30/10 a 01/11
Niterói – RJ

BSB 2018

11th Brazilian Symposium on Bioinformatics

PROCEEDINGS

Luis Antonio Kowada
Ronnie Alves
Daniel de Oliveira (Org.)

SBC Sociedade Brasileira de Computação

Universidade Federal Fluminense

Instituto de Computação

BRAZILIAN SYMPOSIUM ON BIOINFORMATICS
October 30th to November 1st, 2018
Niterói – RJ – Brazil

PROCEEDINGS

Promotion
Sociedade Brasileira de Computação– SBC
Comissão Especial de Biologia Computacional (CE-BioComp) da SBC

Organization
Universidade Federal Fluminense - UFF

Steering Committee
Luis Antonio Kowada (UFF) – Steering Committee Chair
Ronnie Alves (ITV, UFPA)
João Carlos Setubal (IQ-USP)
Maria Emilia Telles Walter (UnB)
Nalvo Franco de Almeida Jr (UFMS)
Guilherme Pimentel Telles (UNICAMP)
Tainá Raiol (FIOCRUZ)
Natália Florencio Martins (EMBRAPA)
Luciana Montera (UFMS)
Sergio Campos (UFMG)

Local Organization Chair
Luis Antonio Kowada (IC-UFF)

Program Committee Chairs
Luis Antonio Kowada (UFF)
Ronnie Alves (ITV, UFPA)
Daniel de Oliveira (UFF) – *Short Papers*

Message from the chairs

Welcome to the Brazilian Symposium on Bioinformatics 2018 and to Niterói, Rio de Janeiro! The Brazilian Symposium on Bioinformatics is the official bioinformatics event of the Brazilian Computer Society (SBC) and one of the largest venues in Latin America for presentation and discussion of research results in the bioinformatics domain. This edition of the symposium (BSB 20188) was held in Niterói, in the state of Rio de Janeiro, from October 30th to November 1st, 2018.

The BSB 2018 program offers a variety of activities, suited for an audience ranging from undergraduate to Ph.D. students, researchers and professors. The excellence of BSB 2018 program is the result of the competence and commitment of the community, which we gratefully acknowledge. We thank the symposium chairs and our colleagues of the local organization committee who donated their precious time to made BSB 2018 a reality. Further, we thank the program committee members for the high quality reviews, and the authors who submitted their papers to this year event. Finally, we are grateful to our sponsors.

We hope you all enjoy BSB 2018 in Niterói!

Luis Antonio Kowada, UFF
Ronnie Alves, ITV, UFPA
Daniel de Oliveira, UFF
BSB 2018 Chairs

Development of a framework for whole-cell model creation

Frederico Chaves Carvalho[1], Paulo Eduardo Ambrósio[1]

[1] Programa de Pós-Graduação em Modelagem Computacional em Ciência e Tecnologia– Universidade Estadual de Santa Cruz (UESC)
Rodovia Jorge Amado, Km 16, Bairro Salobrinho – Ilhéus – Bahia

fcc073@gmail.com, peambrosio@uesc.br

Abstract. *The use of whole-cell models in research has the potential to be a powerful tool for scientific discovery, allowing researchers to test hypotheses faster than using in vitro or in vivo methods. Such models can be considered the equivalent of Computer Aided Design for Biology. However, given their complexity, it is still difficult to employ them as an instrument in investigations. In order to solve this problem, we are developing a framework with the purpose to guide and help scientists through the process of creating whole-cell models faster, enabling them to use these tools as part of their research. This paper brings details of the early stages of the framework's development process.*

1. Introduction and Objectives

Whole-cell models are computational models that aims to simulate the behavior of living cells by representing all the intracellular biochemical processes, as well as the function of most (or all) genes in the cell. Even though the current models are not perfect, they have proved to be powerful tools with the potential to enable scientist to explore new methods for scientific discoveries in fields such as medicine, biology and bioengineering [Covert, 2014].

The first whole-cell model to consider the role of genes in the lifecycle of the cell was created as part of the E-CELL project [Tomita et al., 1999]. Initiated in 1996, soon after the release of the full genome sequence of *Mycoplasma genitalium*, the team developed a computer model of this bacterium using 127 of the 525 genes, which was enough to simulate a cell capable of performing the basic processes necessary for survival. The model was also capable of reproducing some of the behaviors of the cell, such as the peak for intracellular ATP, followed by a sharp decline, when a starvation process begins.

The most comprehensive whole-cell model to date was presented in 2012, and represents all the 525 genes and most of the molecules in the *M. genitalium* [Karr et al., 2012; Goldberg et al., 2016]. The model was able to predict relevant cell behavior, such as the impact of nonessential single-gene disruption in the growth rate of the cell [Sanghivi et al., 2013]. These observations were important to ratify the possibility of using such models to accelerate research and guide in vitro or in vivo experiments.

Despite the aforementioned successes in the creation of whole-cell models, developing new ones remains a challenge. The first reason is the large amount of data necessary to represent a cell with acceptable accuracy. For instance, one needs to model

the proteins and their functions, the metabolic pathways, the compartments, the extracellular environment, etc. Another challenge is the high computational cost for running simulations of increasingly complex models. That is the case of the *Mycoplasma genitalium's* model, which needs approximately 1 core day of an Intel E5520 CPU to complete the simulation [Goldberg et al., 2016].

Moreover, to create and simulate a whole-cell model, one needs to have a strong background in both biology and computer science, which limits the amount of scientists that can use them as a tool for research and prediction. This also restricts the possibility of other uses for whole-cell models, such as in education. To make models accessible to biologists and physicians with average computational knowledge, it would be necessary to simplify the computational part, and one of the ways to do that is to create a software with a user-friendly graphic user interface.

Our goal with this project is to develop a tool that will help the construction of accurate whole-cell models that can be used in research. Therefore, we are creating a framework with graphical user interface that aims to guide the user through the steps of a simplified methodology for model creation. This way, we hope to provide researchers, who otherwise wouldn't be able to build and use whole-cell models, with the possibility of using them as way to accelerate their investigations.

2. Methodology

Bearing in mind that the main objective is to simplify and expedite the creation of accurate whole-cell models, the first step was to define the requirements and specifications for the framework and its GUI. This was done by analyzing two of the most successful models [Karr et al. 2012; Tomita, 2001], their biochemical background and current techniques and paradigms for developing biological models [Fall et al., 2002; Karr et al., 2015; Freddolino and Tavazoie, 2012].

One way to build a whole-cell model is by describing it as a function of chemical species and reactions. By assigning concentrations to each species and rates for each reaction, and creating conditions that determine whether the reaction occurs it is possible to apply a simple ordinary differential equation solver to simulate the behavior of the system. The model can further be refined by adding the function of each gene, protein and organelle. It is possible to do that is by defining each of these components as objects and assigning their behavior to methods associated with them. The complexity, and accuracy of the model can be increased by creating separate compartments or regions within the cell, link reactions to model metabolic pathways, account for the function and structure of the plasma membrane, define the extracellular environment (in terms of concentrations, pH, temperature, osmotic pressure, luminosity, etc.)

With all the above information, we established a simplified model creation methodology, and the framework's interface was built reflecting such procedure and taking into account the need for user-friendliness and versatility. The open source version of Qt Creator was the software used to develop the framework's GUI and its basic functionalities were implemented in Python, due to its versatility and simplicity.

The simulation algorithm used by the most recent model was single-threaded and slow [Goldberg et al., 2016]. In order to remove this bottleneck we are developing a

parallel version of two simulation algorithms, which are ODE and Gillespie's method (for stochastic simulation). To ensure that the model runs in the most efficient way, C language is being used to implement these algorithms. This choice serves a double purpose: enhancing the efficiency of the code, and enabling easier parallel processing techniques that will be done using the Open MP and MPI standards.

In order to validate the framework, a model of the bacterium *M. genitalium* will be created and simulated. The results will be compared to experimental data from the literature as well as to the simulation results from the model developed by Karr et al. (2012) as a benchmark.

3. Framework's specifications and requirements

The input information needed for creating a whole-cell model is diverse and scattered through different databases. For instance, the user will find annotated genome sequences and metabolic pathway maps in KEGG database, but will need to look elsewhere for some specific reaction information. While some organism specific databases, such as WholeCellKB can facilitate the work of the user, they are still not common and, most of the time, incomplete. In order to assist the user in the analysis and selection of data, the framework needs to be able import, organize and display information from different databases that can complement each other.

In cases where the available data is not sufficient to create a comprehensive model, the frameworks must allow the user to create reduced models, using only the information available, to simulate part of the processes involved in a cell, such as a single metabolic pathway or a set of reactions. Another important option the framework must provide is to use prediction tools to estimate the missing values and variables, and thus, assist the user in completing the model.

A living cell can behave in a myriad of ways that makes it difficult to describe its behavior using only one algorithm. For this reason, the framework has to offer the choice to use different types of simulation algorithms. In the first version, three choices are available: partial differential equation, ordinary differential equation, and Gillespie's method algorithm for stochastic simulation. The user must also have the possibility of defining compartments or regions of the cell and assign a specific simulation algorithm to each one of them, provided a clear set of inputs and outputs is possible from each compartment.

The framework is being developed to be versatile regarding the cells that can be modelled, meaning that the user can build models for different organisms, not only for the *M. genitalium*. However, because of the added complexity present in eukaryotic cells, such as the presence of internal membranes and the occurrence of splicing and other more sophisticated events, the initial goal is to enable the creation of prokaryote cell models only.

Bearing in mind the high computational demand reported in the current models, the support for parallel processing and use of GPU as coprocessor must be implemented. Figure 1 shows details of the framework's architecture and operation. Note that the parallelization occur in the "Model processing" phase, where simulations in different compartments are run by different threads.

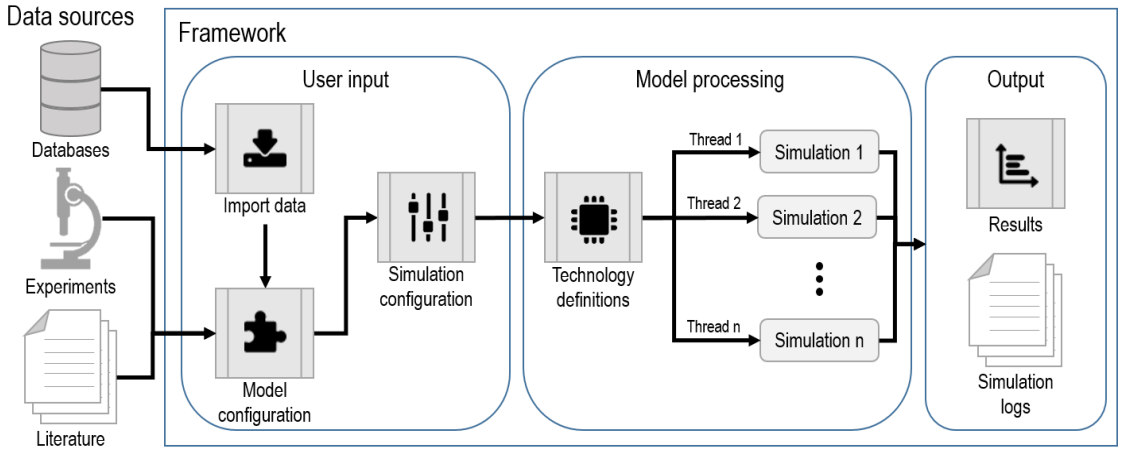


Figure 1. The framework's architecture

4. First results

The first stage of the framework's development process, which is the design of its graphic user interface, is complete. The GUI was designed to reflect the methodology that must be followed to build a whole-cell model that takes into account the genetic information, the molecules inside the cell and the reactions they are involved in, the extracellular environment, and the structure of the cell. To allow the user to go through these steps, the framework is divided into five modules: "Import data", "Model configuration", "Simulation configuration", "Technology definitions" and "Results".

The "Import data" module is implemented and functional, and is currently capable of fetching data from KEGG and WholeCellKB databases. The user can define the data to be imported and used for the model building. Figure 2 shows a preview of how the framework's first module works. It is possible to select the repository from which the data should be imported, the cell/organism, the variant (if more than one is available in the repository), and the data type, which can be the genome sequences, chemical species, reactions or metabolic pathways. The framework creates a separate file containing classes corresponding to every type of information that is allowed to be used for model creation in the framework, and each entry in the table in figure 2 is written in the same file as an object of the corresponding class (data type). These objects will be used alongside the reactions, concentration and positional information (if available) as inputs to the simulation algorithm.

The "Model configuration" module is where the user will use the data imported previously to build the mode and define other important information and parameters for the cell, such as its shape and size, the extracellular environment, membrane structure and composition, concentrations of each species and (optionally) define regions of the cell that has a different behavior (i.e. assign a different simulation algorithm or different concentrations for one or more species). The user is also allowed to manually input new data from other sources (papers, experimental data, information from other databases, etc.), and determine or change the reactions and interactions between each component. Each new piece of information given by the user is also stored as an object in the same file as the imported data, and a new python script is created with the remaining

definitions. The module also allows the user to save the model with all the definitions and inserted data at any time.

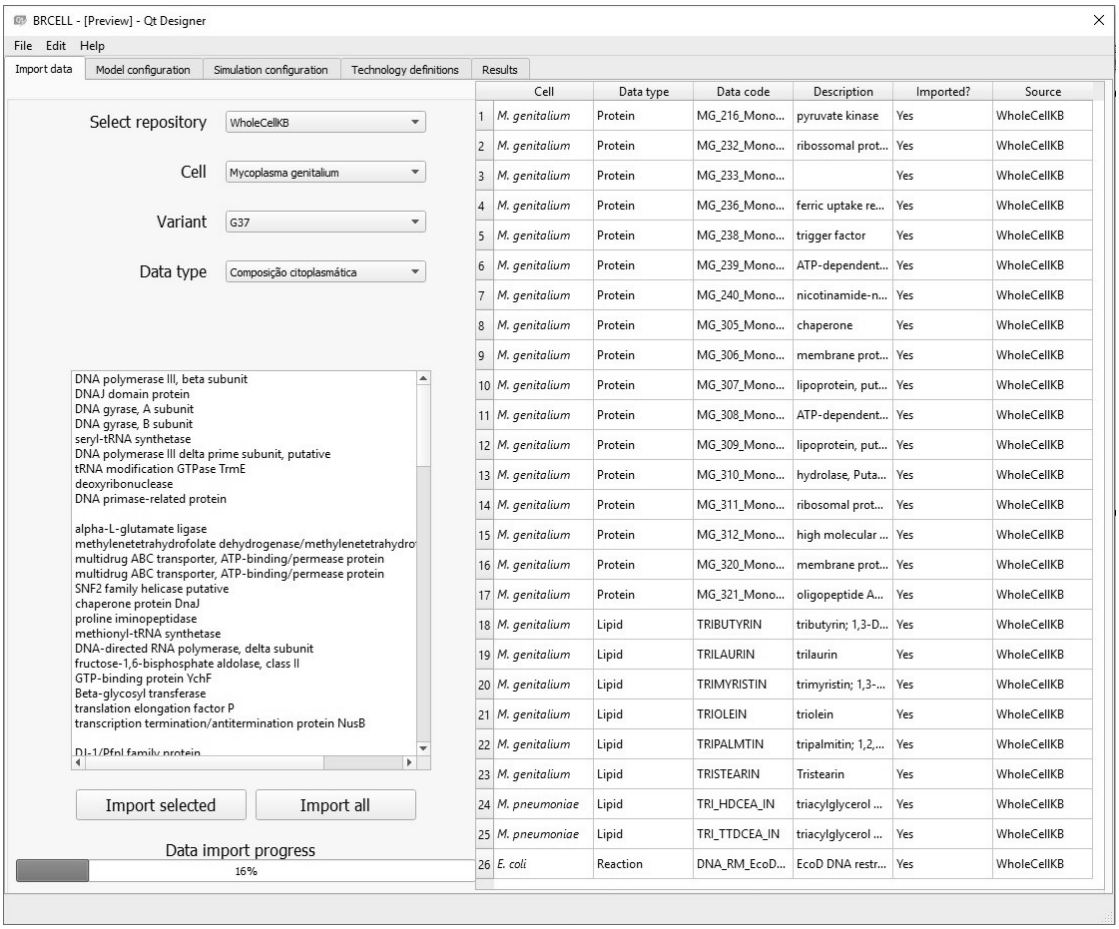


Figure 2. A preview of the "Import data" module of the framework

The Simulation configuration module allows the user to choose the simulation algorithm, what information should be monitored and displayed, the desired results output and to be used. In short, this module will allow the user to opt for a simpler simulation for faster results, or a more complete study.

In the Technology definitions module, the user can decide whether parallelization of the model should occur, and what resources should be used. For instance, the user can turn on the use of graphic card as a coprocessor and select which clusters will be used to run the simulation.

The Results module is designed to show the results of the simulation through graphics in real time. It also provides the option to abort the simulation in the event of unexpected model behavior. Once the simulation is complete, the user will have access to a report containing the main results, simulation definitions used and simulation logs.

With the exception of the data importation module, all the other sections of the framework are currently under development and have either partial functionality or no functionality yet.

5. Acknowledgements

The authors of this work acknowledge FAPESB (Fundação de Amparo à Pesquisa do Estado da Bahia) for its scholarship number BOL0029/2018.

References

CARRERA, J.; COVERT, M. W. (2015) "Why Build Whole-Cell Models?" In: Trends in Cell Biology, v. 25, n. 12, p. 719–722, 2015.

COVERT, M. W. (2014) "Simulating a living cell". Scientific American, v. 310, n. 1, p. 44–51.

FALL, C.P.; MARLAND, E.S.; WAGNER, J.M.; TYSON, J.J. (2002) "Computational cell biology". Springer.

FREDDOLINO, P. L.; TAVAZOIE, S. (2012) "The dawn of virtual cell biology". Cell, v. 150, n. 2, p. 248–250.

GOLDBERG, A. P.; CHEW, Y. H.; KARR, J. R. (2016) "Toward Scalable Whole-Cell Modeling of Human Cells". Proceedings of the 2016 annual ACM Conference on SIGSIM Principles of Advanced Discrete Simulation - SIGSIM-PADS '16, p. 259–262.

KARR, J. R. et al. (2012) "A whole-cell computational model predicts phenotype from genotype". Cell, v. 150, n. 2, p. 389–401.

KARR, J. R.; TAKAHASHI, K.; FUNAHASHI, A. (2015) "The principles of whole-cell modeling", In: Current Opinion in Microbiology, v. 27, p. 18–24.

MACKLIN, D. N.; RUGGERO, N. A.; COVERT, M. W. (2014) "The future of whole-cell modeling", In: Current Opinion in Biotechnology, v. 28, p. 111–115.

PURCELL, O. et al. (2013) "Towards a whole-cell modeling approach for synthetic biology". Chaos, v. 23, n. 2, p. 1–8.

SANGHVI, J. C. et al. (2013) "Accelerated discovery via a whole-cell model". Nature Methods, v. 10, n. 12, p. 1192–1195.

TOMITA, M. (2001) "Whole-cell simulation: A grand challenge of the 21st century, In: Trends in Biotechnology, v. 19, n. 6, p. 205–210.

TOMITA, M. et al. (1999) "E-CELL: Software environment for whole-cell simulation", In: Bioinformatics, v. 15, n. 1, p. 72–84.

Binning de Sequências Anterior à Montagem em Metagenomas: um estudo de caso

Paulo Oliveira[1], **Kleber Padovani**[1], **Raíssa L. da Silva**[1], **Ronnie Alves**[1,2]

[1]Programa de Pós-Graduação em Ciência da Computação (PPGCC)
Universidade Federal do Pará (UFPA)
CEP – 66.075-110 – Belém – PA – Brasil

[2]Instituto Tecnológico Vale (ITV)
CEP – 66.055-090 – Belém – PA – Brasil

{p.paulo.f.oliveira,kleber.padovani,r.lorenna}@gmail.com, ronnie.alves@itv.org

Abstract. *This work, through an empirical study, aimed to answer the following question: Does binning over reads contribute to the production of better assemblies? We evaluated whether quantitative (genome binning) and qualitative (taxonomic binning) approaches bring benefits to the assembly of genomes from metagenome data through statistics which evaluate assemblies considering their sizes and qualities.*

Resumo. *Este trabalho, por meio de um estudo empírico, procurou responder a seguinte questão: O binning sobre reads colabora com a produção de melhores montagens? Buscou-se verificar se o uso das abordagens quantitativa (binning genômico) e qualitativa (binning taxonômico) traz benefícios para a montagem de genomas em metagenomas utilizando estatísticas de avaliação que consideram tamanho e conteúdo das montagens.*

1. Introdução

Na metagenômica, o agrupamento de fragmentos de DNA a um grupo taxonômico correspondente é chamado de *binning*, processo em que cada uma das sequências é alocada em um grupo que representa, de forma ideal, somente os fragmentos pertencentes a determinado táxon [Sedlar 2017]. Nesse sentido, o *binning* é significativo para a reconstrução ou recuperação de genomas de micro-organismos e pode permitir o conhecimento sobre o material genético do metagenoma como um todo.

A maioria dos algoritmos mostram que a precisão do *binning* é aprimorada conforme se aumenta o comprimento das sequências e, por isso, eles normalmente são aplicados após a montagem, sobre os *contigs* [Vollmers 2017]. No entanto, como o processo de montagem é suscetível a erros, as sequências produzidas pelos montadores podem não corresponder a sequências inteiramente corretas, como é o caso dos *contigs* quiméricos [Sedlar 2017].

Como alternativa para contornar este problema, o *binning* pode, também, ser aplicado antes da montagem, e, com isso, pode reduzir a complexidade computacional das montagens subsequentes. Dessa forma, o resultado do processo de *binning* pode ser usado não somente para a avaliação da diversidade taxonômica, mas, também, para facilitar a montagem do genoma [Sedlar 2017].

Nesse sentido, um aspecto importante é que o *binning* pode ser realizado diretamente nas *reads* obtidas do sequenciamento. Ou seja, o *binning* pode ser aplicado antes da montagem particionando as reads em *bins* (grupos) taxonômicos, que podem reduzir de forma significativa a complexidade da montagem de metagenomas.

O *binning* antes da montagem é uma estratégia análoga ao processo de facilitar a montagem de um quebra-cabeça, em que as peças seriam separadas em grupos e cada grupo é montado individualmente, o que pode ser considerado mais fácil do que montar todas peças misturadas. Esta estratégia é conhecida como divisão e conquista. Na investigação de [Constantinescu 2015], o *binning* genômico em conjunto com técnicas de aprendizado de máquina mostrou potencial em reduzir a complexidade e aumento de desempenho na montagem de sequências de DNA.

Este trabalho, por meio de um estudo empírico, procurou responder a seguinte questão: O binning sobre reads colabora com a produção de melhores montagens? Ou seja, buscou-se verificar se a redução de complexidade decorrente do uso das abordagens quantitativa e qualitativa, na análise de *binning* em amostras metagenômicas, traz benefícios ao desempenho da montagem.

2. Materiais e Métodos

2.1. Experimentos

Foram realizados três experimentos para a avaliação do *binning* de *reads* sobre a montagem utilizando subconjuntos de dados de *benchmarking* da iniciativa *Critical Assessment of Metagenomics Interpretation* (CAMI) [Sczyrba et al. 2017]. No primeiro experimento, conforme descrito a seguir, foi utilizada uma abordagem de *binning* taxonômico para a separação das *reads*, cujos *bins* foram posteriormente montados individualmente; no segundo experimento, foi utilizada uma abordagem de *binning* genômico; e, no terceiro, uma abordagem controle, sem a utilização de estratégias de *binning* de *reads*.

2.2. Subamostragem

Em cada experimento, foram utilizadas três subamostragens de dados do CAMI, uma proveniente da única amostra completa de complexidade baixa fornecida pela iniciativa, outra obtida de uma dentre as 4 amostras de complexidade média (RM2, S001) e a terceira provém de uma dentre as 5 amostras de complexidade alta (S001), todas com cerca de 10% do número original de *reads* e o mesmo tamanho de *insert* de 270bp.

Para a avaliação do impacto da subamostragem nos experimentos, foi considerada a estimativa de cobertura de diversidade proposta em [Rodriguez and Konstantinidis 2014] para cada uma das subamostras, bem como para a amostra original, utilizando a ferramenta Prinseq (v0.20.4) [Schmieder and Edwards 2011] para preparação dos dados e a ferramenta Nonpareil (v3 com algoritmo *kmer*) [Rodriguez and Konstantinidis 2014] para o cálculo da estimativa.

Para a geração das subamostras, foi utilizada uma ferramenta para processamento de sequências em formato FASTA/Q disponibilizada gratuitamente na internet, denominada SeqTK (https://github.com/lh3/seqtk), na versão 1.2-r101-dirty. Foram utilizadas, respectivamente e obtidas aleatoriamente, as seeds 15492, 16416 e 30938 como

parâmetros para a ferramenta para a geração de conjuntos aleatórios para construção das subamostras de baixa, média e alta complexidade. Os *scripts* de geração das subamostras e cálculos de cobertura de diversidade, bem como os demais *scripts* utilizados nos métodos descritos a seguir, foram desenvolvidos com a linguagem *python* e estão disponíveis em https://sourceforge.net/p/binning-reads/files/.

2.3. Binning

Para cada subamostra foi realizado o *binning*, com ferramentas de *binning* (*binners*) taxonômico e genômico. No *binning* taxonômico, foi utilizado o *software* Kaiju (v1.4.4) [Menzel et al. 2016], em que foi possível obter os *bins* que correspondem a cada táxon identificado. Com isso, cada *bin* é composto pelas sequências de determinado táxon. Dos diversos *bins* gerados, e para cada um dos três tipos de subamostra, foram escolhidos os seis *bins* que apresentaram maior abundância e relação com os táxons utilizados pelo CAMI. Então, o resultado dessa etapa foi a obtenção de seis *bins* para subamostra de baixa, seis para média e seis para alta complexidade, totalizando dezoito *bins*.

De forma semelhante, foi realizado o *binning* genômico. Inicialmente, foi feita a conversão das amostras FASTQ para FASTA com o *software* SeqTK (adicionalmente, houve a necessidade de desenvolvimento de um *script* para o ajuste dos arquivos FASTA gerados para a correta identificação das *reads*). Em seguida, foi realizado o *binning* genômico, utilizando o *software* MetaProb (v2.0) [Girotto et al. 2016].

Com o MetaProb, as amostras de baixa, média e alta foram analisadas nos tempos de 3h:37m, 3h:33m e 27h, respectivamente, gerando, nesta ordem, 28, 38 e 78 *clusters* para as amostras de complexidades baixa, média e alta. Foi realizada uma análise taxonômica (Kaiju) em cada *cluster* a fim de identificar os táxons mais abundantes. A partir de *script* desenvolvido, os *clusters* com táxons mais abundantes e semelhantes foram mesclados gerando *bins* dos quais foram selecionados os seis mais abundantes e relacionados ao CAMI, de cada tipo de amostra.

Após as análises de *binning* taxonômico e genômico, os *bins* resultantes foram utilizados posteriormente na montagem para as avaliações seguintes.

2.4. Montagem e Obtenção das Estatísticas

Cada *bin* gerado pelos *binners* taxonômico e genômico foram montados com o montador MegaHit (v1.1.2) [Li et al. 2015] em suas configurações padrão. Da mesma forma, foram montadas as subamostras, de baixa, média e alta complexidade, diretamente, sem o uso de *binning*.

Posteriormente, as montagens (*contigs*) foram submetidas a três ferramentas e um *script* para então ser realizada a análise e comparação. Com o intuito de avaliar a qualidade da recuperação do genoma, foram utilizadas as métricas de [Parks 2015] da ferramenta CheckM (v1.0.11, seguindo o *workflow* de identificação automática de táxons *lineage_wf*). Dessas métricas, foram selecionadas as estimativas de completude, qualidade e contaminação do genoma montado.

Para avaliar a montagem dos genomas, foram adotadas as medidas de comparação de desempenho de montagens proposta em [Mikheenko et al. 2015]. Utilizou-se a ferramenta MetaQuast (v4.2) sobre os genomas montados para alinhar com os genomas de

referências, identificando as medidas de fração de recuperação do genoma, fragmentação da montagem e a estatística N50. As medidas de fração de recuperação de genoma e N50 também foram avaliadas nas montagens sem *binning* para as sequências de baixa, média e alta complexidade.

Com o objetivo de auxiliar na análise taxonômica, foi executado a ferramenta de predição de genes FragGeneScan (v1.30), sobre os genomas montados para medir a quantidade de sequências codificadoras [Rho et al. 2010] encontradas em cada montagem do táxon correspondente. Posteriormente foram removidas, via *script*, as sequências redundantes para contagem de sequências codificadoras distintas.

Para a comparação de desempenhos das abordagens com e sem aplicação de *binning*, foram considerados somente táxons comuns nos grupos taxonômicos e genômicos, ou seja, que estavam presentes no dois grupos. Essa comparação pode ser vista na seção seguinte.

3. Resultados

Os resultados a seguir são referentes às métricas adotadas pelas ferramentas CheckM, MetaQuast e FragGeneScan. Além disso, somente foram usados os táxons comuns em todas as abordagens e que estão presentes no estudo disponibilizado pelo CAMI, totalizando oito espécies (genomas)[1].

Foram encontrados os táxons *Albidovulum xiamenense*, *Paracoccus denitrificans* e *Pseudomonas aeruginosa* na amostra de complexidade baixa, identificados na Tabela 1 pelos números 1, 2 e 3, respectivamente; os taxons *Azospirillum brasilense*, *Moorella thermoacetica*, *Phaeospirillum fulvum* e *Sinorhizobium meliloti* na amostra de complexidade média, identificados pelos números 4, 5, 6 e 7; e o táxon *Salegentibacter salarius* na amostra de complexidade alta, identificado pelo número 8.

A Tabela 1 apresenta os valores para as métricas obtidas com as ferramentas CheckM, MetaQuast e FragGeneScan. Na maioria dos *bins*, tanto para as métricas do CheckM quanto para a estatística proveniente da análise do FragGeneScan, o *binning* taxonômico obteve maior qualidade em relação ao genômico.

A partir dos resultados do MetaQuast, também são apresentados na Tabela 1, a fração alcançada do genoma correspondente ao *bin*, o índice de fragmentação das montagens e as taxas de N50 para cada táxon. Considerando, isoladamente, as métricas de fração de genoma e N50, as montagens que não fizeram uso de abordagens de *binning* apresentaram melhores resultados. Contudo, é válido observar que, para alguns táxons, a diferença entre os valores foi muito pequena.

A análise comparativa entre as abordagens com e sem *binning* considerou apenas essas duas métricas (N50 e fração do genoma) pois não foi possível calcular as demais métricas para a montagem sem *binning* sem que isso implicasse em um viés do *binning* pós-montagem, uma vez que o agrupamento por táxon é necessário para a obtenção dessas métricas.

Levando-se em consideração as cinco métricas (Qualidade, Fragmentação, Fração

[1]Informações complementares a respeito das avaliações realizadas estão disponíveis em https://sourceforge.net/projects/binning-reads/files/docs/details.xlsx

Tabela 1. Comparativo dos resultados das ferramentas de análise.

	CheckM		MetaQuast								FragGeneScan	
	Qualidade		Fração do genoma (%)			Fragmentação(%)		N50			CDS	
ID	Taxonômico	Genômico	Taxonômico	Genômico	Sem binning	Taxonômico	Genômico	Taxonômico	Genômico	Sem binning	Taxonômico	Genômico
1	**13,67**	4,17	6,753	2,531	**6,775**	74,66	**71,5**	607	585	**646**	**1477**	1391
2	**48,14**	0	75,557	82,264	**96,224**	23,71	**18,31**	2146	4877	**6864**	5792	**9857**
3	31,87	**33,64**	29,471	67,019	**86,557**	42,67	**12,26**	1016	13218	**116689**	**5319**	4301
4	**12,89**	3,78	**10,048**	4,997	9,466	**57,29**	65,79	**707**	698	**707**	**9870**	6357
5	10,53	**27,99**	35,275	12,143	**37,578**	**53,91**	61,78	721	724	**758**	**3898**	1444
6	**14,96**	8,33	1,02	0,482	**2,371**	**65,21**	70,62	649	670	**681**	**4012**	3207
7	**14,64**	0	49,916	72,934	**73,457**	**55,47**	59,3	812	869	**870**	16442	**54785**
8	0	**14,27**	31,497	8,715	**71,04**	**49,82**	76,83	853	718	**1123**	3084	**3600**

Táxons nas amostras de complexidade: **Baixa** **Média** **Alta**

do Genoma, N50 e número de CDS), o *binning* taxonômico mostrou-se melhor. Esse resultado se confirma sob três perspectivas. Primeiramente, por contagem total de melhor resultado dentre as cinco métricas, onde em 57,5% dos melhores resultados foram obtidos pelo *binning* taxonômico. Analisando por táxon, os melhores resultados foram obtidos pelo *binner* taxonômico em 62,5% dos táxons. Na análise de cada métrica isoladamente, em 62,5% dos casos o *binning* taxonômico obteve melhores resultados em 4 das métricas, e somente com a métrica N50 o *binning* genômico foi superior.

4. Discussão e Conclusões

Nos experimentos realizados, o *binning* pré-montagem não foi benéfico para a montagem subsequente, no entanto, a investigação mais aprofundada de alguns aspectos pode ser válida. Dentre eles, podemos citar a exploração de mais *binners* taxonômicos e genômicos (como os citados em [Sczyrba et al. 2017] e [Mande 2012]), com o intuito de se verificar seus desempenhos na montagem de genomas. Além disso, novos métodos de agrupamento de sequências - incluindo técnicas de aprendizado de máquina, como a ferramenta apresentada por [Vervier et al. 2018] - são outras possibilidades que podem ser investigadas para avaliação de melhorias na montagem decorrentes da aplicação de binning sobre reads.

A investigação do impacto do *binning* pré-montagem nas amostras completas do CAMI, sem a utilização de subamostragem, pode produzir resultados distintos dos alcançados, dado o possível viés contido na escolha aleatória de *reads*. Ainda considerando o conjunto de dados, ressalta-se que os experimentos foram realizados em ambiente controlado, em que se conhecem os táxons contidos nas amostras, porém, o cenário metagenômico é distinto deste e, considerando organismos ainda não sequenciados, o *binning* genômico pode apresentar melhor adequação ao problema. Dessa forma, torna-se importante também a avaliação de *binners* em cenários com escassez de referências.

Embora os resultados com *binning* tenham se apresentado inferiores aos resultados sem *binning*, é válido citar o desempenho do *binning* taxômico contra o desempenho do *binning* genômico, que se mostra mais adequado ao *binning* sobre *reads*, ao se tratar de benefícios à montagem. Contudo, conforme mencionado, acredita-se que essa comparação pode produzir resultados distintos quando a diversidade contida nas amostras

correspondem a organismos ainda não sequenciados, podendo, nesse caso, alcançarmos resultados mais favoráveis aos *binners* genômicos, que não dependem de referência.

Na métrica N50 o melhor resultado foi com o *binning* genômico, isso pode ter ocorrido devido às montagens maiores obtidas, porém, com inconsistências resultantes, por exemplo, de quimeras. Contudo, ainda em direção ao que fora mencionado, também pode ser um indicativo de falta de informação a respeito dos táxons considerados, já que três das cinco métricas utilizadas na avaliação dependem de referência e isso, também, pode ter influenciado negativamente a avaliação do *binning* pré-montagem.

Referências

Constantinescu, R.-I. (2015). A machine learning approach to dna shotgun sequence assembly. Master's thesis, University of the Witwatersrand.

Girotto, S., Pizzi, C., and Comin, M. (2016). Metaprob: accurate metagenomic reads binning based on probabilistic sequence signatures. *Bioinformatics*.

Li, D., Liu, C.-M., Luo, R., Sadakane, K., and Lam, T.-W. (2015). Megahit: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct de bruijn graph. *Bioinformatics*.

Mande, S. S. (2012). Classification of metagenomic sequences: methods and challenges. *Briefings in Bioinformatics*, 13:669–681.

Menzel, P., Ng, K. L., and Krogh, A. (2016). Fast and sensitive taxonomic classification for metagenomics with kaiju. *Nature Communications*.

Mikheenko, A., Saveliev, V., and Gurevich, A. (2015). Metaquast: evaluation of metagenome assemblies. *Bioinformatics*, 32:1088–1090.

Parks, D. H. (2015). Checkm: assessing the quality of microbial genomes recovered from isolates, single cells, and metagenomes. *Genome research*, 25:1043–1055.

Rho, M., Tang, H., and Ye, Y. (2010). Fraggenescan: predicting genes in short and error-prone reads. *Nucleic acids research*, 38(20):191–191.

Rodriguez, L. M. and Konstantinidis, K. T. (2014). Nonpareil: a redundancy-based approach to assess the level of coverage in metagenomic datasets. *Bioinformatics*.

Schmieder, R. and Edwards, R. (2011). Quality control and preprocessing of metagenomic datasets. *Bioinformatics*, 27:863–864.

Sczyrba, A., Hofmann, P., and Belmann, P. (2017). Critical assessment of metagenome interpretation – a benchmark of computational metagenomics software. *Nature methods*, 14(11):1063–1071.

Sedlar, K. (2017). Bioinformatics strategies for taxonomy independent binning and visualization of sequences in shotgun metagenomics. *Computational and Structural Biotechnology Journal*, 15:48–55.

Vervier, K., Mahé, P., and Vert, J.-P. (2018). *MetaVW: Large-Scale Machine Learning for Metagenomics Sequence Classification*, pages 9–20. Springer New York, New York, NY.

Vollmers, J. (2017). Comparing and evaluating metagenome assembly tools from a microbiologist's perspective - not only size matters! *PLoS ONE 12.1*.

Análise de composição de conjunto de treinamento para avaliação de aprendizagem de máquina aplicada à predição de genes

Raíssa Silva[1], Kleber Padovani[1], Wendel Santos[1], Roberto Xavier[1], Ronnie Alves[1,2]

[1]Universidade Federal do Pará (UFPA)
Programa de Pós-Graduação em Ciência da Computação
Rua Augusto Corrêa, 1, Guamá – Belém – PA – Brazil

[2]Instituto Tecnológico Vale (ITV)
Rua Boaventura da Silva, 955, Nazaré – Belém – PA – Brazil

{r.lorenna, kleber.padovani, wendelrenann, rbxjunior}@gmail.com, ronnie.alves@itv.org

Resumo. *A metagenômica realiza o estudo de comunidades microbianas, conhecidas como metagenomas, descrevendo-as por meio de suas composições e das relações e atividades dos microrganismos que ali coabitam, permitindo assim um maior conhecimento a respeito dos fundamentos da vida e da ampla — e ainda pouco conhecida — diversidade microbiológica. Uma das formas de se realizar tal descrição é por meio da análise de informações de genes contidos em (meta)genomas, extraídas através do processo de identificação de genes em sequências de DNA denominado predição de genes. Este trabalho apresenta um estudo de caso que permite a análise do impacto da composição do conjunto de treinamento ao se utilizar aprendizagem de máquina na predição de genes codificadores de proteína.*

Abstract. *Metagenomics allows the study of microbial communities, known as metagenomes, describing them through their compositions and the relation and activities of the microorganisms that coexist there, thus allowing a deeper knowledge about the fundamentals of life and about the broad microbiological diversity, which is still poorly known. Such description can be achieved by the analysis of information from genes contained in (meta) genomes, extracted through the process of identifying genes in DNA sequences, called gene prediction. This work presents a study that allows the analysis of the impact of the training set composition when using machine learning in protein-coding genes prediction.*

1. Introdução

Predição gênica é um processo crucial na área da biologia computacional. Ela consiste na interpretação de sequências genômicas por meio de algoritmos computacionais. Um dos seus objetivos é a identificação de regiões dentro do genoma que podem ser codificadas em proteínas que irão atuar em processos regulatórios no organismo. Essa identificação nos informa a localização dentro do genoma e as características do gene codificante durante esta etapa de análise funcional. Devido a isso, a predição correta dos genes permite

uma boa acurácia na anotação e caracterização do grupo funcional do organismo e das relações entre seus genes[De Filippo et al. 2012].

Dentro da predição de genes, bem como nas outras áreas da genética molecular e bioinformática, utiliza-se com bastante frequência o acrônimo *ORF*, que representa o termo *Open Reading Frame* e é atribuído às sequências genômicas iniciadas e terminadas por determinadas trincas de nucleotídeos, conhecidas como códons. O códon ATG, também conhecida como *start codon*, determina o início de uma ORF em procariotos, enquanto os códons TAG, TGA e TAA, denominadas *stop codon*, encerram a sequência de nucleotídeos da ORF correspondente[Fassetti et al. 2017].

O conceito de ORF é importante dentro da predição de genes pois toda sequência de nucleotídeos correspondente a um gene — comumente referida por *CDS*, de *Co-Ding Sequence* — corresponde a uma ORF. Sendo assim, a detecção das ORFs é um passo importante na busca por genes nos genomas, inclusive em análises com sequências (meta)genômicas altamente fragmentadas. Contudo, a identificação de ORFs não é suficiente para a identificação de genes, uma vez que, embora todo gene corresponda a uma ORF, a recíproca não é verdadeira. Ao longo do genomas dos organismos, existem diversas ORFs que não correspondem a genes[Sieber et al. 2018].

A Figura 1 ilustra um trecho de um genoma fictício de um procarioto em que podemos observar dentro dele a existência de 1 CDS e 3 ORFs, denominadas como *ORF A*, *ORF B* e *ORF C*. Por meio da ilustração, pode-se observar 3 cenários relevantes relacionados com as ORFs: 1) a existência de ORFs que não correspondem a genes — cenário ilustrado pela ORF A; 2) a existência de ORF que corresponde a CDS — representado pela ORF B; e 3) a existência de ORF dentro de outra ORF, aqui denominado sub ORF — representado pela ORF C. Dessa forma, conclui-se que, embora uma ORF tenha potencial para ser uma CDS, é possível a existência de ORFs dentro das regiões do genoma que não correspondem a CDS — chamadas de regiões intergênicas —, bem como a existência de sub ORFs tanto nas regiões intergênicas quanto nas próprias CDS[Sieber et al. 2018].

Figura 1. Ilustração de trecho do genoma contendo 3 ORFs, denominadas como ORF A, B e C, sendo uma delas — a ORF B — também uma CDS. As trincas correspondentes a *start* e *stop codons* estão destacadas em negrito e identificadas, respectivamente, nas cores verde e vermelho, enquanto as regiões intergênicas estão destacadas em azul.

Uma variedade de métodos computacionais do AM têm sido utilizados para a identificação de gene [Rho et al. 2010] [Zhu et al. 2010],[Hoff 2009] [Noguchi et al. 2008]. Embora o estado da arte seja caracterizado principalmente pelos Modelos Ocultos de Markov, outros métodos robustos de aprendizagem, como as árvores aleatórias [Breiman 2001], ainda não foram aprofundadamente explorados e aplicados no problema de predição de genes.

Nesse contexto, o objetivo deste trabalho é apresentar um estudo de caso que pretende avaliar o impacto da composição do conjunto de treinamento no desempenho da

classificação automática de CDS utilizando árvores aleatórias. O primeiro aspecto observado é a utilização complementar de sub ORFs de CDS como instâncias positivas e o segundo aspecto refere-se ao uso de estratégia de balanceamento em casos de desbalanceamento de classes no conjunto de treinamento.

2. Materiais e Métodos

Na construção do conjunto de treino, foram utilizados 20 genomas finalizados de bactérias e, para a elaboração do conjunto de teste, similarmente, foram utilizados outros 5 genomas finalizados de bactérias. O genoma completo, as CDS e as tabelas de características das CDS constituem as anotações utilizadas para cada organismo de treino e de teste, as quais foram baixadas do NCBI [1]. Essas anotações, bem como os códigos utilizados, podem ser encontrados no material suplementar, disponível online em http://sourceforge.net/p/bsb18-genes.

Para a obtenção das instâncias positivas do conjunto de treino, foram utilizadas as CDS de cada organismo e duas de suas sub ORFs: a maior e a menor. Similarmente, as instâncias negativos foram obtidas por meio da extração das ORFs juntamente com a maior e a menor sub ORF das regiões intergênicas. As instâncias positivas do conjunto de teste correspondem apenas a CDS, sem sub ORFs, enquanto as instâncias negativas são extraídas utilizando a mesma metodologia adotada para a confecção do conjunto de treino.

Foi realizada a extração de características para cada ORF das instâncias positivas e negativas. De cada ORF (ou sub ORF), foram extraídos seis atributos, descritos em [Goés et al. 2014]: 1) percentual de bases de guanina e citosina (percentual GC) na ORF, 2) o percentual GC nas primeiras posições de todos os códons da ORF, 3) nas segundas posições dos códons, 4) nas terceiras posições dos códons, 5) o tamanho da ORF e 6) a variância de códon. Ao fim, cada instância, que corresponde a uma ORF, possui seis atributos e a sua classe.

O balanceamento das classes foi realizado somente no conjunto de treino e apenas para os casos em que o número de instâncias positivas era superior ao número de instâncias negativas. Para esse caso, foram adicionadas aleatoriamente sub ORFs extraídas de instâncias negativas, independentemente se seus tamanhos, ao conjunto de treinamento na tentativa de equilibrar as classes.

Foram construídos assim 4 conjuntos de treinamentos distintos, que foram posteriormente utilizados para treinar 4 modelos independentes de árvores aleatórias [Breiman 2001], com 150 árvores por modelo e utilizando pacote *Caret*, do *R*[Kuhn 2008]. Em dois modelos, denominados modelos *A* e *B*, foi utilizada a estratégia de balancemento de classes nos respectivos conjuntos de treinamento, enquanto nos outros 2 modelos, denominados *C* e *D*, não. Adicionalmente, foram utilizadas sub ORFs na produção de instâncias positivas dos conjuntos de treinamento dos modelos *A* e *C* , enquanto nos modelos *B* e *D* não.

O conjunto de teste foi construído seguindo a mesma estratégia adotada pelo modelo D, considerando assim apenas as CDS, sem suas respectivas sub ORFs, juntamente com as ORFs e suas duas ORFs negativas. Para cada sequência do conjunto de teste, as-

[1] http://www.ncbi.nlm.nih.gov/

sim como ocorrido no treinamento, são extraídos os atributos e utilizados seus respectivos rótulos (indicando se corresponde ou não a uma CDS) para avaliação futura de desempenho dos modelos. Como esperado, esses rótulos são removidos do conjunto de teste e as instâncias são submetidas à predição de cada modelo construído.

Para a análise da precisão dos modelos, foram adotadas três métricas: a *acurácia*, que indica o percentual de predições corretas; a *sensibilidade*, que mede o percentual de instâncias positivas corretamente classificados — indicando assim a capacidade do modelo de identificar um gene como tal; e a *especificidade*, que expressa o percentual de instâncias negativas corretamente classificadas — isto é, o percentual acerto do modelo ao analisar sequências que sabidamente não correspondem a genes. As mesmas sequências utilizadas nos testes dos modelos construídos foram também utilizadas para avaliação de desempenho da ferramenta *FragGeneScan*[Rho et al. 2010].

Adicionalmente, foram realizados experimentos considerando o tamanho das sequências do conjunto de teste. A partir da extração dos atributos, foram analisadas separadamente as instâncias consideradas pequenas, médias e grandes. Como sequências pequenas, foram consideradas as sequências de 60 pares de base (pb) a 300pb; como sequências médias, aquelas de 500pb a 800pb; e, como sequências grandes, foram selecionadas sequências de 1000pb a 1220pb.

3. Resultados

A Tabela 1 apresenta os resultados de acurácia, sensibilidade e especificidade obtidos pelos modelos *A*, *B*, *C* e *D* e pela ferramenta FragGeneScan nos experimentos que consideraram todas as instâncias do conjunto de treinamento, com os melhores resultados destacados em negrito. De modo similar, a Tabela 2 apresenta os valores de sensibilidade e especificidade para os modelos propostos que utilizaram árvores aleatórias, considerando separadamente as instâncias do conjunto de testes correspondentes a sequências pequenas, médias e grandes.

Tabela 1. Valores de acurácia, sensibilidade e especificidade para cada modelo utilizando todas as sequências do conjunto de testes

	Balanceamento	**Sub ORFs**	**Acurácia**	**Sens.**	**Espec.**
Modelo A	SIM	SIM	88,45%	94,29%	80,77%
Modelo B	SIM	NÃO	80,57%	98,11%	57,52%
Modelo C	NÃO	SIM	93,20%	93,79%	**92,43%**
Modelo D	NÃO	NÃO	**93,29%**	95,16%	90,83%
FragGeneScan	—	—	69,47%	**99,91%**	29,45%

Os testes de cada modelo construído e para o FragGeneScan também foram executados separadamente para cada organismo. Os resultados de acurácia, sensibilidade e especificidade desses testes também podem ser encontrados no material suplementar.

4. Conclusões

A partir da análise dos resultados apresentados, algumas conclusões são possíveis no que se refere ao uso da estratégia de balaceamento utilizada e o uso de sub ORFs de CDS. Ao considerarmos as taxas de acurácia dos classificadores no conjunto de testes completo, é

Tabela 2. Valores de sensibilidade e especificidade para sequências pequenas (de 60pb a 300pb), médias (de 500pb a 800pb) e grandes (de 1000pb a 1200pb)

	Pequenas		**Médias**		**Grandes**	
	Sens.	**Espec.**	**Sens.**	**Espec.**	**Sens.**	**Espec.**
Modelo A	70,07%	77,41%	96,77%	**87,10**%	98,89%	**86,97**%
Modelo B	**93,08**%	43,43%	**98,28**%	78,74%	**99,31**%	80,07%
Modelo C	63,55%	**96,80**%	96,70%	86,89%	98,62%	86,20%
Modelo D	71,68%	95,11%	97,21%	85,03%	99,10%	85,44%

possível observar que os melhores resultados foram obtidos pelos modelos treinados com conjuntos de treinamento que não utilizaram a estratégia de balanceamento de classes proposta (modelos C e D), que alcançaram taxas superiores a 93%. Essa conclusão permanece válida ao considerarmos as taxas de especificidade, que foram superiores a 90% somente para esses dois modelos.

Considerando que a estratégia de balanceamento utilizada favorece especialmente a classe negativa do conjunto de treinamento — ou seja, com ela, são fornecidas mais informações a respeito de ORFs que não correspondem a CDS —, sua utilização demonstrou-se inapropriada para os conjuntos de dados utilizados, uma vez que as taxas reduziram com a sua utilização, especialmente a especificidade, que retrata a taxa de acerto dentre as instâncias conhecidamente negativas. Entretanto, é válido notar que houve uma melhoria leve na sensibilidade, indicando que os modelos melhoraram suas capacidades de reconhecimento de exemplos positivos após o balanceamento.

Contudo, essas melhorias mostram-se discretas quando comparadas às reduções em acurácia e especificidade. A sensibilidade aumentou em 0,5% quando se utilizou o balanceamento no conjunto de treinamento que utilizou sub ORFs de CDS e 2,95% ao se utilizar balaceamento no conjunto de treinamento sem sub ORFs de CDS. Em contrapartida, as reduções na acurácia e especificidade ao se utilizar o balanceamento foram, respectivamente, de 4,75% e 11,66% para o conjunto de treinamento com sub ORFs e as reduções nos conjuntos sub ORFs foram, nesta ordem, de 12,72% e 33,31%.

De modo similar, o uso de sub ORFs também não se mostrou apropriado para a construção dos modelos propostos, pois, embora tenha melhorado a especificidade, ele reduziu a sensibilidade. Ao se analisar apenas a especificidade — que avalia a taxa de acerto das instâncias conhecidamente negativas —, nota-se que o uso das sub ORFs foi favorável, especialmente quando se utilizou o balanceamento de classes, com as taxas aumentando de 57,52% para 80,77%. Sem utilizar balanceamento, a especificidade também melhorou, porém, de modo mais discreto. Por outro lado, a sensibilidade reduziu ao se utilizar sub ORFs, tanto nos conjuntos com balanceamento quanto no conjunto sem balaceamento.

Outra conclusão relevante com base nos dados é a obtenção de melhores resultados ao se utilizar árvores aleatórias em comparação com os resultados obtidos pela ferramenta FragGeneScan, que utiliza Modelos Ocultos de Markov. Os modelos baseados em árvores aleatórias apresentaram taxas consideravelmente melhores de acurácia e especificidade. Embora o FragGeneScan tenha obtido melhores taxas de sensibilidade, a distância percentual entre o desempenho da ferramenta e os demais modelos é relati-

vamente pequena. Ao compararmos as taxas do FragGeneScan com o modelo C, por exemplo, que obteve a menor sensibilidade, temos uma diferença percentual de 6,12% em favor do FragGeneScan, contra as diferenças percentuais de 23,73% e 62,98% em favor do modelo C para acurácia e especificidade, respectivamente.

Analisando as taxas de especificidade e sensibilidade dos modelos com árvores aleatórias considerando diferentes tamanhos de sequências, observou-se também que a sensibilidade em todos os modelos melhora à medida que o tamanho das sequências aumentam. Já a especificidade apresentou comportamento similar, mas menos acentuado, quando se utilizou balanceamento. Porém, sem utilizar balanceamento, sequências maiores apresentaram menores taxas de especificidade que as sequências pequenas.

Com base no exposto, concluiu-se que, de forma geral, o uso de sub ORFs como instâncias positivas do conjunto de treinamento e o balanceamento de classes por meio de enriquecimento do conjunto de instâncias negativas interferiram negativamente no poder de generalização dos modelos baseados em árvores aleatórias na classificação supervisionada de genes. A exploração de outras formas de balanceamento, o estudo de características extraídas das sequências e a exploração de parâmetros das árvores aleatórias são propostos como trabalhos futuros de investigação nesta área.

Referências

[Breiman 2001] Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

[De Filippo et al. 2012] De Filippo, C., Ramazzotti, M., Fontana, P., and Cavalieri, D. (2012). Bioinformatic approaches for functional annotation and pathway inference in metagenomics data. *Briefings in bioinformatics*, 13(6):696–710.

[Fassetti et al. 2017] Fassetti, F., Giallombardo, C., Leone, O., Palopoli, L., Rombo, S. E., Ruffolo, P., and Saiardi, A. (2017). Automatic simulation of rna editing in plants for the identification of novel putative open reading frames. *PeerJ Preprints*, 5:e3362v1.

[Goés et al. 2014] Goés, F., Alves, R., Corrêa, L., Chaparro, C., and Thom, L. (2014). Towards an ensemble learning strategy for metagenomic gene prediction. In *Advances in Bioinformatics and Computational Biology*, pages 17–24. Springer International Publishing.

[Hoff 2009] Hoff, K. J. (2009). *Gene prediction in metagenomic sequencing reads*. PhD thesis, Georg August University Göttingen.

[Kuhn 2008] Kuhn, M. (2008). Building predictive models inRUsing thecaretPackage. *Journal of Statistical Software*, 28(5).

[Noguchi et al. 2008] Noguchi, H., Taniguchi, T., and Itoh, T. (2008). Metageneannotator: detecting species-specific patterns of ribosomal binding site for precise gene prediction in anonymous prokaryotic and phage genomes. *DNA research*, 15(6):387–396.

[Rho et al. 2010] Rho, M., Tang, H., and Ye, Y. (2010). Fraggenescan: predicting genes in short and error-prone reads. *Nucleic acids research*, page gkq747.

[Sieber et al. 2018] Sieber, P., Platzer, M., and Schuster, S. (2018). The definition of open reading frame revisited. *Trends in Genetics*, 34(3):167–170.

[Zhu et al. 2010] Zhu, W., Lomsadze, A., and Borodovsky, M. (2010). Ab initio gene identification in metagenomic sequences. *Nucleic acids research*, 38(12):e132–e132.

Tratamento e Integração de Metadados Genômicos Mitocondriais em Filogenômica

Gustavo Saboia[1], Jose de Souza[1], Ana Tereza de Vasconcelos[2], André Elias Rodrigues Soares[2], Kary Ocaña[2]

[1]Faculdade de Educação Tecnológica do Estado do Rio de Janeiro (FAETERJ)
[2]Laboratório Nacional de Computação Científica (LNCC). Petrópolis – RJ – Brasil

{gsaboia,jjunior}@faeterj-petropolis.edu.br,
{atrv,asoares,karyann}@lncc.br

Abstract. *This paper presents the tool MPCreator, a workflow that allows the control and automation in the treatment and integration of genomic metadata for later analyzes of phylogenomics. Despite being a work in progress, MPCreator has already been tested and validated by scientists regarding support in the analytical capacity of metadata. MPCreator is available at https://github.com/gustavoSaboia97/MPCreator and the next steps lead to the use of parallel environments and task distribution.*

Resumo. *Este artigo apresenta a ferramenta MPCreator, um workflow que permite o controle e automação no tratamento e integração de metadados genômicos para posteriores análises de filogenômica. Apesar de ser um trabalho em andamento, o MPCreator já foi testado e validado por cientistas no que tange ao apoio na capacidade analítica dos metadados. Ele está disponível em https://github.com/gustavoSaboia97/MPCreator e os próximos passos conduzem ao uso de ambientes paralelos e distribuição de tarefas.*

1. Introdução

Dados genômicos, conhecidos como *biological big data* [Navale e Bourne 2018], são volumosos, heterogêneos e distribuídos em bancos de dados como o GenBank[1]. O processo de obtenção, tratamento e análise da informação (metadados) contida nesses dados é uma fase crítica na Bioinformática e um desafio atual para a Ciência da Computação e Ciência de Dados [Attwood *et al.* 2017; Fingert 2018]. Embora o processo de acesso a essas bases de dados possa ser automatizado computacionalmente [Yin *et al.* 2017], não existe um algoritmo padrão definido para esta tarefa. Este cenário leva ao pesquisador a realizá-lo total ou parcialmente de forma manual, o que é tedioso, demorado e propenso a erros sistemáticos e falhas na reprodutibilidade.

A filogenética é o estudo das relações evolutivas entre organismos através de informação genética. Os genomas mitocondriais são uma fonte altamente viável de informação genética, uma vez que são abundantes nas células, com uma taxa evolutiva acelerada em relação ao genoma nuclear, além de possuir uma estrutura genômica bem conservada [Wang e Wu 2015]. Embora a utilização de material genético da mitocôndria não seja apropriada em análises de grupos muito distintos, é altamente utilizado em diversos campos das ciências da vida, como genética de populações,

[1] https://www.ncbi.nlm.nih.gov/genbank/

biogeografia, demografia histórica, análises forenses e médicas [Rubinoff e Holland 2005].

Para realizar análises filogenômicas utilizando mitocôndrias é preciso organizar e agrupar os dados a serem usados por programas de análise filogenética como RAxML/ExaML, PhyML, IQ-TREE e BEAST. Nesse artigo a ferramenta proposta MPCreator é um *workflow* que oferece a gerência e automação transparente para a obtenção, tratamento e controle no processo de padronização de dados genômicos mitocondriais.

Atualmente existem *workflows* de código aberto baseados na *web* como Tavaxy[2] (Taverna & Galaxy), BioExtract[3] e FeatureExtract [Wernersson 2005]. Eles são uma alternativa para os usuários iniciantes e oferecem plataformas que permitem o uso *naive* destas análises, além de fornecer maior reprodutibilidade em análises em escala genômica. No entanto, especialistas e especialistas na área necessitam de ferramentas altamente escaláveis, acessíveis por interface de texto e que possam ser integrados em seus *workflows* customizados, que geralmente utilizam APIs do BioPerl[4], Bioconductor[5] e Ensembl[6].

O MPCreator é escrito em Python e as atividades foram modeladas na forma de um arcabouço independente. Ele pode ser acoplado como *subworkflow* em outros *workflows* em filogenética e adaptado a diversos ambientes e sistemas de gerência. O MPCreator é uma ferramenta transparente, eficiente e de fácil uso. Ele formata, minera e organiza os metadados das principais *features* do GenBank: CDS, D-loop, rRNA e tRNA e permite ter um melhor domínio no tratamento das anotações e metadados. Ele está disponível em https://github.com/gustavoSaboia97/MPCreator.

Este artigo está organizado em 4 seções, além desta introdução. A Seção 2 apresenta a motivação. A Seção 3 apresenta o *workflow* MPCreator e Seção 4 apresenta os resultados. Finalmente, a Seção 5 conclui este artigo.

2. Motivação

O processo de tratamento e integração de dados genômicos é um passo crítico que muitas vezes demanda esforço manual do cientista, o que pode levar dias e induzir erros sistemáticos *e.g.*, torna-se inviável encontrar a posição exata no meio de uma sequência nucleotídica com mais de 10 mil caracteres. Embora os dados genéticos se apresentem de maneira simplificada, com longas sequências de caracteres que representam a sequência de nucleotídeos presente na molécula de DNA, no contexto biológico esses dados interagem com o organismo de diversas maneiras. As células acessam o material genético baseado na sua sequência, e podem utilizar determinados trechos como base para a construção de proteínas, mas também simplesmente como regiões que irão indicar outros processos biológicos. Esse tipo de informação se encontra nas bases de dados na forma de metadados que irão informar ao pesquisador onde começam e terminam genes, regiões controladoras e demais *features* do genoma.

---

2 http://www.tavaxy.org/

3 http://bioextract.org

4 https://bioperl.org/

5 https://www.bioconductor.org/

6 https://www.ensembl.org/

A base de dados mais comum para armazenamento de dados genômicos, o NCBI, utiliza um formato padrão denominado GenBank. Os registros do GenBank apresentam três seções: um *header* contém identificadores (ID) e versões; *features* com informações de começo, fim, tipo, *etc.,* de cada região gênica e *sequence* com a sequência nucleotídica em si. O MPCreator atua minerando os metadados presentes nesses registros, identificando cada região da sequência nucleotídica, e permitindo ao pesquisador separar essa regiões em suas análises. Esse processo permite que diversas informações *a priori* sobre as sequências possam ser incorporadas nas análises filogenéticas, além de permitir a recuperação de *features* específicas de interesse.

O MPCreator aceita como entrada um grupo de ID, obtém os arquivos GenBank do NCBI (pelo *header*), minera os metadados *source*, *localization*, *organism*, *gene*, *etc.*, (das *features*) e organiza as sequências gerando como resultado arquivos estruturados e organizados por *features* que podem ser usados como entrada por programas de filogenia como RAxML/ExaML, PhyML, IQ-TREE e BEAST.

3. MPCreator: *workflow* para o tratamento de metadados genômicos

O MPCreator é um *workflow* escrito em Python formado por 7 atividades e pode ser acoplado antes da execução de um *workflow* de filogenômica (Figura 1).

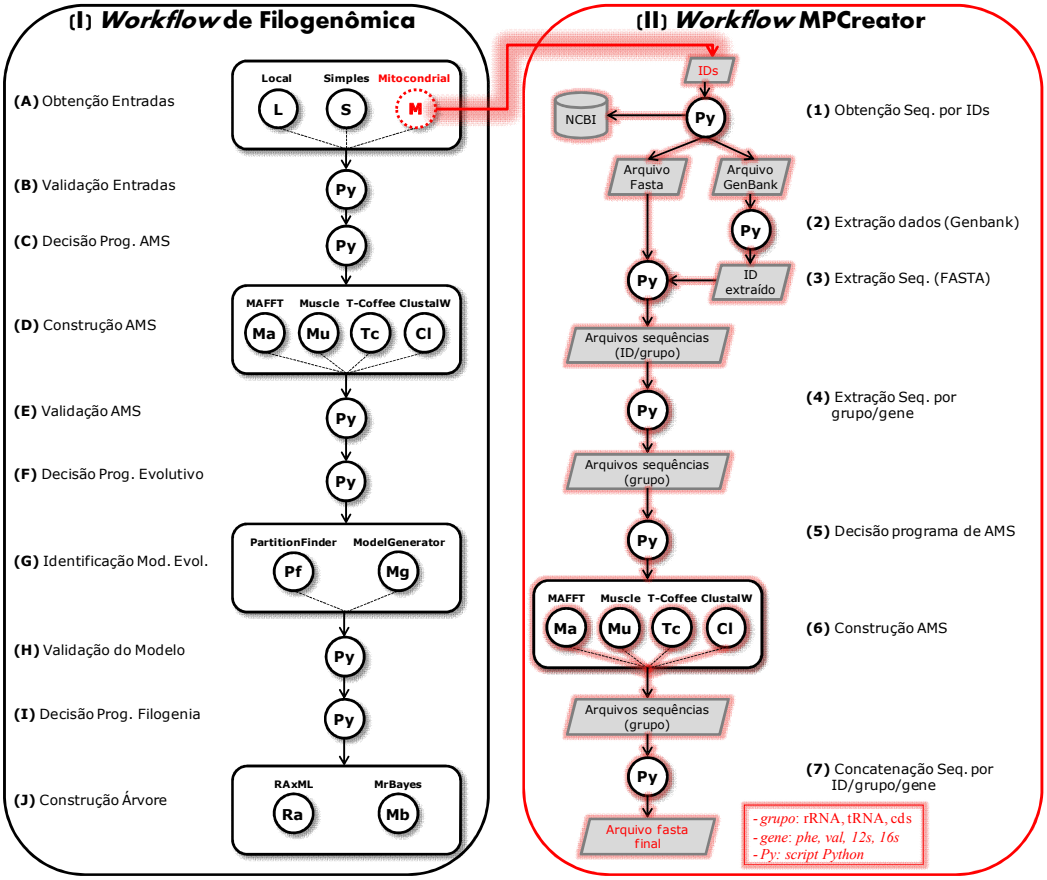


Figura 1. Vista conceitual de um experimento de filogenômica. Na esquerda (I) *Workflow* de filogenômica e na direita (II) *Workflow* MPCreator

O MPCreator recebe como entrada um grupo de ID de genomas mitocondriais, cria acesso ao NCBI (*web* ou local), obtém arquivos, trata e integra dados e gera arquivos agrupados por nomenclatura de *features*. Por questões didáticas nesse artigo, o MPCreator segue 2 níveis de *features* usados para minerar dados seguindo a nomenclatura dos arquivos (Figura 2): por *grupo* (CDS, D-loop, rRNA, tRNA) e por *subgrupo* ("/product" do GenBank *e.g.*, tRNA-Phe, rRNA12s).

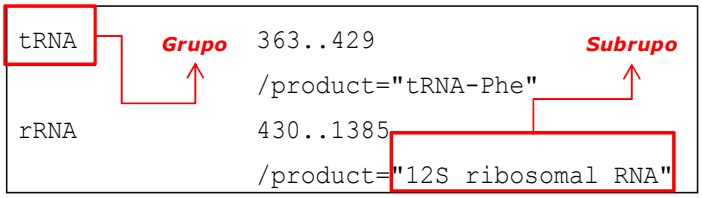


Figura 2. *Grupos* e *subgrupos* no formato GenBank

A atividade 1 obtém os arquivos formatos GenBank e FASTA das sequências dos genomas mitocondriais do NCBI, para cada ID fornecido, usando a API do NCBI - *Entrez Direct: E-utilities*[7]. A atividade 2 extrai para cada ID do arquivo GenBank, os metadados *grupo*, *subgrupo* e índice ou posição da sequência nucleotídica *e.g.*, 363...429.

A atividade 3 extrai, para cada ID, as sequências do arquivo FASTA e o *grupo* e gera arquivos formato FASTA com as sequências agrupadas por ID (*e.g.*, KM926619, KM146616) e *grupo* (Figura 3 A). A atividade 4 extrai as sequências FASTA e as agrupa cruzando com informações do *grupo* e *subgrupo* ao qual pertencem e gera arquivos FASTA (Figura 3 B) a serem alinhados na próxima atividade.

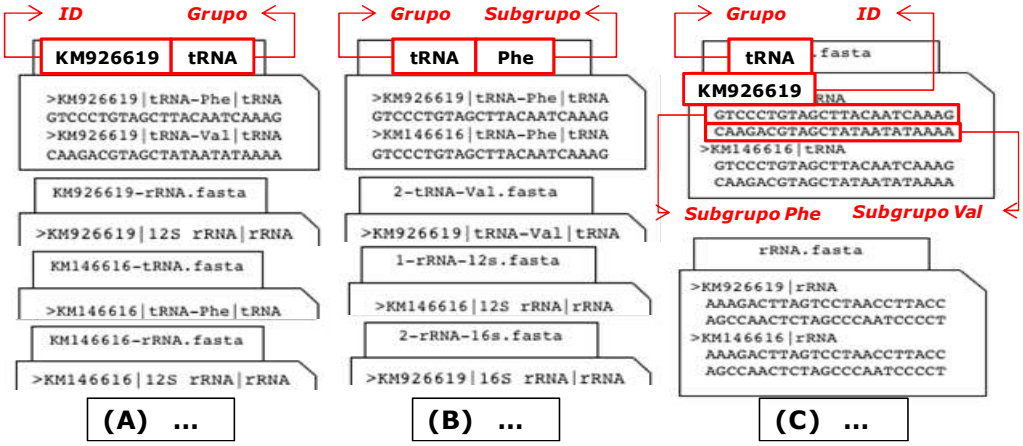


Figura 3. Geração de arquivos FASTA organizados por: (A) ID e *grupos*, (B) *grupos* e *subgrupos* e (C) *subgrupos* concatenados

A atividade 5 toma a decisão sobre qual programa de alinhamento múltiplo de sequências (AMS) será usado (MAFFT, Muscle, ClustalW, T_Coffee). A atividade 6 executa o programa de AMS escolhido pela atividade 5 usando os arquivos FASTA gerados pela atividade 4. A atividade 7 concatena os AMS de forma ordenada por ID, *grupo* e *subgrupo* (Figura 3 A, B) e concatena arquivos com os *subgrupos* (Figura 3 C).

7 https://www.ncbi.nlm.nih.gov/home/develop/api/

4. Resultados

O MPCreator exige apenas como entrada os índices dos arquivos, programa de AMS e diretório de saída. *Transparente*: de código aberto e escrito em Python. *Independente*: pode ser acoplado na composição de qualquer *workflow*. *Eficiente*: executando 1 arquivo (i) na análise quantitativa, o MPCreator levou 1 minuto *versus* 30 minutos pela forma habitual/manual do especialista e (ii) na análise qualitativa, as sequências produzidas por ambos, o MPCreator e o especialista, geraram composições de base e comprimentos idênticos. O processador usado nas execuções é Ryzen 3, geração 2.200, 8 *cores*, 8 GB de RAM e 240 Gb de armazenamento SSD.

O MPCreator é executado via linha de comando com 2 opções e gera as telas da Figura 4: (1) *Automática*, com o comando "*python3 MPCreator.py IDFile.txt*", sendo *IDFile.txt* o arquivo que contém os ID. (2) *Iterativa com Usuário*, o comando "*python3 MPCreator.py*" requer que o usuário forneça as informações: (i) ID ou arquivo contendo ID; (ii) diretório de saída que armazena genomas mitocondriais do GenBank e resultados gerados e (iii) programa de AMS que realizará o alinhamento.

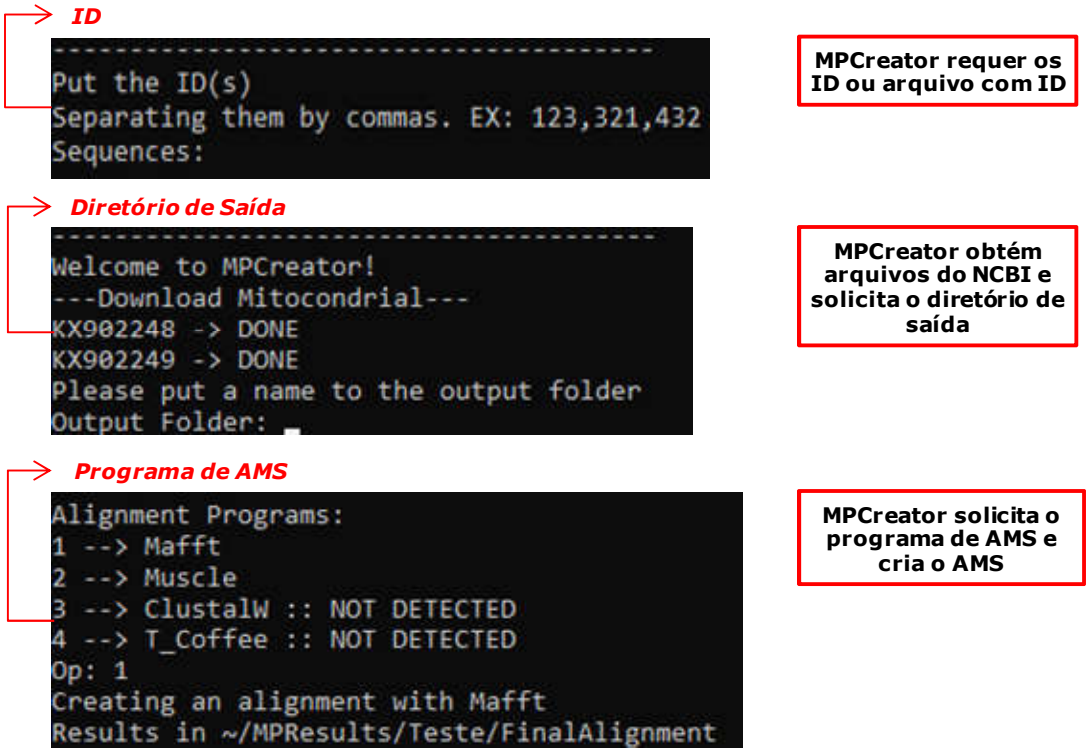


Figura 4. MPCreator requer os ID de interesse

O MPCreator foi testado com 120 genomas mitocondriais da família Columbidae (aves, pombos). A Figura 5 mostra que o tempo de execução é crescente e dependente ao número de entradas. Os algoritmos usados na construção de AMS possuem complexidade de tempo e espaço, o que leva a um alto tempo de execução e uso de memória, pelo que técnicas de paralelismo e distribuição de tarefas são necessárias.

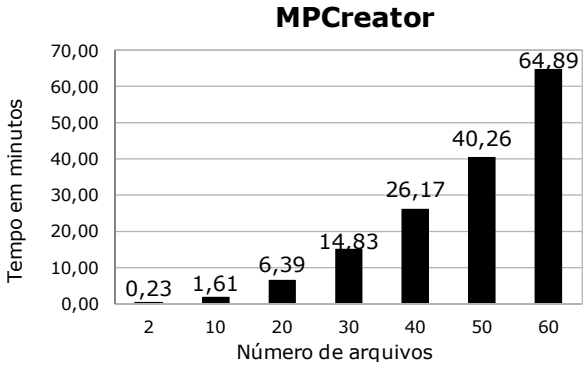


Figura 5. Tempo de execução do MPCreator

5. Conclusões e Trabalhos Futuros

Técnicas de processamento paralelo em plataformas híbridas de GPU/*multicores* e a alocação distribuída de tarefas serão acopladas ao MPCreator para reduzir a demanda de tempo e alocação de memória. O MPCreator v2 está migrando para o sistema de gerência para SAMbA[8], uma extensão do Apache Spark para ambientes de processamento de alto desempenho. Funcionalidades do AnnotationBustR mostram-se também interessantes e potencialmente úteis para serem testadas.

Agradecimentos

Este trabalho foi parcialmente financiado pelos projetos Universal MCTI/CNPq nº 01/2016 processo 429328/2016-8 e JCNE/FAPERJ nº 03/2017 processo 232985. A.E.R.S. é financiado pela CAPES com uma bolsa PNPD.

Referências

Attwood, T. K., Blackford, S., Brazas, M. D., Davies, A. e Schneider, M. V. (2017). A global perspective on evolving bioinformatics and data science training needs. *Briefings in Bioinformatics*, p. bbx100–bbx100.

Fingert, H. J. (2018). Expanding Role of Data Science and Bioinformatics in Drug Discovery and Development. *Clin. Pharmac. & Therap.*, v. 103, n. 1, p. 47–49.

Navale, V. e Bourne, P. E. (2018). Cloud computing applications for biomedical science: A perspective. *PLOS Computational Biology*, v. 14, n. 6, p. e1006144.

Rubinoff, D. e Holland, B. S. (2005). Between Two Extremes: Mitochondrial DNA is neither the Panacea nor the Nemesis of Phylogenetic and Taxonomic Inference. *Systematic Biology*, v. 54, n. 6, p. 952–961.

Wang, Z. e Wu, M. (2015). An integrated phylogenomic approach toward pinpointing the origin of mitochondria. *Scientific Reports*, v. 5, n. 1.

Wernersson, R. (2005). FeatureExtract--extraction of sequence annotation made easy. *Nucleic Acids Research*, v. 33, n. Web Server, p. W567–W569.

Yin, Z., Lan, H., Tan, G., *et al*. (2017). Computing Platforms for Big Biological Data Analytics: Perspectives and Challenges. *Comp. Struct. Biotech. J.*, v. 15, p. 403–411.

[8] https://github.com/UFFeScience/SAMbA

Computational Intelligence applied to Human Genome Data for the Dengue Severity Prognosis

Caio Davi[1], André Pastor[2], Thiego Oliveira[3], Fernando B. Lima Neto[3], Ulisses Braga-Neto[4], Abigail W. Bigham[5], Michael Bamshad[6], Ernesto T. A. Marques[7], Bartolomeu Acioli-Santos[8]

[1]Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco(IFPE)
Paulista, PE – Brazil.

[2]Instituto Federal de Educação, Ciência e Tecnologia do Sertão Pernambucano
Serra Talhada, PE – Brazil

[3]Engenharia de Computação (eComp) – Universidade de Pernambuco (UPE)
Recife, PE – Brazil

[4]Department of Electrical and Computing Engineering – Texas A&M University
College Station, TX – USA

[5]Department of Anthropology – University of Michigan
Ann Arbor, MI – USA

[6]Division of Genetic Medicine – University of Washington
Seattle, WA – USA

[7]Department of Infectious Diseases and Microbiology,
Center for Vaccine Research – University of Pittsburgh
Pittsburgh, PA – USA

[8]Departamento de Virologia, FIOCRUZ-PE
Recife, PE – Brazil

bartacioli@cpqam.fiocruz.br

Abstract. *Dengue has become one of the most important worldwide arthropod-borne diseases around the world. Here, one hundred and two Brazilian dengue virus (DENV) III patients and controls were genotyped for 322 innate immunity gene loci. All biological data (including age, sex and genome background) were analyzed using Machine Learning techniques to discriminate tendency to severe dengue phenotype development. Our current approach produces median values for accuracy greater than 86%, with sensitivity and specificity over 98% and 51%, respectively. Genome data information from 13 key immune polymorphic SNPs was used under different dominant or recessive models. Our approach is a valuable tool for early diagnosis of the severe form of dengue infection and can be used to identify individuals at high risk of developing this form of the disease even in uninfected individuals. The model also identifies various genes involved dengue severity.*

1. Introduction

Dengue is a global public health concern that is caused by dengue virus (DENV), a positive-sense RNA virus belonging to the Flaviviridae family. It is estimated that the annual global incidence is 390 million cases, of which 96 million develop a clinically apparent self-limited disease[Bhatt et al. 2013]. In infected individuals, dengue fever (DF) occasionally progresses to dengue hemorrhagic fever (DHF) and other severe forms, that have been more recently generally classified as Severe Dengue (SD) according to the 2009 World Health Organization[World Health Organization et al. 2009] dengue classification guideline. SD classification includes several life-threatening manifestations including vascular leakage, organ failure, and shock syndrome. The mechanisms leading to the development of SD is an object of intense research. Dengue disease severity has been correlated with viral loads[Paradoa et al. 1987, Soundravally and Hoti 2007, Sakuntabhai et al. 2005], circulating viral proteins[Wang et al. 2003, Libraty et al. 2002] and exacerbated complement activity[Acioli-Santos et al. 2008, Nascimento et al. 2009].

Studies have found associations between single genetic polymorphisms (SNPs) and dengue infection phenotype in multiple genes including dendritic cell-specific inter cellular adhesion molecule 3 (ICAM-3)-grabbing nonintegrin (DC-SIGN), FCcRIIa, Transporter associated with antigen processing (TAP), Vitamin D receptor (VDR), Cytotoxic T lymphocyte-associated antigen-4 (CTLA-4), Acute plasma glycoprotein mannose binding lectin (MBL) and human platelet-specific antigens (HPA), Cytokines (IL, IFN, TNF, etc), in the Fcγ receptor IIA (a pro-inflammatory regulatory Fc receptor) gene and the vitamin D receptor and Human Leukocyte Antigen genes (HLA, i.e human histocompatibility complex)[de Carvalho et al. 2017]. These findings support the hypothesis that both adaptive memory (T-cell responses) and innate immune genes are in the dengue infection disease outcome.

There are a considerable amount of research related to dengue using computational systems[Ali et al. 2017, Muthusamy et al. 2016, Cordeiro et al. 2009]. Indeed, a lot of researches involving Machine Learning (ML) to provide differential diagnostic among DF and DHF has used a variety of techniques, such as decision trees[Tanner et al. 2008] and Support Vector Machines[Gomes et al. 2010]. But almost all of them presents limitations, such as use of clinical data and/or molecular phenotypes that are variable in time and space and/or dependent of human interpretation.

2. Material and Methods

Here we propose a novel approach to dengue infection prediction using (ML) techniques, namely SVM and ANN. This approach could be divided in the four stages described in the subsections bellow: (2.1) Data Acquisition, (2.2) Data Preprocessing, (2.3) Feature Selection, and (2.4) Patient Classification, the entire process is illustrated in Figure 1.

2.1. Data Acquisition

Patients with dengue-related symptoms were screened from three hospitals in the city of Recife, Brazil. The study was reviewed and approved by the Ethics committee of FIOCRUZ-PE: CEP/CPQAM no.11/11, C.A.A.E. 0009.0.095.000-11, IORG0001419. A set of characteristics was investigated (the type of infection, age, sex, and genetic data - 322 loci polymorphisms) over 102 patients already positively diagnosed with DF (n=27) or SD (n=75).

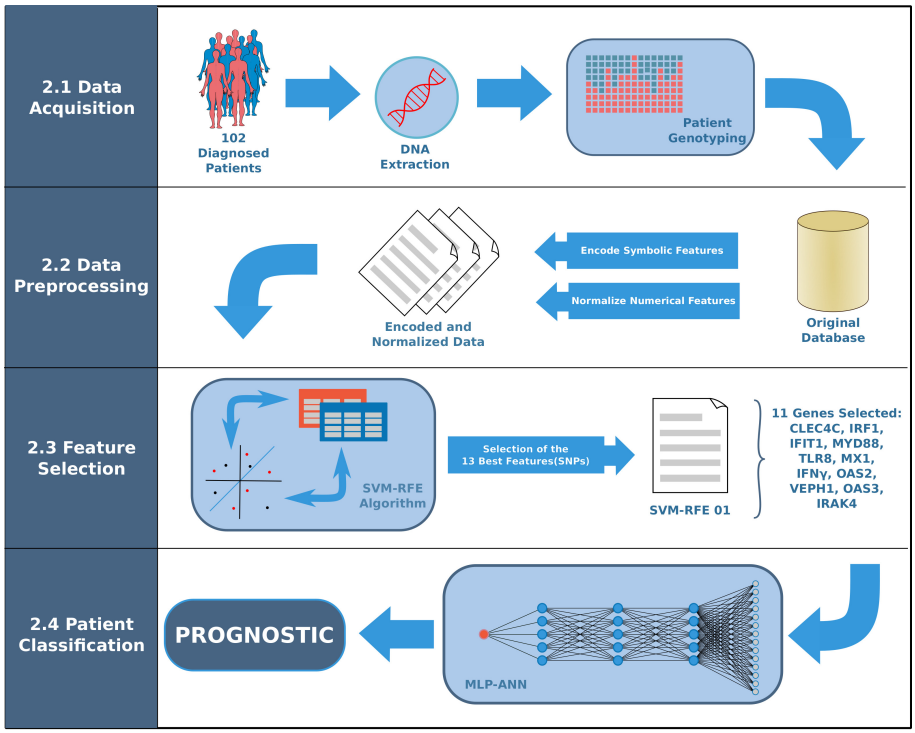


Figure 1. Flowchart of SVM-ANN/genome dengue classifier. 2.1-Data acquisitiohttps://www.overleaf.com/project/5ba94de202f3bd2d56e316dbn was performed by illumina genotyping of all dengue patients and then stored into a database. 2.2- Data preprocessing was performed to encode and normalize data into a suitable format for the ML step. 2.3- Feature selection was performed by keeping the best SNP subset. 2.4- A MLP-ANN classifier was learned for dengue prognosis based on the features previously selected.

2.2. Data Preprocessing

All data was encoded in values between -1 and 1. The genetic data, particularly, were encoded into indicators using a categorical scheme as homozygous dominant, heterozygous or homozygous recessive, resulting in one feature per SNP. Age, as a numeric feature, was normalized also into values between -1 and 1. Missing data was treated as a separate category. Age, the only non-categorical feature, had no missing data.

2.3. Feature Selection

Due to the high dimensionality of the data (325 categorical features and up to 900 features after converted into indicator features) and aiming to avoid the curse of dimensionality[Keogh and Mueen 2011], backward feature elimination using the SVM-RFE algorithm [Guyon et al. 2002] with a linear classification kernel[Fan et al. 2008] was performed.

The SVM-RFE model was implemented using the freely downloadable scikit-learn library provided by Pedregosa *et al.*[Pedregosa et al. 2011]. The process was repeated for all datasets using 3-fold cross-validation[McLachlan et al. 2005] to choose the SVM parameters γ and C from combinations of 0.01, 0.1, 1.0, 10.0 for each one. The process selected the value 1.0 for both. The best subset comprises 13 loci found in 11 genes, as shown in Figure 1.

2.4. Patient Classification

After that, the defined subset was used to train a Multi-Layer Perceptron-Artificial Neural Network (MLP-ANN). The MLP-ANN was implemented using the freely downloadable ML python library[Pedregosa et al. 2011]. The rectified linear unit (ReLU) function was used as the activation function and Limited-memory BFGS[Byrd et al. 1995] was used for weight optimization. The initial value of the parameter α was 0.001 and the optimal topology was found after a search in the bi-dimensional space (layers x neurons) using stratified k-fold cross-validation with k=10. The best topology found for the previously selected subset (SVM-RFE01) consisted of 3 hidden layers of 5 neurons per layer, illustrated in Figure 1 (in the MLP-ANN box).

3. Results

The Feature Selection (described in Section 2.3) found the best subset of features to classify a patient as DF ou SD. This subset comprises 13 SNPs found in 11 genes: CLEC4C, IRF1, IFIT1, MYD88, TLR8, MX1, OAS2, VEPH1, IFNγ, OAS3, IRAK4. Many of those genes have well-known influence in the immune system and the metabolic pathways of each one are subject of further studies in our research.

Those genes are used as input for a MLP-ANN. The accuracy estimation for this subset of features were obtained by the bolstered resubstitution method[Braga-Neto and Dougherty 2004a]. The variance of the bolstering kernels were set using the "Naïve Bayes" method[Jiang and Braga-Neto 2014]. Bolstered estimation is more accurate than cross-validation for small datasets[Braga-Neto and Dougherty 2004b]. The estimated accuracy rates are reported in Table 1. Also displayed, for comparison, are the stratified 10-fold cross-validated accuracy rates obtained in the network selection step (as can be seen, these accuracy rates are inflated by selection bias).

Table 1. Estimated statistics for our two best classifiers.

Subset	Cross-Validation	Bolstered Resubstitution	
	96%	86.1%	Accuracy
SVM-RFE01	100%	98.64%	Sensitivity
	85%	51.85%	Specificity

4. Conclusion

Here we applied ML techniques, namely SVM and ANN, to develop a classifier based on genomic polymorphism to predict the risk of SD. In the Feature Selection step (see Figure 1) we have used a SVM to select the best subset to be used in this classification. This subset consists in 13 SNPs located in 11 innate immune genes: CLEC4C, IRF1, IFIT1, MYD88, TLR8, MX1, OAS2, VEPH1, IFNγ, OAS3, IRAK4. The role of these genes in the immune mechanisms involved in the severe dengue phenotype are very promising, though currently this research is in a preliminary phase. The use of genome data to predict diseases has several advantages, especially due it can be done at any time and in a broad human sample tissue, during early virus infection and/or before the infection itself.

The training dataset used for training the SVM/ANN were well characterized. It was shaped by data from 13 SNPs to produce a classifier with accuracy level greater than

86%, with a sensitivity of 98,64%, and specificity around 51%. This type of diagnostic tool is useful for patient triage, especially during disease outbreaks.

The method presented here provided very robust results for a prognosis (for dengue severity) classifier, as demonstrated by the error assessment calculations. It is able to select the optimal loci combination data and, then, to correctly (pre) classify the patient that will develop severe phenotype based only in its genome background. Our method can be easily replicate for other genetic based/genetic influenced diseases, helping to find optimal loci sets to understand the molecular architecture of different pathologies, and driving to potential therapeutics target genes.

References

Acioli-Santos, B., Segat, L., Dhalia, R., Brito, C. A., Braga-Neto, U. M., Marques, E. T., and Crovella, S. (2008). Mbl2 gene polymorphisms protect against development of thrombocytopenia associated with severe dengue phenotype. *Human immunology*, 69(2):122–128.

Ali, I., Humayun, F., Azam, S., Munir, A., Rizwan, M., et al. (2017). Computational tool for classification of dengue virus. *J Appl Bioinforma Comput Biol 6*, 3:2.

Bhatt, S., Gething, P. W., Brady, O. J., Messina, J. P., Farlow, A. W., Moyes, C. L., Drake, J. M., Brownstein, J. S., Hoen, A. G., Sankoh, O., et al. (2013). The global distribution and burden of dengue. *Nature*, 496(7446):504.

Braga-Neto, U. and Dougherty, E. (2004a). Bolstered error estimation. *Pattern Recognition*, 37(6):1267–1281.

Braga-Neto, U. M. and Dougherty, E. R. (2004b). Is cross-validation valid for small-sample microarray classification? *Bioinformatics*, 20(3):374–380.

Byrd, R. H., Lu, P., Nocedal, J., and Zhu, C. (1995). A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208.

Cordeiro, M. T., Braga-Neto, U., Nogueira, R. M. R., and Marques Jr, E. T. (2009). Reliable classifier to differentiate primary and secondary acute dengue infection based on igg elisa. *PloS one*, 4(4):e4945.

de Carvalho, C. X., Cardoso, C. C., Kehdy, F. d. S. G., Pacheco, A. G., and Moraes, M. O. (2017). Host genetics and dengue fever. *Infection, Genetics and Evolution*.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874.

Gomes, A. L. V., Wee, L. J., Khan, A. M., Gil, L. H., Marques Jr, E. T., Calzavara-Silva, C. E., and Tan, T. W. (2010). Classification of dengue fever patients based on gene expression data using support vector machines. *PloS one*, 5(6):e11267.

Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422.

Jiang, X. and Braga-Neto, U. (2014). A naive-bayes approach to bolstered error estimation in high-dimensional spaces. In *Signal and Information Processing (GlobalSIP), 2014 IEEE Global Conference on*, pages 1398–1401. IEEE.

Keogh, E. and Mueen, A. (2011). Curse of dimensionality. In *Encyclopedia of Machine Learning*, pages 257–258. Springer.

Libraty, D. H., Endy, T. P., Houng, H.-S. H., Green, S., Kalayanarooj, S., Suntayakorn, S., Chansiriwongs, W., Vaughn, D. W., Nisalak, A., Ennis, F. A., et al. (2002). Differing influences of virus burden and immune activation on disease severity in secondary dengue-3 virus infections. *The Journal of infectious diseases*, 185(9):1213–1221.

McLachlan, G., Do, K.-A., and Ambroise, C. (2005). *Analyzing microarray gene expression data*, volume 422. John Wiley & Sons.

Muthusamy, K., Gopinath, K., and Nandhini, D. (2016). Computational prediction of immunodominant antigenic regions & potential protective epitopes for dengue vaccination. *The Indian journal of medical research*, 144(4):587.

Nascimento, E. J., Silva, A. M., Cordeiro, M. T., Brito, C. A., Gil, L. H., Braga-Neto, U., and Marques, E. T. (2009). Alternative complement pathway deregulation is correlated with dengue severity. *PloS one*, 4(8):e6782.

Paradoa, M. P., Trujillo, Y., and Basanta, P. (1987). Association of dengue hemorrhagic fever with the hla system. *Haematologia*, 20(2):83–87.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Sakuntabhai, A., Turbpaiboon, C., Casadémont, I., Chuansumrit, A., Lowhnoo, T., Kajaste-Rudnitski, A., Kalayanarooj, S. M., Tangnararatchakit, K., Tangthawornchaikul, N., Vasanawathana, S., et al. (2005). A variant in the cd209 promoter is associated with severity of dengue disease. *Nature genetics*, 37(5):507.

Soundravally, R. and Hoti, S. (2007). Immunopathogenesis of dengue hemorrhagic fever and shock syndrome: role of tap and hpa gene polymorphism. *Human immunology*, 68(12):973–979.

Tanner, L., Schreiber, M., Low, J. G., Ong, A., Tolfvenstam, T., Lai, Y. L., Ng, L. C., Leo, Y. S., Puong, L. T., Vasudevan, S. G., et al. (2008). Decision tree algorithms predict the diagnosis and outcome of dengue fever in the early phase of illness. *PLoS neglected tropical diseases*, 2(3):e196.

Wang, W.-K., Chao, D.-Y., Kao, C.-L., Wu, H.-C., Liu, Y.-C., Li, C.-M., Lin, S.-C., Ho, S.-T., Huang, J.-H., and King, C.-C. (2003). High levels of plasma dengue viral load during defervescence in patients with dengue hemorrhagic fever: implications for pathogenesis. *Virology*, 305(2):330–338.

World Health Organization, W., for Research, S. P., in Tropical Diseases, T., of Control of Neglected Tropical Diseases, W. H. O. D., Epidemic, W. H. O., and Alert, P. (2009). *Dengue: guidelines for diagnosis, treatment, prevention and control*. World Health Organization.

Specific Substring Problem: an application in bioinformatics

Lucas B. Rocha[1*], **Said Sadique Adi**[1], **Eloi Araujo**[1]

[1]FACOM – Universidade Federal do Mato Grosso do Sul (UFMS)
Campo Grande – MS – Brazil

lucas.lb.rocha@gmail.com, {said,feloi}@facom.ufms.br

Abstract. *Given two sets of sequences A and B, the Substring Specific problem is to find all minimum substrings in A having distance at least k for each subsequence in B. This work addresses three new implementations for the Maaß algorithm when the Hamming distance is considered: a naive cubic-time algorithm and two quadratic-time algorithms. We run tests to compare the running time of these implementations and another recently described algorithm implementation that uses the edit distance. In addition, we conducted preliminary testing on a large Tara Ocean database, looking for efficient and effective strategies for finding unique sequences in a set of sequences comparing with the other.*

1. Introduction

As a result of the advancement of DNA sequencing technologies, there are currently a large number of sequenced genomes. Tara Expedition were journeys where an international group of researchers sequenced the metagenome of over 35,000 different samples from 210 different regions in the world, resulting in a total of 7.2 terabytes of genomic data [Pesant et al. 2015, Bork et al. 2015]. This large amount of data needs now to be processed in searching for meaningful biological information. In this context, an important computational problem is determining markers, that are substrings from a set of sequences that do not occur on sequences of other sets, that is, small substrings found only in the former set. This is the Specific Segment Problem, proposed in [Gusfield 1997].

Other applications for determining markers include finding specific regions for primer design in PCR technology [Montera and Nicoletti 2008], and specific organisms in metagenomes that live in a given diseased tissue for the early diagnosis of cancer [Zitvogel et al. 2018, Shigefuku et al. 2017].

We formulate this problem in a simpler way, considering only two sets of sequences, that is, the problem of finding all the minimum substrings in a set of sequences A that do not appear in another set B. The time required and the quality of the markers found depend on how to define the distance function. Furthermore, since two equal sequences may have few differences due to some mutations or read errors, we consider only a pair of sequences having a distance greater than or equal to a given integer threshold k.

Dobre [Dobre 2017] implemented two versions of an algorithm described by Gusfield [Gusfield 1997] that uses edit distance, and spends time $O(n^3)$ in the worst case. Maaß [Maaß 2003] also described an algorithm that uses Hamming distance and spending time $O(n^2)$. The Hamming distance seems to be, in addition to being easier to calculate, a natural measure of similarity in many biological applications [Lanctot et al. 2003].

*This work is partially sponsored by UFMS and CAPES (scholarship 51001012028D6).

In this work, we describe three implementations of two different algorithms to solve the considered problem using Hamming distance: first, one that implements a naive strategy and, then, two versions of the Maaß's algorithm that use two different data structures; we compare the time spent by each of the three implementations and by that from Dobre's work [Dobre 2017]. In addition, we perform tests with two different samples obtained from the Tara project database, showing the time spent processing them.

This work is organized as follows: in Section 2, we describe the algorithms for the problem considering edit and Hamming distance, and we present the time complexity of each one. In Section 3, we show the practical results comparing our results to those from Dobre [Dobre 2017]; moreover, for two samples of Tara project database, we show the results regarding the time spent in simplified experiments using small subsets of two different samples. This was done to estimate the expected total spent time when comparing whole samples. Finally, in Section 4, we discuss the results and a strategy that we intend to adapt and follow in order to make this work feasible in practice.

2. Preliminaries

An *alphabet* Σ is a finite set of symbols. We denote a sequence s over Σ by $s_1 s_2 \dots s_\ell$ where each symbol $s_i \in \Sigma$. We say the *length* of s, denoted by $|s|$, is ℓ. We say that the sequence $s_i s_{i+1} \dots s_j$, with $i \geq 1$ and $j \leq |s|$ is a *substring* of s and we denote it by $s[i, j]$. A substring $s[1, j]$ is called a *prefix* of s.

Given a distance function, we say that a string s is a *minimal* string that have distance at least k to any string in a set of sequences A if the distance between s and any substring of A is greater than or equal k and, for any prefix s' of s with $|s'| < |s|$, there is a substring t in A such that the distance between s' and t is smaller than k.

Problem 1 (Specific Substring Problem — SSP-$\mathcal{D}$) *Given two sets A and B of sequences such that $|A| = M$, $|B| = N$, find all the minimal substrings in A that have distance at least k to any substring in B for some chosen distance function $\mathcal{D}$.*

The *edit distance (ED)* between two sequences is the minimum number of edit operations *insertions*, *deletions* and *substitutions* required to transform one sequence into another. In [Dobre 2017], Dobre implements an algorithm, called here ***KED,*** described by Gusfield [Gusfield 1997] that solves the SSP-ED in $O(MN \cdot n^3)$ time where n is the length of the longest sequence in $A \cup B$.

Given two n-length sequences s and t, the *Hamming distance (HD)* $d_H(s, t)$ between s and t is defined as

$$d_H(s, t) = \sum_{i=1}^{n} |s_i \neq t_i|,$$

where $|s_i \neq t_i|$ is equals to 1 if $s_i \neq t_i$, and is equal to 0 otherwise.

The *length of the smallest prefix* of s and t with k differences under the Hamming distance is denoted by $p(s, t, k)$ and it can be found by traversing s and t at the same time from left to right until we found k differences; if this number doesn't exist, we define $p(s, t, k) = \min\{|s|, |t|\} + 1$. This process can be done in $O(|s| + |t|)$ time.

Gusfield [Gusfield 1997] describes a naive algorithm for a restricted version of Problem SSP-HD. We implement and call it ***KHD1*** algorithm. This restricted version of

Problem 1 considers $|A|$ and $|B|$ having just one sequence each, saying s and t. Then, this naive algorithm has as input two sequences s and t, where $|s| = m$, $|t| = n$, and an integer k, and computes $r[i] = \max_j\{p(s[i, m], t[j, n], k)\}$ for each i. Each $r[i]$ can be computed in $O(mn)$ time. Thus, this algorithm is computed in $O(m^2n)$ time, i. e, $O(n^3)$ if we consider $n = m$.

For the same restricted version, we also implemented a faster algorithm described by Maaß [Maaß 2003]. This algorithm is summarized in Algorithm 1 and the corresponding implementation is called *KHD2*.

Algorithm 1 [Maaß 2003]

1: $r[i] \leftarrow 0$ for $i = 1, \ldots, m$
2: **for** all $(i, j) \in \{1, \ldots, m\} \times \{1, \ldots, n\}$, such that $i = 1$ or $j = 1$ **do**
3: $\quad max \leftarrow \min\{m, i + n - j\}$
4: $\quad d \leftarrow p(s[i, m], t[j, n], k - 1)$
5: $\quad L \leftarrow 0$
6: $\quad$ **while** $i + L \leq max$ **do**
7: $\quad\quad$ **while** $d + i + L < max \wedge\ s_{d+i+L} = t_{d+j+L}$ **do** $d \leftarrow d + 1$
8: $\quad\quad$ **if** $max \geq d + i + L$ **then** $d \leftarrow d + 1$
9: $\quad\quad$ **while** $TRUE$ **do**
10: $\quad\quad\quad$ **if** $r[i + L] < d$ **then** $r[i + L] \leftarrow d$
11: $\quad\quad\quad L \leftarrow L + 1$
12: $\quad\quad\quad d \leftarrow d - 1$
13: $\quad\quad\quad$ **if** $i + L > max\ \vee\ s_{i-1+L} \neq t_{j-1+L}$ **then break**
14: **return** r

The number of pairs (i, j) where $i = 1$ or $j = 1$ is $n+m-1 = O(n+m)$. The algorithm spends time $O(\min\{n, m\})$ to compute k differences between the $O(\min\{n, m\})$ pairs of sequences in each iteration. Thus, Maaß algorithm spends $O((n + m)^2)$ time, i. e, $O(n^2)$ if we consider $n = m$.

The Longest Common Extension (LCE) of two strings s and t is defined as the length of the longest s and t common prefix. Maaß [Maaß 2003] claims that, since LCE can be found in constant time by using a suffix tree, this structure can be used for executing Line 7 in constant time speeding up the algorithm in practice, although the theoretical complexity is still $O(n^2)$. We implemented the algorithm considering this improvement and we call it *KHD3*.

3. Experiments and results

The experiments reported in Section 3.1 were performed in a server Intel Xeon(R) E5-4650 2.7 GHZ, with 20 MB of cache memory, 95 GB of RAM memory and 8 Processing cores and those reported in Section 3.2 were performed on a cluster with 40 nodes with processors Xeon(R) CPU X3440@2.53GHz of 4 cores, 4 GB of RAM per node. In Section 3.1, we compare the running time of KHD1, KHD2, KHD3 and KED that are implemented in C++. The goal of the test is to find fast algorithms and good parameters for using in the fastest strategies for solving SSP. In Section 3.2, we use the best strategies to estimate the required time to compare two samples from Tara project sequences.

3.1. Tests with Homo sapiens sequences

In order to run the experiments with real data, two sets A and B including sequences obtained from the *Homo sapiens* database were used [Dobre 2017]. Set A includes three sequences, namely $hsarhgdig$ (4398 characters), $hsankr10$ (38530 characters) and $hsaff4$ (90284 characters), and the set B includes 7 sequences, namely $hsascl2$ (4455 characters), $hsankrd43$ (5457 characters), $hsa1bg$ (8694 characters), $hsalg10b$ (14972 characters), $hsbad$ (16877 characters), $hsascc2$ (51655 characters) and $hsasz1$ (66302 characters). The running times obtained for each sequence in A when we set $k \in \{30, 300, 600\}$ is shown in Table 1. We noticed that *KHD2* was the fastest implementation during the tests. This behavior suggests that the use of the Hamming distance to solve SSP-HD is probably a good choice in terms of time when using Tara sequence samples.

sequence in $A \setminus$ implementation	KHD1	KHD2	KHD3	KED
$hsarhgdig$	00:01:10	00:00:36	00:02:07	00:00:40
$hsankr10$	00:07:20	00:02:10	00:04:00	00:04:40
$hsaff4$	00:14:20	00:04:17	00:06:30	00:10:40
sequence in $A \setminus$ implementation	KHD1	KHD2	KHD3	KED
$hsarhgdig$	00:08:03	00:02:18	00:07:30	00:04:06
$hsankr10$	01:06:00	00:18:30	00:28:20	00:36:17
$hsaff4$	02:20:00	00:28:44	00:35:00	01:26:00
sequence in $A \setminus$ implementation	KHD1	KHD2	KHD3	KED
$hsarhgdig$	00:15:44	00:03:42	00:18:40	00:06:41
$hsankr10$	02:18:50	00:33:50	00:55:00	01:11:00
$hsaff4$	03:32:06	01:09:06	02:01:00	02:33:01

Table 1. Average running time (hours:minutes:seconds) obtained with the tests for sequences in A when they are compared to each sequence in B using $k = 30, 300$ and 600 respectively.

3.2. Tests with Tara sequences

From now on, we will use *KHD2* (because it has the best time for Hamming distance) and *KED* in the tests. To run the tests with Tara sequences, two samples were used: ERR599040 and ERR59914. Each sample has thousands of sequences. In order to estimate the time we would spend for comparing the two samples, we did some tests considering only 1, 10 and 100 sequences from ERR599040 (set A) and 2 million sequences from ERR599148 (set B).

Number of sequences considered in ERR599040	KHD2	KED
1	00:10:38	00:10:03
10	01:34:00	01:35:32
100	12:40:00	12:43:00

Table 2. Running time (hours:minutes:seconds) for run sequences in ERR599040 compared to 2,000,000 of sequences in ERR599148, with $k = 30$.

The running times obtained for the two restricted samples of the Tara project database are shown on Table 2. We noticed that *KHD2* and *KED* programs have similar run-times. Moreover, we estimate that it would take a long time to run the entire

sample set. This behavior justifies the reduction of sample sequences with some tool. It will be better discussed in Section 4.

4. Discussion, perspectives and conclusions

This work addresses the problem of specific strings (SSP). Two versions for previously described algorithms for the SSP-HD problem are showed with the second one presenting two variations. We implemented this versions and carried out tests with real *Homo sapiens* sequences data. We also performed a comparison with SSP-ED algorithm implemented by Dobre [Dobre 2017]. According to Table 1, the *KHD2* implementation has the better running time, suggesting that Hamming distance is indeed a promising measure for the considered problem at least concerning time consumption.

However, as well as it can be observed in Table 2, *KHD2* and *KED* show a slight difference in processing time, but both implementations are not fast enough for our main problem. Since we spend a reasonable time to compare a few sequences in ERR599040 with only thousands of sequences in ERR599148, we estimate that it would take hundreds of years to compare all the sequences. In order to overcome this difficult, we need to have a way to reduce the amount of sequences to be processed without giving up our central target that is to find sequences in one set that is not in another. Thus, we plan including a clustering preprocessing step with the CD-HIT tool [Li and Godzik 2006], more specifically the CD-HIT-2D, which is a tool that compares two protein databases and identifies similar sequences considering certain threshold. The FASTA file of non-similar sequences after clustering seems very interesting because it has unique sequences that can be tested using our algorithm. All other sequences can be discarded in this process. The CD-HIT used is [Fu et al. 2012].

Qty. of sequences	Time	Time (8 CPUs)	Clustered	ERR599148 Reduction
10000	00:01:06	00:00:12	3859	38.59%
20000	00:03:11	00:00:44	9957	49.78%
30000	00:08:20	00:01:32	17315	57.71%
40000	00:12:00	00:02:34	25465	63.66%
Qty. of sequences	Time	Time (8 CPUs)	Clustered	ERR599148 Reduction
10000	00:01:19	00:00:10	6360	63.60%
20000	00:04:23	00:00:33	15343	76.72%
30000	00:08:44	00:01:06	25060	85.54%
40000	00:14:16	00:01:48	35269	88.17%

Table 3. CD-HIT-2D Running time (hours:minutes:seconds) to 60% and 58% of similarity, with the same amount of sequences for ERR599040 and ERR599148.

Tests with TARA datasets are used to evaluate the behavior of the two sequential algorithms when it needs to handle a very large amount of data. It can be observed in Table 2 that both algorithms are not fast enough. The runtimes and percentage reductions of the number of sequences in ERR599148 are shown in Table 3. We noticed a significant reduction with 58% of the similarity that would lead us to conclude it is possible to find a good clustering, focusing on reducing the amount of sequences. In addition, we estimate that it takes about one week to cluster the dataset A and B with around 120

million of sequences. In the worst case, if $|A| = N$, $|B| = M$ and $KDH2 = O(n^2)$, it spends $O(NM \cdot n^2)$ for running all sequences. We plan to investigate new ways to cluster sequences to get smaller sets of sequences based on the CD-HIT. In addition, we want to search other tools or techniques for clustering that are faster than CD-HIT. Furthermore, since reducing the sample may not be enough, it is always necessary to improve the *KHD2* run-time. Moreano *et. al* [Feuser and Moreano 2018] describe a parallel approach for the *KED* algorithm achieving good speedups. Hence, we are considering a parallel computing approach for *KHD2* algorithm.

References

Bork, P., Bowler, C., de Vargas, C., Gorsky, G., Karsenti, E., and Wincker, P. (2015). Tara oceans studies plankton at planetary scale. *Science*, 348(6237):873–873.

Dobre, J. A. (2017). O problema da seleção de segmentos específicos: algoritmos e aplicações. Universidade Federal de Mato Grosso do Sul. Master's thesis.

Feuser, L. and Moreano, N. (2018). Parallel solutions to the k-difference primer problem. In *International Conference on Computational Science*, pages 506–523. Springer.

Fu, L., Niu, B., Zhu, Z., Wu, S., and Li, W. (2012). Cd-hit: accelerated for clustering the next-generation sequencing data. *Bioinformatics*, 28(23):3150–3152.

Gusfield, D. (1997). Algorithms on strings, trees and sequences: Computer science and computational biology, 1st editio. *New York, United States*, page 534.

Lanctot, J. K., Li, M., Ma, B., Wang, S., and Zhang, L. (2003). Distinguishing string selection problems. *Information and Computation*, 185(1):41–55.

Li, W. and Godzik, A. (2006). Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22(13):1658–1659.

Maaß, M. G. (2003). A fast algorithm for the inexact characteristic string problem. Technical Report TUM-I0312, Fakultät für Informatik, TU München.

Montera, L. and Nicoletti, M. C. (2008). The PCR primer design as a metaheuristic search process. In *International Conference on Artificial Intelligence and Soft Computing*, pages 963–973. Springer.

Pesant, S., Not, F., Picheral, M., Kandels-Lewis, S., Le Bescot, N., Gorsky, G., Iudicone, D., Karsenti, E., Speich, S., Troublé, R., et al. (2015). Open science resources for the discovery and analysis of tara oceans data. *Scientific data*, 2:150023.

Shigefuku, R., Watanabe, T., Kanno, Y., Ikeda, H., Nakano, H., Hattori, N., Matsunaga, K., Matsumoto, N., Kanno, S.-i., Nosho, K., et al. (2017). Fusobacterium nucleatum detected simultaneously in a pyogenic liver abscess and advanced sigmoid colon cancer. *Anaerobe*, 48:144–146.

Zitvogel, L., Ma, Y., Raoult, D., Kroemer, G., and Gajewski, T. F. (2018). The microbiome in cancer immunotherapy: Diagnostic tools and therapeutic strategies. *Science*, 359(6382):1366–1370.

Avaliação do RAxML no Supercomputador Santos Dumont

Micaella Coelho, Carla Osthoff, Kary Ocaña

Laboratório Nacional de Computação Científica (LNCC), Brasil

{micaella, osthoff, karyann}@lncc.br

Resumo. *Análises filogenéticas apoiam estudos sobre a vida evolutiva dos organismos. Ferramentas otimizadas como RAxML, baseadas em algoritmos de máxima verossimilhança, geram alto custo computacional pelos inúmeros cálculos para processar grande quantidades de dados. O uso eficiente dessas ferramentas em ambientes paralelos é requerido. O presente trabalho visa explorar o desempenho do RAxML em supercomputadores explorando características de configuração do ambiente, programa e dados.*

Abstract. *Phylogenetic analyzes support studies about the evolutionary life of organisms. Optimized tools such as RAxML, based on maximum likelihood algorithms, generate high computational costs to process large amounts of data. Efficient use of these tools in parallel environments is required. The present work aims to explore the performance of RAxML in supercomputers exploring characteristics of environment, program and data.*

1. Introdução

A filogenia faz uso de algoritmos computacionais para representar a história da vida evolutiva de organismos. O algoritmo de máxima verossimilhança (MV) implementa modelos probabilísticos complexos e eficazes, mas que geram alto custo computacional [Stamatakis 2014]. Dentre os programas de MV mais usados estão RAxML/ExaML, PhyML e IQ-TREE.

RAxML [Stamatakis 2014] Sequencial destina-se a conjuntos de dados pequenos a médios (filogenia) e as versões paralelas PThread, MPI e Híbrido são usadas em maior escala (filogenômica). PThread emprega paralelismo interno em arquiteturas de memória compartilhada; MPI paralelismo externo em arquitetura de memória distribuída e Híbrido explora métodos [Abramson *et al*. 2011; Pfeiffer e Stamatakis 2010].

O objetivo deste trabalho visa avaliar o desempenho do RAxML[1] em ambientes de supercomputador através de uma análise comparativa sobre o impacto produzido pela variabilidade nas configurações de ambiente, programas e características dos dados genômicos. O artigo está organizado em 5 seções, além dessa introdução. A Seção 2 discute os trabalhos relacionados. A Seção 3 apresenta o experimento e a Seção 4 apresenta os resultados. Finalmente, a Seção 5 conclui este artigo.

2. Trabalhos Relacionados

[Pfeiffer e Stamatakis 2010] apresentam uma análise de desempenho das versões paralelas implementadas no RAxML, sustentando a versão Híbrida como a mais eficiente. [Zhou *et al*. 2018] apresentam uma análise comparativa entre os programas

[1] https://github.com/stamatak/standard-RAxML

PhyML, IQ-TREE e RAxML/ExaML, concluindo que RAxML, além de apresentar maior escalabilidade, gera topologias de árvores com melhor qualidade.

O presente artigo explorar o desempenho do RAxML no supercomputador Santos Dumont[2] (SDumont), explorando configurações de ambiente, versões do RAxML, valores de *bootstrap* [Felsenstein 1985] e características dos dados (tamanho).

3. Configuração do Experimento

A Figura 1 apresenta a metodologia dividida em três atividades: (i) comparação das versões do RAxML; (ii) execução do RAxML Híbrido variando o tamanho de arquivo e (iii) execução do RAxML híbrido variando o *bootstrap*.

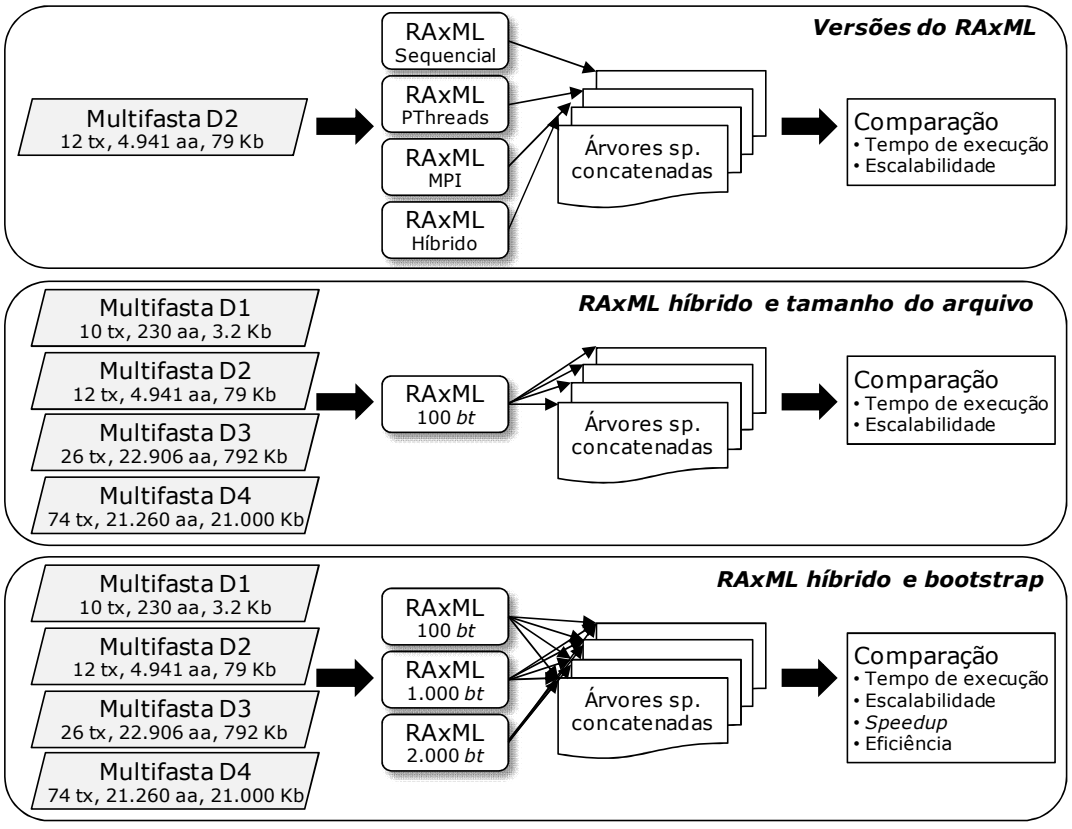


Figura 1. Metodologia do Experimento: Avaliação do RAxML no SDumont

As entradas são quatro arquivos PHYLIP de superalinhamentos de genomas de protozoários [Dávila e Ocaña 2011] com diferente número de táxons, caracteres e tamanho. As versões Sequencial, Pthreads, MPI e Híbrido do RAxML 8.2.12 foram executadas com *bootstraps* 100, 1.000, e 2.000 e os resultados analisados pelo desempenho (tempo, escalabilidade, *speedup* e eficiência).

Os *clusters* Altix[3] e SDumont são gerenciados pelo SINAPAD/LNCC [4]. Altix é composto por 30 nós, cada um possui 2 processadores *quad core*, com 8 núcleos por nó, totalizando 240 núcleos de CPU. SDumont possui uma arquitetura de processamento

[2] http://sdumont.lncc.br/
[3] http://www.lncc.br/altix-xe/
[4] Sistema Nacional de Processamento de Alto Desempenho, https://www.lncc.br/sinapad/

paralelo de configuração híbrida de 1,1 Petaflops com 18.144 núcleos de CPU, 756 nós computacionais (24 núcleos por nó) interconectados pela rede Infiniband FDR. Os gerenciadores de filas utilizados são o SGE (Altix) e Slurm (SDumont).

4. Análise do Desempenho do Experimento

4.1. Versões do RAxML. TTE e Escalabilidade

A versões do RAxML Sequencial, PThreads, MPI e Híbrido foram executadas no Altix e SDumont, usando o arquivo D2, modelo evolutivo JTT (GAMMA) e *bootstrap* 100.

A Figura 2(A) apresenta o Tempo Total de Execução (TTE) em minutos: RAxML MPI apresentou melhor desempenho no Altix com 98,3% de melhoria (285,5 min. em 1 *core* e 4,5 min. em 120 *cores*) e RAxML Híbrido no SDumont com 98,8% de melhoria (161,2 min. em 1 *core* e 1,8 min. em 120 *cores*).

A Figura 2(B) apresenta a escalabilidade em minutos dessas melhores versões até 120 *cores*. O RAxML Híbrido no SDumont apresentou melhor desempenho para todos os *cores*. Até 24 *cores* o TTE teve uma caída drástica e até 120 *cores* a diminuição é menos gradativa, algumas das causas podem ser alocação de memória, afinidade de processo ou escrita em disco.

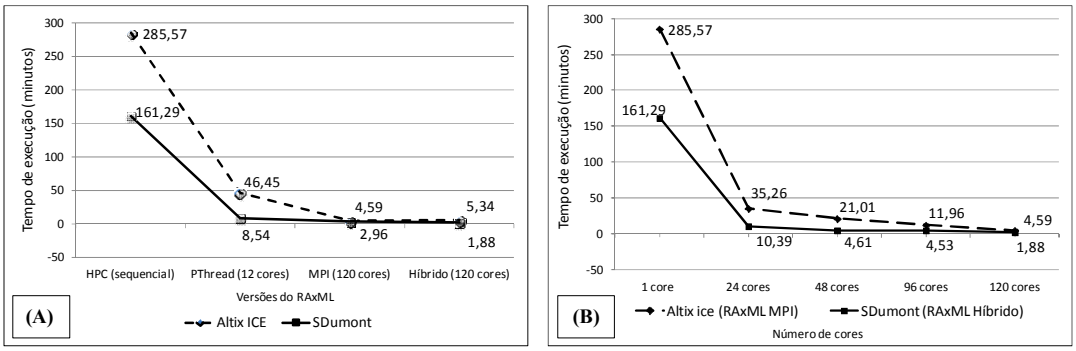


Figura 2. (A) TTE do RAxML no Altix e SDumont. (B) Escalabilidade do melhor desempenho do RAxML MPI no Altix e Híbrido no SDumont

4.2. RAxML híbrido e tamanho do arquivo. TTE e Escalabilidade

A Figura 3 apresenta o desempenho em minutos do RAxML Híbrido no SDumont com variação no tamanho de arquivos e *bootstrap* fixo em 100. Foi observada uma relação entre o incremento de tamanho dos arquivos e a melhoria de desempenho de aprox. 90% entre 1 até 10 nós. D1: 3.2 Kb e 81.3%, D2 79 Kb e 88.6%, D3 792 Kb e 88,4% e D4 21.000 Kb e 89%. Esses valores indicam que a paralelização se beneficia de arquivos maiores e que possuem mais táxons e/ou maior comprimento de sequências.

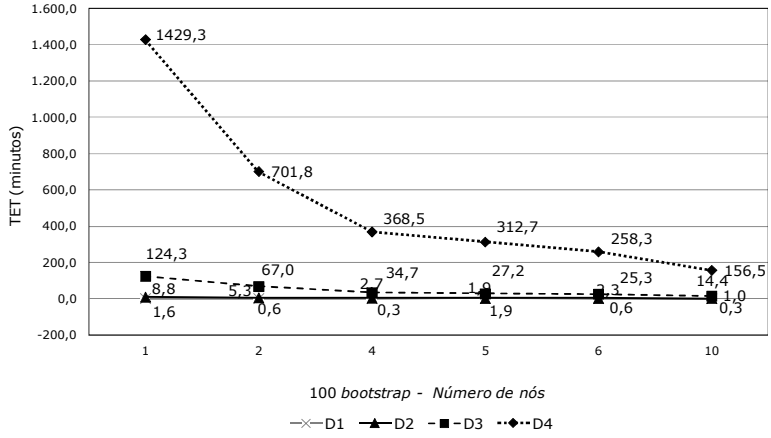


Figura 3. Desempenho do RAxML Híbrido no SDumont variando o tamanho dos arquivos

4.3. RAxML Híbrido e *bootstrap*. TTE, Escalabilidade, *Speedup* e Eficiência

A Figura 4 apresenta o desempenho em minutos do RAxML Híbrido no SDumont executado com os arquivos D1, D2, D3 e D4 e com *bootstraps* 100, 1.000 e 2.000. Em todos os casos, a diminuição do TTE é acentuada até 4 nós e a diminuição entre 5 e 10 nós foi em escala menor. Os resultados de eficiência (Figura 6) corroboram esses resultados. Por exemplo na Figura 6(D4), o arquivo maior D4 (21.000 Kb) apresenta os melhores resultados para todos os *bootstraps*. Já o arquivo menor D1 (3.2 Kb) é mais eficiente com 2 nós e todos os demais arquivos tiveram eficiência melhor com 4 nós. Pode se concluir que RAxML escala melhor com arquivos de tamanho e valor de *bootstrap* maiores. Extrapolando esses resultados a um uso real no SDumont,, este alocaria arquivos e configurações desses tipos diretamente para até 4 nós.

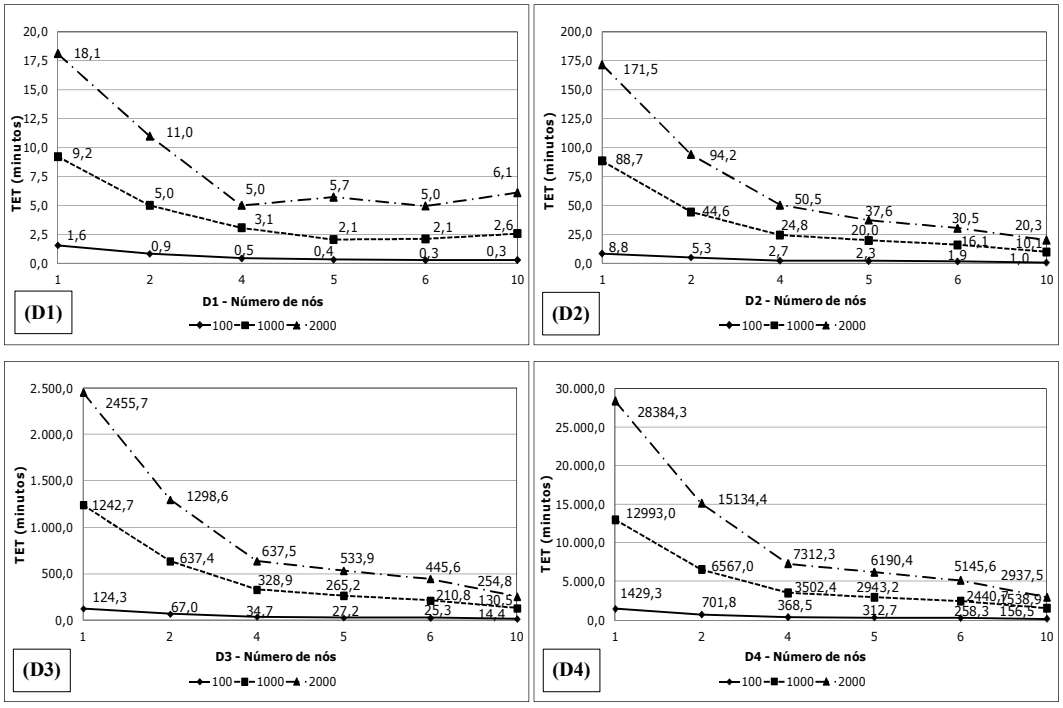


Figura 4. Desempenho do RAxML Híbrido no SDumont variando o valor de *bootstrap* para D1, D2, D3 e D4

A Figura 5 apresenta o *speedup* quase linear até 5 nós para D2, D3 e D4 (o maior dos arquivos), depois disso o *speedup* começa a degradar. O tamanho menor afeta o *speedup* que não permanece constante em comparação aos demais arquivos, em um experimento em que cada execução foi repetida três vezes. O *speedup* é o tempo de execução sequencial dividido pelo tempo de execução paralelo por nós.

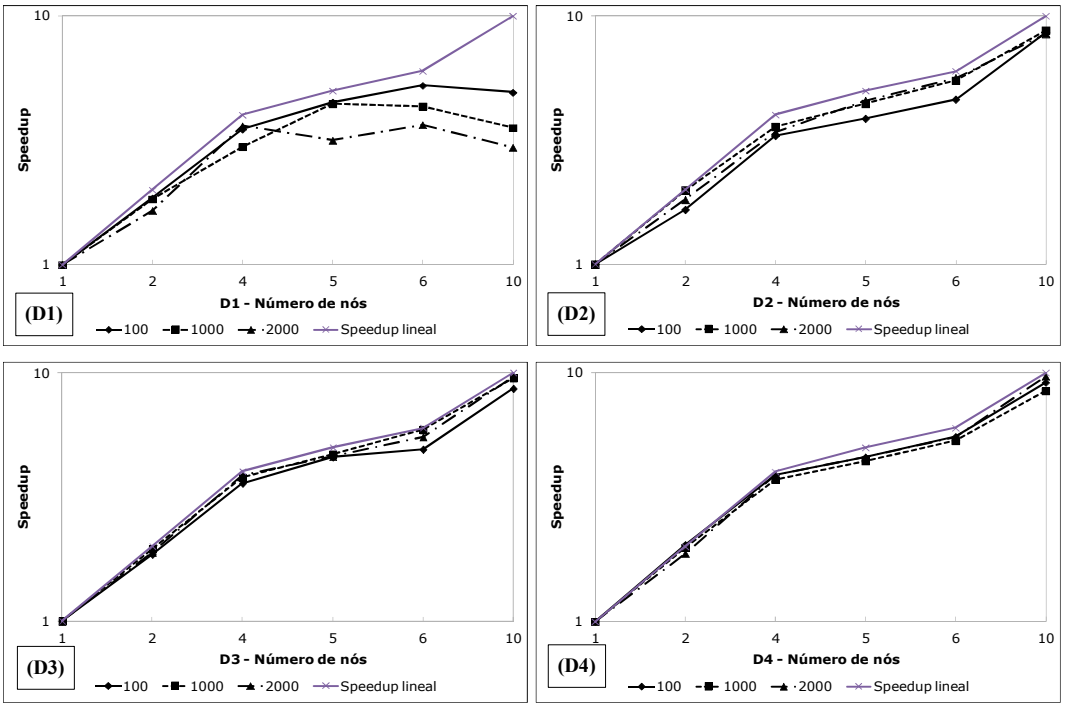
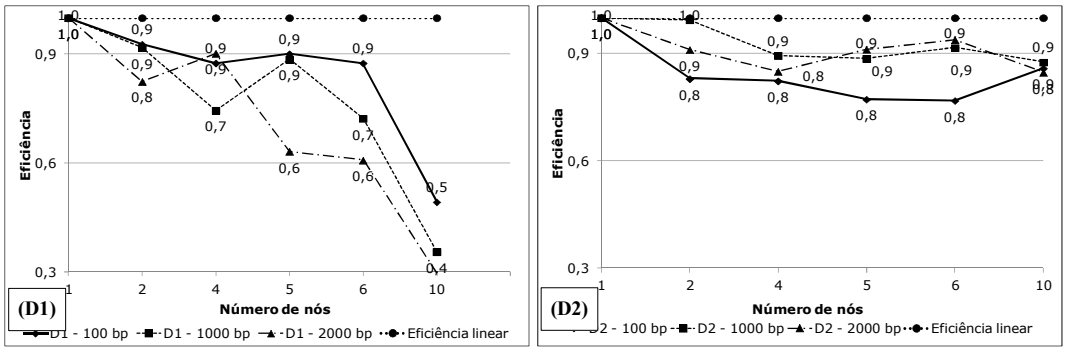


Figura 5. *Speedup* do RAxML Híbrido no SDumont variando os *bootstraps* para D1, D2, D3 e D4

A eficiência ideal (linear) é obtida ao dividir o *speedup* pelo número de nós usado, a curva de eficiência ideal mostra uma forma linear (Figura 6). A eficiência para os arquivos D2, D3 e D4 oscilou entre 0.8 e 1.0 exceto para D1 com valores menores. O arquivo maior D4 é o mais escalável e se beneficia melhor do ambiente e configurações paralelas. Segundo os testes *benchmark* [Pfeiffer e Stamatakis 2010] se espera que na medida em que o *bootstrap* aumente com arquivos de supermatrizes, as execuções se tornem mais escaláveis. Pode se concluir através da Figura 6, que D1 torna-se mais eficiente com 2 nós e todos os demais arquivos com 4 nós.



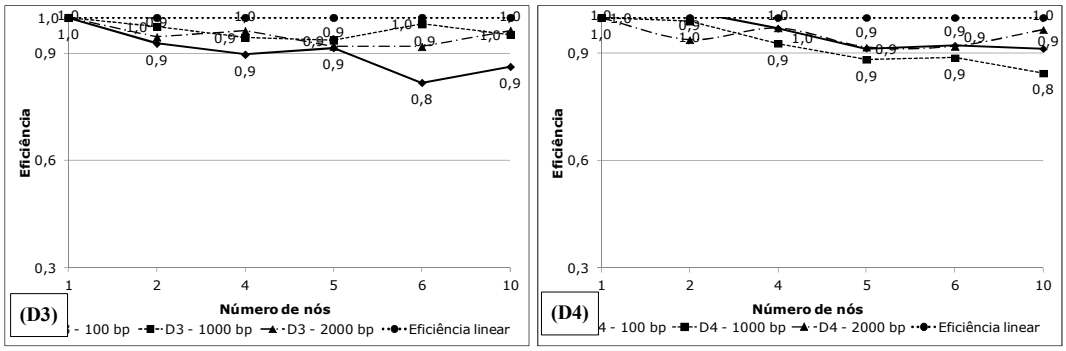


Figura 6. Eficiência do RAxML Híbrido no SDumont variando os *bootstraps* para D1, D2, D3 e D4

5. Conclusão

O supercomputador SDumont aloca portais *Web*[5] e *software* de diversos projetos científicos[6] no Brasil. Por tanto, é de interesse de administradores e usuários ter ciência sobre características de alocação e desempenho para garantir o seu uso efetivo.

Nesse trabalho avaliamos o desempenho do RAxML no SDumont. A eficiência demonstra que características como tamanho de dados e *bootstrap* influem no desempenho. O arquivo maior (21.000 Kb) com o maior valor de *bootstrap* (2.000) foi o mais escalável. Já o arquivo menor (3.2 Kb) teve melhor eficiência com 2 nós e todos os arquivos maiores com 4 nós. Fatores que impactam no desempenho são alocação de memória, afinidade de processo, escrita em disco e uso de gerenciadores de filas e compiladores. Perfiladores para a identificação de gargalhos estão sendo testados.

Agradecimentos. Este trabalho foi financiado pelos projetos Universal MCTI/CNPq nº 01/2016, processo 429328/2016-8 e JCNE FAPERJ nº 03/2017, processo 232985. As execuções foram realizadas no SDumont no SINAPAD/LNCC.

Referências Bibliográficas

Abramson, D., Bethwaite, B., Enticott, C., Garic, S. e Peachey, T. (2011). Parameter Exploration in Science and Engineering Using Many-Task Computing. *IEEE Trans. Parallel Distrib. Syst.*, v. 22, n. 6, p. 960–973.

Dávila, A. e Ocaña, K. (2011). Phylogenomics-Based Reconstruction of Protozoan Species Tree. *Evolutionary Bioinformatics*, p. 107.

Felsenstein, J. (1985). Confidence Limits on Phylogenies: An Approach Using the Bootstrap. *Evolution*, v. 39, n. 4, p. 783–791.

Pfeiffer, W. e Stamatakis, A. (2010). Hybrid MPI/Pthreads parallelization of the RAxML phylogenetics code. IEEE. http://ieeexplore.ieee.org/document/5470900/.

Stamatakis, A. (2014). RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics*, v. 30, n. 9, p. 1312–1313.

Zhou, X., Shen, X.-X., Hittinger, C. T. e Rokas, A. (2018). Evaluating Fast Maximum Likelihood-Based Phylogenetic Programs Using Empirical Phylogenomic Data Sets. *Molecular Biology and Evolution*, v. 35, n. 2, p. 486–503.

[5] BioinfoPortal, http://bioinfo.lncc.br/
[6] http://sdumont.lncc.br/projects_view.php?pg=projects&status=ongoing