 328–343, 2016. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1016/j.asoc.2016.04.034>. Citado na página 56.

WATSON, P. A multi-level security model for partitioningworkflows over federated clouds. In: **2011 IEEE Third International Conference on Cloud Computing Technology and Science**. [s.n.], 2011. p. 180–188. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/CloudCom.2011.33>. Citado nas páginas 31, 84, 86 e 125.

WEN, Y.; LIU, J.; DOU, W.; XU, X.; CAO, B.; CHEN, J. Scheduling workflows with privacy protection constraints for big data applications on cloud. **Future Generation Computer Systems**, 2018. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1016/j.future.2018.03.028>. Citado nas páginas 31 e 86.

WFMC. **Workflow Management Coalition - Terminology & Glossary**. [S.l.], 1999. Acessado em 19 de jun. 2018. Citado na página 41.

WU, C.-M.; CHANG, R.-S.; CHAN, H.-Y. A green energy-efficient scheduling algorithm using the dvfs technique for cloud datacenters. **Future Generation Computer Systems**, v. 37, p. 141 – 147, 2014. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1016/j.future.2013.06.009>. Citado na página 83.

WU, Z.; NI, Z.; GU, L.; LIU, X. A revised discrete particle swarm optimization for cloud workflow scheduling. In: **2010 International Conference on Computational Intelligence and Security**. [s.n.], 2010. p. 184–188. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/CIS.2010.46>. Citado na página 78.

XIAO, P.; HU, Z.-G.; ZHANG, Y.-P. An energy-aware heuristic scheduling for data-intensive workflows in virtualized datacenters. **Journal of Computer Science and Technology**, v. 28, n. 6, p. 948–961, Nov 2013. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1007/s11390-013-1390-9>. Citado na página 84.

XIE, T.; QIN, X. Scheduling security-critical real-time applications on clusters. **IEEE Transactions on Computers**, v. 55, n. 7, p. 864–879, 2006. Acessado em 19 de jun. 2018. Disponível em: <https://dx.doi.org/10.1109/TC.2006.110>. Citado na página 92.

____. Scheduling security-critical real-time applications on clusters. **IEEE Transactions on Computers**, v. 55, n. 7, p. 864–879, 2006. Acessado em 8 de jul. 2018. Disponível em: <https://dx.doi.org/10.1109/TC.2006.110>. Citado na página 143.

YANG, X.-S.; HE, X. Bat algorithm: Literature review and applications. **Int. J. Bio-Inspired Comput.**, Inderscience Publishers, v. 5, n. 3, p. 141–149, jul. 2013. Acessado em 19 de jun. 2018. Disponível em: <http://dx.doi.org/10.1504/IJBIC.2013.055093>. Citado na página 56.

YASSA, S.; CHELOUAH, R.; KADIMA, H.; GRANADO, B. Multi-objective approach for energy-aware workflow scheduling in cloud computing environments. **The Scientific World Journal**, Hindawi Publishing Corporation, v. 2013, 2013. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1155/2013/350934>. Citado nas páginas 83 e 130.

YI, S.; KONDO, D.; ANDRZEJAK, A. Reducing costs of spot instances via checkpointing in the amazon elastic compute cloud. In: **2010 IEEE 3rd International Conference on Cloud Computing**. [s.n.], 2010. p. 236–243. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/CLOUD.2010.35>. Citado na página 82.

YU, J.; BUYYA, R. A taxonomy of scientific workflow systems for grid computing. **SIGMOD Record**, ACM, New York, NY, USA, v. 34, n. 3, p. 44–49, set. 2005. Acessado em 19 de jun. 2018. Disponível em: <http://doi.acm.org/10.1145/1084805.1084814>. Citado na página 38.

____. A budget constrained scheduling of workflow applications on utility grids using genetic algorithms. In: **2006 Workshop on Workflows in Support of Large-Scale Science**. [s.n.], 2006. p. 1–10. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/WORKS.2006.5282330>. Citado na página 79.

____. Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms. **Sci. Program.**, IOS Press, v. 14, n. 4, p. 217–230, 2006. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1155/2006/271608>. Citado na página 78.

YU, J.; KIRLEY, M.; BUYYA, R. Multi-objective planning for workflow execution on grids. In: **Proceedings of the 8th IEEE/ACM International Conference on Grid Computing**. IEEE Computer Society, 2007. (GRID '07), p. 10–17. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/GRID.2007.4354110>. Citado na página 80.

YU, J.; RAMAMOHANARAO, K.; BUYYA, R. **Deadline/Budget-Based Scheduling of Workflows on Utility Grids**. John Wiley & Sons, Inc., 2009. 427–450 p. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1002/9780470455432.ch19>. Citado na página 78.

YU, Z.; SHI, W. A planner-guided scheduling strategy for multiple workflow applications. In: **2008 International Conference on Parallel Processing - Workshops**. [s.n.], 2008. p. 1–8. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/ICPP-W.2008.10>. Citado na página 81.

ZAIDAN, B.; ZAIDAN, A.; AL-FRAJAT, A.; JALAB, H. On the differences between hiding information and cryptography techniques: An overview. **Journal of Applied Sciences(Faisalabad)**, Asian Network for Scientific Information, 308-Lasani Town Sargodha Rd Faislabad 38090 Pakistan, v. 10, n. 15, p. 1650–1655, 2010. Acessado em 14 set. 2018. Citado na página 96.

ZENG, L.; VEERAVALLI, B.; LI, X. Saba: A security-aware and budget-aware workflow scheduling strategy in clouds. **Parallel and Distributed Computing**, v. 75, p. 141 – 151, 2015. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1016/j.jpdc.2014.09.002>. Citado nas páginas 31, 85, 86, 125 e 140.

ZHANG, Q.; CHENG, L.; BOUTABA, R. Cloud computing: state-of-the-art and research challenges. **Journal of Internet Services and Applications**, v. 1, n. 1, p. 7–18, May 2010. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1007/s13174-010-0007-6>. Citado na página 47.

ZHAO, L.; REN, Y.; SAKURAI, K. Reliable workflow scheduling with less resource redundancy. **Parallel Computing**, v. 39, n. 10, p. 567 – 585, 2013. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1016/j.parco.2013.06.003>. Citado na página 53.

ZHENG, W.; SAKELLARIOU, R. Budget-deadline constrained workflow planning for admission control. **Journal of Grid Computing**, v. 11, n. 4, p. 633–651, Dec 2013. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1007/s10723-013-9257-4>. Citado nas páginas 30 e 78.

____. Stochastic dag scheduling using a monte carlo approach. **Journal of Parallel and Distributed Computing**, v. 73, n. 12, p. 1673 – 1689, 2013. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1016/j.jpdc.2013.07.019>. Citado na página 82.

ZHU, Q.; ZHU, J.; AGRAWAL, G. Power-aware consolidation of scientific workflows in virtualized environments. In: **2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis**. [s.n.], 2010. p. 1–12. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.1109/SC.2010.43>. Citado na página 83.

ZHU, X.; ZHA, Y.; JIAO, P.; CHEN, H. Security-aware workflow scheduling with selective task duplication in clouds. In: **Proceedings of the 24th High Performance Computing Symposium**. Society for Computer Simulation International, 2016. (HPC '16), p. 16:1–16:8. Acessado em 19 de jun. 2018. Disponível em: <https://doi.org/10.22360/SpringSim.2016.HPC.048>. Citado nas páginas 85, 86 e 125.

ZITZLER, E.; LAUMANNS, M.; THIELE, L.; ZITZLER, E.; ZITZLER, E.; THIELE, L.; THIELE, L. **SPEA2: Improving the strength Pareto evolutionary algorithm**. [S.l.]: Eidgenössische Technische Hochschule Zürich (ETH), Institut für Technische Informatik und Kommunikationsnetze (TIK) Zürich, Switzerland, 2001. Acessado em 19 de jun. 2018. Citado na página 80.

ICMC
USP
SÃO CARLOS