Análise de Desempenho da Distribuição de *Workflows* Científicos em Nuvens com Restrições de Confidencialidade*

Rodrigo A. P. Silva[1], Esther Pacitti[2], Yuri Frota[1], Daniel de Oliveira[1]

[1]Universidade Federal Fluminense (UFF)

{rprado, yuri, danielcmo}@ic.uff.br

[2]INRIA,University of Montpellier, CNRS, LIRMM, França

Esther.Pacitti@inria.fr

Abstract. *Clouds provide an on-demand environment that allows scientists to migrate their local experiments to an elastic environment. Experiments are modeled as scientific workflows, and many of them are computing and data-intensive. The storage of these data is a concern, as confidentiality can be compromised. Malicious users may infer knowledge of the results and structure of workflows. Data dispersion and encryption can be adopted to increase confidentiality, but these mechanisms cannot be adopted uncoupled from workflow scheduling, at the risk of increasing execution time and financial costs. In this paper, we present SaFER (workflow Scheduling with conFidEntity pRoblem), a scheduling approach that considers data confidentiality constraints.*

Resumo. *As nuvens fornecem um ambiente sob demanda que permite que cientistas migrem seus experimentos locais para um ambiente elástico. Os experimentos são modelados como workflows científicos, e muito deles são intensivos em computação e produção de dados. O armazenamento desses dados preocupa, uma vez que a confidencialidade pode ser comprometida. Usuários maliciosos podem inferir conhecimento dos resultados e da estrutura dos workflows. Dispersão dos dados e criptografia podem ser adotados para aumentar a confidencialidade, mas esses mecanismos não pode ser adotados de forma desacoplada ao escalonamento do workflow, sob o risco de aumentar o tempo de execução e despesas. Nesse artigo, apresentamos a SaFER (workflow Scheduling with conFidEntiality pRoblem), uma abordagem de escalonamento que considera restrições de confidencialidade dos dados.*

1. Introdução

Os *workflows* científicos (doravante denominados apenas como *workflows*) são um arcabouço para modelar as etapas de uma simulação científica [de Oliveira et al. 2019]. Essas etapas envolvem a execução de múltiplas aplicações com parâmetros de entrada e de saída. Os *workflows* são modelados como grafos dirigidos, nos quais os nós representam as *atividades* (*i.e.,* associadas à um programa, *script*, *etc.*) e os arcos representam as *dependências de dados* entre as atividades [Deelman et al. 2015]. Denominamos *ativação* a execução de uma atividade que consome um dado específico (de entrada ou produzido pelo próprio *workflow*). Esses *workflows* geralmente são modelados e executados usando mecanismos complexos chamados de Sistemas de Gerência de *Workflows* (SGWfs). Os

*Os autores agradecem ao CNPq, CAPES a FAPERJ por financiarem parcialmente esse trabalho.

SGWfs apoiam a composição, a execução, o monitoramento e a captura de dados de proveniência [Freire et al. 2008] (histórico de execuções para fins de reprodutibilidade) dos *workflows*. Em muitos domínios da ciência, como a biologia, a química e a astronomia [de Oliveira et al. 2019] é comum que um mesmo *workflow* seja executado múltiplas vezes até que o cientista possa confirmar ou refutar uma hipótese. Essas múltiplas execuções, aliadas ao grande volume de dados a ser processado, requerem ambientes de processamento de alto desempenho ou de processamento escalável em uso de dados, como *clusters* de máquinas *commodity* e supercomputadores, para que os *workflows* produzam resultados em tempo hábil [Deelman et al. 2015].

Muitos cientistas migraram seus *workflows* para nuvens [González et al. 2009], uma vez que as nuvens de computadores oferecem uma ampla variedade de recursos sob demanda, desde máquinas virtuais (VMs), áreas de armazenamento, *etc*. Além disso, as nuvens oferecem elasticidade, o que pode ser valioso para a execução de *workflows* em larga escala. Muitos SGWfs permitem a execução de *workflows* em nuvens como o Pegasus [Deelman et al. 2015] e o SciCumulus [de Oliveira et al. 2010]. Para permitir a execução dos *workflows* de forma otimizada, cada ativação necessita ser escalonada para uma VM. Logo, o problema de escalonamento se torna decidir em qual VM executar cada uma das ativações do *workflow*. O escalonamento de ativações de *workflows* é um conhecido problema NP-Difícil. Os SGWfs já oferecem uma série de algoritmos de escalonamento que exploram características das nuvens. Ao executar *workflows* na nuvem, os SGWfs existentes comumente se baseiam em áreas de armazenamento compartilhado para gravar os dados produzidos (*e.g.*, chamaremos essas áreas de *buckets*), *e.g.*, o Pegasus provê mecanismos para armazenar dados no Amazon S3. Da mesma forma, o SciCumulus usa o *s3fs3* (sistema de arquivos compartilhado sobre o S3) para armazenamento e recuperação de dados. Embora a utilização de serviços de armazenamento na nuvem facilite a gerência dos dados produzidos e consumidos de um *workflow*, a confidencialidade dos resultados é ainda uma questão aberta [Ennajjar et al. 2017]. Nos últimos anos, muitos cientistas evitaram migrar seus experimentos para a nuvem por questões de confidencialidade. Os arquivos produzidos e consumidos por um *workflow* podem conter dados confidenciais, e, se os mesmos forem armazenados em conjunto e um usuário malicioso tiver acesso ao *bucket*, o mesmo pode inferir seus resultados, o que não é desejável.

Para ilustrar esse problema, consideremos um fragmento do *workflow* Montage [Deelman et al. 2015], apresentado na Figura 1. O Montage cria uma imagem em mosaico a partir de múltiplas imagens astronômicas menores. Apesar do Montage possuir 10 atividades, na Figura 1, apresentamos somente duas (*mProjectPP*, em azul, e *mDiffFit*, em amarelo). Cada atividade possuem ativações associadas, cada uma consumindo arquivos de entrada diferentes. Por exemplo, três ativações de *mProjectPP* são apresentadas na Figura 1 (ID00000, ID00001 e ID00002), cada uma consumindo arquivos de entrada diferentes (somente *region.hdr* é consumido pelas três ativações). Os arquivos *p2mass-atlas-ID00000s-jID00000_area.fits* e *p2mass-atlas-ID00000s-jID00000.fits* são gerados pela ativação ID00000 (da atividade *mProjectPP*), que por sua vez consome os arquivos *region.hdr* e *2mass-atlas-ID00000s-jID00000.fits*. Caso os dois arquivos produzidos sejam armazenados em conjunto com os arquivos de entrada, um usuário malicioso pode inferir que tipo de transformação foi aplicada nos dados. Logo, *p2mass-atlas-ID00000s-jID00000_area.fits* e *p2mass-atlas-ID00000s-jID00000.fits* não devem ser armazenados em conjunto. Além disso, não devem ser armazenados em conjunto com os arquivos *region.hdr* e *2mass-atlas-ID00000s-jID00000.fits*. Entretanto, nem todos os arquivos possuem

restrições de armazenamento entre si. Por exemplo, os arquivos *p2mass-atlas-ID00000s-jID00000_area.fits* e *p2mass-atlas-ID00000s-jID00000.fits* podem ser armazenado em conjunto com o arquivo *p2mass-atlas-ID00000s-jID00001_area.fits*, pois não foram gerados pela mesma ativação, e nem a partir dos mesmos arquivos de entrada. Essa definição de quais arquivos serão armazenados em conjunto em um *bucket* se dá o nome de *plano de dispersão*. De fato, a *dispersão* é uma das principais técnicas para garantir confidencialidade de dados em nuvens [Branco-Jr. et al. 2016]. As restrições de armazenamento necessárias para gerar um plano de dispersão podem ser representadas como um *grafo de conflito* (Figura 1(à direita)), onde cada arco ligando dois arquivos representa um conflito (*i.e.*, esses dois arquivos não devem ser armazenados no mesmo *bucket*).

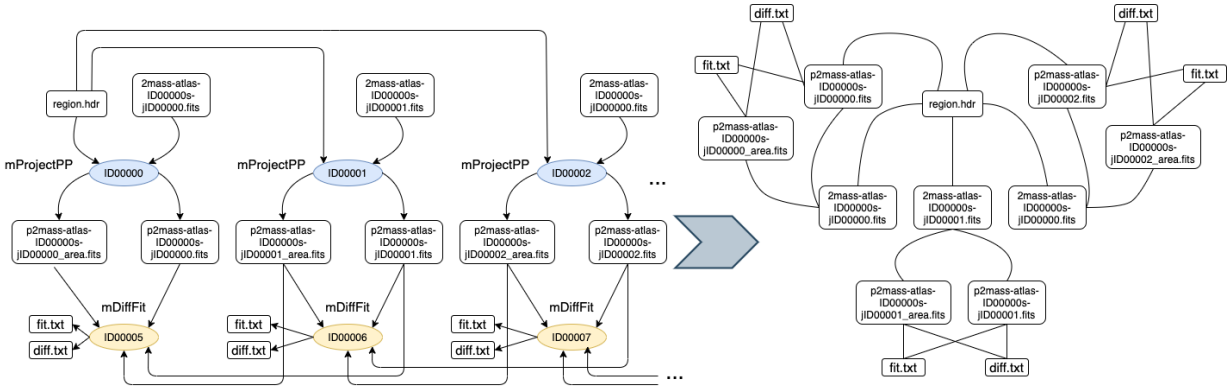

Figura 1. Montage (esquerda) e exemplo de grafo de conflitos do Montage (direita).

Um bom plano de dispersão de dados deve considerar conflitos entre os arquivos e a capacidades/custo de armazenamento nos *buckets*. Abordagens existentes já tratam do problema de dispersão de dados para garantir a confidencialidade [Guerine et al. 2019]. Porém, tais abordagens dispersam os dados desacoplados do escalonamento das ativações do *workflow* na nuvem. Essa dispersão sem considerar a VM que a ativação que consumirá os dados irá executar, pode inserir um *overhead* de transferência desnecessário, *e.g.*, um arquivo pode ser armazenado em um *bucket* da Amazon S3 em São Paulo e a ativação que o consome executar no Japão. Logo, o plano de dispersão dos dados integrado ao escalonamento das ativações é fundamental. Além disso, [Branco-Jr. et al. 2016] mencionam que a criptografia nas VMs aumenta o nível de segurança nos dados, e devem ser consideradas complementarmente ao plano de dispersão.

Nesse artigo, de forma a garantir a confidencialidade dos dados produzidos e ao mesmo tempo reduzir o tempo total e o custo de execução do *workflow*, propomos uma abordagem denominada SaFER (*workflow Scheduling with conFidEntiality pRoblem*) para escalonamento de *workflows*, que leva em consideração a variedade e a heterogeneidade de VMs na nuvem (por exemplo, diferentes larguras de banda, taxas de transferência e capacidades de processamento), os mecanismos de segurança oferecidos por cada VM (*e.g.*, criptografia) e o grafo de conflito entre os arquivos, todos integrados. Assim, o escalonador do SGWf define a distribuição de ativações nas VMs e o plano de dispersão dos arquivos entre os múltiplos *buckets* minimizando o tempo total de execução e a chance de um usuário malicioso acessar os dados e inferir o resultado do experimento. As contribuições do presente artigo são: (i) a formulação do problema SaFER como um problema de programação matemática mista, (ii) a definição de um algoritmo de escalonamento de *workflows* com restrições de confidencialidade, e (iii) uma avaliação experimental da abordagem proposta

usando *benchmarks* tradicionais da área de *workflows* científicos. Esse artigo está organizado em cinco seções além desta introdução. Na Seção 2 apresentamos os trabalhos relacionados. Na Seção 3 é apresentada a formulação matemática, e na Seção 4 a heurística de escalonamento proposta. Na Seção 5, discutimos a avaliação experimental, e na Seção 6 concluímos o presente artigo.

2. Trabalhos Relacionados

O escalonamento de ativações de *workflows* em nuvem é um tópico amplamente discutido na literatura. Existem diversas abordagens de escalonamento que consideram tempo de transferência, custo financeiro, redução de *makespan*, *etc*. [Liu et al. 2015]. Entretanto, poucas abordagens focam no escalonamento de *workflows* considerando questões de confidencialidade e/ou segurança [Sujana et al. 2019, Abazari et al. 2019, Tawfeek and AbdulHamed 2018, Shishido et al. 2018, Guerine et al. 2019]. A abordagem OPTIC [Guerine et al. 2019] propõe uma heurística para dispersão de dados produzidos por *workflows* em *buckets* de provedores de nuvem. Apesar de representar um avanço no que tange questões de confidencialidade de dados, a OPTIC não considera o escalonamento do *workflow* para planejar a dispersão dos dados, *i.e.*, um dado pode ser armazenado em um *bucket* geograficamente distante da VM em que será processado. Isso pode ser um problema tanto em termos de *makespan* (já que o tempo de transferência será impactado) quanto em termos financeiros (custos de transferência e pagamento do tempo em que a VM se mantém inativa aguardando a transferência dos dados.

Por outro lado, existem trabalhos que propõem algoritmos de escalonamento de *workflows* que consideram questões de segurança no processo. [Shishido et al. 2018] propõem uma abordagem de escalonamento de *workflows* em nuvem que considera restrições de níveis de segurança para as VMs envolvidas na execução, *i.e.*, o algoritmo só executa determinadas ativações em VMs que oferecem os requisitos de segurança necessários. Apesar de ser um avanço [Shishido et al. 2018] não consideram questões de dispersão de dados, *i.e.*, o requisito de segurança é considerado no processamento da ativação, mas não onde os dados são armazenados. De forma similar, [Tawfeek and AbdulHamed 2018] resolve o problema de escalonamento de *workflows* em nuvem por meio de um algoritmo genético que considera as restrições de segurança, prazo e custo. Assim como o trabalho de [Shishido et al. 2018], as restrições de segurança não consideram questões de armazenamento de dados. [Sujana et al. 2019] e [Abazari et al. 2019] propõem abordagens de escalonamento multi-critério que consideram as demandas de segurança das ativações e as interações na distribuição de ativações seguras na nuvem. De maneira complementar, o i-OBJECT é uma abordagem que visa aumentar o nível de confidencialidade dos dados usando fragmentação, decomposição e dispersão de informações para dividir os dados em partes irreconhecíveis e armazená-los hosts distribuídos na nuvem [Branco-Jr. et al. 2016].

3. Formulação Matemática

O problema tratado pela abordagem SaFER pode ser definido como um problema de programação matemática mista como descrito a seguir. Seja N o conjunto de ativações de um *workflow* a serem executadas. Vamos definir $\overline{M} = M \cup B$ como o conjunto de todos os recursos computacionais envolvidos na execução, onde M é o conjunto de todas as VMs com capacidade de processamento/armazenamento local, e B é o conjunto de *buckets* que possuem apenas capacidade de armazenamento (*i.e.*, não processam dados). Cada VM

$j \in M$ possui uma capacidade de armazenamento máxima cm_j e um custo financeiro de c_j^M (por unidade de tempo). Similarmente, cada *bucket* $j \in B$ possui uma faixa de intervalos de uso de dados $\{0...|L_j|\}$, onde cada intervalo $l \in L_j$ possui um tamanho sm_{jl} e um custo financeiro de uso c_{jl}^B (L_0 representa que o *bucket* está vazio). Essa definição de uso de intervalo de dados no *bucket* segue a organização de provedores como Amazon AWS e Azure. Vamos também definir $D = D_s \cup D_d$ como o conjunto de todos os dados produzidos/consumidos em uma execução do *workflow*, onde cada dado $d \in D$ tem tamanho $W(d)$ e pode ser tanto estático (D_s), previamente armazenado no seu recurso computacional de origem $O(d) \in \overline{M}$ (dados de entrada do *workflow*), ou dinâmico (D_d), *i.e.*, o dado será gerado durante a execução do *workflow*. Além disso, para cada ativação $i \in N$, definimos um conjunto de dados de entrada $\Delta_{in}(i) \subseteq D$ necessário para sua execução, e um conjunto de dados de saída $\Delta_{out}(i) \subseteq D_d$ gerados pela execução da ativação. Definimos também os requisitos do usuário c_{max} como o custo financeiro máximo permitido (*i.e., budget*)), e t_{max} como o tempo máximo de processamento do *workflow* (*i.e.*, *deadline*), onde $T = \{1, \ldots, t_{max}\}$ é definido como o conjunto de períodos de tempo. Denotamos t_{ij} como o tempo de processamento da ativação $i \in N$ executada na VM $j \in M$. Similarmente, denotamos $\overrightarrow{t}_{djp}$ como o tempo gasto pela VM $j \in M$ para ler o dado $d \in D$ armazenado no recurso computacional $p \in \overline{M}$, e $\overleftarrow{t}_{djp}$ como o tempo gasto pela VM $j \in M$ para escrever o dado $d \in D_d$ no recurso computacional $p \in \overline{M}$.

Em relação aos requisitos de confidencialidade do *workflow*, definimos o grafo de conflito de dados (nesse artigo usamos arquivos como a unidade mínima, mas caso uma fragmentação dos dados fosse aplicada, cada fragmento seria considerado um vértice do grafo) $G_c = (D, E_h \cup E_s)$, formado pelo conjunto de vértices (dados) D e o conjunto de arestas $E_h \cup E_s$. O conjunto E_h é definido como o conjunto de pares de dados (d_1, d_2) onde $d_1 \in D$ e $d_2 \in D$ que não podem ser armazenados no mesmo recurso computacional por razões de confidencialidade. Similarmente, o conjunto de arestas E_s é definido como o conjunto de pares de dados (d_1, d_2) que se forem armazenados no mesmo recurso computacional, uma penalidade de confidencialidade $p_{d_1 d_2}$ deve ser cobrada. Definimos também R como o conjunto dos requisitos de segurança presentes nas ativações do *workflow*, onde cada requisito $r \in R$ pode variar de intensidade no intervalo $\{0...l_{max}^r\}$. Um requisito de segurança pode ser a ativação $i \in N$ requisitar execuções em VMs que ofereçam alto nível de criptografia (o nivel pode ser definido de acordo com as intensidades dos intervalos, explicadas a seguir). Dessa forma, definimos que cada ativação $i \in N$ possui uma necessidade de l_{task}^{ri} em relação ao requisito $r \in R$, e que cada VM $j \in M$ pode oferecer l_{vm}^{rj} de segurança em relação a este requisito (onde $0 \leq l_{task}^{ri}, l_{vm}^{rj} \leq l_{max}^r$). Dessa forma, podemos definir o grau de exposição máxima de confidencialidade de um escalonamento como $s_{max} = \sum_{r \in R} |N| \cdot l_{max}^r + \sum_{(d_1,d_2) \in E_s} p_{d_1 d_2}$. A Tabela 3 descreve o conjunto de variáveis binárias e contínuas do modelo descrito a seguir:

Var.	***Descrição das variáveis binárias***
x_{ijt}	Indica se a ativação $i \in N$ começa sua execução na VM $j \in M$ no período $t \in T$, ou não.
$\overrightarrow{x}_{idjpt}$	Indica se a ativação $i \in N$ executando na VM $j \in M$ começa a ler o dado $d \in \Delta_{in}(i)$ armazenado no recurso $p \in \overline{M}$ no período $t \in T$, ou não.
$\overleftarrow{x}_{idjpt}$	Indica se a ativação $i \in N$ executando na VM $j \in M$ começa a escrita do dado $d \in \Delta_{out}(i)$ no recurso $p \in \overline{M}$ no período $t \in T$, ou não.
y_{djt}	Indica se o dado $d \in D$ está armazenado no recurso $j \in \overline{M}$ no período $t \in T$, ou não.
$\overline{y}_{dj}$	Indica se o dado $d \in D$ está armazenado no recurso $j \in \overline{M}$ em algum momento.
$w_{d_1 d_2}$	Indica se os dados d_1 e d_2 estão armazenados no mesmo local ou não, onde $(d_1, d_2) \in E_s$.
b_{jl}	Indica se o *bucket* $j \in B$ está sendo usado no intervalo $l \in L_j$, ou não.
v_{jt}	Indica se a VM $j \in M$ está contratada no período $t \in T$, ou não.
Var.	***Descrição das varíaveis contínuas***
e_{ri}	Indica o nível de exposição da ativação $i \in N$ em relação ao requisito de segurança $r \in R$.
q_{jl}	Contem a quantidade (tamanho) total de dados alocada no *bucket* $j \in B$ no intervalo $l \in L_j$.
z_T^j	Indica o tempo total de uso da VM $j \in M$.
z^T	Indica o tempo total de processamento do *workflow* (*makespan*).

A função objetivo procura minimizar 3 fatores: o *makespan* (1), o custo financeiro (2) e o nível de exposição de confidencialidade (3), onde os pesos α_t, α_b e α_s definem a relevância de cada um destes 3 objetivos, respectivamente. Além disso, o valor de cada um dos objetivos é normalizado por seus valores máximos definidos por t_{max}, c_{max} e s_{max} respectivamente.

$$\min \quad \alpha_t \cdot (\frac{z^T}{t_{max}}) + \tag{1}$$

$$\alpha_b \cdot (\sum_{j \in M} \frac{c_j^M z_j^T}{c_{max}} + \sum_{j \in B} \sum_{l \in L_j} \frac{c_{jl}^B q_{jl}}{c_{max}}) \tag{2}$$

$$\alpha_s \cdot (\sum_{r \in R} \sum_{i \in N} \frac{e_{ri}}{s_{max}} + \sum_{(d_1,d_2) \in E_s} \frac{p_{d_1 d_2} w_{d_1 d_2}}{s_{max}}) \tag{3}$$

As Restrições (4) garantem que cada ativação será executada, enquanto que as Restrições (5) e (6) obrigam que cada operação de leitura e escrita deve ser realizada.

$$\sum_{j \in M} \sum_{t \in T} x_{ijt} = 1, \qquad \forall i \in N \tag{4}$$

$$\sum_{\substack{j \in M \\ p \in \overline{M}}} \sum_{t \in T} \overrightarrow{x}_{idjpt} = 1, \qquad \forall i \in N, \forall d \in \Delta_{in}(i) \tag{5}$$

$$\sum_{\substack{j \in M \\ p \in \overline{M}}} \sum_{t \in T} \overleftarrow{x}_{idjpt} = 1, \qquad \forall i \in N, \forall d \in \Delta_{out}(i) \tag{6}$$

As Restrições (7) garantem que cada dado $d \in \Delta_{out}(i)$ só pode ser escrito, se a ativação i foi executada anteriormente. Além disso, as Restrições (8) obrigam que cada dado d não possa ser escrito antes da execução da ativação i (responsável por sua escrita).

$$\overleftarrow{x}_{idjpt} \leq \sum_{q=1}^{t-t_{ij}} x_{ijq}, \qquad \forall i \in N, \forall d \in \Delta_{out}(i), \forall j \in M, \forall p \in \overline{M}, \forall t = (t_{ij}+1) \cdots T_M \tag{7}$$

$$\overleftarrow{x}_{idjpt} = 0 \qquad \forall i \in N, \forall d \in \Delta_{out}(i), \forall j \in M, \forall p \in \overline{M}, 1 \leq t \leq t_{ij} \tag{8}$$

As Restrições (9) estabelecem que uma ativação só pode ser executada se todas suas leituras de dados de entrada foram anteriormente concluídas.

$$x_{ijt} \leq \sum_{p \in \overline{M}} \sum_{q=1}^{t-\overrightarrow{t}_{djp}} \overrightarrow{x}_{idjpq}, \qquad \forall i \in N, \forall d \in \Delta_{in}(i), \forall j \in M, \forall t \in T, \text{ such } (t - \overrightarrow{t}_{djp}) \geq 1 \tag{9}$$

As Restrições (10) garantem que apenas uma ação ativa (execução, leitura ou escrita) possa ser realizada em cada período de tempo em cada VM. Por outro lado, as Inequações (11) definem que ações passivas (um dado ser lido ou ser escrito) podem ser realizadas em paralelo. Note que se uma ação (ativa ou passiva) é realizada por uma VM j no período t, a variável de contratação v_{jt} terá valor 1, indicando que a máquina está contratada.

$$\sum_{i \in N} \sum_{q=\max(1,t-t_{ij}+1)}^{t} x_{ijq} + \sum_{i \in N} \sum_{d \in \Delta_{out}(i)} \sum_{p \in \overline{M}} \sum_{r=\max(1,t-\overleftarrow{t}_{djp}+1)}^{t} \overleftarrow{x}_{idjpr} +$$

$$\sum_{i \in N} \sum_{d \in \Delta_{in}(i)} \sum_{p \in \overline{M}} \sum_{r=\max(1,t-\overrightarrow{t}_{djp}+1)}^{t} \overrightarrow{x}_{idjpr} \leq v_{jt}, \qquad \forall j \in M, \forall t \in T \tag{10}$$

$$\sum_{i \in N} \sum_{d \in \Delta_{out}(i)} \sum_{p \in M} \sum_{r=\max(1,t-\overleftarrow{t}_{dpj}+1)}^{t} \overleftarrow{x}_{idpjr} +$$

$$\sum_{i \in N} \sum_{d \in \Delta_{in}(i)} \sum_{p \in M} \sum_{r=\max(1,t-\overrightarrow{t}_{dpj}+1)}^{t} \overrightarrow{x}_{idpjr} \leq |M|.v_{jt}, \qquad \forall j \in M, \forall t \in T \tag{11}$$

As Restrições (12) estabelecem que não existem dados dinâmicos gerados no início da execução, enquanto que as Restrições (13) e (14) estabelecem que os dados estáticos estão armazenados em seus respectivos recursos origem.

$$y_{dj1} = 0, \qquad \forall d \in D_d, \forall j \in \overline{M} \quad (12)$$

$$y_{dj1} = 0, \qquad \forall d \in D_s \mid j \in (\overline{M} \backslash O(d)) \quad (13)$$

$$y_{djt} = 1, \qquad \forall d \in D_s \mid j \in O(d), \forall t \in T \quad (14)$$

As Restrições (15) e (16) conectam as variáveis de armazenamento y com as variáveis de escrita $\overleftarrow{x}$ e as de leitura $\overrightarrow{x}$, garantindo uma leitura e escrita consistente com os dados armazenados tanto em VMs quanto em *buckets* ($\overline{M}$).

$$y_{dp(t+1)} = y_{dpt} + \sum_{j \in M} \overleftarrow{x}_{idjp(t-\overleftarrow{t}_{djp}+1)} \qquad , \forall d \in D, \forall p \in \overline{M}, \forall t \in \{1 \dots t_{max} - 1\},$$

$$\text{such } (t - \overleftarrow{t}_{djp} + 1) \geq 1 \text{ and } d \in \Delta_{out}(i) \quad (15)$$

$$\sum_{j \in M} \overrightarrow{x}_{idjpt} \leq y_{dpt} \qquad , \forall i \in N, \forall d \in \Delta_{in}(i), \forall p \in \overline{M}, \forall t \in T \quad (16)$$

A capacidade de armazenamento das máquinas é limitado pelas Restrições (17). As Restrições (18) conectam a última operação de escrita em cada máquina com a variável do tempo total de execução do *workflow* z^T. Além disso, as Restrições (19) relacionam as variáveis de contratação v_{jt} e z_j^T com o objetivo de estabelecer o custo financeiro de cada VM j, enquanto que as Restrições (20) asseguram que os custos financeiros não ultrapassem o orçamento máximo disponível.

$$\sum_{d \in D} y_{djt} W(d) \leq cm_j \qquad , \forall j \in \overline{M}, \forall t \in T \quad (17)$$

$$\overleftarrow{x}_{idjpt} \cdot (t + \overleftarrow{t}_{djp}) \leq z^T \qquad , \forall i \in N, \forall d \in \Delta_{out}(i), \forall j \in M, \forall p \in \overline{M}, \forall t \in T \quad (18)$$

$$v_{jt} \cdot t \leq z_j^T, \qquad \forall j \in M, \forall t \in T \quad (19)$$

$$\sum_{j \in M} c_j^M z_j^T + \sum_{j \in B} \sum_{l \in L_j} c_{jl}^B q_{jl} \leq c_{max} \quad (20)$$

As seguintes restrições operacionais (21) precisam ser satisfeitas: uma ativação i só pode iniciar sua execução se todos seus arquivos de entrada $d \in \Delta_{in}(i)$ já estiverem disponíveis para leitura.

$$\sum_{p \in \overline{M}} \overrightarrow{x}_{idjpt} \cdot |\Delta_{in}(i)| \leq \sum_{g \in \Delta_{in}(i)} \sum_{p \in \overline{M}} y_{gpt} \qquad , \forall i \in N, \forall d \in \Delta_{in}(i), \forall j \in M, \forall t \in T \quad (21)$$

As Restrições (22) conectam as variáveis de armazenamento (temporais) y com as variáveis de armazenamento (atemporais) $\overline{y}$. Restrições (23) asseguram que dados conflitantes não sejam armazenados no mesmo recurso) (arestas fortes). De forma similar, as Inequações (24) garantem que se dois dados conflitantes são armazenados no mesmo recurso, a variável de penalização w terá valor 1 (arestas fracas). As Restrições (25) medem o nível de exposição (risco) de se executar uma ativação em relação aos requisitos de segurança.

$$y_{djt} \leq \overline{y}_{dj} \qquad , \forall d \in D, \forall j \in \overline{M}, \forall t \in T \quad (22)$$

$$\overline{y}_{d_1 j} + \overline{y}_{d_2 j} \leq 1 \qquad , \forall (d_1, d_2) \in E_h, \forall j \in \overline{M} \quad (23)$$

$$\overline{y}_{d_1 j} + \overline{y}_{d_2 j} \leq 1 + w_{d_1 d_2} \qquad , \forall (d_1, d_2) \in E_s, \forall j \in \overline{M} \quad (24)$$

$$l_{task}^{ri} - \sum_{j \in M} \sum_{t \in T} l_{vm}^{rj} x_{ijt} \leq e_{ri} \qquad , \forall i \in N, \forall r \in R \quad (25)$$

As Equações (26) definem o tamanho total dos dados armazenados em cada *bucket*. As Restrições (27) estabelecem que apenas um intervalo de armazenamento $l \in L_j$ é usado em cada *bucket* $j \in B$, enquanto as Restrições (28) limitam a quantidade de dados armazenados em cada intervalo no *bucket*. Note que as Restrições (27) e (29) estabelecem em qual intervalo de uso de dados (b_{jl}) o *bucket* se encontra, de acordo com a quantidade de dados total (q_{jl}) armazenada nele. Por fim, as Restrições (30) impõe que se um intervalo $l \in L_j \backslash \{0\}$ do *bucket* $j \in B$ está em uso, então deve existir pelo menos um dado armazenado no *bucket* j.

$$\sum_{l \in L_j} q_{jl} = \sum_{d \in D} \overline{y}_{dj} W(d) \quad , \forall j \in B \quad (26)$$

$$\sum_{l \in L_j} b_{jl} = 1 \quad , \forall j \in B \quad (27)$$

$$\sum_{d \in D} \overline{y}_{dj} W(d) \leq \sum_{l \in L_j} sm_{jl} b_{jl} \quad , \forall j \in B \quad (28)$$

$$sm_{j(l-1)}.b_{jl} \leq q_{jl} \leq sm_{jl}.b_{jl} \quad , \forall j \in B, \forall l \in L_j \backslash \{0\} \quad (29)$$

$$b_{jl} \leq \sum_{d \in D} \overline{y}_{dj} \quad , \forall j \in B, \forall l \in L_j \backslash \{0\} \quad (30)$$

4. Abordagem Heurística para Escalonamento de *Workflows* com Restrições de Confidencialidade

O problema modelado na Seção 3 pode ser resolvido por métodos exatos para instâncias pequenas (10 ativações ou menos). Porém, são impraticáveis para instâncias do mundo real pois consomem muitos recursos computacionais elevando demasiadamente o tempo de execução. As Metaheurísticas são métodos que orquestram uma interação entre procedimentos de melhorias locais e estratégias de nível superior para criar um processo capaz de escapar dos ótimos locais efetuando buscas no espaço de solução [Gendreau et al. 2010], apresentando-se como uma boa alternativa aos métodos exatos. Logo, propomos uma heurística construtiva gulosa e estocástica como parte da abordagem SaFER, denominada *SaFER-hcg*, para resolver o escalonamento de ativações com restrição de confidencialidade. Nosso objetivo é encontrar boas soluções para instâncias de grande porte.

No Algoritmo 1, construtivo, o laço mais externo (linha 2) itera até que todas as ativações sejam executadas em alguma VM. Nesse laço, é criada uma Lista de Candidatos (LC) combinando cada ativação ainda não escalonada $i \in \overline{N}$ com todas as VMs disponíveis $j \in M$ (linhas 4 a 9). Porém, nem todas as ativações estão prontas para serem adicionadas à LC porque dependem se as dependências de dados foram satisfeitas, *i.e.*, se seus arquivos de entrada já estão disponíveis (linha 6). Para calcularmos o "custo" de cada candidato, o inserimos na solução atual S e calculamos o valor da função objetivo (FO) descrita pelas Equações (1-3). Dentro do cálculo da função objetivo, definimos também os recursos computacionais $W \subseteq \overline{M}$ que irão armazenar os dados de saída da ativação i da seguinte forma: para cada dado $d \in \Delta_{out}(i)$ selecionamos aleatoriamente β recursos computacionais dentre o conjunto $\overline{M}$ e escolhemos o de menor acréscimo ao valor da FO que não irá gerar inviabilidade em relação às restrições de capacidade e confidencialidade para realizar a gravação do dado. Em seguida, ordenamos a LC em função do valor do custo (FO) de cada solução (linha 9) e criamos uma Lista Restrita de Candidatos (LRC) contendo os melhores candidatos usando o parâmetro α_{LRC} (linha 10). Posteriormente, um candidato é selecionado aleatoriamente de LRC e adicionado ao escalonamento atual S (linhas 11 e 12). O procedimento é repetido até que todas as ativações do *workflow* sejam escalonadas. Dadas as características aleatórias do algoritmo, o procedimento construtivo pode gerar diferentes soluções para um mesmo dado de entrada, *i.e.*, para uma mesma instância do *workflow*.

Em nossos experimentos, executamos o algoritmo da *SaFER-hcg* 100 vezes armazenando, sempre, o melhor resultado encontrado.

Algoritmo 1: Heurística Construtiva Gulosa-Aleatória *SaFER-hcg*

Dados: $\alpha_{LRC}, \beta, \alpha_t, \alpha_b, \alpha_s, t_{max}, c_{max}$
Resultado: escalonamento S
$S \leftarrow \emptyset, \overline{N} \leftarrow N$
enquanto $\overline{N} \neq \emptyset$ **faça**
  $LC \leftarrow \emptyset$
  para cada $i \in \overline{N}$ **faça**
    para cada $j \in M$ **faça**
      se $\Delta_{in}(i)$ *já foi gerado* **então**
        $W, FO \leftarrow CalculaFO(S, i, j, t_{max}, c_{max}, \beta, \alpha_t, \alpha_b, \alpha_s)$
        $LC \leftarrow LC \cup (i, j, W)$
  ordena LC **por** FO
  $LRC \leftarrow ListaRestritaDeCandidatos(LC, \alpha_{LRC})$
  $(i^*, j^*, W^*) \leftarrow Sorteia(LRC)$
  $S \leftarrow S \cup (i^*, j^*, W^*)$
  $\overline{N} \leftarrow \overline{N} \backslash \{i^*\}$
retorna S

Tabela 1. Detalhes das instâncias utilizadas nos testes experimentais

Instância	#Ativações	#Arquivos	t_{max}	c_{max}	Instância	#Ativações	#Arquivos	t_{max}	c_{max}
Sintética_5_A	2	3	32	120	Epigenomics_24	24	38	5.000	1.500
Sintética_5_B	2	3	32	120	Epigenomics_46	46	71	10.000	3.000
Sintética_5_C	1	4	40	80	Epigenomics_100	100	152	50.000	30.000
Sintética_10_A	4	6	50	168	Inspiral_30	30	47	1.500	1.000
Sintética_10_B	3	7	36	198	Inspiral_50	50	77	2.000	1.500
Sintética_10_C	4	6	60	440	Inspiral_100	100	151	3.000	2.000
Sintética_15_A	5	10	40	154	Montage_25	25	54	250	125
Sintética_15_B	5	10	30	54	Montage_50	50	107	500	250
Sintética_15_C	5	10	40	136	Montage_100	100	215	1.000	500
CyberShake_30	30	49	400	200	Sipht_30	29	963	3.500	2.000
CyberShake_50	50	79	500	250	Sipht_60	58	1.100	4.000	2.500
CyberShake_100	100	154	600	300	Sipht_100	97	1.223	6.000	3.500

5. Avaliação Experimental

Nesta seção, comparamos a solução heurística *SaFER-hcg* (Seção 4) com a solução exata fornecida pelo modelo matemático (*i.e., baseline*). Para compararmos as soluções, foram criadas 9 instâncias *sintéticas* de execuções de *workflows*, pequenas o suficiente para que o método exato encontre alguma solução. Além disso, executamos a solução heurística *SaFER-hcg* para instâncias reais de *workflows* com características distintas e inviáveis para execução com o método exato. De forma a garantir a reprodutibilidade da abordagem proposta, o código-fonte e as configurações de ambiente e dados de entrada se encontram disponibilizados em `https://github.com/UFFeScience/Wf-Security`. Os métodos heurísticos e exatos foram implementados com GCC 4.8.5 e IBM ILOG CPLEX 12.8; executados em um computador com processador Intel®i5-7300U 2.60GHz, com 16GB de RAM e SO Linux Fedora 31. Além disso, cada teste foi repetido 10 vezes (com sementes aleatórias diferentes) armazenando-se a médias dos resultados. O valor dos parâmetros de entrada $\alpha_{LRC} = 0.5$ e $\beta = 4$ foram definidos de forma empírica e as ponderações da função objetivo α_t, α_b, α_s foram definidas como 0,3, 0,3 e 0,4, respectivamente, baseados na experiência em trabalhos anteriores do grupo [Teylo et al. 2017].

Considerando o recursos computacionais necessários para a viabilidade da solução, utilizamos seguintes parâmetros para execução dos *workflows*: 4 VMs com diferentes capacidade de armazenamento (80, 120, 160 e 200 GBs); *slowdown* (quanto menor, maior o poder de processamento) (1,53, 0,77, 0,38 e 0,19); velocidade do enlace (4, 9, 10 e 10 Mbps); e o custo, por minuto, de processamento em dólares (US$ 0,02, US$ 0,09, US$ 0,16 e US$

0,33). Em se tratando da quantidade de *buckets*, definimos 2 para as instâncias *sintéticas* e variados números para as demais instâncias (valores apresentados a seguir). Quanto a capacidade de armazenamento, velocidade do enlace e preço cobrado por GBs de dados armazenados de cada *bucket*, definimos como 50 TB, 25 Mbps e US$ 0,023, respectivamente. Para as instâncias públicas de *workflows* tradicionais (*i.e., benchmarks*) foram utilizadas as seguintes: (i) Montage, que cria uma imagem em mosaico a partir de fotos astronômicas menores, (ii) Cybershake, usado para identificar terremotos na Califórnia, (iii) Inspiral, usado para identificar ondas gravitacionais, (iv) Epigenomics, para sequenciamento genômico e (v) Sipht, que busca por RNA não transcrito no NCBI. Foram determinadas as seguintes quantidades de *buckets* para as instâncias CyberShake_30 (o número no nome da instância define a quantidade de ativações a serem escalonadas), CyberShake_50, CyberShake_100, Epigenomics_24, Epigenomics_46, Epigenomics_100, Inspiral_30, Inspiral_50, Inspiral_100, Montage_25, Montage_50, Montage_100, Sipht_30, Sipht_50, Sipht_100: 2, 8, 19, 6, 6, 44, 3, 8, 9, 6, 11, 28, 10, 10 e 16 , respectivamente. A Tabela 1 apresenta os valores t_{max}, c_{max}, a quantidade de ativações e a quantidade de arquivos (tanto os de entrada quanto os gerados durante a execução do *workflow*) escolhidos para cada instância utilizada nos experimentos.

Criamos um *script* para ler os arquivos que contém as instâncias dos *workflows* (em formato XML) e atribuir restrições (arestas fortes e fracas) entre os arquivos automaticamente, (*i.e.,* o grafo de conflitos). Para o presente experimento, utilizamos a seguinte heurística para criação do grafo de conflitos: (i) arquivos de entrada de uma ativação são penalizados (com um valor padrão de 1) se armazenados junto dos arquivos de saída desta mesma ativação (arestas fracas), (ii) arquivos gerados por ativações em um mesmo nível do *workflow* (*i.e.*, gerados por ativações "irmãs") não podem ser armazenados juntos (arestas fortes). No caso do Montage, esta última restrição evita tanto que arquivos cabeçalho e corpo sejam armazenados juntos, e que arquivos que componham partes diferentes da solução também sejam armazenados juntos (*e.g.,* dados gerados por diferentes ativações que formam um mosaico). Quanto aos requisitos definidos na Seção 3, utilizamos valores que permitissem que todas as ativações pudessem ser executadas em qualquer VM, pois focamos nas restrições de confidencialidade, uma vez que, requisitos como autenticação ou criptografia são funcionalidades já estudadas em trabalhos anteriores (vide Seção 2).

A Tabela 2 apresenta os resultados das execuções das heurística *SaFER-hcg* e a solução exata. As colunas *Instância*, F. O., *Makespan*, Custo, Confid., Tempo representam, respectivamente, as instâncias de *workflows*, função objetivo, tempo de execução, custo e risco de exposição, das execuções da heurística *SaFER-hcg* e do método exato. Para uma melhor interpretação, a coluna *Confid.* foi normalizada pelo fator S_{max}. Os resultados indicam que o modelo matemático foi capaz de retornar soluções ótimas para todas as instâncias *sintéticas*, com exceção das instâncias Sintética_15_B e Sintética_15_C que interromperam sua execução após uma hora (tempo limite configurado para a execução do CPLEX). Por outro lado, a *SaFER-hcg* conseguiu, para as mesmas instâncias, produzir boas soluções com diferença de, em média, 14% em relação a solução encontrada pelo método exato, mas não conseguiu encontrar a solução ideal para 6 das instâncias sintéticas. Uma explicação para a diferença entre o método exato e a *SaFER-hcg* é que o método exato permite que as leituras de todas as ativações iniciais ocorram antes de qualquer execução, enquanto que a heurística *SaFER-hcg* agrega leitura, execução e escrita atomicamente; isso faz com que o recurso que contém os arquivos de entrada tenha que ser alocado por mais tempo. É interessante notar que os tempos de execução para a *SaFER-hcg* são inferiores aos executados pelo CPLEX (em torno de um centésimo de segundo).

Considerando que quanto menor os valores de *Confid.* menor o risco de exposição de dados confidenciais, os valores baixos demonstram a eficiência da abordagem proposta. Por último, somente a heurística *SaFER-hcg* conseguiu encontrar soluções para as instâncias maiores em tempo hábil, comprovando, assim, sua importância.

Tabela 2. Avaliação Experimental - *SaFER-hcg versus* Solução Exata

	Heurística (SaFER-hcg)					**Solução Exata**				
Instância	F. O.	*Makespan* (s)	Custo (US$)	Confid.	Tempo (s)	F. O.	*Makespan* (s)	Custo (US$)	Confid.	Tempo (s)
Sintética_5_A	0,25	24,00	8,32	0,00	0,01	0,25	24,00	8,32	0,00	0,16
Sintética_5_B	0,28	27,00	9,35	0,00	0,01	0,28	27,00	9,21	0,00	1,29
Sintética_5_C	0,32	36,00	12,12	0,00	0,01	0,32	36,00	12,12	0,00	0,21
Sintética_10_A	0,33	48,00	22,42	0,00	0,01	0,24	35,00	16,81	0,00	1.039,48
Sintética_10_B	0,28	31,00	14,24	0,00	0,01	0,26	29,00	11,68	0,00	60,68
Sintética_10_C	0,31	58,00	24,78	0,00	0,01	0,28	53,00	22,97	0,00	18,44
Sintética_15_A	0,15	19,00	4,90	0,00	0,01	0,11	14,00	4,59	0,00	285,60
Sintética_15_B	0,29	20,00	11,60	0,07	0,01	0,24	16,00	8,88	0,07	3.740,48 *
Sintética_15_C	0,34	36,80	17,81	0,05	0,01	0,29	29,00	16,63	0,09	3.604,57 *
CyberShake_30	0,27	158,30	93,02	0,02	0,19					
CyberShake_50	0,39	294,20	172,95	0,01	0,78					
CyberShake_100	0,46	420,40	249,78	0,01	7,15					
Epigenomics_24	0,42	2.698,50	1.204,41	0,04	0,12					
Epigenomics_46	0,43	4.928,40	2.773,21	0,02	0,40					
Epigenomics_100	0,53	43.722,10	26.285,44	0,02	7,24					
Montage_25	0,34	857,90	522,29	0,02	0,11					
Montage_50	0,42	1.532,50	944,01	0,01	0,41					
Montage_100	0,49	2.509,20	1.566,27	0,00	2,25					
Inspiral_30	0,33	130,30	68,70	0,02	0,15					
Inspiral_50	0,35	266,60	155,43	0,01	1,59					
Inspiral_100	0,37	561,90	329,79	0,00	19,13					
Sipht_30	0,49	2.869,70	1.440,52	0,06	3,26					
Sipht_60	0,55	3.664,10	2.105,33	0,06	17,16					
Sipht_100	0,46	4.622,00	2.571,43	0,03	52,63					

* A execução foi interrompida após uma hora de processamento.

6. Conclusões e Trabalhos Futuros

Diversos experimentos são modelados como *workflows*. Esses *workflows* são, geralmente, compostos por programas caixa preta com dependência de dados entre eles. Muitos dados são produzidos durante a execução de *workflows* de larga-escala. Quando essa execução é feita na nuvem, comumente os SGWfs existentes armazenam os dados produzidos em um *storage* compartilhado (*i.e.*, *bucket*). Ao fazer isso, a confidencialidade do experimento pode ser comprometida. Caso um usuário malicioso venha a acessar esses arquivos, existe o risco do mesmo inferir os resultados e/ou a estrutura do experimento. Esse risco deve ser evitado, ou, pelo menos, minimizado. Nesse artigo, formalizamos o problema de escalonamento de ativações de *workflows* com restrição de confidencialidade, modelando como um problema de otimização combinatória para preservar a confidencialidade dos dados produzidos e armazenados na nuvem. Introduzimos uma formulação matemática e um procedimento heurístico chamado *SaFER-hcg* para resolver o problema de otimização. A abordagem proposta utiliza um grafo de conflitos para integrar o escalonamento de ativações com o plano de dispersão dos dados determinando as áreas de armazenamento para os arquivos produzidos durante a execução, respeitando os limites de capacidade. A abordagem proposta foi avaliada com uma série de instâncias de *workflows* sintéticos e reais, e se mostrou promissora no que tange a garantia da confidencialidade ao mesmo tempo em que foi capaz de minimizar o *makespan*. Como trabalho futuro, pretendemos implementar uma metaheurística (*e.g.*, GRASP) a fim de melhorar a qualidade da solução heurística proposta, além de implementar o *SaFER-hcg* no SGWf SciCumulus.

Referências

Abazari, F., Analoui, M., Takabi, H., e Fu, S. (2019). Mows: multi-objective workflow scheduling in cloud computing based on heuristic algorithm. *Simulation Modelling Practice and Theory*, 93:119–132.

Branco-Jr., E. C., Monteiro, J. M., Reis, R., e Machado, J. C. (2016). A new mechanism to preserving data confidentiality in cloud database scenarios. In *ICEIS*, volume 291, pages 261–283. Springer.

de Oliveira, D., Liu, J., e Pacitti, E. (2019). *Data-Intensive Workflow Management: For Clouds and Data-Intensive and Scalable Computing Environments*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers.

de Oliveira, D., Ogasawara, E. S., Baião, F. A., e Mattoso, M. (2010). Scicumulus: A lightweight cloud middleware to explore many task computing paradigm in scientific workflows. In *IEEE CLOUD 2010, Miami*, pages 378–385.

Deelman, E., Vahi, K., Juve, G., Rynge, M., Callaghan, S., Maechling, P., Mayani, R., Chen, W., da Silva, R. F., Livny, M., e Wenger, R. K. (2015). Pegasus, a workflow management system for science automation. *FGCS*, 46:17–35.

Ennajjar, I., Tabii, Y., e Benkaddour, A. (2017). Securing data in cloud computing by classification. BDCA'17, New York, NY, USA. ACM.

Freire, J., Koop, D., Santos, E., e Silva, C. T. (2008). Provenance for Computational Tasks: A Survey. *Computing in Science & Engineering*, pages 20–30.

Gendreau, M., Potvin, J.-Y., et al. (2010). *Handbook of metaheuristics*, volume 2. Springer.

González, L. M. V., Rodero-Merino, L., Caceres, J., e Lindner, M. A. (2009). A break in the clouds: towards a cloud definition. *Comp. Comm. Rev.*, 39(1):50–55.

Guerine, M., Stockinger, M. B., Rosseti, I., Simonetti, L. G., Ocaña, K. A., Plastino, A., e de Oliveira, D. (2019). A provenance-based heuristic for preserving results confidentiality in cloud-based scientific workflows. *FGCS*, 97:697 – 713.

Liu, J., Pacitti, E., Valduriez, P., e Mattoso, M. (2015). A survey of data-intensive scientific workflow management. *J. Grid Comput.*, 13(4):457–493.

Shishido, H. Y., Estrella, J. C., e Toledo, C. F. M. (2018). Multi-objective optimization for workflow scheduling under task selection policies in clouds. In *CEC*, pages 1–8. IEEE.

Sujana, J. A. J., Revathi, T., Priya, T. S., e Muneeswaran, K. (2019). Smart pso-based secured scheduling approaches for scientific workflows in cloud computing. *Soft. Comp.*, 23(5):1745–1765.

Tawfeek, M. A. e AbdulHamed, A. A. (2018). Service flow management with multi-objective constraints in heterogeneous computing. In *ICCES*, pages 258–263. IEEE.

Teylo, L., de Paula Junior, U., Frota, Y., de Oliveira, D., e de A. Drummond, L. M. (2017). A hybrid evolutionary algorithm for task scheduling and data assignment of data-intensive scientific workflows on clouds. *Future Gener. Comput. Syst.*, 76:1–17.