Verificação da Reprodução de *Workflows* Científicos por meio de Algoritmos de Detecção de Plágio*

Filipe Tadeu Santiago, Daniel de Oliveira

Instituto de Computação – Universidade Federal Fluminense (IC/UFF)
Niterói – RJ – Brazil

filipe_santiago@id.uff.br, danielcmo@ic.uff.br

Resumo. *Atualmente, diversos experimentos científicos são realizados por intermédio de simulações computacionais. Além da especificação, execução e monitoramento desses experimentos, um dos principais desafios da comunidade científica é como podemos verificar a reprodução de tais experimentos. Para que um experimento possa ser classificado como científico, seus resultados devem ser passíveis de reprodução por terceiros em condições similares. Em experimentos modelados como workflows científicos e executados em sistemas de gerência de workflows científicos, os dados de proveniência capturados são capazes de descrever tanto a especificação do workflow quanto o histórico da execução do mesmo. Dessa forma, esses dados oferecerem a base para verificar se uma determinada execução de um experimento é a reprodução de uma execução anterior. Entretanto, como comparar os dados de proveniência de execuções distintas de workflows não é uma tarefa trivial. Para verificar a reprodução desses workflows, nesse artigo fazemos uso de algoritmos de detecção de plágio. Tais algoritmos são capazes de verificar a semelhança entre execuções bastando ter acesso à base de proveniência prospectiva e retrospectiva do experimento.*

1. Introdução

Os avanços científicos em ciência da computação nas últimas décadas possibilitaram que os cientistas de diversas áreas, como a biologia e a química, se tornassem capazes de utilizar simulações computacionais em suas pesquisas, e com isso obter resultados com mais qualidade e em menor tempo [1][2]. Muitas dessas simulações são compostas por uma série de programas que formam um fluxo coerente onde a saída produzida por um programa é a entrada do próximo no fluxo. Tal encadeamento de programas pode ser modelado como um *workflow* científico. Um *workflow* científico é uma abstração capaz de representar experimentos, onde o cientista escolhe os programas que serão executados, a ordem em que tais programas serão executados, as dependências de dados entre os programas, e após isto, submete o *workflow* a um Sistema de Gerência de *Workflows* Científicos (SGWfC). Os SGWfCs são responsáveis por gerenciar a complexa execução do *workflow*, distribuindo a execução do mesmo em ambientes de processamento de alto desempenho, além de coletar dados de proveniência [3], que descrevem tanto a especificação do *workflow* quanto o seu histórico de execução.

Em um mesmo experimento científico, um *workflow* pode ser executado *n* vezes. Cada execução do *workflow* pode consumir um subconjunto de dados e valores

* Os autores gostariam de agradecer ao CNPq, CAPES e FAPERJ por financiarem parcialmente esse trabalho

de parâmetros de forma a confirmar ou refutar uma hipótese científica. Entretanto, mesmo que uma determinada hipótese científica seja confirmada ou refutada por meio de diversas execuções de um *workflow*, o seu resultados só será considerado de fato *científico* caso o mesmo seja passível de reprodução por terceiros em condições semelhantes. Isto é, o experimento só é considerado científico caso existam garantias de reprodutibilidade [4].

Dessa forma, para que o experimento e os resultados obtidos sejam válidos perante a comunidade científica, devemos ser capazes de comparar diferentes execuções de *workflows*. Uma das formas de verificar se uma execução de um *workflow* é uma reprodução de uma execução anterior é comparar os dados de proveniência prospectiva (a partir daqui chamada de *p-prov*) e proveniência retrospectiva (a partir daqui chamada de *r-prov*) das execuções. Entretanto, essa comparação não é trivial. Além de informações que normalmente são armazenadas em bancos de dados de proveniência (*p-prov* e *r-prov*), se torna necessário comparar os conteúdos de um grande volume de arquivos gerados pelas execuções dos *workflows* (que contém efetivamente o resultado do experimento). Os SGWfCs apesar de oferecerem um arcabouço completo para especificação, execução e monitoramento de *workflows*, ainda carecem de ferramentas de análise, principalmente no que tange a verificação da reprodução de experimentos. Dessa forma, se torna crucial que novas soluções para verificação da reprodução de *workflows* sejam propostas [1].

Um problema similar é considerado na área de mineração de dados, mais especificamente na área de detecção de plágio [5]. O plágio, ou simplesmente a cópia de um conteúdo desenvolvido por terceiros sem a devida autorização, é um problema crescente na academia e na indústria [5]. Comparar conteúdos para se calcular a similaridade entre os mesmos é uma tarefa crucial na detecção de plágio. Assim, diversas técnicas e algoritmos diferentes de detecção de plágio foram propostos, alguns focados na detecção de plágio em código fonte e outras em documento de texto. Tais algoritmos e técnicas oferecem um arcabouço interessante que pode ser usado para comparar execuções de *workflows*.

Nesse artigo, é proposta uma abordagem para verificação da reprodução de *workflows* científicos chamada Sličnost. A Sličnost se beneficia de algoritmos e técnicas de detecção de plágio já existentes para verificar a reprodução de *workflows*. A ideia principal por trás da abordagem é incorporar a um SGWfC um componente que importa do banco de dados de proveniência a *p-prov*, a *r-prov* e o conteúdo de arquivos de diferentes execuções de um *workflow*, os formata e submete a algoritmos de detecção de plágio. A saída do algoritmo é então um percentual de similaridade entre as execuções do *workflow*. Todo o arcabouço foi implementado no SGWfC SciCumulus [6].

Esse artigo se encontra organizado em 4 seções além dessa introdução. A seção 2 apresenta o referencial teórico sobre detecção de plágio. A Seção 3 apresenta a abordagem proposta e a validação realizada. A Seção 4 discute trabalhos relacionados e, finalmente, a Seção 5 conclui esse artigo.

2. Referencial Teórico: Detecção de Plágio em Documentos

Com a ascensão da Internet, a quantidade de ocorrências de plágio aumentou consideravelmente, visto que apenas acessando a *Web* e buscando por um determinado assunto tem-se acesso a diversos materiais sobre os mesmos [5]. O plágio é considerado um crime, semelhante à pirataria e à falsificação [5]. Dessa forma, detectar documentos plagiados se tornou uma prioridade em todo setor acadêmico.

Devido ao grande número de informação disponível na internet, e ao grande número de trabalhos publicados na academia, é impossível que uma abordagem manual seja capaz de detectar todos os possíveis casos de plágio que existam em um documento apresentado. Com isto surge a necessidade de um procedimento automatizado que execute tal tarefa. Para cumprir esta tarefa, os algoritmos de detecção de plágio foram propostos, e até hoje, infelizmente, são necessários. Existem dois métodos para analisar se um documento foi plagiado ou não [7]: extrínseco e intrínseco.

O primeiro método se baseia basicamente em comparar um documento específico com uma base de dados de documentos. São realizados vários testes que verificam a similaridade entre o documento entregue e cada um dos que estão disponíveis na base de dados, e, uma vez que que seja encontrado um trecho semelhante a um documento da base que não fora citado, é detectado um nível de plágio [7]. Tal detecção é influenciada por diversos fatores, como por exemplo, o idioma em que o documento entregue foi redigido, o idioma que os documentos presentes na base foram redigidos, presença de sinônimos, e até mesmo a ordem em que uma frase tenha sido escrita.

O segundo método, também chamado de detecção de plágio interna, propõe verificar se no próprio documento que está sendo analisado existem trechos com padrões de escrita diferentes. Um mesmo texto com mais de um padrão de escrita pode indicar que partes foram plagiadas. Entretanto o documento deverá ser melhor analisado para atestar se é realmente um plágio ou apenas uma modificação brusca do estilo de escrita do autor. Os dois métodos são complementares, uma vez que o método intrínseco não é capaz de dizer a partir de qual documento o plágio foi realizado.

Para cada um dos métodos, diversos algoritmos para detecção de plágio já foram propostos [7]. O *N-Gram* [8] é um dos mais famosos entre os algoritmos de detecção de plágio. Ele tem como ideia principal criar uma espécie de impressão digital de um documento, e esse documento poderia ser identificado por essa impressão. Uma vez que outro documento contiver um fragmento de tal impressão, este fragmento é considerado como plágio do primeiro. O primeiro desafio desse algoritmo é como deve-se montar os *N-Grams*. O primeiro passo é remover os caracteres especiais do documento, visto que eles sempre existem em textos, e não é desejado que eles influenciem a semelhança entre os resultados. Após isto devemos separar as palavras em *substrings* de tamanho *n*.

Cada um desses *N-Grams* é submetido a um algoritmo de *hash*, formando um conjunto de *hashes*, e com isto obtêm-se um identificador para o documento, ou sua impressão digital. Se submetermos outro documento ao mesmo processo e comparamos suas impressões digitais, poderemos observar a similaridade entre ambos. Um ponto crítico nesse algoritmo é o parâmetro *n*, pois este definirá a granularidade de sua impressão digital, e se este for muito pequeno, palavras comuns em nosso vocabulário como “de”, “as”, *etc*. Seriam consideradas no texto que é usado para comparação. Por outro lado, se *n* for demasiadamente grande, utilizaremos *substrings* muito grandes, o que ignoraria muitas semelhanças entre os arquivos a serem comparados.

Para diminuir o impacto do parâmetro *n*, foi proposta a técnica dos múltiplos *N-Grams*. Nessa técnica, é utilizada uma faixa de valores para o parâmetro *n*, de modo que a taxa de semelhança entre dois documentos é dada pela média das taxas de semelhança encontradas por cada *N-Gram*. Dessa forma, o algoritmo de múltiplos *N-Grams* não fixa um único valor de *N-Gram*, podendo assim se livrar se singularidades para um

determinado valor do parâmetro *n*. Porém, esse algoritmo é computacionalmente custoso, uma vez que realiza o cálculo de similaridade entre documentos diversas vezes.

Uma alternativa menos custosa computacionalmente é o algoritmo *Winnowing* [9]. Esse algoritmo, também é baseado no uso de *N-Grams*, mas tem como objetivo principal reduzir o tamanho da impressão digital, sem fazê-la perder a qualidade, visto que existem textos muito extensos, e apenas a comparação de suas impressões digitais se tornaria deveras custosa. Basicamente, o *Winnowing* remove os caracteres não representativos do documento, forma todas as *N-Grams* possíveis e essas são submetidas a um algoritmo de *hash*. Porém, de posse dos *hashes*, o *Winnowing* procura reduzir a quantidade de informação na impressão digital do texto. Para alcançar tal objetivo ele introduz uma ideia de janela e esta janela irá conter *w hashes*, e dentre estes o de menor valor é selecionado. Uma vez escolhido esse valor de *hash*, a janela se desloca uma unidade para a direita, e seleciona o *hash* de menor valor da mesma, se não for o mesmo escolhido anteriormente. Deste modo, a comparação entre dois documentos se torna mais eficaz, pois o tamanho de sua impressão digital é menor. Porém, isto insere uma preocupação nova que é definir o valor do parâmetro *w*, visto que quanto maior a janela, mais informação estamos dispostos a perder. Esses três algoritmos foram usados na abordagem proposta nesse artigo para calcular a similaridade entre execuções de *workflows* e, assim, verificar a reprodução.

3. Abordagem Proposta: Sličnost

Nessa seção apresentamos a abordagem proposta, denominada Sličnost (similaridade de origem em Bósnio). O Sličnost é um componente que pode ser acoplado a um SGWfC e que se baseia nos dados de proveniência coletados pelo SGWfC e presentes na base de proveniência para comparar execuções diferentes de um *workflow*. Uma visão abstrata da arquitetura do Sličnost é apresentada na Figura 1. Em sua versão atual o Sličnost foi configurado para ser acoplado ao SGWfC SciCumulus e sua base de proveniência que segue o modelo PROV-Wf [10].

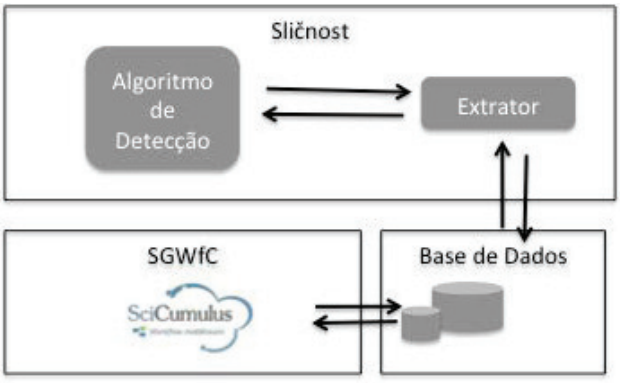

Figura 1 Arquitetura simplificada do Sličnost

O Sličnost é composto por dois componentes principais: o componente extrator e o algoritmo de detecção de plágio. O extrator é o responsável por consultar o banco de dados de proveniência do SGWfC escolhido (nesse caso o SciCumulus) e obter tanto a *r-prov* quanto a *p-prov*. Esse componente deve ser estendido para cada um dos SGWfCs que se desejar trabalhar, uma vez que cada um dos SGWfCs existentes possui um modo diferente de armazenar dados de proveniência. No caso do SciCumulus, toda a base de proveniência se encontra representada em um banco de dados relacional. Para se obter

tanto a *r-prov* quanto a *p-prov* foram realizadas consultas simples em SQL como a apresentada a seguir onde as informações das execuções das atividades são obtidas:

```
SELECT * FROM eworkflow ew, cworkflow cw, eactivity ea
WHERE ew.tagexec like '%'||cw.tag||'%'
      AND ea.wkfid = ew.ewkfid
      AND Cw.tag = %Nome do workflow%
      AND ew.tagexec = %Nome do Experimento%
      AND ca.tag = %Nome da Atividade%
```

Uma vez que esses dados são obtidos, o componente extrator identifica quais arquivos foram produzidos pela execução do *workflow* (por meio de consultas a *r-prov*) e extrai o conteúdo de cada um desses arquivos como texto plano (*i.e., plain text*). Para conseguirmos realizar as comparações entre as execuções dos *workflows* utilizando algoritmos de detecção de plágio é necessário que todas as informações estejam contidas em apenas um documento. Dessa forma, todos os dados extraídos (*r-prov*, *p-prov* e conteúdo de arquivos) são representados em um documento com um *template* bem definido. Sendo assim, nessa primeira versão do Sličnost as seguintes informações foram consideradas e inseridas no documento: nome do *workflow*, quantidade de atividades, tempo de execução do *workflow*. Para cada atividade do *workflow* lista-se o nome da atividade, a quantidade de execuções da mesma e quais os valores de parâmetros associados a cada execução. Além disso, cada arquivo produzido é listado e seu conteúdo inserido no documento. Dessa forma, é gerado um documento textual que descreve por completo uma execução de um *workflow*, de forma que ela possa ser analisada por um algoritmo de detecção de plágio conforme apresentado na Figura 2.

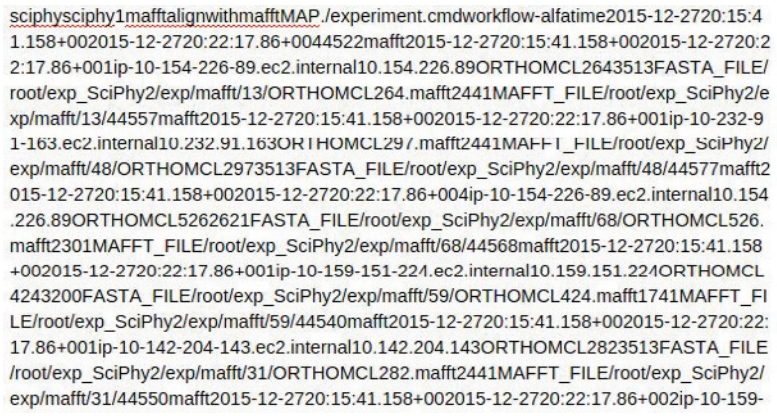
sciphysciphy1mafftalignwithmafftMAP./experiment.cmdworkflow-alfatime2015-12-2720:15:41.158+002015-12-2720:22:17.86+0044522mafft2015-12-2720:15:41.158+002015-12-2720:22:17.86+001ip-10-154-226-89.ec2.internal10.154.226.89ORTHOMCL2643513FASTA_FILE/root/exp_SciPhy2/exp/mafft/13/ORTHOMCL264.mafft2441MAFFT_FILE/root/exp_SciPhy2/exp/mafft/13/44557mafft2015-12-2720:15:41.158+002015-12-2720:22:17.86+001ip-10-232-91-163.ec2.internal10.232.91.163ORTHOMCL297.mafft2441MAFFT_FILE/root/exp_SciPhy2/exp/mafft/48/ORTHOMCL2973513FASTA_FILE/root/exp_SciPhy2/exp/mafft/48/44577mafft2015-12-2720:15:41.158+002015-12-2720:22:17.86+004ip-10-154-226-89.ec2.internal10.154.226.89ORTHOMCL5262621FASTA_FILE/root/exp_SciPhy2/exp/mafft/68/ORTHOMCL526.mafft2301MAFFT_FILE/root/exp_SciPhy2/exp/mafft/68/44568mafft2015-12-2720:15:41.158+002015-12-2720:22:17.86+001ip-10-159-151-224.ec2.internal10.159.151.224ORTHOMCL4243200FASTA_FILE/root/exp_SciPhy2/exp/mafft/59/ORTHOMCL424.mafft1741MAFFT_FILE/root/exp_SciPhy2/exp/mafft/59/44540mafft2015-12-2720:15:41.158+002015-12-2720:22:17.86+001ip-10-142-204-143.ec2.internal10.142.204.143ORTHOMCL2823513FASTA_FILE/root/exp_SciPhy2/exp/mafft/31/ORTHOMCL282.mafft2441MAFFT_FILE/root/exp_SciPhy2/exp/mafft/31/44550mafft2015-12-2720:15:41.158+002015-12-2720:22:17.86+002ip-10-159-

Figura 2 Exemplo de fragmento de documento gerado contendo *r-prov*, *p-prov* e dados de domínio extraídos de arquivos produzidos

Uma vez que o componente de extração gera os documentos, o algoritmo de detecção de plágio pode então ser executado. Nessa primeira versão do Sličnost foram implementados os algoritmos *N-Grams*, múltiplos *N-Grams* e *Winnowing*. Todos os algoritmos descritos na Seção 2 foram implementados em C++ e o algoritmo de *hash* escolhido foi o djb2 que é conhecido como um dos melhores algoritmos de *hash*, apesar de sua eficiência não ter sido formalmente provada. Todo o código desenvolvido pode ser acessado em https://bitbucket.org/danielcmo/deteccao_plagio.

Para avaliar o Sličnost foi realizada uma série de experimentos, com cada algoritmo de detecção de plágio implementado, e para determinados cenários possíveis em uma situação de experimentação real. Escolhemos como caso de estudo o workflow SciPhy [11] que gera árvores filogenéticas a partir de sequências de DNA, RNA e

aminoácidos. Para realizar as comparações, escolhemos uma execução do SciPhy como execução-base. A execução-base escolhida consumiu um arquivo multi-fasta contendo informações do *Plasmodium falciparum*, que é um protozoário e causa a malária em humanos. A partir da execução-base (exp01) extraímos informações de 3 outras execuções com diferentes características: (i) uma segunda execução (exp02) que consome o mesmo arquivo multi-fasta de entrada e há variação no parâmetro *bootstrap replicate* da atividade RAxML; (ii) uma terceira execução (exp03) que consome o mesmo arquivo multi-fasta e há variação no modelo evolutivo e no parâmetro *bootstrap replicate* e (iii) uma quarta execução (exp04) que consome um arquivo multi-fasta diferente (que não corresponde ao *Plasmodium falciparum)*. Os parâmetros utilizados nos algoritmos foram *n=4* para o *N-Gram* (gerando o chamado *4-Gram*), o parâmetro *n* variando de 4 a 6 para o algoritmo de múltiplos *N-Grams* e *w=3* para o tamanho de janela do algoritmo *Winnowing*. Os resultados das comparações são apresentados na Tabela 1, na Tabela 2 e na Tabela 3, onde são apresentadas a similaridade entre as execuções e o tempo em milissegundos decorrido para realizar a comparação.

Tabela 1 Resultados *4-Gram*

Comparação *4-Gram*			
Experimento Base	Experimento Comparado	Similaridade (%)	Tempo Decorrido (ms)
exp01	exp02	96.65	981
exp01	exp03	66.97	883
exp01	exp04	25.46	865
exp02	exp01	96.68	888
exp03	exp01	66.99	886
exp04	exp01	27.37	867

Tabela 2 Resultado Múltiplos *N-Gram*

Comparação Multiplos *N-Gram* [N = 4,5,6]			
Experimento Base	Experimento Comparado	Similaridade (%)	Tempo Decorrido (ms)
exp01	exp02	89.99	2861
exp01	exp03	58.80	2847
exp01	exp04	19.56	2780
exp02	exp01	90.02	2792
exp03	exp01	58.81	2863
exp04	exp01	21.03	3089

Tabela 3 Resultados do algoritmo *Winnowing*

Comparação *Winnowing* N = 4 W = 3			
Experimento Base	Experimento Comparado	Similaridade (%)	Tempo Decorrido (ms)
exp01	exp02	95.79	668
exp01	exp03	69.24	639
exp01	exp04	26.07	611
exp02	exp01	95.92	667
exp03	exp01	68.61	613
exp04	exp01	28.62	551

Conforme podemos observar na Tabela 1, na Tabela 2 e na Tabela 3, a suposição realizada de que podemos verificar a reprodução de experimentos foi comprovada após a execução dos diversos algoritmos. Nesse experimento consideramos que a similaridade acima de 90% indica uma reprodução (tal *threshold* irá variar de experimento para experimento). A execução com pouca variação nos dados (exp02) em relação ao experimento-base (exp01) tem alta taxa de similaridade (aproximadamente 95%), e quanto maior a variação nos dados do experimento menos similares os resultados se tornam. Porém, esse resultado depende também dos valores de parâmetros escolhidos para os algoritmos de detecção de plágio. Dependendo da escolha desses valores o resultado pode não condizer com a realidade.

Podemos notar também que como o esperado o *Winnowing* foi a técnica mais eficaz em termos de tempo de execução, uma vez que otimiza as escolhas dos *N-Grams* que compõem a impressão digital do documento que descreve a execução do *workflow*.

A técnica menos eficaz, como esperado, foi a de múltiplos *N-Grams* por fazer a verificação do documento para mais de um valor do parâmetro *n*. É importante ressaltar que os resultados foram bastante similares utilizando as 3 técnicas, ocorrendo apenas uma queda maior no valor da similaridade quando utilizamos a técnica de múltiplos *N-Grams*, o que era relativamente esperado, pois utilizamos apenas dois valores diferentes para o parâmetro *n* do que fora usado nas execuções do *4-Gram* e do *Winnowing*. Ainda sim, os valores se mantiveram relativamente estáveis, o que comprova que a similaridade não é apenas uma peculiaridade para o parâmetro *n=4*.

Outro ponto que merece esclarecimento é que o valor da similaridade entre um documento A em comparação com um documento B pode diferir do valor entre a comparação do B em relação ao documento A, visto que os dados de um experimento podem estar "contidos" no outro. Dessa forma, da maneira que os algoritmos foram implementados o documento que se encontra contido será semelhante ao que o contém, mas o contrário não será necessariamente verdade.

4. Trabalhos Relacionados

Existem algumas abordagens na literatura que visam garantir a reprodutibilidade de experimento e comparar execuções de *workflows* científicos para verificar a reprodução de experimentos. Nenhuma delas utiliza algoritmos de detecção de plágio para executar essa verificação. Dessa forma, apresentamos as abordagens mais semelhantes nessa seção. O ReproZip [12] é uma ferramenta que consegue copiar o *workflow*, e suas dependências, assim como os dados que o mesmo gerou utilizando uma máquina virtual. O ReproZip possibilita que o *workflow* possa ser executado *a posteriori* com configurações idênticas, de forma a garantir a reprodutibilidade. Entretanto, o ReproZip não compara as execuções do *workflow*. O ReproZip apenas mantém o mesmo ambiente para garantir que uma será reprodução fiel da outra.

Missier *et al*. [13] propõem um algoritmo para verificar a semelhança entre os dados de proveniência entre um experimento base e suas possíveis reproduções. A abordagem de Missier *et al*. se baseia na hipótese que a reprodução do experimento tenha uma especificação similar ao original, de modo que a similaridade é medida comparando-se cada atividade de um experimento reproduzido com a sua correspondente no original. Se uma execução possuir uma atividade diferente, é identificada uma nova ramificação e indica-se uma diferença entre as execuções. Apesar de realizar uma comparação estrutural eficaz, a abordagem de Missier *et al*. não utiliza o conteúdo dos arquivos produzidos na comparação, o que faz com que o resultado se torne menos eficaz.

5. Conclusão

Há muito tempo a ciência se preocupa não só em realizar grandes descobertas, mas também validar a capacidade dos resultados de experimentos serem passíveis de reprodução por terceiros em condições semelhantes. Com o advento da computação no processo de experimentação científica e a adoção de *workflows* como abstrações para modelar experimentos baseados em simulação, novas limites para as pesquisas surgiram, com isso aumentando a produtividade, e ao mesmo tempo dificultando a forma como se valida a reprodução desses experimentos baseados em simulação.

Como tal problema é, em termos científicos, uma prioridade, várias instituições incentivam a pesquisa sobre a verificação da reprodução de experimentos baseados em simulações. Nesse artigo apresentamos uma abordagem para verificação da reprodução

de experimentos modelados como *workflows* chamada Sličnost. A Sličnost utiliza dados de proveniência extraídos de bases de dados de proveniência (*r-prov* e *p-prov*) e algoritmos de detecção de plágio para verificar se uma execução de *workflow* é uma reprodução (ou não) de uma execução anterior. A avaliação da abordagem proposta foi realizada com o *workflow* SciPhy de análise filogenética e os resultados obtidos mostraram que a Sličnost é capaz de verificar a reprodução e a similaridade entre execuções de *workflows*. Trabalhos futuros incluem a análise de escalabilidade da Sličnost, uma vez que dependendo do tamanho do *workflow* analisado o documento gerado pode se tornar demasiadamente grande e a comparação se tornar pouco eficaz.

6. Referências Bibliográficas

[1] E. Deelman, D. Gannon, M. Shields, and I. Taylor, "Workflows and e-Science: An overview of workflow system features and capabilities," *Future Generation Computer Systems*, vol. 25, no. 5, pp. 528 – 540, 2009.

[2] Y. Gil, E. Deelman, M. Ellisman, T. Fahringer, G. Fox, D. Gannon, C. Goble, M. Livny, L. Moreau, and J. Myers, "Examining the Challenges of Scientific Workflows," *Computer*, vol. 40, no. 12, pp. 24–32, 2007.

[3] J. Freire, D. Koop, E. Santos, and C. T. Silva, "Provenance for Computational Tasks: A Survey," *Computing in Science and Engineering*, vol. 10, pp. 11–21, May 2008.

[4] C. Goble, "The Reality of Reproducibility in Computational Science," *Fapesp*, May-2013. [Online]. Available: http://www.fapesp.br/eventos/2013/05/escience/Goble.pdf. [Accessed: 27-Jan-2014].

[5] A. M. E. T. Ali, H. M. D. Abdulla, and V. Snasel, "Survey of Plagiarism Detection Methods," in *2011 Fifth Asia Modelling Symposium*, 2011, pp. 39–42.

[6] D. Oliveira, E. Ogasawara, F. Baião, and M. Mattoso, "SciCumulus: A Lightweight Cloud Middleware to Explore Many Task Computing Paradigm in Scientific Workflows," in *International Conference on Cloud Computing*, Washington, DC, USA, 2010, pp. 378–385.

[7] S. M. zu Eissen and B. Stein, "Advances in Information Retrieval: 28th European Conference on IR Research, ECIR 2006, London, UK, April 10-12, 2006. Proceedings," M. Lalmas, A. MacFarlane, S. Rüger, A. Tombros, T. Tsikrika, and A. Yavlinsky, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 565–569.

[8] W. B. Cavnar, "Using An N-Gram-Based Document Representation With A Vector Processing Retrieval Model.," in *TREC*, 1994.

[9] S. Schleimer, D. S. Wilkerson, and A. Aiken, "Winnowing: Local Algorithms for Document Fingerprinting," in *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, New York, NY, USA, 2003, pp. 76–85.

[10] W. Oliveira, D. Oliveira, and V. Braganholo, "Experiencing PROV-Wf for Provenance Interoperability in SWfMSs," in *IPAW*, Cologne, German, 2014, pp. 294–296.

[11] K. A. C. S. Ocaña, D. de Oliveira, E. Ogasawara, A. M. R. Dávila, A. A. B. Lima, and M. Mattoso, "SciPhy: A Cloud-Based Workflow for Phylogenetic Analysis of Drug Targets in Protozoan Genomes," in *Advances in Bioinformatics and Computational Biology*, 2011, pp. 66–70.

[12] F. Chirigati, D. Shasha, and J. Freire, "ReproZip: Using Provenance to Support Computational Reproducibility," in *Proceedings of the 5th USENIX Conference on Theory and Practice of Provenance*, Berkeley, CA, USA, 2013, pp. 1–4.

[13] Z. Bao, S. Cohen-Boulakia, S. B. Davidson, A. Eyal, and S. Khanna, "Differencing Provenance in Scientific Workflows," in *IEEE International Conference on Data Engineering*, 2009, pp. 808–819.