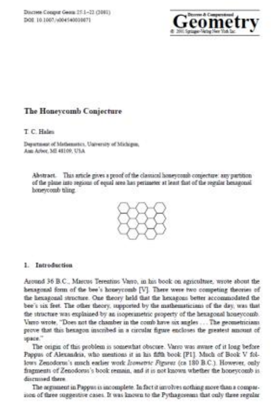
CEFET/RJ

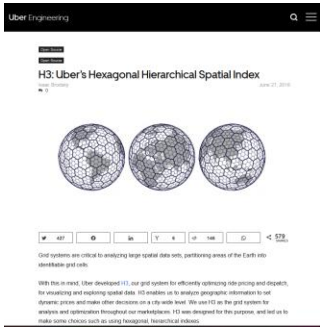
# Pesquisa em Computação

Eduardo Ogasawara
eduardo.ogasawara@cefet-rj.br
https://eic.cefet-rj.br/~eogasawara

# *Ciência e tecnologia*

- Ciência busca conhecimento e explicações
  - Constrói teorias para explicar fatos observados
- Tecnologia aplica conhecimento para resolver problemas práticos
  - Seu foco não é explicar fenômenos naturais, mas sim desenvolver soluções que impactam o mundo
- Na Computação, ciência e tecnologia caminham juntas:
  - Trabalhos científicos explicam como e por que algo funciona
  - Trabalhos tecnológicos resultam em ferramentas, algoritmos e processos
- Exemplo:
  - Ciência: A conjectura do favo de mel
  - Tecnologia: Estrutura espacial hexagonal hierarquia do Uber




[1] T.C. Hales, 2001, The Honeycomb Conjecture, *Discrete & Computational Geometry*, v. 25, n. 1 (Jan.), p. 1–22.
[2] I. Brodsky, 2018. H3: Uber’s Hexagonal Hierarchical Spatial Index. *Uber Engineering Blog*. URL: https://eng.uber.com/h3

# *Tipos de Pesquisa em Computação*

- A pesquisa pode ser classificada pela:
  - Fonte de dados
    - Primária
    - Secundária
  - Métodos de investigação
    - Quantitativa
    - Qualitativa
    - Analítica

# *Pesquisa primária vs. secundária*

- Pesquisa Primária
  - Produz novos dados (experimentos, entrevistas, observações)
  - Metodologias: qualitativa, quantitativa, analítica
  - Exemplo:
    - Desenvolvimento de um novo algoritmo de aprendizado de máquina
    - Novo método de normalização de séries temporais
- Pesquisa Secundária
  - Sistematiza o conhecimento existente
  - Inclui revisões sistemáticas e meta-análises
  - ◆ Exemplo:
    - Revisão sistemática sobre técnicas de aprendizado de máquina
    - Revisão sistemática sobre métodos de normalização para séries temporais

# *Pesquisa quantitativa vs. qualitativa*

- Pesquisa Quantitativa
  - Trabalha com dados numéricos
  - Utiliza métodos estatísticos
  - Exemplo:
    - Avaliação da acurácia de um modelo de IA com métricas como precisão e recall
- Pesquisa Qualitativa
  - Trabalha com dados textuais ou subjetivos
  - Utiliza métodos descritivos e exploratórios
  - ◆ Exemplo:
    - Entrevistas com programadores sobre a adoção de uma nova tecnologia

# *Pesquisa Analítica*

- Constrói teorias e apresenta provas matemáticas
- Comum em áreas como criptografia, algoritmos e lógica formal
- Exemplo:
  - Demonstração da complexidade do algoritmo de Dijkstra

# *Comparação e Validação de Contribuições Científicas em Computação*

- Ao apresentar uma nova contribuição científica, é fundamental contextualizá-la e avaliá-la de forma adequada. Dependendo da natureza do trabalho, a validação pode seguir diferentes abordagens:
  - Apresentação de Algo Diferente: Quando não há trabalhos semelhantes, é necessário construir uma argumentação forte
  - Apresentação de Algo Melhor: Se há soluções existentes, a nova abordagem deve ser comparada com benchmarks reconhecidos
  - Apresentação de uma Prova: Em áreas teóricas, a validade se dá por meio de demonstrações matemáticas
  - Apresentação de um Artefato Computacional: Quando o foco é um software ou framework, a avaliação pode envolver testes práticos
  - Apresentação de Artigos de Dados: Trabalhos que organizam e disponibilizam datasets devem descrever claramente seu impacto
- Pergunta-chave para cada caso:
  - Como posso demonstrar o valor da minha pesquisa?

# *Apresentação de algo diferente*

- Pesquisa que não tem trabalhos relacionados direto
- Avaliação qualitativa quando não há dados quantitativos disponíveis
- Estudos de caso são comuns, pois fornecem evidências qualitativas, mas não provas estatísticas
- Exemplo:
  - Desenvolvimento de um novo paradigma de programação sem comparação direta com modelos existentes

Contexto

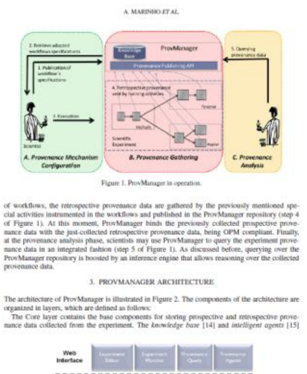

A. MARINHO *ET AL.*



Figure 1. ProvManager in operation.

of workflows, the retrospective provenance data are gathered by the previously mentioned special activities instrumented in the workflows and published in the ProvManager repository (step 4 of Figure 1). At this moment, ProvManager binds the previously collected prospective provenance data with the just-collected retrospective provenance data, being OPM compliant. Finally, at the provenance analysis phase, scientists may use ProvManager to query the experiment provenance data in an integrated fashion (step 5 of Figure 1). As discussed before, querying over the ProvManager repository is boosted by an inference engine that allows reasoning over the collected provenance data.

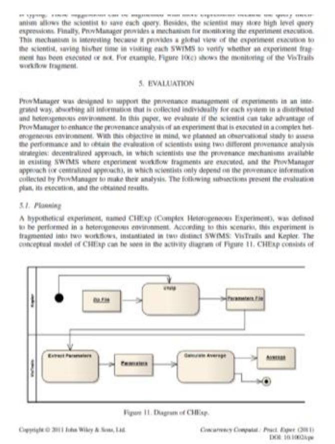
3. PROVMANAGER ARCHITECTURE

The architecture of ProvManager is illustrated in Figure 2. The components of the architecture are organized in layers, which are defined as follows:

The Core layer contains the base components for storing prospective and retrospective provenance data collected from the experiment. The *knowledge base* [14] and *intelligent agents* [15]



Argumentação


anism allows the scientist to save each query. Besides, the scientist may store high level query expressions. Finally, ProvManager provides a mechanism for monitoring the experiment execution. This mechanism is interesting because it provides a global view of the experiment execution to the scientist, saving his/her time in visiting each SWIMS to verify whether an experiment fragment has been executed or not. For example, Figure 10(c) shows the monitoring of the VisTrails workflow fragment.

5. EVALUATION

ProvManager was designed to support the provenance management of experiments in an integrated way, absorbing all information that is collected individually for each system in a distributed and heterogeneous environment. In this paper, we evaluate if the scientist can take advantage of ProvManager to enhance the provenance analysis of an experiment that is executed in a complex heterogeneous environment. With this objective in mind, we planned an observational study to assess the performance and to obtain the evaluation of scientists using two different provenance analysis strategies: decentralized approach, in which scientists use the provenance mechanisms available in existing SWIMS where experiment workflow fragments are executed, and the ProvManager approach (or centralized approach), in which scientists only depend on the provenance information collected by ProvManager to make their analysis. The following subsections present the evaluation plan, its execution, and the obtained results.

*5.1. Planning*

A hypothetical experiment, named CHExp (Complex Heterogeneous Experiment), was defined to be performed in a heterogeneous environment. According to this scenario, this experiment is fragmented into two workflows, instantiated in two distinct SWIMS: VisTrails and Kepler. The conceptual model of CHExp can be seen in the activity diagram of Figure 11. CHExp consists of

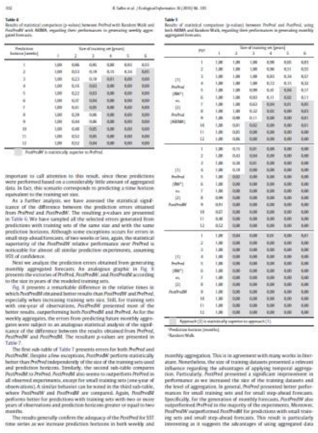
Figure 11. Diagram of CHExp.

[1] A. Marinho, L. Murta, C. Werner, V. Braganholo, S.M.S.D. Cruz, E. Ogasawara, and M. Mattoso, 2012, ProvManager: A provenance management system for scientific workflows, *Concurrency and Computation: Practice and Experience*, v. 24, n. 13, p. 1513–1530.

# *Apresentação de algo melhor*

- Exige comparação com a literatura
- Pode usar benchmarks ou criar seus próprios testes
- Se não houver benchmarks disponíveis, o autor pode criar experimentos específicos, mas isso exige cuidado para evitar viés na avaliação
- Importante definir métricas claras
- Exemplo:
  - Um novo algoritmo de aprendizado de máquina testado em datasets padrão como MNIST e ImageNet

Comparação



[1] R. Salles, P. Mattos, A.-M.D. Iorgulescu, E. Bezerra, L. Lima, and E. Ogasawara, 2016, Evaluating temporal aggregation for predicting the sea surface temperature of the Atlantic Ocean, *Ecological Informatics*, v. 36, p. 94–105.

# Apresentação de uma prova

- Construção de uma teoria baseada em definições formais
- Uso comum em áreas matematicamente rigorosas
- Atualmente, experimentação também é necessária para demonstrar aplicabilidade
- Exemplo:
  - Demonstração formal da convergência de um algoritmo de otimização

## Formalização

optimum solution for the probabilities $\alpha_0, \alpha_1, \ldots, \alpha_{i-1}$ and $\beta_1, \ldots, \beta_{i-1}$, its right subtree is an optimum solution for the probabilities $\alpha_i, \ldots, \alpha_n$ and $\beta_{i+1}, \ldots, \beta_n$. Therefore, we can get a bottom up algorithm for building an optimal binary search tree for a set of probabilities $(\alpha_0, \ldots, \alpha_n, \beta_1, \ldots, \beta_n)$. We can build up optimal trees $T_{i,j}$ for all the probabilities $\alpha_i, \ldots, \alpha_j$ and $\beta_{i+1}, \ldots, \beta_j$ where $i \leqslant j$ starting from the smallest intervals and working toward the largest.

Let $P_{i,j}$ and $W_{i,j}$ denote the weighted path length and the total weight of an optimal binary search tree for all words $K_i < X \leqslant K_{j+1}$ where $i \leqslant j$. Let $R_{i,j}$ denote the index of the root of this tree when $i < j$. The following formulae determine the cubic time algorithm:

$$P_{i,i} = W_{i,i} = \alpha_i \quad \text{for } 0 \leqslant i \leqslant n, \tag{1}$$

$$W_{i,j} = W_{i,j-1} + \beta_j + \alpha_j, \tag{2}$$

$$P_{i,j} = W_{i,j} + \min_{i<k\leqslant j} (P_{i,k-1} + P_{k,j}) \quad \text{for } 0 \leqslant i < j \leqslant n. \tag{3}$$

Since we choose $R_{i,j}$ from among $j - i$ pairs for each $i$, $j$ such that $0 \leqslant i < j \leqslant n$ the algorithm runs in $O(n^3)$ time, as there are only $(n+1)(n+2)/2$ choices of $0 \leqslant i \leqslant j \leqslant n$, the space required being $O(n^2)$.

*2.1.2. The monotonicity of roots and consequent* $O(n^2)$ *algorithm*

Knuth [85] observed that the $R_{i,j}$'s satisfy the condition $R_{i,j-1} \leqslant R_{i,j} \leqslant R_{i+1,j}$. We will look at the proof in the next subsection. This condition means that we only have to search all the indices between $R_{i,j-1}$ and $R_{i+1,j}$ to compute $R_{i,j}$. The running time

## Prova

**Lemma 2.2.** *If $w$ satisfies the quadrangle inequality and is monotone, then the function $c$ defined above also satisfies the quadrangle inequality, i.e. $c(i,j) + c(i',j') \leqslant c(i,j') + c(i',j)$ for $i \leqslant i' \leqslant j \leqslant j'$.*

**Proof.** We use induction on the length $l = j' - i$ to prove the result.

This inequality is trivially true if $i = i'$ or $j = j'$. This proves the quadrangle inequality for $c$ for $l \leqslant 1$. For the induction step we distinguish two cases $i' = j$, $i' < j$.

*Case* 1: $i < i' = j < j'$. In this case the quadrangle inequality for $c$ reduces to

$$c(i,j) + c(j,j') \leqslant c(i,j'). \tag{9}$$

Let $k = R(i,j')$. We distinguish two symmetric subcases: $k \leqslant j$, $k \geqslant j$.

*Case* 1.1: $k \leqslant j$. We have

$$\begin{aligned} c(i,j) + c(j,j') &\leqslant w(i,j) + c(i,k-1) + c(k,j) + c(j,j') \\ &\quad \text{(by definition of } c(i,j)) \qquad (10) \\ &\leqslant w(i,j') + c(i,k-1) + c(k,j) + c(j,j') \\ &\quad \text{(by monotonicity of } w) \qquad (11) \\ &\leqslant w(i,j') + c(i,k-1) + c(k,j') \\ &\quad \text{(by the induction hypothesis)} \qquad (12) \\ &= c(i,j') \quad \text{(by definition of } c(i,j') \text{ and } k). \qquad (13) \end{aligned}$$

*Case* 1.2: $k \geqslant j$. As this case is symmetric to case 1.1 the proof is similar.

*Case* 2: $i < i' < j < j'$. Let $y = R(i',j)$ and $z = R(i,j')$. We have to distinguish two symmetric cases: $z \leqslant y$ or $z \geqslant y$. We only consider the case $z \leqslant y$. We note that $z \leqslant y \leqslant j$ by the definition of $y$ and $i < z$ by the definition of $z$. We have

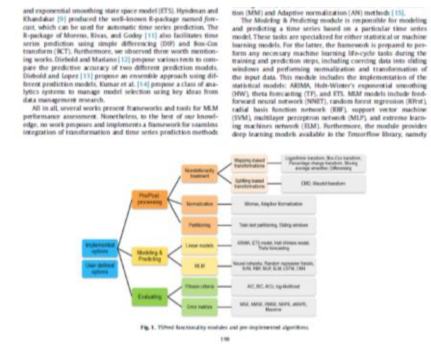
$$c(i',j') + c(i,j) \tag{14}$$

[1] S.V. Nagaraj, 1997, Optimal binary search trees, *Theoretical Computer Science*, v. 188, n. 1 (Nov.), p. 1–44.

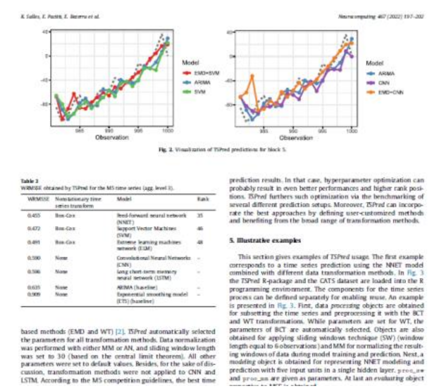
# *Apresentação de um artefato computacional*

- O foco está na criação de um produto novo (software, ferramenta, framework)
  - Comum em TCCs e projetos aplicados
  - Pode não exigir rigor científico na apresentação dos resultados, mas deve demonstrar impacto prático
- Exemplo:
  - Desenvolvimento de um framework para detecção de anomalias em séries temporais

Arquitetura



Uso

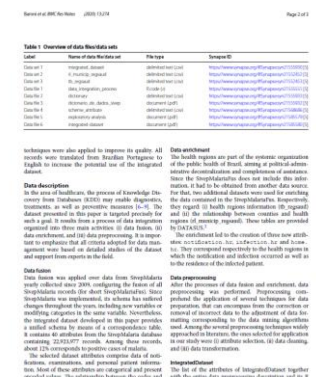


[1] R. Salles, E. Pacitti, E. Bezerra, F. Porto, and E. Ogasawara, "TSPred: A framework for nonstationary time series prediction," *Neurocomputing*, vol. 467. pp. 197–202, 2022. doi: 10.1016/j.neucom.2021.09.067.

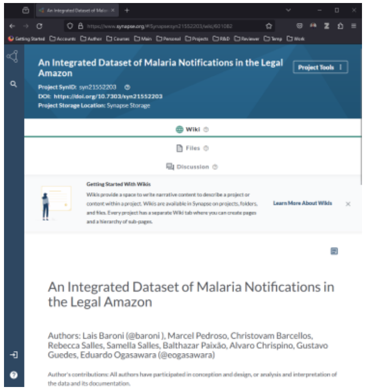
# *Apresentação de artigos de dados*

- Documento revisado por pares que descreve um conjunto de dados
- Valoriza o esforço para organizar e descrever dados
- Exemplo:
  - Google's Open Images Dataset: um dataset publicado para treinamentos de modelos de visão computacional

ETL



Uso



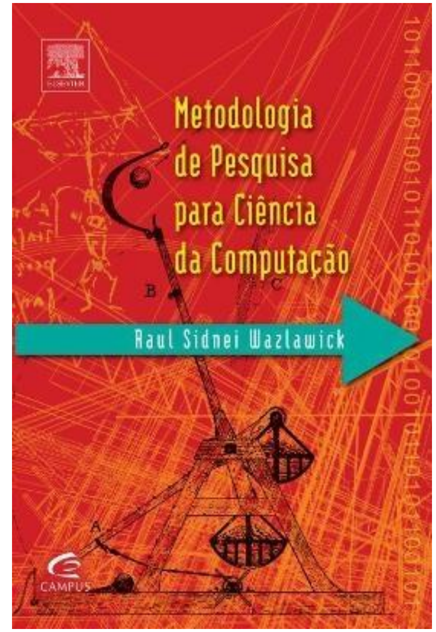

[1] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov, T. Duerig, and V. Ferrari, "The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale," Int J Comput Vis, vol. 128, no. 7, pp. 1956–1981, Jul. 2020, doi: 10.1007/s11263-020-01316-z.

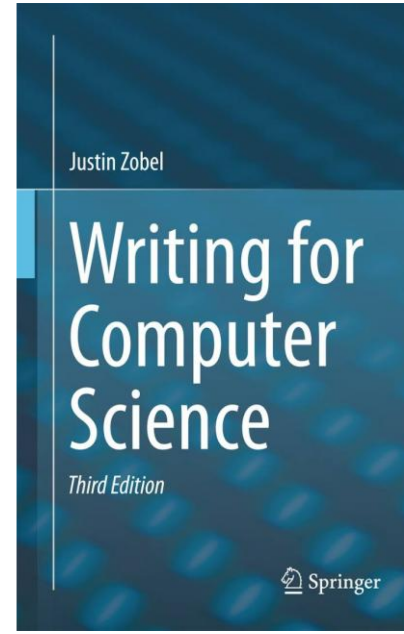
# *Considerações*

- A pesquisa em Computação pode assumir diversas abordagens
- Escolher a metodologia correta depende do problema estudado
- Trabalhos podem focar na criação de conhecimento, na análise de dados, ou na prototipagem de soluções
  - Apesar do foco, pode apresentar em menor grau as outras abordagens
  - Exemplo:
    - Um artigo pode combinar aspectos de pesquisa analítica (prova formal), quantitativa (avaliação experimental) e tecnológica (implementação de ferramenta)

# *Referências*






[1] D. G. Perovano, Manual de metodologia da pesquisa científica. Editora Intersaberes, 2016.
[2] A. L. Cervo, P. A. Bervian, e R. da Silva, Metodologia Científica. Pearson Universidades, 2006.
[3] R. Wazlawick, 2017, Metodologia de Pesquisa para Ciência da Computação. Elsevier Brasil.
[4] J. Zobel, 2015, Writing for Computer Science. Springer.