BRAZILIAN SYMPOSIUM ON BIOINFORMATICS

BSB & EBB 2011

VI BRAZILIAN SYMPOSIUM ON BIOINFORMATICS
IV BRAZILIAN SCHOOL ON BIOINFORMATICS

Digital Proceedings

Osmar Norberto de Souza • Guilherme P. Telles • Mathew J. Palakal
Editors

Maria Emília Walter • Marcelo M. Brígido
Conference Chairs

Promotion
Brazilian Computer Society – SBC

Brazilian Symposium on Bioinformatics (2011 : Brasília-DF)
BSB 2011 digital proceedings, August, 10th to 12th, Brasília, Distrito Federal, Brasil / conference chairs Maria Emília Walter and Marcelo M. Brígido ; volume editors Osmar Norberto de Souza, Guilherme P. Telles and Mathew J. Palakal -- [Brasília] : Brazilian Computer Society, [2011].

1 CD-ROM.

ISSN 2178-5120

1. Bioinformática – Congresso. I. Walter, Maria Emília. II. Brígido, Marcelo M. III. Norberto de Souza, Osmar. IV Telles, Guilherme P. V Palakal, Mathew J. VI. Título.

Preface

This is the second volume of full papers and extended abstracts selected by peer review for presentation at BSB 2011 and at EBB IV. The first volume was published as a special issue of the series Lecture Notes in Bioinformatics - Springer.

The 6th Brazilian Symposium on Bioinformatics, BSB 2011, was held in Brasília-DF, Brazil, from August 10 to 12, 2011. BSB was co-located with EBB IV - the 4th Brazilian School on Bioinformatics. BSB 2011 is indeed the 9th meeting of the series, which started in 2002 and was called WOB (Workshop on Bioinformatics). In 2005, the conference name was changed from WOB to its current name BSB.

The Program Committee Chairs are thankful to the Program Committee and to the additional reviewers for their careful work and to the authors that submitted their papers.

The members of BSB organization committees thank the keynote speakers, Professors Mathew J. Palakal, João Setubal, Peter F. Stadler and Sergio Verjovski-Almeida, for their participation in the event. They also thank the SBC CEBioComp steering committee members for their support to the event and to supporting institutions Brazilian Computer Society (SBC), CNPq, CAPES and FAPDF.

Our sincerest thanks go to all who have supported the Brazilian Symposium on Bioinformatics along all these years. In 2011, we are in debt to the indispensable close help and assistance of Prof. Maria Emília Walter. Mia, it did make a difference. Our gratitude also goes to Professors Marcelo de Macedo Brígido and to the team at UnB: Tainá Raiol, Daniel Saad Nunes, Halian Vilela, Felipe Lessa, Maria Beatriz W. Costa, Paulo Alvarez, and Tulio Conrado da Silva.

August 2011

Osmar Norberto de Souza
Guilherme P. Telles
Mathew J. Palakal

Organization

BSB 2011 was promoted by the Brazilian Computer Society (SBC) and was organized by the Institute of Biology and by the Institute of Exact Sciences of the University of Brasília (UnB), Brasília-DF, Brazil.

Conference Chairs

Marcelo de Macedo Brígido	Laboratory of Molecular Biology, Institute of Biology, UnB, Brazil
Maria Emília M. T. Walter	Departament of Computer Science, Institute of Exact Sciences, UnB, Brazil

Program Chairs

Osmar Norberto de Souza	Faculty of Informatics–PUCRS, Brazil
Guilherme P. Telles	Institute of Computing–Unicamp, Brazil
Mathew J. Palakal	Indiana University Purdue University Indianapolis, USA

Steering Comittee

Marcelo de Macedo Brígido	University of Brasília, Brazil
André C.P.L.F. de Carvalho	University of São Paulo–São Carlos, Brazil
Carlos E. Ferreira	University of São Paulo, Brazil
Katia Guimarães	Federal University of Pernambuco, Brazil
Sergio Lifschitz	Pontifical Catholic University of Rio de Janeiro, Brazil
Osmar Norberto de Souza	Pontifical Catholic University of Rio Grande do Sul, Brazil
Maria Emília M. T. Walter	University of Brasília, Brazil

Program Committee

Nalvo F. Almeida	Federal University of Mato Grosso do Sul, Brazil
Ana Lúcia C. Bazzan	Federal University of Rio Grande do Sul, Brazil
Marcelo Brigido	University of Brasília, Brazil
Mario Caccamo	TGAC-BBSRC, UK
André C.P.L.F. de Carvalho	University of São Paulo–São Carlos, Brazil
Ivan G. Costa	Federal University of Pernambuco, Brazil
Mark Craven	University of Wisconsin, USA
Alberto Dávila	Fiocruz, Brazil
Zanoni Dias	University of Campinas, Brazil
Alan Durham	University of São Paulo, Brazil
Fazel Famili	Institute for Information Technology, Canada
Katia S. Guimarães	Federal University of Pernambuco, Brazil
Ronaldo Hashimoto	University of São Paulo, Brazil
Paul Horton	AIST, Japan
Maricel Kann	UMBC, USA
Paula Kuser-Falcão	Embrapa, Brazil
Alair Pereira do Lago	University of São Paulo, Brazil
Ney Lemke	IBB-Unesp, Brazil
Sergio Lifschitz	Pontifical Catholic University of Rio de Janeiro, Brazil
Ion Mandoiu	University of Connecticut, USA
Natalia F. Martins	Embrapa, Brazil
Wellington Martins	Federal University of Goiás, Brazil
Anna Panchenko	NIH, USA
Fábio Passeti	INCA, Brazil
Duncan Ruiz	Pontifical Catholic University of Rio Grande do Sul, Brazil
Alexander Schliep	Rutgers University, USA
João Setubal	Virginia Bioinformatics Institute, USA
Marcilio M.C.P. de Souto	Federal University of Pernambuco, Brazil
Rainer Spang	University of Regensburg, Germany
Peter F. Stadler	University of Leipzig, Germany
Jerzy Tiuryn	University of Warsaw, Poland
Maria Emilia M. T. Walter	University of Brasília, Brazil
Alex Zelikovsky	Georgia State University, USA

Additional Reviewers

Said Adi	Federal University of Mato Grosso do Sul, Brazil
Aleteia P.F. de Araujo	University of Brasília, Brazil
Francisco Eloi Araújo	Federal University of Mato Grosso do Sul, Brazil
Katti Faceli	Federal University of São Carlos, Brazil
Mileidy Gonzalez	NIH, USA
Carlos Higa	University of São Paulo, Brazil
Maristela T. de Holanda	University of Brasília, Brazil
Renato P. Ishii	Federal University of Mato Grosso do Sul, Brazil
David Martins-Jr.	Federal University of ABC, Brazil
Edson Matsubara	Federal University of Mato Grosso do Sul, Brazil
Mariana Mendoza	Federal University of Rio Grande do Sul, Brazil
Roberto Riga	Embrapa, Brazil
John Spouge	NIH, USA
David Swarbreck	The Genome Analysis Center, UK
Manoj Tyagi	NCBI/NIH, USA
Michel Yamagishi	Embrapa, Brazil

Sponsoring Institutions

Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq)
Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES)
Fundação de Apoio à Pesquisa do DF (FAPDF)
Brazilian Computer Society (SBC)
University of Brasília (UnB)

Table of Contents

Full Papers

Extended Abstracts

Comparing Correlation Coefficients as Dissimilarity Measures for Cancer Classification in Gene Expression Data

Pablo A. Jaskowiak and Ricardo J. G. B. Campello

Department of Computer Sciences
University of São Paulo at São Carlos
São Carlos, Brazil
{pablo,campello}@icmc.usp.br

Abstract. An important analysis performed in gene expression data is sample classification, e.g., the classification of different types or subtypes of cancer. Different classifiers have been employed for this challenging task, among which the k-Nearest Neighbors (kNN) classifier stands out for being at the same time very simple and highly flexible in terms of discriminatory power. Although the choice of a dissimilarity measure is essential to kNN, little effort has been undertaken to evaluate how this choice affects its performance in cancer classification. To this extent, we compare seven correlation coefficients for cancer classification using kNN. Our comparison suggests that a recently introduced correlation may perform better than commonly used measures. We also show that correlation coefficients rarely considered can provide competitive results when compared to widely used dissimilarity measures.

Keywords: Correlation Coefficients, kNN, Cancer Classification

1 Introduction

Microarray technology enables expression level measurement for thousands of genes in a parallel fashion. The genomic picture obtained with the technology can help to discriminate among different classes of samples, which are usually associated with distinct types of cancer. Sample classification is not only essential to successful cancer diagnosis, but also to help choosing the best treatment for different patients, minimizing collateral effects of the treatment [19]. A wide range of classifiers have been applied to this task [2, 8, 12]. Among these, k-Nearest Neighbors (kNN) [5] is of main interest for our work, once it has shown quite good results in cancer classification problems [2, 8, 15]. In addition, kNN is fairly simple and straightforward to implement, which makes it very appealing.

In brief, kNN has two parameters that must be set or adjusted, which can, in turn, directly affect the classification outcomes. The first one is the number of neighbors (k), while the second one is the dissimilarity measure that induces the neighboring relationships. Parameter k and the effect of its choice in cancer

classification have been explored in some works [12, 13]. In what concerns the dissimilarity measure, on the other hand, Euclidean distance and Pearson correlation have been widely adopted as rules of thumb [2, 8, 12]. Adding to the fact that these measures are not the only alternatives to quantify dissimilarities in gene expression data, kNN can be quite sensitive to the dissimilarity choice [1]. In spite of that, little effort has been made to try establishing guidelines to the choice of a proper dissimilarity measure in this particular context. Parry et al. [13] evaluated three different dissimilarities with kNN. The evaluation, however, was based on a few datasets and did not consider any correlation coefficient, regardless their common use in gene expression data.

In the present work, to the best of our knowledge, we present the first comparison of correlation coefficients as dissimilarity measures for cancer classification using kNN. We evaluate seven correlation coefficients on 35 publicly available datasets, from both single and double channel microarrays [16]. Our investigation is motivated by sensitivity differences exhibited by the measures, such as, robustness to outliers. In addition, the correlations considered in our analysis take into account different characteristics of the data, as will be discussed later.

The remainder of this paper is organized as follows. In Section 2 the correlation coefficients considered for comparison are reviewed. In Section 3 the experimental setup is presented, whereas in Section 4 the results obtained are discussed. Finally, in Section 5 the main conclusions of our work are addressed.

2 Correlation Coefficients

When considering gene expression data and comparing any two objects (samples in our case), it turns out that these objects should be regarded as similar if they exhibit similarity in shape, rather than in absolute differences from their values [19]. Correlation coefficients have been widely used in this context, since they capture such a kind of similarity. In this sense, any two samples can be seen as sequences of real values $\mathbf{a}$ and $\mathbf{b}$, in the form $\mathbf{a} = (a_1, \ldots, a_n)$ and $\mathbf{b} = (b_1, \ldots, b_n)$, for which a correlation coefficient can be directly applied. Such coefficients produce values between -1 and 1. High absolute values indicate a stronger relationship between sequences, while values close to 0 indicate non-correlated sequences. Bearing the above considerations in mind, in the following we review the seven correlation coefficients considered in our comparison.

2.1 Pearson - ρ

The Pearson correlation [14] allows the identification of linear correlations between two sequences of numbers. It is described in (1), where $\bar{a}$ and $\bar{b}$ stand for the means of the sequences in hand. Although widely used in gene expression data, Pearson is sensitive to the presence of outliers and may not be robust when both sequences do not come from an approximately normal distribution [19, 10].

$$\rho(\mathbf{a}, \mathbf{b}) = \frac{\sum_{i=1}^{n}(a_i - \bar{a})(b_i - \bar{b})}{\sqrt{\sum_{i=1}^{n}(a_i - \bar{a})^2}\sqrt{\sum_{i=1}^{n}(b_i - \bar{b})^2}} \quad (1)$$

2.2 Jackknife - ϱ

Jackknife correlation [10] is defined in order to minimize the effect that single outliers may have in the final correlation value. This is achieved by removing one value at a time from both sequences. Jackknife is defined in (2), where $\rho^i(\mathbf{a}, \mathbf{b})$ stands for the Pearson correlation between $\mathbf{a}$ and $\mathbf{b}$ with their i^{th} values removed.

$$\varrho(\mathbf{a}, \mathbf{b}) = min\{\rho^1(\mathbf{a}, \mathbf{b}), \ldots, \rho^n(\mathbf{a}, \mathbf{b}), \rho(\mathbf{a}, \mathbf{b})\} \tag{2}$$

2.3 Goodman-Kruskal - γ

The Goodman-Kruskal correlation [9] takes into account only the ranks of sequences $\mathbf{a}$ and $\mathbf{b}$, and is defined according to the number of concordant (S_+), discordant (S_-), and neutral pairs in the sequences. A pair is said to be concordant if the same relative order applies in the two sequences ($a_i < a_j$ and $b_i < b_j$ or $a_i > a_j$ and $b_i > b_j$). Similarly, discordant pairs are those in which the inverse relative order applies ($a_i < a_j$ and $b_i > b_j$ or $a_i > a_j$ and $b_i < b_j$). All other pairs are deemed as neutrals. Goodman-Kruskal is defined by Equation (3).

$$\gamma(\mathbf{a}, \mathbf{b}) = \frac{S_+ - S_-}{S_+ + S_-} \tag{3}$$

2.4 Kendall - τ

The Kendall correlation [11] is based on the same concepts previously defined for Goodman-Kruskal. The difference between these two measures is due to the fact that Kendall, defined in (4), takes into account all the $n(n-1)/2$ pairs in its normalization term, reaching its extrema only in the absence of neutrals.

$$\tau(\mathbf{a}, \mathbf{b}) = \frac{S_+ - S_-}{n(n-1)/2} \tag{4}$$

2.5 Spearman - $\hat{\rho}$

The Spearman correlation [17] can be seen as a particular case of the Pearson correlation, provided that the values of both $\mathbf{a}$ and $\mathbf{b}$ are replaced with their ranks in the respective sequences. By doing such a replacement, the Spearman correlation can also be defined by (1). As only the ranks of the sequences are considered, Spearman is more robust to the presence of outliers than Pearson [19].

2.6 Rank-Magnitude - r

The Rank-Magnitude correlation [4] was introduced as an asymmetric measure, for cases in which one of the sequences is composed by ranks and the other by real values. The correlation is defined by (5), with $R(a_i)$ denoting the rank of the i^{th} element of sequence $\mathbf{a}$. In (5), $r^{min} = \sum_{i=1}^{n}(n+1-i)\bar{b}_i$ and $r^{max} = \sum_{i=1}^{n} i\bar{b}_i$.

Value $\overline{b}_i$ is the i^{th} element of the sequence, which is obtained by rearranging sequence $\mathbf{b}$ so that it gets sorted in ascending order.

$$\hat{r}(\mathbf{a}, \mathbf{b}) = \frac{2\sum_{i=1}^{n} R(a_i)b_i - r^{max} - r^{min}}{r^{max} - r^{min}} \tag{5}$$

As previously mentioned, Rank-Magnitude is asymmetric. To be used in cases in which both sequences are constituted of real values, the measure must be symmetrized. Any mention to the Rank-Magnitude correlation in the remainder of this paper refers to its symmetric version, given by $r(\mathbf{a}, \mathbf{b}) = (\hat{r}(\mathbf{a}, \mathbf{b}) + \hat{r}(\mathbf{b}, \mathbf{a}))/2$.

2.7 Weighted Goodman-Kruskal - $\hat{\gamma}$

The Weighted Goodman-Kruskal correlation [4] takes into account ranks and magnitudes of both sequences by considering that concordance and discordance are both a matter of degree. The complete Weighted Goodman-Kruskal formulation is presented in [4] and is omitted here due to space restrictions.

3 Experimental Setup

We compared different variants of the kNN classifier, each of which employing one of the correlation coefficients described in Section 2. All correlation coefficients were adapted as dissimilarities in the form: $Dissimilarity(\mathbf{a}, \mathbf{b}) = 1 - correlation(\mathbf{a}, \mathbf{b})$. For completeness, we also included the Euclidean distance (represented by letter 'e') in our comparison. Regarding kNN parameter k, we considered four values during our evaluation: 1 (1NN), 3 (3NN), 5 (5NN) and 7 (7NN). These values were chosen based on the work of Dudoit et al. [8], in which the authors show that for small sample cancer classification values of k smaller than 7 are usually preferred. Each kNN variant was evaluated by its generalization capability (error rates), which was estimated using Leave One Out Cross Validation (LOOCV). It is worth noticing that the choice of error estimators in the case of small sample sizes is still under debate [3, 7].

To evaluate each one of the 32 kNN variants considered we used a set of publicly available benchmark datasets proposed in [16]. Briefly, this benchmark set encompass 35 microarray datasets from cancer gene expression experiments and comprehend the two flavors in which the technology is generally available: single channel (21 datasets) and double channel (14 datasets) [19, 18]. Hereafter we refer to single channel microarrays as Affymetrix and double channel microarrays as cDNA, since the data was collected using either of these technologies [16]. Detailed information about these datasets can be obtained in [16].

Finally, to provide reassurance about the validity of our results, we used the Friedman and Nemenyi statistical tests (with a 95% confidence level), which are more appropriate when comparing multiple classifiers on multiple datasets [6].

4 Results

Once we are dealing with two different microarray technologies, i.e., cDNA and Affymetrix, we chose to analyze the results obtained for each technology independently. In Fig. 1 and 2 we present classification error boxplots for each one of the 32 kNN variants in cDNA and Affymetrix datasets respectively.

Considering cDNA datasets (Fig. 1), the largest variabilities were found with Euclidean distance (e), which showed the worse results among all compared measures. It is interesting to note that the use of Jackknife (ϱ) provided some decrease in variability when compared to Pearson (ρ) (except for 5NN). Regarding the rank-based correlation coefficients, Goodman-Kruskal (γ), Kendall (τ), and Spearman ($\hat{\rho}$) showed comparable results, regardless of the value of k used. These results are quite interesting and show that rank-based measures rarely used in gene expression data, such as, Goodman-Kruskal (γ) and Kendall (τ), can provide competitive results when compared to the more commonly used Spearman ($\hat{\rho}$). Considering Rank-Magnitude (r), comparable results were observed in comparison to rank-based correlations.

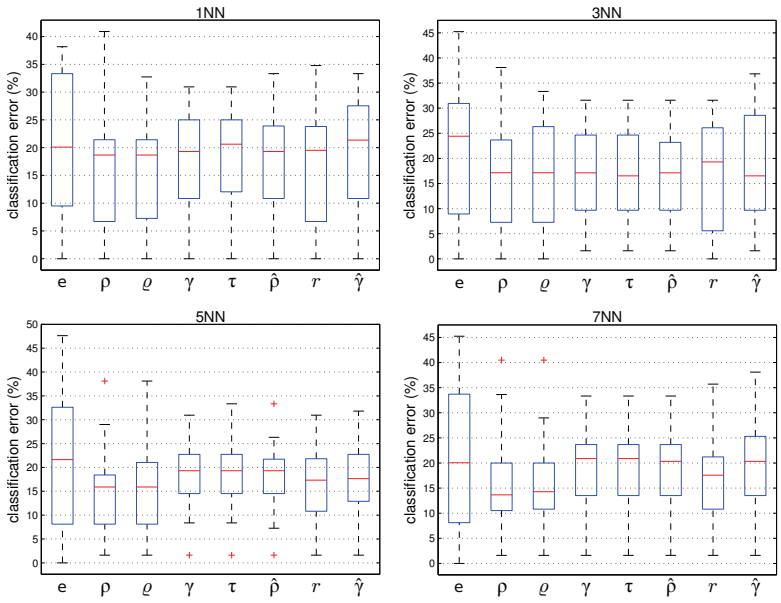

Fig. 1. cDNA datasets - boxplots showing classification errors obtained when comparing kNN with different correlation coefficients (Euclidean distance also included).

The results obtained for Affymetrix datasets are shown in Fig. 2. All measures produced outliers but the Euclidean distance (e). These outliers can, to some extent, be justified by the large number of classes present in some datasets. In this sense, the observed outliers approximate the errors that would be found

with a majority voting classifier, i.e., a classifier that assigns each unlabeled sample to the majority class observed in the training data. Regardless of the value of k, Weighted Goodman-Kruskal ($\hat{\gamma}$) displayed the largest variability among the compared dissimilarity measures. Considering the rank-based correlations, Goodman-Kruskal (γ), Kendall (τ), and Spearman ($\hat{\rho}$) showed comparable results. It is interesting to note that the commonly used Euclidean distance (e), Pearson (ρ) and Jackknife (ϱ) performed slightly worse than rank-based correlations. Rank-Magnitude correlation (r) displayed the lowest variabilities among the compared measures considering 5NN and 7NN. This correlation also performed well for 1NN and 3NN, showing better results when compared to the commonly employed Pearson (ρ) and Euclidean distance (e).

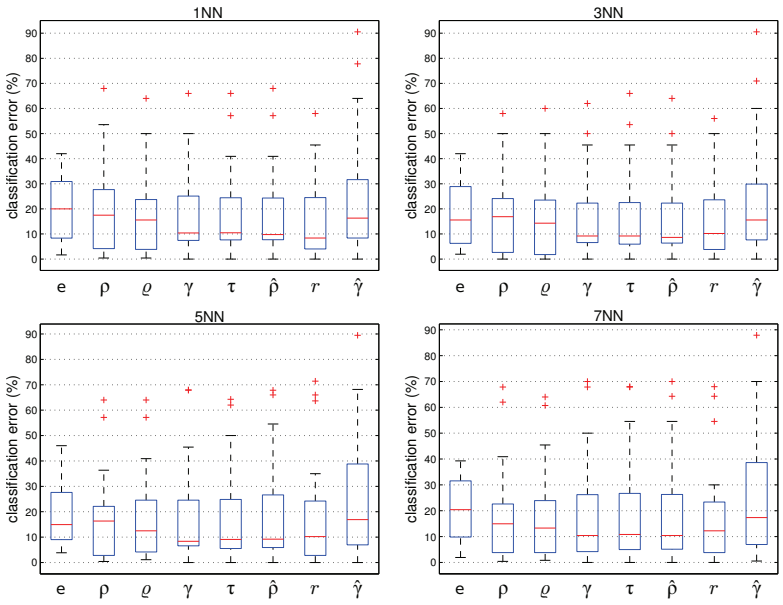

Fig. 2. Affymetrix datasets - boxplots showing classification errors obtained when comparing kNN with different correlation coefficients (Euclidean distance also included).

Fig. 1 and Fig. 2 provide an overview but hide some interesting results. In order to elucidate such cases, we also present results for each individual dataset considering only 1NN classifier[1], as shown in Table 1. In *west-2001* and *ramaswamy-2001* datasets, for which the commonly used Euclidean distance (e) and Pearson (ρ) led to larger error rates, the use of rank-based measures and Rank-Magnitude (r) can decrease their errors in almost 25%. Considering *bitner-2000*, the use of Pearson (ρ) and Jackknife (ϱ) led to a difference of almost 20% in error when compared to the Euclidean distance (e). It is important to note

[1] Results concerning the other values of k are omitted due to space restrictions.

that, in some datasets, e.g., *nutt-2003-v1, v2* and *v3*, the Euclidean distance (e) can provide better results when compared to the other measures.

Table 1. 1NN classification errors (%) for all evaluated dissimilarity measures. Shades of gray indicate relative performance of the dissimilarities (columns) for each dataset (rows). The lighter the cell, the lower the error for the respective dataset.

		e	ρ	ϱ	γ	τ	$\hat{\rho}$	r	$\hat{\gamma}$
cDNA	alizadeh-2000-v1	19.1	21.4	21.4	23.8	23.8	23.8	23.8	23.8
	alizadeh-2000-v2	0.0	1.6	1.6	1.6	1.6	1.6	1.6	1.6
	alizadeh-2000-v3	17.7	19.4	19.4	14.5	14.5	16.1	19.4	14.5
	bittner-2000	34.2	13.2	13.2	18.4	21.1	18.4	15.8	29.0
	bredel-2005	12.0	18.0	18.0	18.0	18.0	18.0	16.0	20.0
	chen-2002	9.5	6.7	7.3	9.5	9.5	8.9	6.7	9.5
	garber-2001	24.2	19.7	19.7	21.2	21.2	21.2	19.7	22.7
	khan-2001	0.0	0.0	0.0	10.8	12.1	10.8	1.2	10.8
	lapointe-2004-v1	37.7	37.7	31.9	26.1	26.1	27.5	34.8	27.5
	lapointe-2004-v2	38.2	40.9	32.7	26.4	25.5	26.4	26.4	30.0
	liang-2005	2.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	risinger-2003	33.3	28.6	26.2	31.0	31.0	33.3	28.6	33.3
	tomlins-2006	21.2	16.4	16.4	20.2	20.2	20.2	20.2	18.3
	tomlins-2006-v2	23.9	19.6	20.7	25.0	25.0	23.9	21.7	23.9
Affymetrix	armstrong-2002-v1	8.3	2.8	1.4	1.4	1.4	1.4	0.0	6.9
	armstrong-2002-v2	11.1	4.2	5.6	4.2	5.6	5.6	4.2	9.7
	bhattacharjee-2001	14.3	13.3	13.8	7.9	6.4	6.4	8.4	8.9
	chowdary-2006	1.9	2.9	2.9	3.9	4.8	3.9	2.9	3.9
	dyrskjot-2003	20.0	17.5	17.5	20.0	22.5	20.0	20.0	15.0
	golub-1999-v1	6.9	4.2	2.8	8.3	8.3	8.3	2.8	5.6
	golub-1999-v2	8.3	5.6	4.2	8.3	8.3	8.3	4.2	6.9
	gordon-2002	1.7	2.2	2.2	0.0	0.0	0.0	0.0	0.0
	laiho-2007	13.5	24.3	24.3	10.8	10.8	10.8	18.9	10.8
	nutt-2003-v1	42.0	68.0	64.0	66.0	66.0	68.0	58.0	64.0
	nutt-2003-v2	32.1	53.6	46.4	50.0	57.1	57.1	39.3	57.1
	nutt-2003-v3	31.8	45.5	50.0	45.5	40.9	40.9	45.5	40.9
	pomeroy-2002-v1	32.4	38.2	35.3	17.7	17.7	17.7	32.4	17.7
	pomeroy-2002-v2	23.8	28.6	21.4	26.2	23.8	23.8	23.8	28.6
	ramaswamy-2001	34.2	27.4	22.6	24.7	26.3	25.8	18.4	90.5
	shipp-2002-v1	22.1	22.1	15.6	10.4	10.4	9.1	6.5	22.1
	singh-2002	23.5	26.5	23.5	22.6	23.5	23.5	19.6	21.6
	su-2001	15.5	11.5	10.3	10.3	8.1	9.8	5.2	11.5
	west-2001	30.6	16.3	14.3	6.1	8.2	8.2	8.2	16.3
	yeoh-2002-v1	2.0	0.4	0.4	9.3	10.5	8.9	3.6	17.3
	yeoh-2002-v2	25.8	23.8	21.4	40.7	38.3	38.3	26.6	77.8
		e	ρ	ϱ	γ	τ	$\hat{\rho}$	r	$\hat{\gamma}$

Finally, the statistical tests suggest that for Affymetrix datasets, Rank-Magnitude was statistically superior to Pearson, when considering 1NN. Still concerning Affymetrix datasets, both Rank-Magnitude and Jackknife were statistically superior to Euclidean distance, when considering 7NN. Regarding the results for cDNA datasets, no statistically significant differences were found.

5 Conclusions

We presented a comparison of seven correlation coefficients for cancer classification with kNN. Although no correlation performed best in all datasets, some interesting results were found. First of all, we showed that there are competitive alternatives to Euclidean distance and Pearson in both cDNA and Affymetrix datasets. Considering only Affymetrix data, the recently proposed Rank-Magnitude is, in some cases, statistically superior to Euclidean distance and Pearson, common choices in gene expression analysis. Regarding rank-based correlations, our results suggest that Goodman-Kruskal and Kendall are possible alternatives to the more commonly used Spearman. As in individual datasets

large differences were found with different correlations, in real applications an exploratory analysis considering different measures is seemingly the best choice.

As future work, it would be interesting to investigate possible relations between characteristics of the datasets and the results produced by the correlations coefficients in particular cases, e.g., *west-2001*, where greater differences among results from different coefficients were observed.

Acknowledgments. The authors would like to acknowledge the Brazilian research agencies CAPES, CNPq and FAPESP for financial support to this work.

References

1. Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. Mach. Learn. 6(1), 37–66 (1991)
2. Ben-Dor, A., et al.: Tissue classification with gene expression profiles. J. Comput. Biol. 7(3-4), 559–583 (2000)
3. Boulesteix, A.L., Strobl, C., Augustin, T., Daumer, M.: Evaluating microarray-based classifiers: An overview. Cancer Informatics 6, 77–97 (2008)
4. Campello, R.J.G.B., Hruschka, E.R.: On comparing two sequences of numbers and its applications to clustering analysis. Inform. Sciences 179(8), 1025–1039 (2009)
5. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE T. Inform. Theory 13(1), 21 – 27 (Jan 1967)
6. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30 (2006)
7. Dougherty, E.R., Sima, C., Hua, J., Hanczar, B., Braga-Neto, U.M.: Performance of error estimators for classification. Curr. Bioinf. 5, 53–67 (2010)
8. Dudoit, S., et al.: Comparison of discrimination methods for the classification of tumors using gene expression data. J. Amer. Stat. Assoc. 97(457), 77–87 (2002)
9. Goodman, L., Kruskal, W.: Measures of association for cross-classifications. J. Amer. Stat. Assoc. 49, 732764 (1954)
10. Heyer, L.J., Kruglyak, S., Yooseph, S.: Exploring expression data: Identification and analysis of coexpressed genes. Genome Res. 9(11), 1106–1115 (1999)
11. Kendall, M.G.: Rank correlation methods. Griffin, London, 4 edn. (1970)
12. Lu, J., et al.: MicroRNA expression profiles classify human cancers. Nature 435(7043), 834–838 (2005)
13. Parry, R.M., et al.: k-nearest neighbor models for microarray gene expression analysis and clinical outcome prediction. Pharmac. J. 10(4), 292–309 (2010)
14. Pearson, K.: Contributions to the mathematical theory of evolution. iii. regression, heredity, and panmixia. P. Roy. Soc. Lond. A Mat. 59, 69–71 (1895)
15. Singh, D., et al.: Gene expression correlates of clinical prostate cancer behavior. Cancer Cell 1(2), 203 – 209 (2002)
16. de Souto, M., Costa, I., de Araujo, D., Ludermir, T., Schliep, A.: Clustering cancer gene expression data: a comparative study. BMC Bioinformatics 9(1), 497 (2008)
17. Spearman, C.: The proof and measurement of association between two things. Am. J. Psychol. 100(3/4), 441–471 (1904)
18. Tarca, A.L., Romero, R., Draghici, S.: Analysis of microarray experiments of gene expression profiling. Am. J. Obstet. Gynecl. 195(2), 373 – 388 (2006)
19. Zhang, A.: Advanced analysis of gene expression microarray data. World Scientific Publishing Company, 1 edn. (2006)

Paired Segments Alignment and its Application to the Gene Prediction Task

Ronaldo Fiorilo dos Santos, Rodrigo Mitsuo Kishi, and Said Sadique Adi*

Faculdade de Computação (FACOM)
Universidade Federal de Mato Grosso do Sul (UFMS),
CP 549, 79070-900 Campo Grande-MS, Brazil
ronaldo.santos@ufms.br
rodrigokishi@gmail.com
said@facom.ufms.br
http://www.facom.ufms.br

Abstract. Given the increasing number of available genomic sequences, we now face the task of identifying their functional parts, like the protein coding regions. The gene prediction problem can be addressed in several ways, and one of the most promising methods is based on the comparison of evolutionary related sequences. Despite the number of comparative-based gene predictions tools available, the identification of coding regions in eukaryotic genomic sequences remains an open problem. In this paper we present a mathematical formulation for the gene prediction problem and propose a dynamic programming algorithm to solve it. We confirm the validity of our approach on a benchmark including 185 pairs of vertebrate genomic sequences.

Keywords: Pairwise Alignment, Gene Prediction, Bioinformatics

1 Introduction

The **Gene Prediction Problem** can be defined as the task of finding the protein-coding genes of a genomic sequence of interest. More specifically, given a DNA sequence, we would like to correctly locate the boundaries of its coding regions. Like the search for promoters, CpG islands and other functional genomic regions, the search for genes, that takes place at the annotation phase of any genomic project, has an undeniable practical importance.

To predict the genes in an eukaryotic DNA sequence of interest is a difficult task due to the fact that they are separated by long stretches of intergenic DNA. Furthermore, the coding fragments of a DNA, called **exons**, are intervened by non-coding ones, the **introns**. Besides this, there is no an evident pattern of bases that distinguishes the exons from the other regions of the sequence.

Gene prediction methods can be roughly classified into two main categories, named *ab initio* or intrinsic methods and *similarity-based* or extrinsic methods

* During the development of this work, the authors were supported by FUNDECT (Proc. No. 23/200.184/2007).

(see [6] for a review on this topic). The first ones rely on statistical information that alone, or in conjunction with some signals previously identified in the input sequence, allows the identification of its coding regions. The most recent intrinsic methods make use of Hidden Markov Models in order to combine both signal and statistical information concerning the target genes. The similarity-based methods make use of homology information between the genomic sequence and a fully annotated transcript sequence, like cDNAs, ESTs or proteins, in order to accomplish the gene prediction task.

Recently, with the huge amount of newly sequenced genomes, new similarity-based methods are being successfully applied in the task of gene prediction. In some way different from traditional extrinsic methods, the so-called *comparative-based methods*, pioneered by Batzoglou *et al.* with Rosetta [1], rely on local similarities between regions of two or more unannotated genomic sequences in order to find the genes encoded in each of them. The main assumption of these methods is that the functional parts of the eukaryotic genomic sequences tend to be more conserved than the non-functional ones. Despite the enormous progress made to date (see Brent and Guigó [2] for a survey on this topic), the gene identification problem remains open.

In this work we present a mathematical formulation for the gene prediction problem in its comparative context. In the proposed combinatorial optimization problem, we are interested in finding two ordered subsets of non-overlapping segments, with the same number of elements, such that their corresponding segments are very similar. Based on an efficient solution for this problem, we have developed a new comparative-based gene prediction tool, that was validated on a set of human genomic sequences.

This paper is organized as follows. In the next section we introduce the paired segments alignment problem and present a dynamic programming algorithm to solve it. The computational results obtained on a benchmark including human genomic sequences are shown in Section 3. The concluding remarks are presented in the last section.

2 Paired Segments Alignment Problem

We start with the formal statement of the problem used to model the gene prediction task by comparison of two genomic sequences. This problem is called *Paired Segments Alignment Problem*, and to a better understanding of it consider the following. Let s be a string over an alphabet Σ. We say that a segment $s_1[i..j]$ of s **ends** before a segment $s_2[k..l]$, also of s, if $j < k$, and denote this fact by $s_1 \prec s_2$. Let B be a set of segments of s. A subset $\Gamma_B = \{s_1, s_2, \ldots, s_p\}$ of B is a **chain of segments** if $s_1 \prec s_2 \ldots \prec s_p$. Finally, given two strings s and t, $sim_w(s,t)$ denotes the score of an optimal alignment between s and t under a scoring function w. With these definitions in mind, the paired segments alignment problem can be stated as follows:

Paired Segments Alignment Problem (PSAP): given two strings s and t, a set B of segments of s, a set C of segments of t and a scoring function w,

find two chains of segments $\Gamma_B = \{b_1, \ldots, b_q\}$ and $\Gamma_C = \{c_1, \ldots, c_q\}$ such that $\sum_{i=1}^{q} sim_w(b_i, c_i)$ is maximum among all possible chains of segments of B and C.

The relation between the PSAP and the gene prediction problem becomes clear taking s and t as two evolutionary related DNA sequences, and the segments of B and C as potential exons of s and t, respectively. With this, a solution for the PSAP will correspond to two chains of exons that best fit one another. Given the supposed high rate of conservation of the protein-coding regions, a solution for the PSAP will probably includes the real exons of both sequences.

The PSAP is similar to the fragment-chaining problem (FCP) [7], but there is a subtle difference between them. While the input to the FCP is a set of highly similar segments of s and t, the input to the PSAP does not carry any information about the similarities of the input segments.

A naive brute-force solution for the PSAP would generate all possible chains of segments of B and C and calculate the value of $\sum_{i=1}^{q} sim_w(b_i, c_i)$ for that ones with the same number of segments q. It is easy to see that this approach is hardly suitable when the number of segments is high. Fortunately, a deeper look at the problem reveals that a solution for a given instance of PSAP can be constructed efficiently from optimal solutions to its subproblems. Moreover, a solution for a subproblem may be reused several times in computing a solution for the original instance. These observations lead to a dynamic programming algorithm for the PSAP.

To a fully understanding of the proposed solution, consider the following. Given a string s over an alphabet Σ and a set $B = \{b_1, b_2, \ldots, b_u\}$ of segments of s, we call $first(i)$ and $last(i)$ the start and end positions, respectively, of the segment b_i relative to s. Let $\Gamma_B = \{b_x, \ldots, b_y, \ldots, b_z\}$ be a chain of segments of B such that some segment b_y contains the position i of s and define $\Gamma_B(y[i]) = \{b_x, \ldots, b_y[i]\}$ as an ordered subset of Γ_B that includes the segments prior to b_y and the segment b_y from its beginning until the positions i (of s). With these definitions in mind, let M be a matrix such that $M[i, j, k, l]$ ($0 \leq i \leq |s|$, $0 \leq j \leq |t|$, $0 \leq k \leq |B|$, $0 \leq l \leq |C|$) stores the value of a solution for the PSAP that includes segments prior to b_k and c_l and the segments b_k and c_l from their beginning until the positions i and j of s and t, respectively.

From the above definitions, the computation of M can be done by thinking in four distinct cases according to the positions of i and j relative to the segments b_k and c_l, respectively. When we are inside both segments b_k and c_l ($i \neq first(k)$ and $j \neq first(l)$), the position $M[i, j, k, l]$ is calculated by using the same recurrence proposed by Needleman and Wunsch [8] to compute the global similarity of two strings. When we are at the beginning of both segments b_k and c_l ($i = first(k)$ and $j = first(l)$), the position $M[i, j, k, l]$ is calculated by searching for a solution that makes use of segments prior to b_k and c_l and choosing the best option among align $s[i]$ with $t[j]$, $s[i]$ with a space or $t[j]$ with a space. Finally, when we are at the beginning of a segment and inside the other one ($i = first(k)$ and $j \neq first(l)$ or $i \neq first(k)$ and $j = first(l)$), the position $M[i, j, k, l]$ is calculated by extending the current alignment (align $s[i]$ with space

or $t[j]$ with space), or by searching for a solution that makes use of segments prior to b_k and c_l and choosing the best option among align $s[i]$ with $t[j]$, $t[j]$ with a space or $s[i]$ with a space. In these last two cases, indels are inserted at the beginning of s or t. Recurrence 1 gives the details to compute M.

$$
M[i,j,k,l] = \max \begin{cases}
\text{if } i \neq \mathit{first}(k) \text{ and } j \neq \mathit{first}(l): \\
\begin{cases}
M[i-1,j-1,k,l] + \omega(s[i],t[j]), \\
M[i,j-1,k,l] + \omega(-,t[j]), \\
M[i-1,j,k,l] + \omega(s[i],-)
\end{cases} \\
\text{if } i = \mathit{first}(k) \text{ and } j = \mathit{first}(l): \\
\begin{cases}
\max_{k'<k,l'<l} M[\mathit{last}(k'),\mathit{last}(l'),k',l'] + \omega(s[i],t[j]), \\
\max_{k'<k,l'<l} M[\mathit{last}(k'),\mathit{last}(l'),k',l'] + \mathit{indel} + \omega(-,t[j]), \\
\max_{k'<k,l'<l} M[\mathit{last}(k'),\mathit{last}(l'),k',l'] + \mathit{indel} + \omega(s[i],-)
\end{cases} \\
\text{if } i = \mathit{first}(k) \text{ and } j \neq \mathit{first}(l): \\
\begin{cases}
M[i,j-1,k,l] + \omega(-,t[j]), \\
\max_{k'<k,l'<l} M[\mathit{last}(k'),\mathit{last}(l'),k',l'] + \\
\qquad (j - \mathit{first}(l)) \times \mathit{indel} + \omega(s[i],t[j]), \\
\max_{k'<k,l'<l} M[\mathit{last}(k'),\mathit{last}(l'),k',l'] + \\
\qquad (j - \mathit{first}(l)+1) \times \mathit{indel} + \omega(s[i],-)
\end{cases} \\
\text{if } i \neq \mathit{first}(k) \text{ and } j = \mathit{first}(l): \\
\begin{cases}
M[i-1,j,k,l] + \omega(s[i],-), \\
\max_{k'<k,l'<l} M[\mathit{last}(k'),\mathit{last}(l'),k',l'] + \\
\qquad (i - \mathit{first}(k)) \times \mathit{indel} + \omega(s[i],t[j]), \\
\max_{k'<k,l'<l} M[\mathit{last}(k'),\mathit{last}(l'),k',l'] + \\
\qquad (i - \mathit{first}(k)+1) \times \mathit{indel} + \omega(-,t[j])
\end{cases}
\end{cases}
\tag{1}
$$

The first position of M is initialized to 0 ($M[0,0,0,0] = 0$) and after its completion, the score of an optimal paired segments alignment can be found by searching for the maximum value at the positions $M[last(k), last(l), k, l]$ with $1 \leq k \leq |B|$ and $1 \leq l \leq |C|$. A solution for the PSAP can be constructed by a traceback procedure. Starting from the position that stores the score of an optimal paired segments alignment, we proceed to the cell that stores the value used to calculate the optimum score, and continue in this way until the position $M[0,0,0,0]$ is reached.

A naive implementation of Recurrence 1 gives rise to an algorithm whose space complexity is $\mathcal{O}(|s| \times |t| \times |B| \times |C|)$ and that runs in $\mathcal{O}(|s| \times |t| \times |B| \times |C| + |B|^2 \times |C|^2 \times (bmax + cmax))$, where $bmax$ and $cmax$ are the length of the largest segment in B and C, respectively. Since the length of the input genomic sequences as well as the number of their potential exons can be rather large, these complexities are prohibitive. An efficient implementation of the Recurrence 1 takes into account the fact that $M[i,j,k,l]$ is defined only if i is a position of b_k and j is a position of c_l. As a consequence, only a portion of the matrix M needs

to be computed. The overall number of such entries is $\sum_{k=1}^{k\leq|B|} |b_k| \times \sum_{l=1}^{l\leq|C|} |c_l|$. This observation leads to an efficient implementation of the Recurrence 1 that runs in time $\mathcal{O}(bmax \times cmax \times |B| \times |C| + |B|^2 \times |C|^2 \times (bmax + cmax))$ and requires $\mathcal{O}(bmax \times cmax \times |B| \times |C|)$ space.

3 Experimental Results

The above ideas were implemented in a new comparative-based gene prediction program called GenePredictor. It takes two sequences in FASTA format as input and returns the locations of the exons predicted in each of them.

In order to evaluate the applicability of our formulation, GenePredictor was tested on a benchmark including 185 pairs of single gene vertebrate sequences. This benchmark was constructed from the human chromosome sequences of the ENCODE (ENCyclopedia of DNA elements) project [13]. More specifically, from the sequences annotated by the ENCODE project, we chose all the protein-coding genes with typical structure, maximum length 250000bp and no alternative splicing. At a second step, a homologous to each of these genes was taken from the HomoloGene Database [11]. For each chosen gene, we left 1000bp of intergenic region at their $5'$ and $3'$ side. About the corresponding set of potential exons, they were obtained by means of an algorithm, encoded on a gene prediction tool called Genscan [3], that finds the potential exons of a given DNA sequence. More details about each instance of this benchmark can be found at `http://www.facom.ufms.br/~said/segmalignment/benchmark.html`.

To a better insight into the accuracy of our gene prediction tool, its results were compared with that achieved by other comparative-based gene prediction programs: Agenda [12], Progen [9], SGP2 [10] and Twinscan [5]. This comparison was done by means of the following measures introduced by Burset and Guigó in [4]:

1. Specificity at the nucleotide level (Sp_n): proportion of nucleotides predicted as coding that are really coding;
2. Sensitivity at the nucleotide level (Sn_n): proportion of really coding nucleotides that have been correctly predicted as coding;
3. Specificity at the exon level (Sp_e): proportion of predicted exons that have been correctly predicted;
4. Sensitivity at the exon level (Sn_e): proportion of annotated exons that have been correctly predicted.

The quantity approximate correlation (AC), that evaluates the correlation between the predicted and real coding nucleotides, summarizes the sensitivity and specificity at the nucleotide level in a single measure. At the exon level, the average value of Sp_e and Sn_e (A_v) summarizes the sensitivity and specificity at the exon level.

The average values of specificity and sensitivity achieved by the programs, at both nucleotide and exon levels, are shown in Table 1.

Tools	Sp_n	Sn_n	AC	Sp_e	Sn_e	A_v
GenePredictor	0,93	0,94	0,93	0,62	0,76	0,69
AGenDA	0,85	0,67	0,67	0,45	0,43	0,44
Progen	0,81	0,96	0,87	0,47	0,60	0,54
Sgp2	0,86	0,80	0,78	0,55	0,51	0,53
Twinscan	0,91	0,77	0,79	0,65	0,52	0,59

Table 1. Average values of Sp, Sn, AC and A_v achieved by the programs.

The values in Table 1 show that 93% of the nucleotides predicted as coding by GenePredictor are in fact coding nucleotides. Furthermore, 94% of the really coding nucleotides were correctly identified as such by our program. At the exon level, 62% of the exons predicted by GenePredictor has a corresponding annotated exon. Beside this, 76% of the annotated exons was correctly predicted by our program. It is important to observe that, on average, our program outperforms the other tools at both nucleotide and exon level.

A deeper look at the instances on which GenePredictor presented the worst results reveals that the related mispredictions is mainly due to the absence of some segments at the input corresponding to annotated exons of the searched genes. This can be seen in Figure 1, where the last exon of bet1 sequence was missed by our program.

Other reason to the mispredictions is related to the formulation of the problem itself. Since we are interested in finding a set of segments that maximize the objective function, some of them can be erroneously identified as a real exon, leading to overpredictions. In Figure 1, the first two predicted exons of zbtb45, for example, correspond to overpredictions. The need to maximize the sum of the similarity of each pair of corresponding segments also leads to an erroneous prediction of segments a little bit bigger than annotated exons. Non-coding bases tend to be predicted at the boundaries of annotated exons in the sequences, where the similarity rate is still very high when compared with that of the intronic and intergenic regions. The first predicted exon of the sequence rasl10b, for example, is 147bp larger than the corresponding annotated one (Figure 1).

It is important to observe that the formulation proposed by us considers that the target genes have the same structure, i.e., the same number of exons. In order to evaluate how this constraint affects the results of our program, additional tests were made on a benchmark including 37 pairs of sequences that encode genes with different structures. The average values of specificity and sensitivity achieved in this new benchmark are shown in Table 2.

As can be seen in Table 2, the results presented by GenePredictor are still better than those achieved by the other tools. Taking as input sequences whose genes include different number of exons, our program tends to miss some annotated exons. This fact decreases the sensitivity of our program. But even with this drawback, the related results remains good, suggesting that our formulation is robust and applicable in practice.

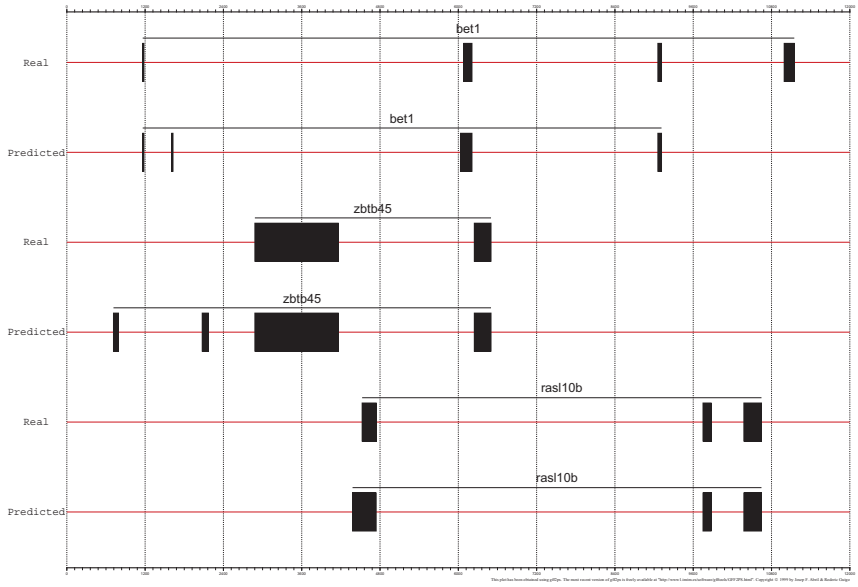

Fig. 1. Examples of exons mispredicted by GenePredictor.

4 Discussion

Despite its practical importance and the number of methods developed to date, the gene identification remains an open and interesting problem. Given the increasing number of homologous sequences in the databases and the assumption that the exons tend to be more conserved than the introns inside a genome, comparative-based gene prediction programs starts to be extensively used in the task of gene identification. In this work we presented a formulation to the gene prediction problem and, based on it, a new gene prediction tool.

Our approach is based on a combinatorial optimization problem whose objective is to find two ordered subsets of highly similar non-overlapping segments of the input strings. This problem was solved using a dynamic programming algorithm and the proposed solution was implemented and tested on a benchmark including 185 pairs of single common structure gene sequences. The results achieved by our program were better than those achieved by other comparative-based gene prediction tools. This occurred at all levels of evaluation, and even when the input sequences include genes with different structures.

The main drawback of our approach is related to the formulation itself and the existence of well conserved regions outside the searched genes. This leads to a number of mispredicted exons and additional bases identified as coding at the

Tools	Sp_n	Sn_n	CA	Sp_e	Sn_e	A_v
GenePredictor	0,94	0,96	0,94	0,60	0,75	0,68
AGenDA	0,84	0,66	0,66	0,44	0,42	0,43
Progen	0,86	0,96	0,89	0,48	0,57	0,53
Sgp2	0,89	0,85	0,83	0,52	0,51	0,51
Twinscan	0,92	0,80	0,81	0,63	0,52	0,58

Table 2. Average values of Sp, Sn, AC and A_v achieved by the programs in the benchmark including genes with different structures.

5-UTR and 3-UTR regions of the annotated genes. One way to overcome this problem is by making use of additional biological information that can give us better insights about the first and last exons of the input sequences. This work is in progress in the hope that better values of specificity and sensitivity at both nucleotide and exon level can be achieved in the future.

References

1. Batzoglou, S., Pachter, L., Mesirov, J., Berger, B., Lander, E.S.: Human and Mouse Gene Structure: Comparative Analysis and Application to Exon Prediction. Genome Research 10(7), 950-958 (2000)
2. Brent, M.R., Guigó, R.: Recent Advances in Gene Structure Prediction. Curr. Opin. Struct. Biol. 14(3), 264-272 (2004)
3. Burge, C., Karlin, S.: Prediction of Complete Gene Structures in Human Genomic DNA. Journal of Molecular Biology 268(1), 78-94 (1997)
4. Burset, M., Guigó, R.: Evaluation of Gene Structure Prediction Programs. Genomics 34(3), 353-367 (1996)
5. Korf, I., Flicek, P., Duan, D. Brent, M.R.: Integrating Genomic Homology into Gene Structure Prediction. Bioinformatics 17(Suppl 1), S140-S148 (2001)
6. Mathé, C., Sagot, M.-F., Schiex, T., Rouzé, P.: Current Methods of Gene Prediction, their Strengths and Weaknesses. Nucleic Acids Research 30(19), 4103-4117 (2002)
7. Myers, G., Miller, B.: Chaining Multiple-Alignment Fragments in Sub-Quadratic Time. SODA 38-47 (1995)
8. Needleman, S.B., Wunsch, C.D.: A General Method Applicable to the Search for Similarities in the Amino acid Sequence of Two Proteins. Journal of Molecular Biology 48(3), 433-453 (1970)
9. Novichkov, P.S., Gelfand, M.S., Mironov, A.A.: Gene Recognition in Eukaryotic DNA by Comparison of Genomic Sequences. Bioinformatics 17(11), 1011-1018 (2001).
10. Parra, G., Agarwal, P., Abril, J.F., Wiehe, T., Fickett, J.W., Guigó, R.: Comparative Gene Prediction in Human and Mouse. Genome Research 13(1), 108-117 (2003)
11. Sayers, E.W. *et al.*: Database Resources of the National Center for Biotechnology Information. Nucleic Acids Research 37(Suppl 1), D5-D15 (2008)
12. Taher, L., Rinner, O., Garg, S., Sczyrba, A., Brudno, M., Batzoglou, S., Morgenstern, B.: AGenDa: Homolgy-based Gene Prediction. Bioinformatics 19(12), 1575-1577 (2003)
13. The ENCODE Project Consortium: The ENCODE (encyclopedia of DNA elements) Project. Science 306(5696), 636-640 (2004)

A Structured-Population Genetic Algorithm for the 3-D Protein Structure Prediction Problem*

William W. Gonçalves, Márcio Dorn, Luciana S. Buriol, and Luis C. Lamb

Institute of Informatics, Federal University of Rio Grande do Sul,
Porto Alegre, Brazil
{wwgoncalves,buriol}@inf.ufrgs.br, {marcio.dorn,luislamb}@acm.org

Abstract. The Protein Structure Prediction (PSP) problem is one of the most important problems in Structural Bioinformatics. Algorithms have been proposed over the last years. However, the problem still remains challenging because the complexity and dimensionality of the protein conformational search space. We present a new strategy to predict native-like protein structures based on genetic algorithms that use a structured population. Different selection and population organization are used in order to guarantee the diversity of the population during the simulation. We illustrate the efficacy of our method in six instances.

Keywords: Genetic Algorithms, 3-D PSP problem, Bioinformatics

1 Introduction

The protein structure prediction (PSP) problem is one of the most important problems in Structural Bioinformatics [8]. The protein structure dictates the function of protein in the cell. The PSP problem consists in determining the three-dimensional (3-D) structure of a protein given only its amino acid sequence [8]. Several methods have been recently applied to the PSP problem. These methods can be classified as [8]: (a) *Ab initio* methods without data base information [15]; (b) first principle methods with database information [17]; (c) comparative modeling methods [13] and (d) fold recognition methods [11].

Ab initio methods without data base information are thermodynamics based and rely on the fact that the native structure of a protein corresponds to the global minimum of its free energy. The goal is to find the global minimum of its free energy which corresponds to the protein native structure or their functional conformation [15]. The main advantage of methods belonging at this group is that they can predict new folds of proteins because they are not limited to templates from the Protein Data Bank (PDB) [2]. In methods that use database information, structural patterns are extracted from the PDB and are used to build 3-D protein structures. Methods belonging to group 'b', frequently do not compare an entire target sequence against the sequence of template protein with known 3-D structure, but they compare short fragments of amino acid residues

* The authors thank MCT/CNPq (Brazil) for the financial support.

and combine them in order to find the 3-D protein structure with the lowest potential energy. In comparative modeling, the target sequence is aligned to the sequence of an evolutionary related template with known 3-D structure in the PDB [13]. Once homology is detected, usually above the 30% identity threshold, modeling can proceed by copying coordinates of the template or the average of multiples templates, or using the distance and torsion angles and inter-atomic distances from aligned regions from template as modeling restraints. *Ab initio* methods can predict new folds of proteins, but they have to deal with the problem of the complexity and high dimensionality of the conformational search space [14]. In optimization, the PSP is classified as a NP-Complete problem [5]. First principle methods that use database information have also to deal with the magnitude of the search space and have as prerequisite the necessity of finding a clever way to manipulate the structural data from the PDB and use it to predict 3-D native-like protein structures.

We describe a new method based on genetic algorithms to predict structures of polypeptides or proteins. We explore the application of a structured population in order to guarantee the diversity during the simulation phase and the convergence of the method to find native-like protein structures. The main goal of the developed method is to construct in a relative short time approximate structures of polypeptides. This structure is expected to be good enough to be further refined by means of *ab initio* methods such as Molecular Dynamics simulation. Section 2 presents preliminaries on protein structure concepts, while Section 3 presents our proposed genetic algorithm with structured population. Experiments and results are presented in Section 4. Section 5 discusses the obtained results. Finally, Section 6 concludes the paper.

2 Preliminaries

A polypeptide is a molecule composed of two or more amino acid residues chained by a peptide bond (Fig. 1). A peptide bond is a covalent chemical bond formed between two molecules when the carboxyl group of one molecule reacts with the amino group of the other molecule, thereby releasing a molecule of water ($H2O$). The amino acid residues differ from each other through the group R that is attached to the C_α. A peptide has three main chain torsion angles, namely phi (ϕ), psi (ψ) and omega (ω). In the model peptide (Fig. 1) the bonds between N and C_α, and between C_α and C are free to rotate. These rotations are described by the ϕ and ψ torsion angles, respectively. The main chain ϕ and ψ torsion angles are the most responsible for determining the conformation or fold of a polypeptide [3]. In this work we represent the polypeptide structure based only on the backbone (ϕ, ψ, and ω) and side-chain (χ_i) torsion angles. Fig. 1 represents schematically all information about the model peptide adopted. All the peptide bond torsion angles (ω) has a fixed value at their ideal value 180°.

The internal energy of the protein and its interactions with the environment in which it is inserted is described by an energy function. Frequently, energies function are used in *ab initio* methods in order to evaluate the conformation

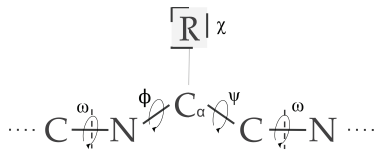

Fig. 1. Schematic representation of a model peptide. N is nitrogen, C and C_α are carbons and R is an arbitrary side-chain. Dotted lines identify the peptide bond.

along the simulation and find the conformation with the global minimum of its free energy. A potential energy function incorporates two types of terms: bonded and non-bonded. The bonded terms (*bonds*, *angles* and *torsion's*) are covalently linked. The bonded terms constrain bond lengths and angles near their equilibrium values. The bonded terms also include a torsional potential (*torsion*) that models the periodic energy barriers encountered during bond rotation. The non-bonded potential includes: ionic bonds, hydrophobic interactions, hydrogen bonds, van der Waals forces, and Dipole-dipole bonds. In the propose GA the AMBER-94 [4] potential energy function is used as the objective function. AMBER is one of the most widely used force fields whose functional form is given by Eq. 1.

$$Energy = \sum_{\text{bonds}} \frac{1}{2}K_b(b-b_0)^2 + \sum_{\text{angles}} \frac{1}{2}K_\theta(\theta-\theta_0)^2 + \sum_{\text{torsions}} \frac{1}{2}K_\eta(1+\cos(\eta_\omega-\gamma)) + \sum_{\text{j=1}}^{N-1}\sum_{\text{i=j+1}}^{N-1}\left\{\epsilon_{i,j}\left[\left(\frac{R_{0ij}}{r_{ij}}\right)^{12} - 2\left(\frac{R_{0ij}}{r_{ij}}\right)^{6}\right] + \frac{q_i q_j}{4\pi\epsilon_0 r_{ij}}\right\} \quad (1)$$

where:
Bonds represent the energy between covalently bonded atoms. *Angles* represent the energy due to the geometry of electron orbitals involved in covalent bonding. *Torsions* represent the energy for twisting a bond due to bond order (e.g. double bonds) and neighboring bonds or lone pairs of electrons. The last term represents the non-bonded energy between all atom pairs, which can be decomposed into van der Waals (first term of the double summation) and electrostatic energies (second term of the double summation).

3 Proposed Method

Genetic algorithms have gained steady recognition as useful computational tools for addressing optimization tasks related to protein structures and in particular to the PSP problem [16].We developed a structured-population genetic algorithm for the PSP problem. In our proposal, a genetic algorithm is combined with a structured population. The solution structure is a set of n genes, where each gene corresponds to a set of angles of the protein. Each set of angles are comprised

of two dihedral angles (ϕ, ψ) from the protein backbone and a number of angles χ that varies according with the type of amino acid residue. Each chromosome is represented by a list of amino acids structures. Each amino acid structure consists of informations of amino acid type, secondary structure and number of side-chain angles, and their current values for ϕ, ψ and χ_i angles, when appropriate. To evaluate the conformation of a solution, we use the AMBER potential energy function (1). To calculate the potential energy of each solution (fitness value) we use routines from the TINKER molecular mechanics program package. The function returns a value for the energy based on the molecule conformation provided by the solution. Our algorithm searches for a global minimum energy, so the lower the energy, better is the solution. The population is structured in "castes" as originally proposed in [7]. A structured-population provides a way to organize solutions, which allows a better selection of individuals for crossover. The overall goal is to obtain better solutions and, simultaneosly, to reduce the computational time. After sorting the individuals according to their fitness values, we divide the population into three castes. The fittest 40% of the individuals compose caste A, the 40% next fittest ones compose caste B, and the remaining 20% least fit ones compose caste C. The algorithm combines one parent from the elite caste (caste A) and the other parent from a non-elite caste (either caste B or C). As illustrated in Fig. 2, for producing a new generation the algorithm keeps the caste A entirely to next generation, replaces solutions fro caste C by randomly generated solutions, and complete the population with solutions originated from the crossover.

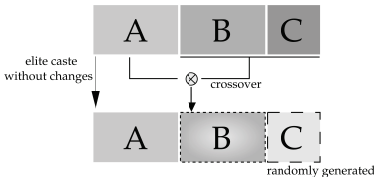

Fig. 2. Schematic representation of a new generation breeding.

The *random-key* crossover procedure [1] combines two parent solutions p_1 (elite) and p_2 (non-elite) to generate a new solution. For each crossover gene, a number is randomly generated and if it is $\leq K$, then the corresponding gene (amino acid structure) is inherited from p_1, otherwise from p_2. We adopted a cutoff $K = 0.7$ which experimentally produced good convergence results. This *random-key* crossover procedure gives genes from the elite parent more chances to be selected. To avoid premature convergence, we do not accept two solutions having the same energy value. When a solution is evaluated, if another solution has the same value, then it is replaced by a random solution.

4 Experiments and Results

Using the proposed algorithm we predict the approximate 3-D protein structure of six proteins sequences from the PDB. We run the GA four times for each instance. For the biochemical analysis we selected the best solution among the four results. In our experiments amino acid residues are bounded by torsion angles (ϕ and ψ) intervals. Each conformational state has a different interval of torsion angles [9]. Side-chain torsion angles are constrained in regions derived from the backbone-independent rotamer library of Dunbrack [6]. We use routines of the Tinker molecular modeling package (`http://dasher.wustl.edu/tinker`) to convert torsion angles into 3D atomic representation.

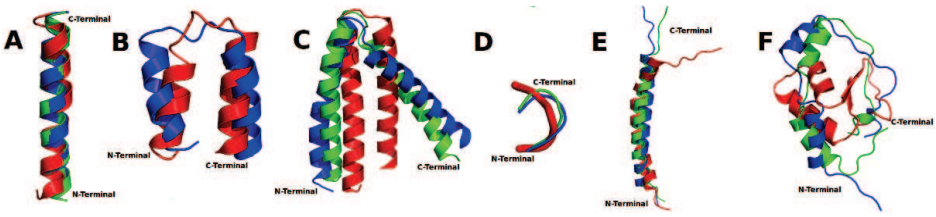

Fig. 3. Ribbon representation of the experimental (red), the predicted structures (blue) and the structures with the minor RMSD found by the GA for each instance (green). The C_α of the experimental and predicted structures are fitted. A, B, C, D, E present, respectively, the experimental and predicted 3-D structures of the protein with PDB ID: 1A11 (A), 1ZDD (B), 1ROP (C), 1PLW (D), 1JR8 (E), and 1CRN (F). Amino acid side chains are not shown for clarity. Graphics were generated by `www.pymol.org`.

5 Analysis of Results

In order to measure the quality of the predicted 3-D structures we calculate the root mean square deviation (RMSD) value of the predicted conformation with the lowest potential energy. Table 5 presents the C_α RMSD of the predicted approximate conformations with respect to their experimental structures. We observe that for case studies 1A11, 1ZDD, 1PLW, and 2JR8 we obtain accurate results (1.09Å, 4.48Å, 0.25Å, 6.14Å, respectively). The case studies 1CRN and 1ROP present higher RMSD (10.80Å and 8.27Å, respectively), what is expected since 1CRN has a more complex folding pattern when compared to the other test cases. In oder to analyze the secondary structure arrangement of the predicted structures (Fig. 3 A, B, C, D, E and F- Blue - structure with the lowest energy), we run the PROMOTIF [10] . Table 2 shows the number (%) of amino acid residues that occur in one of four conformational states (Regular conformational state: β-sheet, α-helix, 3^{10}-helix and others (coil or loop regions)). We observe that the secondary structure formation of the predicted structures are closely related to their experimental structures.

Table 1. Quality of the generated solutions. The best solution found in the initial population (*Initial*) and the best final solution (*Final*) are presented. Potential energy and the RMSD values are also presented. The extension A, B, C and D indicates, respectively, each one of the four runs of each instance. A '*' marks the lowest potential energy of the four runs.

Instance	Energy ($kcal/mol$)		C_α RMSD (Å)	
	Initial	Final	Initial	Final
1A11_A	$2.47 \cdot 10^6$	-475.74	1.09	1.00
1A11_B	$9.18 \cdot 10^6$	-477.13	0.83	0.80
1A11_C	$1.48 \cdot 10^6$	-488.46	1.82	0.99
1A11_D *	407980	-491.33	1.51	1.09
1CRN_A	$4.81 \cdot 10^7$	-84.45	8.53	9.66
1CRN_B	$1.77 \cdot 10^6$	-256.59	13.43	9.13
1CRN_C *	$1.04 \cdot 10^7$	-374.32	13.14	10.80
1CRN_D	$1.62 \cdot 10^8$	-297.81	9.84	10.87
1PLW_A	102.79	-92.96	0.51	0.15
1PLW_B *	94.17	-102.84	0.09	0.25
1PLW_C	35.96	-102.19	0.17	0.13
1PLW_D	68.29	-93.03	0.47	0.13
1ROP_A	$5.18 \cdot 10^8$	-658.97	13.34	12.70
1ROP_B *	$9.53 \cdot 10^8$	-710.74	14.56	8.27
1ROP_C	$2.77 \cdot 10^9$	-707.01	12.77	8.19
1ROP_D	$5.18 \cdot 10^8$	-680.28	12.43	7.42
1ZDD_A	$2.25 \cdot 10^8$	-1022.56	5.32	5.92
1ZDD_B	$4.53 \cdot 10^7$	-997.85	8.99	5.83
1ZDD_C	$1.59 \cdot 10^8$	-824.22	8.09	4.91
1ZDD_D *	$5.85 \cdot 10^8$	-1049.17	5.51	4.48
2JR8_A	$2.52 \cdot 10^7$	1153.76	3.77	6.43
2JR8_B	$5.20 \cdot 10^8$	1081.93	6.12	5.86
2JR8_C	$1.15 \cdot 10^8$	1137.59	4.49	5.14
2JR8_D *	$4.42 \cdot 10^7$	1017.37	4.68	6.14

We analyse with PROCHECK [12] the distribution of the amino acid residues in the Ramachandran Plot. These results are summarized in Table 3. We observe that in predicted structures $\geq 95\%$ of the amino acid residues are located in the most favorable regions of the Ramachandran plot. This means that these structures have a small number of bad contacts. When we compare the results obtained with the predicted structure against the experimental structures we observe that the structures are very similar.

Table 2. Analysis of the secondary structure arrangement of the predicted approximate conformations. -P denotes the predicted structures with the lowest energy solution among the four runs. Values are expressed in %.

PDB ID	Strand	α-helix	3^{10}-helix	Other	Residues
1A11	0.0	92.0	0.0	8.0	25
1A11-P	0.0	92.0	0.0	8.0	25
1ZDD	0.0	73.5	0.0	26.5	34
1ZDD-P	0.0	76.5	0.0	23.5	34
1ROP	0.0	89.3	0.0	10.7	56
1ROP-P	0.0	89.3	0.0	10.7	56
2JR8	0.0	73.8	0.0	26.2	42
2JR8-P	0.0	73.8	0.0	26.2	42
1CRN	8.7	41.3	6.5	43.5	46
1CRN-P	0.0	43.5	(6.5	50.0	46

Table 3. Numerical Ramachandran plot values for the experimental and predicted conformations. -P denotes the predicted structures with the lowest energy in the four group of solutions. Values are expressed in %.

	Most Favorable	Additional	Generously	Disallowed
PDB	Region	Allowed	Allowed	
1A11	91.3	4.3	4.3	0.0
1A11-P	100.0	0.0	0.0	0.0
1ZDD	87.1	12.9	0.0	0.0
1ZDD-P	87.1	12.9	0.0	0.0
1ROP	98.1	1.9	0.0	0.0
1ROP-P	96.3	1.9	0.0	1.9
2JR8	85.3	8.8	2.9	2.9
2JR8-P	85.3	14.7	0.0	0.0
1CRN	94.3	5.7	0.0	0.0
1CRN-P	82.9	17.1	0.0	0.0

6 Conclusions

In this article, we proposed a new algorithm for the 3-D Protein Structure Prediction problem. The algorithm combines a genetic algorithm principle with a structured population procedure. Our method allows efficient mechanisms for protein structure prediction. The experiments described in the paper show that the developed method can indeed produce accurate predictions. The main contributions of the paper are: First, the use of computational techniques effective algorithms for a relevant biological problem (the 3-D PSP problem). Second, the use of genetic algorithms with a structured population. This may open several research avenues, with potential applications in computational biology. For example, the application of the developed method to other classes of proteins.

Further, it would be interesting to combine the developed algorithm with a conjugate gradient which could also lead to efficient algorithms for the PSP problem.

References

1. Bean, J.C.: Genetic algorithms and random keys for sequencing and optimization. ORSA J. on Comp. 6, 154–160 (1994)
2. Berman, H., Westbrook, J., Feng, Z., Gilliland, G., Bath, T., Weissig, H., Shindyalov, I., Bourne, P.: The protein data bank. Nucleic Acids Res. 28(1), 235–242 (2000)
3. Branden, C., Tooze, J.: Introduction to protein structure. Garlang Publishing Inc., New York, USA, 2 edn. (1998)
4. Cornell, W., Cieplak, P., Bayly, C., Gould, I., Merz Jr., K., Ferguson, D., Spellmeyer, D., Fox, T., Caldwell, J., Kollman, P.: A second generation force field for the simulation of proteins, nucleic acids, and organic molecules. J. Am. Chem. Soc. 117(19), 5179–5197 (1995)
5. Crescenzi, P., Goldman, D., Papadimitriou, C., Piccolboni, A., Yannakakis, M.: On the complexity of protein folding. J. Comput. Biol. 5(3), 423–466 (1998)
6. Dunbrack Jr., R., Cohen, F.: Bayesian statistical analysis of protein side-chain rotamer preferences. Protein Sci. 6(8), 1661–1681 (1997)
7. Ericsson, M., Resende, M., Pardalos, P.: A genetic algorithm for the weight setting problem in ospf routing. J. Comb. Optim. 6, 299–2002 (2002)
8. Floudas, C., Fung, H., McAllister, S., Moennigmann, M., Rajgaria, R.: Advances in protein structure prediction and de novo protein design: A review. Chem. Eng. Sci. 61(3), 966–988 (2006)
9. Hovmöller, T., Ohlson, T.: Conformation of amino acids in protein. Acta Crystallogr. 58(5), 768–776 (2002)
10. Hutchinson, E., Thornton, J.: Promotif: A program to identify and analyze structural motifs in proteins. Protein Sci. 5(2), 212–220 (1996)
11. Jones, D.: Successful ab initio prediction of the tertiary structure of nk-lysin using multiple sequences and recognized supersecondary structural motifs. Proteins S1, 185–191 (1997)
12. Laskowski, R., MacArthur, M., Moss, D., Thornton, J.: Procheck: a program to check the stereochemical quality of protein structures. J. Appl. Crystallogr. 26(2), 283–291 (1993)
13. Martí-Renom, M., Stuart, A., Fiser, A., Sanchez, A., Mello, F., Sali, A.: Comparative protein structure modeling of genes and genomes. Annu. Rev. Biophys. Biomol. Struct. 29(16), 291–325 (2000)
14. Ngo, J., Marks, J., Karplus, M.: The protein folding problem and tertiary structure prediction. In: Merz Jr, K., Grand, S. (eds.) Computational complexity, protein structure prediction and the Levinthal Paradox, pp. 435–508. Birkhauser, Boston, USA (1997)
15. Osguthorpe, D.: Ab initio protein folding. Curr. Opin. Struct. Biol. 10(2), 146–152 (2000)
16. Pedersen, J., Moult, J.: Protein folding simulations with genetic algorithms and a detailed molecular description. Journal of Molecular Biology 269(2), 240–259 (1997)
17. Rohl, C., Strauss, C., Misura, K., Baker, D.: Protein structure prediction using rosetta. Methods Enzymol. 383(2), 66–93 (2004)

Constraint Logic Programming Models for Reversal and Transposition Distance Problems

Victor de Abreu Iizuka and Zanoni Dias

Institute of Computing, University of Campinas
Av. Albert Einstein 1251, Cidade Universitária, Campinas-SP, Brazil
victor.iizuka@students.ic.unicamp.br, zanoni@ic.unicamp.br

Abstract. Genome Rearrangements research appeared in the last years to deal with problems such as to find the minimum number of rearrangement events, for example, transposition, reversals, fusion and fissions, needed to transform one genome into another. In this paper we follow the research line of Dias and Dias [12] and we introduce Constraint Logic Programming (CLP) models for sorting by reversals and transpositions, based on Constraint Satisfaction Problems (CSP) theory and Constraint Optimization Problems (COP) theory, for unsigned and linear permutations. We made a comparison between the CLP models and the ILP polynomial models described in Dias and Souza [13].

1 Introduction

Genome rearrangements field focus on the comparison of the positions of the same blocks of genes on distinct genomes and on the rearrangement events that possibly transformed one genome into another. Previous studies show that rearrangements are more suitable than nucleotide (or amino acid) comparison when the objective is to compare the genome of two species [6, 19].

The transposition exchanges two adjacent blocks of any size in a chromosome. The transposition distance problem is to find the minimum number of transpositions that transform one genome into another. This problem is NP-hard, the proof was presented by Bulteau, Fertin e Rusu [10]. The best approximation algorithm ratio is 1.375 and was presented by Elias and Hartman [14],

The reversal inverts a block of any size in a chromosome. The reversal distance problem is to find the minimum number of reversals that transform one genome into another. The problem of reversals with unsigned permutations is NP-hard, the proof was presented by Caprara [11]. The best approximation algorithm is 1.375 and was presented by Berman, Hannenhalli and Karpinski [9].

The reversal and transposition distance problem is to find the minimum number of reversals and transpositions that transform one genome into another. Walter, Dias and Meidanis [17, 20] and Lin and Xue [15] studied this problem.

In this paper we will focus on sorting by reversals and transpositions with unsigned and linear permutations and we want to find the exact distance necessary to transform one genome into another. We follow the research line of Dias

and Dias [12] and we introduce Constraint Logic Programming (CLP) models for sorting by reversals and transpositions, based on Constraint Satisfaction Problems (CSP) theory and Constraint Optimization Problems (COP) theory. We made a comparison between the CLP models and the ILP polynomial models described in Dias and Souza [13].

This paper is divided as follow. Section 2 presents the CLP model for sorting by reversal and transposition problem. Section 3 discusses the computational tests. Finally, Section 4 exhibits the conclusions and future works.

2 CLP Model for Sorting by Reversals and Transpositions

In this Section we present a basic CLP model for sorting by reversals and transpositions. We will use the Constraint Satisfaction Problem (CSP) and Constraint Optimization Problems (COP) theories to formulate the problem. We define the formulations using the prolog-like Marriott's notation [16] as much as possible.

The representation of permutation (1) and the effects of reversal (2) and transposition (3) can be seen as the same way we described the problem. In this model the permutation π is a list of elements $(\pi_1, \pi_2, \cdots, \pi_n)$ where $\pi_i \in \mathbb{N}$, $0 < \pi_i \leq n$ and $\pi_i \neq \pi_j$ for $i \neq j$. The identity permutation ι is defined as $\iota = (1\ 2\ 3\ \cdots\ n)$

$$\begin{aligned} &\mathit{permutation}(\pi, N)\ \text{:-} \\ &\quad \mathit{length}(\pi, N), \\ &\quad \pi\ ::\ [1\ ..\ N], \\ &\quad \mathit{all\_different}(\pi). \end{aligned} \tag{1}$$

Note that in prolog variables are denoted by strings starting with an upper letter or "_" (the underscore) if the variable is anonymous. The greek letters π and σ are lists in this notation. The construction $X\ ::\ [i\ ..\ j]$ means that X (or every element of X if X is a list) ranges over the interval $[i\ ..\ j]$.

A reversal $\rho(i, j)$, $0 < i < j \leq n$, split the list in three sub-lists $C_1C_2C_3$ where $C_1 = (\pi_1..\pi_{i-1})$, $C_2 = (\pi_i..\pi_j)$ and $C_3 = (\pi_{j+1}..\pi_n)$. After that, we do a reverse on the sub-list C_2 and the result is the sub-list R_{C_2}. Finally we join the new sub-list R_{C_2} with the sub-lists C_1 and C_3 to form $\rho\pi = C_1R_{C_2}C_3$.

$$\begin{aligned} &\mathit{reversal}(\pi, \sigma, I, J)\ \text{:-} \\ &\quad \mathit{permutation}(\pi, N), \\ &\quad \mathit{permutation}(\sigma, N), \\ &\quad 1 \leq I < J \leq N, \\ &\quad \mathit{split}(\pi, I, J, C_1, C_2, C_3), \\ &\quad \mathit{reverse}(C_2, R_{C_2}), \\ &\quad \sigma = C_1, R_{C_2}, C_3. \end{aligned} \tag{2}$$

A transposition $\rho(i, j, k)$, $0 < i < j < k \leq n$, split the list in four sublists $C_1C_2C_3C_4$ where $C_1 = (\pi_1..\pi_{i-1})$, $C_2 = (\pi_i..\pi_{j-1})$, $C_3 = (\pi_j..\pi_{k-1})$ and $C_4 = (\pi_k..\pi_n)$. After we join them to form $\rho\pi = C_1C_3C_2C_4$. Note that C_1 and C_4 could be empty.

$$
\begin{aligned}
&\mathit{transposition}(\pi, \sigma, I, J, K) \text{ :-} \\
&\quad \mathit{permutation}(\pi, N), \\
&\quad \mathit{permutation}(\sigma, N), \\
&\quad 1 \leq I < J < K \leq N, \\
&\quad \mathit{split}(\pi, I, J, K, C_1, C_2, C_3, C_4), \\
&\quad \sigma = C_1, C_3, C_2, C_4.
\end{aligned}
\tag{3}
$$

We first model the problem as CSP, but the number of variables is unknown because we need the value of distance $d_{r\_t}(\pi)$ to set the constraints and variables that represent the permutations. For this reason we pick a candidate value for distance N such that $N \in [LB..UB]$, where LB is a known lower bound and UB is a known upper bound for the problem, and try to find the appropriate combination of N reversals and transpositions. If the CSP fails with the candidate N, we choose another value for N just incrementing its value. We check the value of N using a bottom-up strategy and for definition we don't check any value higher than any upper bound UB. This behaviour is described by *rev_trans_dist/3* predicate (4).

The *event/2* predicate chooses the best event between the *reversal/4* predicate (2) and the *transposition/5* predicate (3) to minimize the distance.

The *indomain(X)* predicate gets the domain of the variable X and chooses the minimum element in it. If a fail backtracks to *indomain*, the element that generated the fail will be removed from the domain and another value will be chosen.

$$
\begin{aligned}
&\mathit{rev\_trans\_dist}(\iota, 0, \_Model). \\
&\mathit{rev\_trans\_dist}(\pi, N, Model) \text{ :-} \\
&\quad \mathit{bound}(\pi, Model, LB, UB), \\
&\quad N :: [LB..UB], \\
&\quad \mathit{indomain}(N), \\
&\quad \mathit{event}(\pi, \sigma), \\
&\quad \mathit{rev\_trans\_dist}(\sigma, N-1, Model).
\end{aligned}
\tag{4}
$$

The CSP models have the above structure changing only the bounds we used. We call **def_csp** the model that doesn't use any lower bounds, **r_t_br_csp** the model that chooses the best bound between the reversal breakpoint lower and upper bounds described by Bafna and Pevzner [7] and the transposition breakpoint lower and upper bounds described by Bafna and Pevzner [8], and

r_t_bc_csp the model that chooses the best bound between the reversal breakpoint lower and upper bounds and the transposition edge-colored cycle graph lower and upper bounds described by Bafna and Pevzner [8].

Another approach is to model the problem as a COP. This approach needs an upper bound and some changes on previous predicates. We use the binary variables B to indicate whether a event, reversal or transposition, has modified the permutation.

The first predicate that we need to create is *reversal_cop/5* (5). First of all, given a permutation $\rho(i, j)$, we add a new clause to allow $(i, j) = (0, 0)$. If $(i, j) = (0, 0)$ then $\pi\rho = \pi$. We add a new argument to the *reversal_cop/5* predicate that receive the variable B.

$$\begin{aligned} &reversal\_cop(\iota, \iota, 0, 0, 0). \\ &reversal\_cop(\pi, \sigma, I, J, 1) \text{ :- } reversal(\pi, \sigma, I, J). \end{aligned} \tag{5}$$

The equivalent predicate for transposition is *transposition_cop/6* (6). In this case, given a permutation $\rho(i, j, k)$, we add a new clause to allow $(i, j, k) = (0, 0, 0)$. If $(i, j, k) = (0, 0, 0)$ then $\pi\rho = \pi$.

$$\begin{aligned} &transposition\_cop(\iota, \iota, 0, 0, 0, 0). \\ &transposition\_cop(\pi, \sigma, I, J, K, 1) \text{ :- } transposition(\pi, \sigma, I, J, K). \end{aligned} \tag{6}$$

To calculate the reversal and transposition distance in the COP model we implemented the *rev_trans_dist_cop/3* predicate (7), which set the variables B using the upper bound and constrains the permutations by making $\pi_k = \pi_{k-1}\rho_k$. The predicate *length/2* is a prolog built-in and is used to create a list of non instantiated variables of a given size. The cost function $Cost$ is the sum of variables B associated with each ρ_k, $Cost = \sum_{k=1}^{UB} B_k$, where UB is a known upper bound. The reversal and transposition distance is the minimum value of the cost function $d_{r\_t} = \min Cost$. To avoid unnecessary processing, the value of $Cost$ must be greater or equal to any lower bound.

$$\begin{aligned} &rev\_trans\_dist\_cop(\pi, N, Model) \text{ :-} \\ &\quad bound(\pi, Model, LB, UB), \\ &\quad length(B, UB), \\ &\quad upperbound\_constraint\_rev\_trans(\pi, B, Model, UB), \\ &\quad sum(B, Cost), \\ &\quad Cost \geq LB, \\ &\quad minimize(Cost, N). \end{aligned} \tag{7}$$

The *upperbound_constraint_rev_trans/4* predicate (8) applies the effects of ρ_k in permutation and returns the value of B for every reversal or transposition

ρ_k. An important constraint is to check if it is possible to sort the permutation using the remaining number of reversals and transpositions to avoid unnecessary processing. The *event_cop/3* predicate chooses the best event between reversal, using the *reversal_cop/5* predicate (5), and transposition, using the *transposition_cop/6* predicate (6), to minimize the distance.

$$
\begin{aligned}
&upperbound\_constraint\_rev\_trans(\iota, [\,], \_Model, \_UB).\\
&upperbound\_constraint\_rev\_trans(\pi, [B|Bt], Model, UB) :-\\
&\quad event\_cop(\pi, \sigma, B),\\
&\quad bound(\pi, Model, LB, \_UB),\\
&\quad UB \geq LB,\\
&\quad upperbound\_constraint\_rev\_trans(\sigma, Bt, Model, UB - 1),
\end{aligned}
\tag{8}
$$

The COP models have the above structure changing only the bounds we used. We use the same CSP bounds modified for COP models. So we have the following bounds: **def_cop**, **r_t_br_cop** and **r_t_bc_cop**.

3 Computational Experiments

All the constraint logic programming models were implemented using the open source programming system *ECLiPSe* [2] and the proprietary C++ package *IBM® ILOG® CPLEX® CP Optimizer* [3]. We recommend Apt and Wallace [5] and Marriott and Stuckey [16] as introduction for CLP using *ECLiPSe*.

All the integer programming formulations were implemented using the open source system *GLPK*[1], the models were written in *GNU MathProg modeling language* intended for describing linear mathematical programming models, and the proprietary C++ package *IBM® ILOG® CPLEX® Optimizer*[4].

We carried our tests on a microcomputer equipped with Intel® Core™ 2 Duo 2.33GHz, with 3 GB of RAM, running under a Ubuntu Linux operating system with kernel 2.6.31, *ECLiPSe-6.0*, *GLPK-4.35*, *IBM ILOG CPLEX CP Optimizer v 2.3* and *IBM ILOG CPLEX Optimizer v 12.1*.

The CLP results used the model described in previous section. The ILP results used the formulations described in Dias and Souza [13].

The Table 1 summarize the results. The column **size** represents the length of permutations used in the tests. The CPU times (in seconds) reported refers to an average of 50 instances where the permutation π was randomly generated. In all models we made a comparison between the permutation π and the identity permutation ι. The times are given in seconds and grow very fast as the instance size increases. This behavior occur due to the exponential search space, in case of CLP models, and occur due to model sizes, in case of ILP formulations. The timeout is printed when the model could not finish the entire tests within a time limit of 25 hours. Instances with $|\pi| \geq 13$ printed timeout in all models, except for the model **r_t_bc_csp**.

Table 1. Average time (in seconds) for sorting by reversals and transpositions models. The "-" means that the model could not finish the entire data test within a time limit of 25 hours.

Reversals and Transpositions Models															
		CP						ILOG CP						ILP	
size	ECLiPSe													GLPK	ILOG CPLEX
	def_cop	r.t_br_cop	r.t_bc_cop	def_csp	r.t_br_csp	r.t_bc_csp	def_cop	r.t_br_cop	r.t_bc_cop	def_csp	r.t_br_csp	r.t_bc_csp		
3	0.034	0.012	0.003	0.004	0.003	0.002	0.004	0.002	0.001	0.004	0.002	0.003	0.001	0.002
4	7.370	12.288	0.341	0.028	0.007	0.004	0.008	0.008	0.007	0.012	0.009	0.005	0.001	0.012
5	-	-	26.047	0.343	0.020	0.010	0.026	0.024	0.021	0.031	0.021	0.013	0.396	0.055
6	-	-	409.079	16.742	0.122	0.031	0.268	0.232	0.103	0.085	0.066	0.046	4.062	0.808
7	-	-	-	593.666	0.670	0.104	1.896	1.967	1.179	0.533	0.400	0.255	3.660	94.429
8	-	-	-	-	2.579	0.149	12.851	10.589	5.566	3.088	2.655	1.531	-	-
9	-	-	-	-	13.958	0.339	468.581	422.396	102.687	61.973	60.465	19.984	-	-
10	-	-	-	-	64.208	1.318	-	-	-	-	-	1189.290	-	-
11	-	-	-	-	167.423	3.327	-	-	-	-	-	-	-	-
12	-	-	-	-	1058.050	11.044	-	-	-	-	-	-	-	-
13	-	-	-	-	-	20.961	-	-	-	-	-	-	-	-
14	-	-	-	-	-	51.294	-	-	-	-	-	-	-	-
15	-	-	-	-	-	164.994	-	-	-	-	-	-	-	-
16	-	-	-	-	-	188.704	-	-	-	-	-	-	-	-
17	-	-	-	-	-	1046.984	-	-	-	-	-	-	-	-

We can see that CLP models based on CSP theory have better times in comparison to ILP formulation and the CLP models based on COP theory have the worst times. In terms of CLP models the model **r_t_bc_csp** have the best times (especially in large instances), due to breakpoint bounds with transposition edge-colored cycle graph bounds. This is an expected result if we consider the fact that the cycle graph lower bound is tighter than the breakpoint lower bound. Another interesting fact is that the ILOG models were faster than ECLiPSe at first, but in the end the ILOG models became slower. In terms of ILP models we can see that GLPK models, in average, had better times than ILOG models.

We can observe that even with the ILP polynomial model, the times grows very fast due to the order of polynomials, making the ILP models prohibitive in practice [13].

Note that in the experiment the CLP model based on COP theory have the worst CPU time. We did not use any heuristic that could improve the results, but the search mechanism of COP models is to find a solution and try to improve it by looking for new solutions with improved value of the cost function. In the end it will lead to a greater space search than CSP models that use a bottom-up strategy.

4 Conclusion and Future Works

In this paper we introduced a Constraint Logic Programming model for sorting by reversals and transpositions, based on Constraint Satisfaction Problem theory and Constraint Optimization Problem Theory. We made a comparison between the CLP models and the ILP polynomial models described in Dias and Souza [13]. The analysis shows that the CLP models based on CSP theory achieved better performance than ILP formulations.

This approach is still not viable in practice. For future works we plan to improve the models with better bounds for the problem, reducing the search space for them, or with some procedure to choose a set of reversals or transpositions that will be branched firstly.

In terms of ILP models we plan to improve the bounds of models with techniques like *Lagrangian Relaxation* or writing a new model in order to apply *Column Generation* or *Branch-and-Cut* [18, 21].

Acknowledgments. This work was supported by CNPq (472504/2007-0, 479207/2007-0, 483177/2009-1). Victor de Abreu Iizuka receives a MSc scholarship from CNPq.

References

1. GNU Linear Programming Kit. `http://www.gnu.org/software/glpk/` (2010)
2. The ECLiPSe Constraint Programming System. `http://www.eclipseclp.org/` (April 2011)
3. IBM® ILOG® CPLEX® CP Optimizer. `http://www-01.ibm.com/software/integration/optimization/cplex-cp-optimizer/` (April 2011)

4. IBM® ILOG® CPLEX® Optimizer. http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/ (April 2011)
5. Apt, K., Wallace, M.: Constraint Logic Programming using Eclipse. Cambridge University Press, New York, NY, USA (2007)
6. Bafna, V., Pevzner, P.A.: Sorting by Reversals: Genome Rearrangements in plant and organelles and evolutionary history of X chromossome. Molecular Biology and Evolution 12(2), 239–246 (1995)
7. Bafna, V., Pevzner, P.A.: Genome rearrangements and sorting by reversals. SIAM Journal on Computing 25(2), 272–289 (1996)
8. Bafna, V., Pevzner, P.A.: Sorting by transpositions. SIAM Journal on Discrete Mathematics 11(2), 224–240 (1998)
9. Berman, P., Hannenhalli, S., Karpinski, M.: 1.375-approximation algorithm for sorting by reversals. In: Proceedings of the 10th Annual European Symposium on Algorithms (ESA'02). pp. 200–210. Springer-Verlag, London, UK (2002)
10. Bulteau, L., Fertin, G., Rusu, I.: Sorting by transpositions is difficult. Computing Research Repository abs/1011.1157 (2010)
11. Caprara, A.: Sorting by reversals is difficult. In: Proceedings of the first annual international conference on Computational molecular biology (RECOMB'97). pp. 75–83. ACM, New York, NY, USA (1997)
12. Dias, U., Dias, Z.: Constraint programming models for transposition distance problem. Lecture Notes on Bioinformatics 5676, 13–23 (2009)
13. Dias, Z., de Souza, C.: Polynomial-size ILP models for rearrangements distance problems. In: Proceedings of the Brazilian Symposium on Bioinformatics (BSB'2007). pp. 74–85 (2007)
14. Elias, I., Hartman, T.: A 1.375-approximation algorithm for sorting by transpositions. IEEE/ACM Transactions on Computational Biology Bioinformatics 3(4), 369–379 (2006)
15. Lin, G.H., Xue, G.: Signed genome rearrangement by reversals and transpositions: Models and approximations. Lecture Notes in Computer Science 1627, 71–80 (1999)
16. Marriott, K., Stuckey, P.J.: Programming with constraints: an introduction. MIT Press (1998)
17. Meidanis, J., Walter, M.E.M.T., Dias, Z.: A lower bound on the reversal and transposition diameter. Journal of Computational Biology 9(5) (2002)
18. Nemhauser, G., Wolsey, L.: Integer and Combinatorial Optimization. Wiley-Interscience (1988)
19. Palmer, J.D., Herbon, L.A.: Plant mitochondrial DNA evolves rapidly in structure, but slowly in sequence. Journal of Molecular Evolution 27, 87–97 (1988)
20. Walter, M.E.M.T., Dias, Z., Meidanis, J.: Reversal and transposition distance of linear chromosomes. In: Proceedings of the String Processing and Information Retrieval (SPIRE'98) (1998)
21. Wolsey, L.: Integer Programming. Wiley-Interscience (1998)

A Flexible Framework for Computing Rearrangement Distance of Every Permutation in the Symmetric Group

Gustavo Rodrigues Galvão and Zanoni Dias

University of Campinas, Institute of Computing, Brazil
gustavo.galvao@students.ic.unicamp.br
zanoni@ic.unicamp.br

Abstract. We consider the problem of computing rearrangement distance of every permutation in the symmetric group and present a framework for doing it. The framework can be easily adapted to compute the rearrangement distance of all permutations in the symmetric group regarding any set of rearrangement operations that acts over a single permutation, thus it could be used by anyone who needs to run experiments on approximation algorithms or heuristics for permutation sorting problems.

1 Introduction

Since it was shown that rearrangement operations, such as reversals and transpositions, are more appropriate to analyze genome evolution than sequence comparison [14], researchers from Computational Biology have been addressing the problem of finding a shortest sequence of operations that transform the genome of one specie into another. Representing the order of the genes in a genome as permutations, the previous problem can be equivalently stated as the combinatorial problem of sorting a permutation using rearrangement operations.

In general, it is computationally difficult to compute the minimum number of operations that sorts a permutation (for instance, sorting unsigned permutations by reversals and sorting unsigned permutations by transpositions are NP-hard problems [2, 3]). Consequently, the best known solutions for these problems are mostly approximations algorithms [1, 6, 7, 9, 15]. It means that if we want to compute the exact rearrangement distance of a given permutation of size n, we will have to perform an exhaustive search on the space of all permutations of size n, probably using some method to reduce the search space, such as a branch-and-bound algorithm.

In this work we present a framework which, given a set of rearrangement operations, computes the rearrangement distance of all permutations of size n. This framework may be useful for anyone who needs to run experiments on approximation algorithms or heuristics for permutation sorting problems, as it done by Walter *et al.* [17] and Dias and Dias [5].

The rest of this paper is organized as follows. In the next section, we give the definitions and notations used in this work. The algorithm for computing rearrangement distance of every permutation in S_n is given in Sect. 3. Section 4 contains some remarks on the implementation of the framework and on its performance. The final section concludes the paper.

2 Definitions and Notation

A genome is composed by chromosomes, which are represented as a n-tuple, where each element represents a gene. Assuming there are no duplicated genes, this n-tuple is a permutation $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_n)$, where $1 \leq |\pi_i| \leq n$ and $|\pi_i| \neq |\pi_j|$ if $i \neq j$. If the orientation of the genes is known, then each π_i has a sign, $+$ or $-$, which indicates its orientation. A permutation whose elements are signed is called a signed permutation, otherwise it is called a unsigned permutation (we distinguish between signed and unsigned permutations when it is pertinent).

A conservative rearrangement operation occurs when a chromosome is broken in segments and then is joined in such a way that the genes remain unaltered, but its order or orientation possibly do not (if the rearrangement operation is not conservative, then the set of genes of the chromosome may be altered). Given a permutation π, which represents a chromosome, and an operation ρ, we represent ρ occurring over π by $\rho \cdot \pi$.

In this work, we just consider unichromosomal genomes and we are only interested in conservatives operations, such as reversals and transpositions. For more background in such operations, the reader is referred to [8].

Given two permutations π and σ, transforming π into σ consists in finding a sequence of operations $\rho_1, \rho_2, \ldots, \rho_t$ such that $(\rho_t \cdots (\rho_2 \cdot (\rho_1 \cdot \pi))) = \sigma$. When this sequence is of minimum length, we say that its size is the rearrangement distance between π and σ and we denote it by $d(\pi, \sigma)$. As it is equivalent to the distance between $\pi\sigma^{-1}$ and the identity permutation $I(n) = (1, 2, \ldots, n)$, the problem of transforming a permutation into another reduces to the problem of sorting a permutation. We denote the distance between a permutation π and $I(n)$ by $d(\pi)$.

The symmetric group S_n is the group of all permutations of $\{1, 2, \ldots, n\}$. By abuse of notation, we also denote the group of all permutations of $\{\pm 1, \pm 2, \ldots, \pm n\}$ by S_n. A slice of S_n is the subset $S_n^i = \{\pi : d(\pi) = i \text{ and } \pi \in S_n\}$, $0 \leq i \leq D(n)$, where $D(n)$ is the greatest distance between two permutations of size n. Note that the slices of S_n depend on the set of operations acting on the permutations in it.

The sign value of a permutation π, denoted by $s(\pi)$, is the number given by the binary representation of its signs, with 0 and 1 standing for $+$ and $-$ respectively. For instance, 1100 is the binary representation of the signs of permutation $\pi = (-1\ -2\ 3\ 4)$, thus $s(\pi) = 12$. Note that $s(\sigma) = 0$ for any unsigned permutation σ.

3 Algorithm for Computing Rearrangement Distances

The simplest way of computing rearrangement distance of all permutation in S_n is the straightforward one: initialize a permutation queue Q with the identity permutation $I(n)$ and set its distance to 0. While Q is not empty, remove a permutation π from Q, report π and $d(\pi)$, and compute all permutations that can be generated from π by applying every possible operation under consideration on it. The ones that have not been generated yet are added to Q and their distances are set to $d(\pi) + 1$.

Besides its exponential time complexity, which is inherent to the computation of rearrangement distances of all permutations in S_n, it is expected that the algorithm also has exponential space complexity, because the generation rate of new permutations is much larger than the removal rate from the queue (for instance, there are $\binom{n}{2}$ possible reversals and $\binom{n}{3}$ possible transpositions to apply on a permutation of size n). Therefore, the main concern when implementing such algorithm is how to represent permutations in the most concise form.

By the definition of a permutation, one concludes that a permutation in S_n uses $O(n)$ data units to be stored (one data unit per element). The problem is that using a data unit to store an element of a permutation incur a reasonable waste of space. Assuming that a data unit has b bits and $n = 2^{b'}$, $2 \leq b' \leq b$, the bit waste is given by $\sum_{i=1}^{b'} (2^i - 2^{i-1})(b - i) \geq 2^{b'} - 2$, which means that the bit waste is $\Omega(n)$. Thus, it is necessary to consider another approach for representing permutations.

A possible approach would be to represent a permutation of size n as a unique natural number in $\{0, \ldots, n! - 1\}$; then each permutation in S_n would require $O(\lceil \frac{\lg n!}{b} \rceil)$ data units to be stored. Since it is assumed that $n = 2^{b'}$, we conclude that a permutation would need $O(\lceil \frac{b'n}{b} \rceil)$ data units to be stored. Although this approach is asymptotically equivalent to the former, it is not hard to realize that it is better in practice. Furthermore, the bit waste is at most $b - 1$.

It remains to be seen how to construct a bijective function $f : S_n \to \{0, \ldots, n! - 1\}$. Myrvold and Ruskey [13] presented simple *ranking* and *unranking* algorithms for permutations of size n that runs in $O(n)$ time. A ranking algorithm assigns a unique integer number in the range $[0, n! - 1]$ to a unsigned permutation of size n, while the corresponding unranking algorithm performs the inverse. For signed permutations of size n, one can assign a unique integer in the range $[0, n!2^n - 1]$ to it as follows: compute the rank of the permutation as if it was not signed; multiply the rank by 2^n and then sum the result with $s(\pi)$. With both ranking and unranking algorithms in hand, it is possible to present the algorithm to compute rearrangement distances of all permutations in S_n (Algorithm 1).

Observing the algorithm sketched at the beginning of this section, we conclude that it would need two data structures: a queue, for storing the generated permutations, and an oracle, for both answering if a given permutation had already been generated and what is the distance of a given permutation. As a permutation π can be represented as a unique non negative integer i, we imple-

mented the oracle as a vector (V). All elements of the vector are initialized to -1. When π is placed into the queue, the i-th entry of the vector is set to $d(\pi)$. Similarly, the queue is also implemented as a vector (Q) and the permutations are both retrieved and placed in it following the generation order. It is achieved by using two auxiliary variables: one indicating which is the next permutation to be processed ($firstElement$) and the other one indicating in which position the next generated permutation must be placed ($lastElement$).

Data: n, the size of permutations
Result: Output all permutations of size n and its distances

```
1  Let Q and V be two vectors of size |S_n|;
2  Initialize V such that V[i] = −1, i ∈ {1, 2, ..., |S_n|};
3  r ← rank(I(n));
4  Q[0] ← r;
5  V[r] ← 0;
6  firstElement ← 1;
7  lastElement ← 2;
8  while lastElement ≤ |S_n| do
9      d ← V[Q[firstElement]];
10     π ← unrank(Q[firstElement]);
11     firstElement ← firstElement + 1;
12     Print out π and d;
13     for every possible combination of operation ρ on π do
14         r ← rank(ρ · π);
15         if V[r] = −1 then
16             Q[lastElement] = r;
17             lastElement ← lastElement + 1;
18             V[r] = d + 1;
19         end
20     end
21 end
```

Algorithm 1: Computing rearrangement distances

Regarding the complexity of the Algorithm 1, each of lines 1, 4-7, 9, 11-12, and 16-18 takes constant time; line 3 and line 10 takes $O(n)$ time; the while loop of lines 8-20 is executed $O(|S_n|)$ times; finally, the for loop of lines 13-20 takes $O(p(n))$ time, where $p(n)$ is the least degree polynomial which is an asymptotically upper bound for the sum of combinations of the operations being considered. Therefore, Algorithm 1 runs in $O(p(n)|S_n|)$ time, which means that it runs in exponential time on the size of permutations, but in polynomial time on the number of permutations.

Notice that, in order to use vectors, one must allocate all the memory needed to store every permutation in S_n since the maximum queue size is not known. Therefore, one could say that a vector is not the best data structure for implementing the queue, because the latter is dynamic and it may be the case that

just a small number (compared to $|S_n|$) of permutations is waiting to be processed. Firstly, one must observe that using dynamic data structures (like linked lists) for implementing a queue implies allocating extra space for the references between units (nodes) of the queue. Secondly, in the specific case of computing rearrangement distances of every permutation in S_n, it must be seen that, at some point of the algorithm, the queue will be storing, at least, as much permutations as there are in a slice of S_n. As one can verify in a recent work regarding distribution of rearrangement distance in S_n [11], it is common that more than 50% of all permutations of size n are in just one slice of S_n. Combining these two facts, one realizes that using a dynamic data structure for implementing the queue would not bring a significant improvement and, in some cases, it would require more memory than a vector.

Finally, we remark on the possibility of parallelization of Algorithm 1. It could be achieved by encapsulating the while loop in a thread and then creating multiple threads. As a result, it would create a race condition on vectors Q and V, as well as on the variables $indexFirst$ and $indexLast$. Thus, it would be necessary to synchronize all threads, avoiding that two or more of them write on the same common data at the same time (this could be accomplished by using a binary semaphore). But this could lead to a scenario where just one thread is running and the other ones are blocked, waiting for the former to use the common data. In order to overcome this, we could make the algorithm generate all possible permutations and store them in vector, say P, before updating the queue Q. If each thread has its own instance of vector P, one minimizes the race condition on the common data, because while some thread is accessing it, the other ones may be filling their instance of vector P.

Besides preventing race condition, we must also guarantee that all threads work at the same slice of S_n in order to maintain the queue coherent. We say that a thread is *working* at a slice S_n^i if it is generating permutations from a permutation $\pi \in S_n^i$. Therefore, as long as there are threads working at slice S_n^i, any thread working at slice S_n^{i+1} must wait until those threads start working at slice S_n^{i+1} before trying to update the queue.

4 Implementation and Discussion

We implemented two versions of the framework: in one version we represented each permutation as an unsigned integer of 32 bits and in the other, as an unsigned integer of 64 bits. That is, the former can handle all unsigned permutations of up to 12 elements and all signed permutations of up to 10 elements, and the latter can handle all unsigned permutations of up to 20 elements and all signed permutations of up to 16 elements. Thus, the 64 bit version can handle more permutations, but needs twice as memory. Both versions were implemented in C, using *pthread* library for dealing with threads. The source code is available for download [10].

Both versions of the framework support a number of sets of operations considered in the literature for generating all permutations of size n and computing its

distances from $I(n)$, such as: reversals [1]; prefix reversals [9]; transpositions [7]; prefix transpositions [6]; signed reversals and transpositions [16]; prefix reversals and prefix transpositions [15]. Furthermore, given the flexibility of the algorithm for computing the rearrangement distances, we remark that it is not hard to extend the framework and make it support any other set of operations that act over a single permutation.

Dias and Meidanis [6] computed the prefix transposition distance of all unsigned permutations of up to 11 elements and stated that their method would need 30GB of physical memory to compute prefix transposition distances of all unsigned permutations of 12 elements, what made the computation for $n = 12$ impossible (they used a computer with 8GB of RAM). Our 32 bit implementation needs 2.4GB of physical memory to accomplish it, what make the computation for $n = 12$ possible in almost any desktop computer nowadays (and even in the computer they used). One can estimate the memory usage (in bytes) of the 32 and the 64 bit implementations by multiplying $|S_n|$ by 5 and 9 respectively.

There are further publications where the rearrangement distance of all permutations (signed or unsigned) of up to n elements were computed for a particular set of operations [4,5,12,17], but they do not contain a detailed description of the algorithm used to compute it nor any remark on its performance, therefore we are not able to present a comparison between our approach and theirs. Nevertheless, since such publications do not present the rearrangement distance of permutations with more than 11 elements, we infer that the methods used in them for computing rearrangement distances had reached a limit very much like the method used by Dias and Meidanis [6] had.

Although one can choose the number of threads used to run the implementations (we advise that this number do not exceed the number of cores of the CPU), we remark that the speed gain is not linear on the number of threads. We ran some tests for different sets of operations and observed that the speed gain reaches a limit, which is due to the race condition discussed in Sect. 3. It is directly related to the degree of $p(n)$: the lower the degree, the greater the speed gain. Besides, we also observed that the speed gain is not easy predictable, that is, it is possible to obtain different speed gains by running one implementation with the same parameters (i.e. set of operations and number of threads). It happens because the threads basically run at random, thus the degree of parallelization may vary considerably.

We executed the 32 bit implementation 10 times for each pair (n, t), where n is the size of the permutations and t is the number of threads. Figure 1 shows the average speed gain obtained. The source code was compiled with gcc version 4.5.0 and the resulting program was executed on a desktop PC featuring 16 Intel Xeon(R) CPU E5520 at 2.27GHz and 64GB of RAM running GNU/Linux 2.6.34.

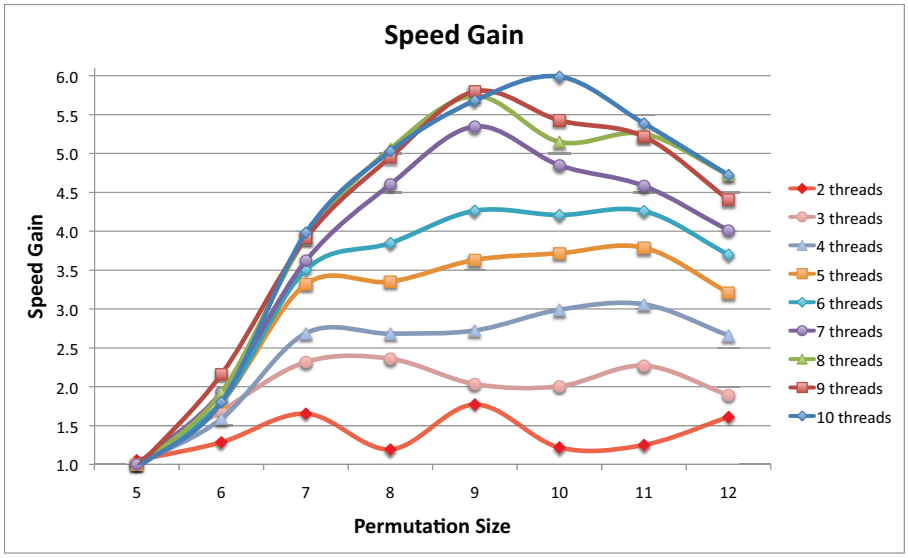

Fig. 1. Average speed gain on the execution of the 32 bit implementation with multiple threads. The set of operations chosen covers reversals only, thus $p(n) = \binom{n}{2}$.

5 Concluding Remarks

We considered the problem of computing the distance of all permutations in S_n and presented a framework for solving it. It basically performs a breadth first search in S_n, starting at $I(n)$. As we discussed, the framework runs in exponential time on the size of permutations, but in polynomial time on the number of permutations. We showed how to reduce its memory usage by representing each permutation as an unique natural number and how its performance can be enhanced by means of parallelization.

This framework was used in a recent work [11] to investigate the rearrangement distance distribution in S_n regarding reversals and transpositions. Moreover, it could be used by anyone who needs to run experiments on approximation algorithms or heuristics for permutation sorting problems.

Acknowledgments. This work was supported by CNPq (processes 483177/2009-1, 200815/2010-5 and 135212/2010-3).

References

1. Berman, P., Hannenhalli, S., Karpinski, M.: 1.375-approximation algorithm for sorting by reversals. In: Proceedings of the 10th Annual European Symposium on Algorithms (ESA'2002). pp. 200–210. Springer-Verlag, Rome, Italy (2002)
2. Bulteau, L., Fertin, G., Rusu, I.: Sorting by transpositions is difficult. The Computing Research Repository abs/1011.1157 (2010)
3. Caprara, A.: Sorting by reversals is difficult. In: Proceedings of the 1st Annual International Conference on Computational Molecular Biology (RECOMB'97). pp. 75–83. ACM Press, Santa Fe, New Mexico, United States (1997)
4. Cohen, D.S., Blum, M.: On the problem of sorting burnt pancakes. Discrete Applied Mathematics 61(2), 105–120 (1995)
5. Dias, U., Dias, Z.: Extending Bafna-Pevzner algorithm. In: Proceedings of the International Symposium on Biocomputing (ISB'10). pp. 23:1–23:8. ACM Press, Calicut, Kerala, India (2010)
6. Dias, Z., Meidanis, J.: Sorting by prefix transpositions. In: Proceedings of the 9th International Symposium on String Processing and Information Retrieval (SPIRE'2002). pp. 65–76. Springer-Verlag, Lisbon, Portugal (2002)
7. Elias, I., Hartman, T.: A 1.375-approximation algorithm for sorting by transpositions. IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB) 3(4), 369–379 (2006)
8. Fertin, G., Labarre, A., Rusu, I., Tannier, E., Vialette, S.: Combinatorics of Genome Rearrangements. The MIT Press (2009)
9. Fischer, J., Ginzinger, S.W.: A 2-approximation algorithm for sorting by prefix reversals. In: Proceedings of the 13th Annual European Symposium on Algorithms (ESA'2005). pp. 415–425. Springer, Heidelberg, Mallorca, Spain (2005)
10. Galvão, G.R., Dias, Z.: Rearrangement distance database. `http://mirza.ic.unicamp.br:8080/bioinfo`
11. Galvão, G.R., Dias, Z.: Computing rearrangement distace of every permutation in the symmetric group. In: Proceedings of the 26th ACM Symposium on Applied Computing (SAC'2011). pp. 106–107. ACM Press, Taichung, Taiwan (2011)
12. Labarre, A.: Combinatorial aspects of genome rearrangements and haplotype networks. Ph.D. thesis, Université Libre de Bruxelles, Brussels, Belgium (2008)
13. Myrvold, W., Ruskey, F.: Ranking and unranking permutations in linear time. Information Processing Letters 79(6), 281–284 (2001)
14. Palmer, J.D., Herbon, L.A.: Plant mitochondrial dna evolved rapidly in structure, but slowly in sequence. Journal of Molecular Evolution 28(1-2), 87–97 (1988)
15. Sharmin, M., Yeasmin, R., Hasan, M., Rahman, A., Rahman, M.S.: Pancake flipping with two spatulas. Electronic Notes in Discrete Mathematics 36, 231–238 (2010)
16. Walter, M., Dias, Z., Meidanis, J.: Reversal and transposition distance of linear chromosomes. In: Proceedings of the 5th International Symposium on String Processing and Information Retrieval (SPIRE'1998). pp. 96–102. IEEE Computer Society, Santa Cruz, Bolivia (1998)
17. Walter, M.E.M.T., Sobrinho, M.C., Oliveira, E.T.G., Soares, L.S., Oliveira, A.G., Martins, T.E.S., Fonseca, T.M.: Improving the algorithm of Bafna and Pevzner for the problem of sorting by transpositions: a practical approach. Journal of Discrete Algorithms 3(2-4), 342–361 (2005)

On the Distribution of Rearrangement Distances

Gustavo Rodrigues Galvão and Zanoni Dias

University of Campinas, Institute of Computing, Brazil
gustavo.galvao@students.ic.unicamp.br, zanoni@ic.unicamp.br

Abstract. We analyze the rearrangement distance distribution in the symmetric group regarding sets of rearrangement operations that cover reversals or transpositions and present or reinforce conjectures on its diameter, traversal diameter and longevity.

1 Introduction

In genome rearrangements, the evolutionary distance between two genomes is assumed to be closely related to the rearrangement distance between them, which is the size of the minimum length sequence of rearrangement operations that transforms one genome into the other. Representing both genomes as permutations, where the genes appear as elements, and assuming that one of them is the identity permutation, one can obtain the rearrangement distance by computing the minimum number of rearrangement operations that sorts a permutation. For more background on genome rearrangements, the reader is referred to [8].

In this paper, we present and analyze the distribution of rearrangement distances in S_n and in $S_n^{\pm}$, extending a preliminary work [10]. Our main goal is to unveil patterns that could shed some light on diameter problems (see Sect. 2 for details), which are briefly reviewed below.

Bafna and Pevzner [1] proved that the reversal diameter of S_n is $n - 1$ and Meidanis, Walter and Dias [16] proved that the reversal diameter of $S_n^{\pm}$ is $n + 1$. Gates and Papadimitriou [11] gave a $\frac{5n}{3}$ upper bound on the prefix reversal diameter of S_n, further improved to $\frac{11n}{8}$ by Chitturi *et al.* [3]. Cohen and Blum [5] gave an $\frac{3n}{2}$ lower bound and a $2n - 2$ upper bound on the prefix reversal diameter of $S_n^{\pm}$. Bafna and Pevzner [2] proved a $\lceil \frac{n}{2} \rceil$ lower bound and a $\lfloor \frac{3n}{4} \rfloor$ upper bound on transposition diameter of S_n. The former was improved to $\frac{17n}{33} + \frac{1}{33}$ by Lu and Yang [15] and the second was improved to $\lfloor \frac{2n-2}{3} \rfloor$ by Eriksson *et. al* [7]. Meidanis and Dias [6] showed that the prefix transposition diameter of S_n lies between $\frac{n}{2}$ and $n - 1$. Chitturi and Sudborough [4] improved the upper bound to $n - \log_8 n$ and Labarre [14] improved the lower bound to $\lfloor \frac{3n+1}{4} \rfloor$. Meidanis, Walter and Dias [17] demonstrated that $\lfloor \frac{n}{2} \rfloor + 2$ is a lower bound of the reversal and transposition diameter of $S_n^{\pm}$.

The rest of this paper is organized as follows. In the next section, we give the definitions and notations used in this work. The distribution of rearrangement distances computed for different set of operations are presented in Sect. 3 and analyzed in Sect. 4. The final section concludes the paper.

2 Definitions and Notation

The symmetric group S_n is the group of all permutations of $\{1, 2, \ldots, n\}$ and the symmetric group (also known as hyperoctahedral group) $S_n^{\pm}$ is the group of all permutations of $\{\pm 1, \pm 2, \ldots, \pm n\}$.

A reversal $r(i, j)$, $1 \leq i \leq j \leq n$, is an operation that acts over a permutation $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_n)$ in such a way that $r(i, j) \cdot (\pi_1\ \ldots\ \pi_{i-1}\ \underline{\pi_i\ \ldots\ \pi_j}\ \pi_{j+1}\ \ldots\ \pi_n) = (\pi_1\ \ldots\ \pi_{i-1}\ \underline{\pi_j\ \ldots\ \pi_i}\ \pi_{j+1}\ \ldots\ \pi_n)$. If the permutation is signed, then the respective signs are also reversed. The reversal $r(1, j)$ is said to be a prefix reversal.

A transposition $t(i, j, k)$, $1 \leq i < j < k \leq n + 1$, is an operation that acts over a permutation $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_n)$ in such a way that $t(i, j, k) \cdot (\pi_1\ \ldots\ \pi_{i-1}\ \underline{\pi_i\ \ldots\ \pi_{j-1}}\ \underline{\pi_j\ \ldots\ \pi_{k-1}}\ \pi_k\ \ldots\ \pi_n) = (\pi_1\ \ldots\ \pi_{i-1}\ \underline{\pi_j\ \ldots\ \pi_{k-1}}\ \underline{\pi_i\ \ldots\ \pi_{j-1}}\ \pi_k\ \ldots\ \pi_n)$. A transposition does not change the sign of an element. The transposition $t(1, j, k)$ is said to be a prefix transposition.

Given two permutations π and σ, transforming π into σ consists in finding a sequence of operations $\rho_1, \rho_2, \ldots, \rho_t$ such that $(\rho_t \cdots (\rho_2 \cdot (\rho_1 \cdot \pi))) = \sigma$. When this sequence is of minimum length, we say that its size is the rearrangement distance between π and σ and we denote it by $d(\pi, \sigma)$. As it is equivalent to the distance between $\pi\sigma^{-1}$ and the identity permutation $I(n) = (1\ 2\ \ldots\ n)$, the problem of transforming a permutation into another reduces to the problem of sorting a permutation. We denote the distance between a permutation π and $I(n)$ by $d(\pi)$.

The greatest distance between two permutations in S_n is said to be the diameter of S_n and we denote it by $D(n)$. A slice of S_n is the subset $S_n^i = \{\pi |\ d(\pi) = i \text{ and } \pi \in S_n\}$, $0 \leq i \leq D(n)$. We say that $S_n^i \leq S_n^j$ if $|S_n^i| < |S_n^j|$ or if $|S_n^i| = |S_n^j|$ and $i \leq j$. Let S_n^t be the slice of S_n such that $S_n^i \leq S_n^t$, $0 \leq i \leq D(n)$. We define t as the traversal diameter of S_n and it is denoted by $T(n)$. We establish a relation between $D(n)$ and $T(n)$ by defining a function $L : \mathbb{N} \to \mathbb{N}$ such that $D(n) = T(n) + L(n)$. This function is said to be the longevity of S_n.

Note that the definitions of the diameter, of a slice, of the traversal diameter, and of the longevity of S_n extend naturally to $S_n^{\pm}$. Besides, their values may vary depending on the set of operations acting over the permutations.

3 Distribution of Rearrangement Distances

In a recent work [10], we computed the rearrangement distance distribution in S_1, S_2, ..., S_{12} and in $S_1^{\pm}$, $S_2^{\pm}$, ..., $S_9^{\pm}$ with respect to eight sets of operations that cover reversals or transpositions. Using the same algorithmic approach and using computers with more memory than the one we used, we computed the rearrangement distance distribution in S_{13} and in $S_{10}^{\pm}$ regarding nine sets of operations: seven of the eight sets considered previously [10] plus two other sets that cover transreversals, which were posed by Gu *et al.* [12] and Hartman and Sharan [13].

For reason of brevity, we present in tables 1 to 7 only the distributions related to the sets of operations cited in Sect. 1, which are, to the best of our knowledge, the most studied ones when it comes to diameter problems. The rest of them are available at a website [9]. As far as we know, such distributions had never been presented before.

In order to facilitate the analysis of diameter, traversal diameter and longevity of $S_n^{\pm}$ and S_n obtained from each table, we compiled them in Table 8 and Table 9 respectively.

Table 1. Reversal distance distribution in $S_n^{\pm}$

	n									
d	1	2	3	4	5	6	7	8	9	10
1	1	3	6	10	15	21	28	36	45	55
2		3	16	50	120	245	448	756	1200	1815
3		1	25	170	700	2170	5586	12600	25740	48675
4				145	1554	8820	35612	115254	318600	781968
5				8	1447	19495	138229	684525	2670025	8760653
6					3	15148	262688	2295786	13699020	63317771
7						180	202464	4198049	43476100	303929186
8							64	3006846	73952585	893848890
9								8067	51649124	1462987482
10									2120	981609816
11										604888

Table 2. Prefix reversal distance distribution in $S_n^{\pm}$

	n									
d	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2		2	6	12	20	30	42	56	72	90
3		2	12	36	80	150	252	392	576	810
4		1	18	90	280	675	1386	2548	4320	6885
5			6	124	680	2340	6230	14056	28224	51960
6			2	96	1214	6604	24024	68656	166740	359928
7				18	1127	12795	71568	276136	843822	2193534
8				3	389	15519	159326	901970	3636954	11738418
9					40	6957	222995	2195663	12675375	53257425
10					4	959	136301	3531887	33773653	198586153
11						43	21951	2743477	60758618	570362563
12						1	1021	562095	57953163	1138201788
13							15	24627	15244962	1282857296
14							1	347	701298	435390455
15								1	6721	22703532
16									51	179828
17									1	523
18										1

Table 3. Reversal and transposition distance distribution in $S_n^{\pm}$

	n									
d	1	2	3	4	5	6	7	8	9	10
1	1	4	10	20	35	56	84	120	165	220
2		3	33	157	518	1379	3178	6594	12624	22671
3			4	201	2334	14079	59420	199539	570642	1447413
4				5	952	28668	351053	2503693	12745196	51513506
5						1897	231248	6941233	93986201	773679172
6							136	670740	78428542	2611596863
7									51189	277631156
8										198

Table 4. Prefix reversal distance distribution in S_n

	n											
d	2	3	4	5	6	7	8	9	10	11	12	13
1	1	2	3	4	5	6	7	8	9	10	11	12
2		2	6	12	20	30	42	56	72	90	110	132
3		1	11	35	79	149	251	391	575	809	1099	1451
4			3	48	199	543	1191	2278	3963	6429	9883	14556
5				20	281	1357	4281	10666	22825	43891	77937	130096
6					133	1903	10561	38015	106461	252737	533397	1030505
7					2	1016	15011	93585	377863	1174766	3064788	7046318
8						35	8520	132697	919365	4126515	14141929	40309555
9							455	79379	1309756	9981073	49337252	184992275
10								5804	814678	14250471	118420043	639783475
11									73232	9123648	169332213	1525125357
12										956354	111050066	2183056566
13										6	13032704	1458653648
14											167	186874852
15												2001

Table 5. Prefix transposition distance distribution in S_n

	n											
d	2	3	4	5	6	7	8	9	10	11	12	13
1	1	3	6	10	15	21	28	36	45	55	66	78
2		2	14	50	130	280	532	924	1500	2310	3410	4862
3			3	55	375	1575	4970	12978	29610	61050	116325	208065
4				4	194	2598	18096	85128	308988	933108	2456256	5812092
5					5	562	15532	188386	1364710	7030210	28488724	96641974
6						3	1161	74183	1679189	19713542	148968371	827628815
7								1244	244430	11759676	242448896	2832043750
8									327	416845	56288493	2323157040
9										3	231058	141492748
10												31375

Table 6. Reversal distance distribution in S_n

	n											
d	2	3	4	5	6	7	8	9	10	11	12	13
1	1	3	6	10	15	21	28	36	45	55	66	78
2		2	15	52	129	266	487	820	1297	1954	2831	3972
3			2	55	389	1563	4642	11407	24600	48204	87758	150707
4				2	184	2539	16445	69863	228613	626677	1510973	3304457
5					2	648	16604	169034	1016341	4398136	15240603	44997104
6						2	2111	105365	1686534	14313789	81531167	354920337
7							2	6352	654030	16584988	198802757	1489090761
8								2	17337	3900116	159650162	2717751441
9									2	42878	22073230	1499706234
10										2	102050	116855950
11											2	239756
12												2

Table 7. Transposition distance distribution in S_n

	n											
d	2	3	4	5	6	7	8	9	10	11	12	13
1	1	4	10	20	35	56	84	120	165	220	286	364
2		1	12	68	259	770	1932	4284	8646	16203	28600	48048
3			1	31	380	2700	13467	52512	170907	484440	1231230	2864719
4					45	1513	22000	191636	1183457	5706464	22822293	78829491
5							2836	114327	2010571	21171518	157499810	910047453
6									255053	12537954	265819779	3341572727
7											31599601	1893657570
8												427

Table 8. Diameter, traversal diameter and longevity of $S_n^{\pm}$

Table 1				**Table 2**				**Table 3**			
n	**D(n)**	**T(n)**	**L(n)**	**n**	**D(n)**	**T(n)**	**L(n)**	**n**	**D(n)**	**T(n)**	**L(n)**
2	3	2	1	2	4	3	1	2	2	1	1
3	3	3	0	3	6	4	2	3	3	2	1
4	5	3	2	4	8	5	3	4	4	3	1
5	6	4	2	5	10	6	4	5	4	3	1
6	7	5	2	6	12	8	4	6	5	4	1
7	8	6	2	7	14	9	5	7	6	4	2
8	9	7	2	8	15	10	5	8	6	5	1
9	10	8	2	9	17	11	6	9	7	5	2
10	11	9	2	10	18	13	5	10	8	6	2

Table 9. Diameter, traversal diameter and longevity of S_n

Table 4			
n	**D(n)**	**T(n)**	**L(n)**
2	1	1	0
3	3	2	1
4	4	3	1
5	5	4	1
6	7	5	2
7	8	6	2
8	9	7	2
9	10	8	2
10	11	9	2
11	13	10	3
12	14	11	3
13	15	12	3

Table 5			
n	**D(n)**	**T(n)**	**L(n)**
2	1	1	0
3	2	1	1
4	3	2	1
5	4	3	1
6	5	3	2
7	6	4	2
8	6	4	2
9	7	5	2
10	8	6	2
11	9	6	3
12	9	7	2
13	10	7	3

Table 6			
n	**D(n)**	**T(n)**	**L(n)**
2	1	1	0
3	2	1	1
4	3	2	1
5	4	3	1
6	5	3	2
7	6	4	2
8	7	5	2
9	8	5	3
10	9	6	3
11	10	7	3
12	11	7	4
13	12	8	4

Table 7			
n	**D(n)**	**T(n)**	**L(n)**
2	1	1	0
3	2	1	1
4	3	2	1
5	3	2	1
6	4	3	1
7	4	3	1
8	5	4	1
9	5	4	1
10	6	5	1
11	6	5	1
12	7	6	1
13	8	6	2

4 Conjectures

As one can verify, the conjecture of Dias and Meidanis [6] on prefix transposition diameter of S_n, which is $n - \lfloor \frac{n}{4} \rfloor$, holds for $n = 13$. This is an important result because Eriksson *et al.* [7] showed that the first deviation of the transposition diameter of S_n from its original conjecture occurred when $n = 13$. Moreover, the conjecture on reversal and transposition diameter of $S_n^{\pm}$ [10] holds for $n = 10$ and the conjectures on traversal diameter and longevity of S_n and $S_n^{\pm}$ [10] hold for $n = 13$ and $n = 10$ respectively.

Once the transposition distance distribution in S_{13} and the reversal distance distribution in $S_{10}^{\pm}$ are known, it seems reasonable to conjecture that the transposition traversal diameter of S_n is $\lfloor \frac{n}{2} \rfloor$ and that the prefix reversal traversal diameter of $S_n^{\pm}$ is $\lfloor \frac{5n+2}{4} \rfloor$. All the conjectures and known results regarding $D(n)$,

$T(n)$ and $L(n)$ of S_n and $S_n^{\pm}$ are assembled together in Table 10 along with the respective reference if the conjecture or result was not provided by us.

Table 10. Conjectures and known results (assigned with an asterisk) regarding $D(n)$, $T(n)$ and $L(n)$ of S_n and $S_n^{\pm}$.

Set of Operations	**Group**	**D(n)**	**T(n)**	**L(n)**	$n \geq$
Reversals	S_n	$n - 1^*$ [1]	$\lceil \frac{2n}{3} \rceil - 1$	$\lfloor \frac{n}{3} \rfloor$	3
Prefix reversals	S_n		$n - 1$		1
Transpositions	S_n		$\lfloor \frac{n}{2} \rfloor$		1
Prefix transpositions	S_n	$n - \lfloor \frac{n}{4} \rfloor$ [6]	$n - \lceil \frac{2n}{5} \rceil$	$\lceil \frac{2n}{5} \rceil - \lfloor \frac{n}{4} \rfloor$	4
Reversals	$S_n^{\pm}$	$n + 1^*$ [16]	$n - 1$	2	4
Prefix reversals	$S_n^{\pm}$		$\lfloor \frac{5n+2}{4} \rfloor$		1
Reversals and transpositions	$S_n^{\pm}$	$n - \lfloor \frac{n-2}{3} \rfloor$	$n - \lceil \frac{n-2}{2} \rceil$	$\lceil \frac{n-2}{2} \rceil - \lfloor \frac{n-2}{3} \rfloor$	3

5 Concluding Remarks

We conducted an experimental investigation on the rearrangement distance distribution regarding a number of sets of operations. As a result, we presented rearrangement distance distributions that, as far as we know, had never been published before, reinforced some prior conjectures on diameter, traversal diameter and longevity of S_n and $S_n^{\pm}$, and presented some conjectures on traversal diameter of S_n and $S_n^{\pm}$. The next step is finding ways of proving such conjectures.

In future, we will use the rearrangement distance database [9] generated in this work to develop a framework that will support the process of evaluating approximation algorithms for genome rearrangements. The idea of the framework is to compare, for each permutation in S_n (or $S_n^{\pm}$), the distance outputted by a given approximation algorithm with the related rearrangement distance and to produce statistics that can be used to analyze its performance. Such analysis is essential, for instance, when comparing two approximations algorithms that have the same approximation ratio or an approximation algorithm with an heuristic.

Acknowledgments. This work was supported by CNPq (processes 483177/2009-1, 200815/2010-5 and 135212/2010-3). We would like to thank Felipe Rodrigues da Silva from Embrapa and João Carlos Setubal from Virginia Bioinformatics Institute for helping us with the computation of the rearrangement distances.

References

1. Bafna, V., Pevzner, P.A.: Genome rearrangements and sorting by reversals. SIAM Journal on Computing 25(2), 272–289 (1996)

2. Bafna, V., Pevzner, P.A.: Sorting by transpositions. SIAM Journal on Discrete Mathematics 11(2), 224–240 (1998)
3. Chitturi, B., Fahle, W., Meng, Z., Morales, L., Shields, C., Sudborough, I., Voit, W.: An (18/11)n upper bound for sorting by prefix reversals. Theoretical Computer Science 410(36), 3372–3390 (2009)
4. Chitturi, B., Sudborough, I.: Bounding prefix transposition distance for strings and permutations. In: Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS'2008). p. 469. IEEE Computer Society, Waikoloa, Big Island, Hawaii, USA (2008)
5. Cohen, D.S., Blum, M.: On the problem of sorting burnt pancakes. Discrete Applied Mathematics 61(2), 105–120 (1995)
6. Dias, Z., Meidanis, J.: Sorting by prefix transpositions. In: Proceedings of the 9th International Symposium on String Processing and Information Retrieval (SPIRE'2002). pp. 65–76. Springer-Verlag, Lisbon, Portugal (2002)
7. Eriksson, H., Eriksson, B., Karlander, J., Svensson, L., Wästlund, C.: Sorting a bridge hand. Discrete Mathematics 241(1-3), 289–300 (2001)
8. Fertin, G., Labarre, A., Rusu, I., Tannier, E., Vialette, S.: Combinatorics of Genome Rearrangements. The MIT Press (2009)
9. Galvão, G.R., Dias, Z.: Rearrangement distance database. `http://mirza.ic.unicamp.br:8080/bioinfo`
10. Galvão, G.R., Dias, Z.: Computing rearrangement distance of every permutation in the symmetric group. In: Proceedings of the 26th ACM Symposium on Applied Computing (SAC'2011). pp. 106–107. ACM Press, Taichung, Taiwan (2011)
11. Gates, W., Papadimitriou, C.: Bounds for sorting by prefix reversal. Discrete Mathematics 27, 47–57 (1979)
12. Gu, Q., Peng, S., Sudborough, H.: A 2-approximation algorithm for genome rearrangements by reversals and transpositions. Theoritical Computer Science 210(2), 327–339 (1999)
13. Hartman, T., Sharan, R.: A 1.5-approximation algorithm for sorting by transpositions and transreversals. Journal of Computer and System Sciences 70(3), 300–320 (2005)
14. Labarre, A.: Edit distances and factorisations of even permutations. In: Proceedings of the 16th annual European symposium on Algorithms (ESA'08). pp. 635–646. Springer-Verlag, Karlsruhe, Germany (2008)
15. Lu, L., Yang, Y.: A lower bound on the transposition diameter. SIAM Journal on Discrete Mathematics 24(4), 1242–1249 (2010)
16. Meidanis, J., Walter, M., Dias, Z.: Reversal distance of signed circular chromosomes. Technical Report IC-00-23, Institute of Computing, University of Campinas, Brazil (2000)
17. Meidanis, J., Walter, M., Dias, Z.: A lower bound on the reversal and transposition diameter. Journal of Computational Biology 9(5), 743–745 (2002)

Improving reliability by reducing input information required by a non-coding RNA identifier based on Support Vector Machine

Victor H. H. Taira[1], Tulio C. C. Silva[1], Pedro A. Berger[1], Maria Emília Walter[1], and Marcelo M. Brígido[2]

[1] University of Brasilia
Department of Computer Science
mariaemilia@unb.br
[2] University of Brasilia
Institute of Biology
brigido@unb.br

Abstract Non-coding RNA (ncRNA) participate in many cellular mechanisms, such as regulation, gene expression and supression, and many other control activities. Several methods and software have been developed to accomplish the identification of ncRNA. Particularly, Portrait [2] is an *ab initio* method based in the Support Vector Machine (SVM) algorithm to decide if a given sequence is indeed ncRNA. SVM's running time relies heavily on the number of numerical attributes extracted from any given RNA sequence. Early implementations of SVM models have repeatedly used more than 80 different attributes, which strongly compromise training and running speed, not necessarily increasing the SVM reliability. The objective of this work is to reduce the number of input attributes of the Portrait ncRNA identifier improving reliability, whilst also achieving faster training speed.

1 Introduction

Identifying non-coding RNA (ncRNA) is a challenging problem because the role of ncRNA in cellular regulation, gene expression and supression, as well as many other control activities, is not entirely understood. Methods such as Portrait [2], SOM-Portrait [19], CPC [8] and Infernal [14] have different approaches to accomplish the identification of ncRNA.

Particularly, Portrait is an *ab initio* ncRNA identifier based in Support Vector Machines (SVM). SVM is a machine learning algorithm, based on dimensionality transformation and pattern identification [7] of a set of numerical attributes, or characteristics. In Portrait, these attributes are computed for each RNA sequence. These attributes heavily influence the time complexity of the SVM algorithm. Current implementations of SVM models use more than 80 different numerical attributes, heavily compromising training and running speeds, and causing lower quality ncRNA predictions. In this work, we trained a SVM

model using the same training data of Portrait, reducing the number of numerical attributes that are extracted from RNA sequences losing an average $\approx 2\%$ of training accuracy. However, further results using three different *fungi* show that the new identifier agrees with Portrait ncRNA predictions and gives a more reliable prediction probability. The obtained final number of input attribute reduction was of about 60% less than original Portrait, what significantly reduced training execution time. This is an important result, because reliable ncRNA classifiers lead to better identification of ncRNA candidates. Also, SVM training is slow, committing good computational resources for up to several days. Reducing training complexity and execution time for SVM-based identifiers is a key achievement for further implementations of ncRNA identifiers, using less computational resources for reaching similar or even better experimental results.

This paper is divided as follows. In Section 2 we briefly address the biological problem of identifying ncRNA, the SVM algorithm used in Portrait, and describe the Portrait ncRNA identifier. In Section 3 we describe the methodology used for complexity reduction and for testing the trained SVM models. In Section 4 we discuss obtained results. Finally, we conclude and suggest further work in Section 5.

2 About SVM and Portrait

Identification of ncRNA through software helps biologists to precisely and quickly establish if a given RNA sequence could be non-coding. Among other possible applications, this analysis helps to establish new criteria for identification of these molecular structures. In this context, machine learning techniques such as SVM have the ability to gather several known ncRNA identification criteria, such as thermodinamical or structural information, allowing biologists to quickly adapt better identification methods. More information about known criteria for identification can be found in CONC [11] and Portrait [2] papers. Readers can refer to [12] for a detailed explanation of known methods for ncRNA identification.

Support Vector Machine

SVM is a special kind of learning algorithm that implements a solution for the statistical problem of structural risk minimization [7]. Structural risk minimization states that an optimal separation between two distinct sets is obtained by choosing the largest separation between them, in order to minimize the risk of erroneously separating members of the same set. Figure 1 represents a two-dimensional decision plane for the problem of optimally separating circles from crosses, representing, respectively, positive and negative training sets.

Each point in the decision space represents an attribute vector i_j, which consists of $X_1, X_2, .., X_k$ numerical data collected from input information. The optimal separation of these two sets is accomplished by discovering an optimal hiperplane function f, such that the distance ρ_0 between i_0 and f is as maximum as possible. In this case, i_0 is called a support vector for the optimal hiperplane.

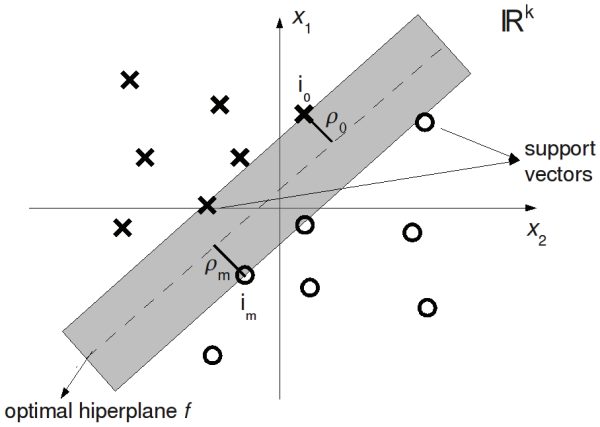

Figure 1. An example of the decision plane, the members of two training sets (circles and crosses) and the optimal hyperplane.

The training complexity is based on many factors, including the number of dimensions in the decision space, which directly affects the optimal hiperplane function. In general, a radial basis transformation, called a kernel function, is applied for each input instance, and calculated for each support vector. In this case, we can estimate the time complexity of the algorithm as $O(k \cdot n^2)$, where n is the number of positive and negative training instances, and k is the number of numerical attributes extracted from each input data.

Training SVM models is achieved by providing the algorithm with known positive and negative attribute vectors. These are built using several auxiliary programs and routines to collect information from the input data. The algorithm then divides the training data into m equal parts (m usually between 4 and 10), repeatedly using $m-1$ parts to train the model and the remaining one to conduct a cross-validation check on the trained model. According to the kernel function, the algorithm finds the best hiperplane in the decision space. In radial basis functions, this value consists of a numerical value γ. Another numerical value C is used as a penalty factor for missclassified attribute vectors. SVM is not prone to overfitting, a situation where the algorithm excessively learns the training data and loses the capacity of correctly identifying other experimental data.

The Portrait ncRNA identifier

Portrait [2] is a software developed to identify ncRNA using *ab initio* information extracted from a given RNA sequence. Previously trained SVM model then tries to determine the set (coding or non-coding) in which the sequence fits best. It is developed in Perl, and uses the LIBSVM version 3.00 [3]. Figure 2 provides the program workflow.

In Step 1, Portrait filters sequences having a number of nucleotides different from A, T, C and G higher than 10% of the nucleotides total number for each

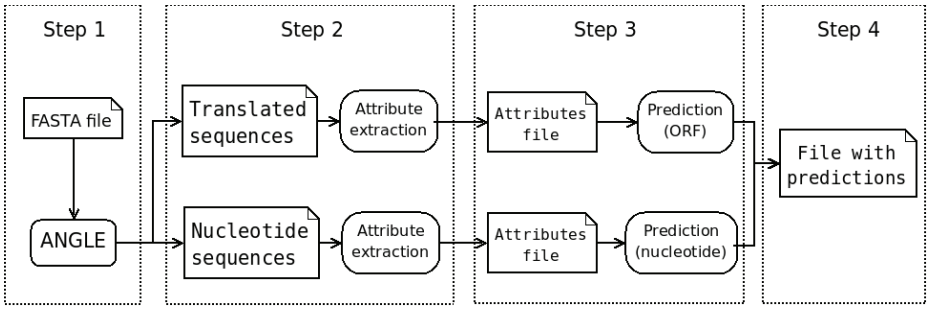

Figure 2. Workflow for the Portrait method.

sequence. Sequences are translated using ANGLE [18], an ORF finding software designed for low quality sequences. In the next step, the software creates two files, one composed of sequences with ORFs predicted by ANGLE (*Translated sequences* file), and another file containing sequences that had no ORF predicted by ANGLE (*Nucleotide sequences* file). In Step 3, Portrait extract several structural numerical attributes from both files, and only *Translated sequences* file undergoes another extraction of attributes depending on the predicted ORFs. Attributes extracted from each sequence are listed in the second column of Table 1. The algorithm then sends the file with the attributes for prediction by LIBSVM using the previously trained SVM model. Sequences from *Translated sequences* file use an ORF-dependent model, while *Nucleotide sequences* file uses a nucleotide-dependent model requiring any ORF information. Finally, in Step 4, the program gathers the predictions results for each input sequence, which are then printed to the user.

Table 1. Attributes extracted from RNA sequences for SVM input in Portrait (*extracted only for the ORF-dependent model) (†extracted only for the nucleotide-dependent model).

Attribute name	Variables Portrait	Variables Light-Portrait
Trinucleotide frequency	64	5
Dinucleotide frequency	16	16
Nucleotide frequency	4	4
ORF length*	4	1
Sequence length†	4	1
Aminoacid frequency*	20	20
Protein isoeletric point (IEP-EMBOSS)* [17]	1	1
Protein complexity (CAST)* [16]	1	1
Protein hydrophobicity* [9]	1	1
Predicted ORF quality score*	0	1

The training set used by Portrait is a compilation of several different ncRNA databases. In total, 12,555 known ncRNA sequences collected from the Rfam [6], RNAdb [15] and NONCODE [10] ncRNA databases form the negative set. The positive set comprises 20,000 sequences collected from SwissProt [1] protein database.

3 Reducing the number of attributes in Portrait

In order to reduce the number of attributes, we used the same training set and the same kernel function of Portrait, a radial basis function. First, we executed a specially designed algorithm in LIBSVM to retrieve a "relevance factor" for each attribute. This factor is calculated by running the SVM model training with and without the attribute, and calculating the resulting training cross-validation accuracy. Using this data, we selected attributes that contributed less for the training accuracy. However, this algorithm can not verify interactions between different attributes. Some attributes, although having no relevant contribution to SVM accuracy, interacted with other more important attributes.

We had to empirically determine which attributes could be safely removed without losing training accuracy. The third column of Table 1 summarizes the new set of attributes, called Light-Portrait. Sequence length was reduced to only one variable having four possible values: 0.25 if length $< 100nt$, 0.5 if $100nt \leq$ length $< 400nt$, 0.75 if $400nt \leq$ length $< 900nt$ and 1 if length $\geq 900nt$. ORF length contributes with only two variables, sequences with ORF length $\geq 300nt$ or $< 300nt$. Trinucleotide frequency was calculated only for TAG, TGT, ACG, TCG and CGA trinucleotides.

For the training, we used 8 Intel®Xeon®E5506 $2.13GHz$ with $20.0GB$ RAM. Portrait was trained using the same LIBSVM configurations and parameters described by its authors, running in the machine above. We trained two models, one protein-dependent, using ORF information obtained from ANGLE, and another nucleotide-dependent, using only structural attributes. Training time for the protein-dependent and nucleotide-dependent models totalized, in Portrait, 1273.2min and 987.6min, respectively. For Light-Portrait, we assessed 449.3min and 394.5min, achieving a percentual gain in speed (speedup) of 64.71% and 60.07%. The assessed accuracy of the two trained models, using ten-fold cross-validation, suffered a reduction. For the protein-dependent model, Portrait achieved cross-validation accuracy of 93.20% against 91.44% of Light-Portrait. For the nucleotide-dependent model, Portrait obtained 90.90% against 83.35% of Light-Portrait. This significant reduction is a result of complex interactions between chosen parameters, that individually may deliver lower or higher accuracies than interacting with other parameters. By reducing the number of such interations, the trained SVM model has a less complex decision surface, which reduces cross-validation accuracy, but allows the SVM model to be more general, and, therefore, reduces overfitting risks. We further analyze the trained SVM model in terms of reliability in Section 4. The $(\gamma; C)$ values for SVM radial

basis kernel function found for protein and nucleotide-dependant models were, respectively, (0.5; 8) and (8; 2).

4 Three case studies with *fungi*

To validate Light-Portrait, we used RNA sequences from three different *fungi* transcriptome: the *Paracoccidiodes brasiliensis* [4], the *Coccidiodes immitis* [5] and the *Aspergillus oryzae* [13]. The *P. brasiliensis* has 6,022 assembled ESTs, the *C. immitis* comprises 9,757 transcripts and the *A. oryzae* totalizes 9,051 transcripts. *C. immitis* and *P. brasiliensis* share close phylogenies, while *A. oryzae* differ from both organisms. With these *fungi*, we are able to correctly assess the trained model against organisms that were not previously presented in the training stage.

A Perl script was developed to filter this data set, accepting a sequence if it had length with at least 80 nucleotides and at most 20% of characters different from A, C, G and T. This filter script discarded 9 sequences, from the 6,022 *P. brasiliensis* sequences, generating a final testing set containing 6,013 sequences. No sequences from *C. immitis* and *A. oryzae* transcriptomes were discarded.

In order to analyze the light-Portrait reliability $R(c)$, we use the following equation, where $ncRNA(c)$ is the number of putative predicted ncRNA with prediction probability above a given cutoff c, and $ncRNA$ is the total number of predicted ncRNA by the identifier. In the experiments we used $c \geq 70\%$.

$$R(c) = \frac{ncRNA(c)}{ncRNA}$$

Table 2. Results for *P. brasiliensis* using Portrait and Light-Portrait.

	Portrait	Light-Portrait	Portrait ∩ Light-Portrait
mRNA	5,044	5,384	4,956
ncRNA	969	629	541
Total	6,013	6,013	5,497 (91.42%)

Table 2 shows the results of the execution of Portrait (first column) and Light-Portrait (middle column), and the number of matching predictions found for each sequence (last column) for the *P. brasiliensis* fungus. The percentual of matching sequences for the whole set used is shown. The percentual of matching sequences clearly demonstrates that, despite having reduced the number of input attribute at about 60%, results were very similar to the Portrait method. Comparatively, light-Portrait predicted less ncRNA than the original Portrait. Analyzing the reliability measured for light-Portrait, we could determine that $R(0.7) = 91.09\%$. Portrait, in comparison, measured only $R(0.7) = 68.44\%$. This indicates that light-Portrait has fewer predicted ncRNA with low predicted probability, which increases the reliability.

Table 3. Results for *C. immitis* using Portrait and Light-Portrait.

	Portrait	Light-Portrait	Portrait ∩ Light-Portrait
mRNA	8,919	9,375	8,877
ncRNA	838	382	340
Total	9,757	9,757	9,217 (94.47%)

In Table 3 we summarize the results obtained for the *C. immitis* fungus. It is noteworthy that the number of ncRNA predicted by Light-Portrait drastically shrinks, when compared to Portrait. Analyzing the reliability of Portrait, however, we measured only $R(0.7) = 65.27\%$. In contrast, we obtained $R(0.7) = 96.58\%$ for Light-Portrait. This again indicates that the method identifies ncRNA more precisely than Portrait. We note that results for *C. immitis* and *P. brasiliensis* are similar, what states the correctness of Light-Portrait.

Table 4. Results for *A. oryzae* using Portrait and Light-Portrait.

	Portrait	Light-Portrait	Portrait ∩ Light-Portrait
mRNA	7,690	7,905	7,514
ncRNA	1,361	1,146	970
Total	9,051	9,051	8,484 (93.74%)

In Table 4, results for the *A. oryzae* organism are shown. Again, the measured reliability for Light-Portrait is $R(0.7) = 97.29\%$, in contrast with $R(0.7) = 70.79\%$ for Portrait. This states that, even for organisms not so close phylogenetically, the trained model is able to identify ncRNA improving reliability for putative predicted ncRNA.

5 Conclusions and future work

The results obtained by Light-Portrait clearly demonstrate that, by reducing the number of attributes in trained SVM models, we reduced training costs and improved the behaviour of the trained model, having assessed good reliability of the ncRNA prediction of light-Portrait. We trained our method using RNA sequences mostly from *Neurospora crassa* and *Aspergillus niger*, and validated light-Portrait using other three different species. Although using different species for training and validation, the results showed that the method succesfully applied the acquired knowledge from the training set to other species. This aspect is useful when analyzing new species or species with poor or no genetic data.

Future work involves further validation of the method reliability, confronting with other SVM classifiers, such as CONC [11] and CPC [8], and non-SVM classifiers, such as SOM-Portrait [19] and Infernal [14]. Other tools to validate our parameter choice will be used to confirm our methodology and we will extend our

results to improve training accuracy and identifier reliability for non-supervised methods.

References

1. Apweiler, R., *et al*: Uniprot: the universal protein knowledgebase. NAR 32, D115–119 (2004)
2. Arrial, R., *et al*: Screening non-coding RNAs in transcriptomes from neglected species using PORTRAIT: case study of the pathogenic fungus *Paracoccidioides brasiliensis*. BMC Bioinformatics 10(1), 239–247 (2009)
3. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines (2001), software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
4. Felipe, M.S.S., *et al*: Transcriptional profiles of the human pathogenic fungus *Paracoccidioides brasiliensis* in mycelium and yeast cells. Journal of Biological Chemistry 280(26), 24706–24714 (2005)
5. Galgiani, J.N.: Coccidioidomycosis: A Regional Disease of National Importance: Rethinking Approaches for Control. Annals of Internal Medicine 130(1), 293–300 (1999)
6. Griffiths-Jones, S., *et al*: Rfam: annotating non-coding RNAs in complete genomes. NAR 33, D121–D124 (2005)
7. Haykin, S.: Neural Networks. Macmillan College Publishing Company, New York (1999)
8. Kong, L., *et al*: CPC: assess the protein-coding potential of transcripts using sequence features and support vector machine. NAR 35, 345–349 (2007)
9. Kyte, J., Doolittle, R.F.: A Simple Method for Displaying the Hydropathic Character of a Protein. J. Mol. Biol. 157(1), 105–132 (1982)
10. Liu, C., *et al*: NONCODE: an integrated knowledge database of non-coding RNAs. NAR 33, D112–D115 (2005)
11. Liu, J., *et al*: Distinguishing protein-coding from non-coding rnas through support vector machines. PLoS Genet 2(e), 29–36 (2006)
12. Machado-Lima, A., *et al*: Computational methods in noncoding RNA research. Journal of Mathematical Biology 56, 15–49 (2008)
13. Machida, M., *et al*: Genome sequencing and analysis of *Aspergillus oryzae*. Nature 438(7071), 1157–1161 (2005)
14. Nawrocki, E.P., *et al*: Infernal 1.0: inference of RNA alignments. Bioinformatics 25(10), 1335–1337 (2009)
15. Pang, K., *et al*: RNAdb - a comprehensive mammalian noncoding RNA database. NAR 33, D125–D130 (2005)
16. Promponas, V.J., *et al*: Cast: an iterative algorithm for the complexity analysis of sequence tracts. Bioinformatics 16(10), 915–922 (2000)
17. Rice, P., *et al*: EMBOSS: The European molecular biology open software suite. Trends Genet. 16(6), 276–277 (2000)
18. Shimizu, K., *et al*: ANGLE: a sequencing errors resistant program for predicting protein coding regions in unfinished cDNA. Journal of Bioinformatics and Computational Biology 4(3), 649–664 (2006)
19. Silva, T.C.C., *et al*: SOM-PORTRAIT: Identifying non-coding RNAs using Self-Organizing Maps. In: Advances in Bioinformatics and Computational Biology. vol. 5676, pp. 73–85 (2009)

ALGAe: A Test-bench Environment for a Genetic Algorithm-based Multiple Sequence Aligner

Sergio J. R. Ordine, Alex B. Grilo, André Atanasio M. Almeida, and Zanoni Dias

Institute of Computing, University of Campinas
{ra921298, ra070018}@students.ic.unicamp.br
{atanasio, zanoni}@ic.unicamp.br

Abstract. Multiple Sequence Alignment (MSA) is one of the most important problems on Computational Biology and Bioinformatics. This manuscript describes ALGAe, an application to test a parametrized Genetic Algorithm approach in order to look for a good set of parameters to solve this problem.

Keywords: Multiple Sequence Alignment, Genetic Algorithm

1 Introduction

Multiple Sequence Alignment (MSA) is a very important tool for Computational Biology and Bioinformatics fields. It is applied on a wide range of uses within such as: phylogenetic analysis, identification of conserved motifs and domains and structure prediction [3].

MSA is a complex problem considering not just biological [3] but also computational aspects, as it is known to be a NP-hard problem as proved by Just [2].

Considering both the importance and complexity of solving the Multiple Sequence Alignment problem, many different approaches were tried to present an approximation algorithm or heuristic that provides good results for this problem. It is possible to classify these heuristics into two major groups: progressive and iterative algorithms.

Progressive algorithms solves the MSA problem by progressively aligning pairs of sequences or pairs of sub-alignments up to align all of the original sequences. Usually this algorithms runs quickly, but are not able to recover from a bad decision during the process, what can jeopardize the final result. One of the most important progressive alignment tools is ClustalW [6].

Iterative algorithms approach is to incrementally improve a original alignment (or set of alignments) during its execution. It can be achieved through a range of strategies, which includes genetic algorithms (SAGA [4]).

Genetic Algorithm (GA) [1] is inspired on the Darwinian idea of Natural Selection. This heuristic is based on sets of possible solutions, called population. Each possible solution (an individual on the population) is represented on a discrete set of data called chromosome. Those chromosomes are classified according to its "fitness" to solve the proposed problem, mixed to generate new solutions and kept or eliminated on a next iteration.

The current work presented on this manuscript is an initial version of a Genetic Algorithm-based application to solve MSA problems. This application is named `ALGAe` ("ALigning by Genetic Algorithm environment") and it is structured as a test environment where several Genetic Algorithm parameters can be changed and combined in order to search a good set of them.

The Section 2 describes briefly the `ALGAe` solution and its initial results and the Section 3 outilines the current status and next steps of this project.

2 Current Work and Results

The presented solution is based on the Algorithm 1.

Algorithm 1: Genetic Algorithm

```
Data: Set of Sequences (S), number of generations (n)
Result: chromosome c (the best alignment for S after n generations)
Generation ← 1
Population ← CreateInitialPopulation(S)
while Generation ≤ n do
    BreedingPopulation ← SelectForBreeding(Population)
    Population ← Population ∪ Offspring(BreedingPopulation)
    Population ← Select(Population)
    Generation ← Generation + 1
c ← Max(Population)
return c
```

An initial population is created based on the given input sequences. The individuals are mixed and submitted to a selection process based on a fitness criteria until a stop criteria is reached. On this project the number of generations is being currently used as the stop criteria.

There are several parameters on this algorithm that can be adjusted and tested to improve the output alignment quality: the initial population creation, the population size, the breeding selection, the operations used to mix the current population and generate offspring, the survival (selection) function, the fitness criteria and the stop criteria.

The following parameters were tested up to this moment.

Fitness Function This function provides the quality of a individual and is used as a decision factor for breeding and survival.

- **Sum of Pairs (SP)** - This function considers, based on a score function, the sum of each pair of letters on the same column. Considering an alignment of m sequences with size n and c_{ik} representing the k-esimal column of the i-esimal sequence on the alignment, this function can be represented as:

$$SP = \sum_{k=1}^{n} \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} score(c_{ik}, c_{jk}) \quad (1)$$

– **Gap Affinity - Sum of Pairs (GASP)** - This function is similar to the previous one but considers a smaller penalties for sequences of gaps due to its biological meaning.

Offspring Operations This operations are executed in order to include new characteristics into the population, increasing population variability and providing possible better solutions.

– **Crossing-Over Operators** - Crossing-Over operators merge characteristics from two different chromosomes in order to generate one or more new individuals. Three different crossing-over operators where tested. The Single Point operator, which splits one of the parents vertically at a random point and splits the second using the same residues of the first split as its reference, merging the complementary sub-alignments (from different parents) to generate its offspring. The Best Partial Alignment operator selects the best aligned sequences for each parent (without repetition) up to select all sequences and merges the sub-alignments built using a consensus global alignment. Finally, the Sequence Similarity operator performs the same kind of operation of the Best Partial Alignment operator, but using a phylogenetic tree to guide this process.
– **Mutation Operators** - Mutation operators change part of an already existent chromosome, creating a new individual with characteristics not yet existent on the current population. Three mutation operators were also created. The Simple Mutation, which inserts a randomly positioned gap into one of the sequences (adjusting the alignment size); the Shift Gap Block operator that shifts a gap block within one of the sequences to the left or to the right and the Change Gap Block operator that randomly selects part of both a gap block and its joint sequence and swaps them.

In order to test this initial approach, the developed application was benchmarked against GAADT, a genetic algorithm-based sequence aligner based on abstract types developed by Santos [5]. In order to make this test ALGAe was executed 100 times for 2000 generations (with a population of 250 individuals), using BLOSUM62 score matrix. The scores were gotten based on BAliBASE 2 for ALGAe and from Santos [5] results for GAADT (Table 1). These test used five protein families from BAliBASE 2 using the SP score metric, with measures the alignment quality from 0 to 1 (the bigger the better).

Table 1. ALGAe and GAADT results for 1aab and related protein groups

	1aab	1fjlA	1hpi	1csy	1tgxA	average
GAADT (avg)	0.881	0.814	0.708	0.703	0.692	0.760
ALGAe (avg)	0.884	0.938	0.883	0.762	0.685	0.830
ALGAe (max)	0.896	1.000	0.962	0.807	0.772	0.887

`ALGAe` was also tested against BAliBASE 3.0 [7] protein families sets RV11 and RV12 (Table 2). This tests were executed using 650 generations with a population of 100 individuals.

Table 2. `ALGAe` results for BAliBASE 3.0 References RV11 and RV12

Reference	BLOSUM62	BLOSUM80	PAM100	PAM250
RV11 (SP)	0.336	0.349	0.289	0.289
RV11 (GASP)	0.356	0.381	0.305	0.327
RV12 (SP)	0.739	0.757	0.742	0.752
RV12 (GASP)	0.732	0.759	0.728	0.761

3 Conclusion and Future Works

Up to the writing of this work, the authors have developed `ALGAe` basic architecture and a set of operators, score point functions and initial population creators. The results are being run against BAliBASE 3.0 references to check its effectiveness.

The authors are considering the following tasks to improve the current results. First of all, different initial population building methods and offspring operators are being considered to be implemented. It is expected that some of them improves the overall score results. Besides that, a different scoring method considering different score matrices for each sequence pair on the alignment is being implemented and tested. This scoring method may improve the alignment score, reflecting biological similarity or differences more precisely.

The `ALGAe` also provides logging capabilities. This gathered data can be used on a fine grained analysis in order to improve the operators or even highlight changes that can trigger new operators development.

References

1. J. Holland. *Adaptation in natural and artificial systems.* University of Michigan Press, 1975.
2. W. Just. Computational complexity of multiple sequence alignment with SP-score. *Journal of Computational Biology*, 8:615–623, 1999.
3. C. Notredame. Recent progress in multiple seq. alignment: a survey. *Pharmacogenomics*, 3(1):131–144, 2002.
4. C. Notredame and D. G. Higgins. SAGA: Sequence Alignment by Genetic Algorithm. *Nucleic Acids Res.*, 24(8):1515–1524, 1996.
5. D. Santos. Alinhamento multiplo de proteinas via algoritmo genetico baseado em tipos abstratos de dados. Master's thesis, Universidade Federal de Alagoas, 2008.
6. J. D. Thompson. et al. CLUSTAL W: improving the sensitivity of progressive multiple seq. alignment through seq. weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.*, 22(22):4673–4680, 1994.
7. J. D. Thompson, P. Koehl and O. Poch. Balibase 3.0: latest developments of the multiple sequence alignment benchmark. *Proteins*, 61:127–136, 2005.

Two new whole-genome distance measures

Ulisses Dias *, Zanoni Dias, and João C. Setubal

Institute of Computing, University of Campinas
{udias,zanoni}@ic.unicamp.br
Virginia Bioinformatic Institute, Virginia Tech
{setubal,zanoni}@vbi.vt.edu

Abstract. We propose two new whole genome distance measures, called NUCMi and PROMi. These measures are easy to compute, thus providing an efficient tool with which to quickly compare whole genome sequences. The measure NUCMi is meant for closely related genomes, and in our tests performs better than or equal to the previously proposed measure MUMi. The measure PROMi is meant for more distant genomes.

Keywords: Whole-Genome Comparison

1 Introduction

With the increasing availability of prokaryotic genome sequences the need for methods that can quickly and effectively compare large sets of genome sequences becomes more pressing. One way to compare two genomes is to compute a distance between them. Ideally this distance should accurately reflect the evolutionary divergence between the two genomes.

A quick and accurate genome distance method has several applications. For example, given a new draft genome, use of a distance method can determine what is the closest complete genome that could be used for alignment and reference purposes. For the case of closely related genomes, Deloger et al. [2] have proposed the MUM index, or MUMi, as a genomic distance measure. This measure uses results from a pairwise genome alignment as given by the program MUMmer [7]. The authors show that MUMi values agree with those obtained by other methods, such as ANI [5, 6], while being considerably easier and faster to obtain.

In this work we propose two new distance measures to compare two whole genome sequences. One of them is aimed at comparing closely related genomes, just like MUMi. The other is meant to compare more distant genomes. Both measures can be computed efficiently.

* This work was made possible in part by a PhD fellowship from FAPESP to UD (2007/05574-4), a sabbatical fellowship from CNPq to ZD (200815/2010-5) and by project funding from CNPq to ZD (483177/2009-1 and 473867/2010-9).

2 Description of the distance measures

MUMi relies on the number of maximal unique matches (MUMs) found by MUMmer when comparing two genomes. The computation of MUMi is based on the following formula:

$$\text{MUMi}(A, B) = 1 - \frac{L_{\text{mum}}}{L_{\text{av}}}$$

where L_{mum} is the sum of the lengths of all regions present in some MUM, and L_{av} is the mean length of the two genomes being compared. MUMi values are in the interval $[0, 1]$, with values closer to 0 denoting closer genomes and values closer to 1 denoting more distant genomes.

The two measures that we propose are called NUCMi and PROMi. Both are based on scripts that are part of the MUMmer package [7]. For the NUCMi computation we use the script `nucmer`, which performs two additional steps after running MUMmer. It clusterizes the MUMs found and extends the alignments that can connect MUMs that are close enough. We also process the results of `nucmer` with the utility `delta-filter` with the -1 parameter, which keeps only the best reciprocal alignments. For the PROMi computation we use the script `promer`, which performs the comparison using conceptual translations of the input DNA sequences.

In both cases the index computation relies on the projection of matches found onto each of the genomes being compared. In Fig. 1 we show in schematic fashion what we mean by projections.

Both measures use the following formula:

$$\text{Dist}(A, B) = 1 - \frac{Px + Py}{L_A + L_B}$$

where P_x is the sum of all match projections on the x axis, and P_y is the analogous for the y axis. L_A and L_B are the lengths of genome A and B respectively. This measure has the same range and meaning as MUMi: its values are in the interval $[0, 1]$, and close genomes have a value closer to 0 and distant genomes have a value closer to 1.

3 Results

We have applied our distance measures to five different sets of genomes. In order to evaluate their performance we used maximum likelihood (ML) phylogenetic trees obtained from the literature through the following methodology.

On any given set of genomes we obtain all pairwise distances using some method. The resulting matrix is used to build a phylogenetic tree using the neighbor-joining algorithm [9] as implemented by PHYLIP [3]. We then compare the tree so generated with the reference ML tree using an algorithm to obtain the maximum agreement subtree, or MAST [4]. We used the implementation of MAST given by Vienne et al. [11]. The MAST algorithm results in a number

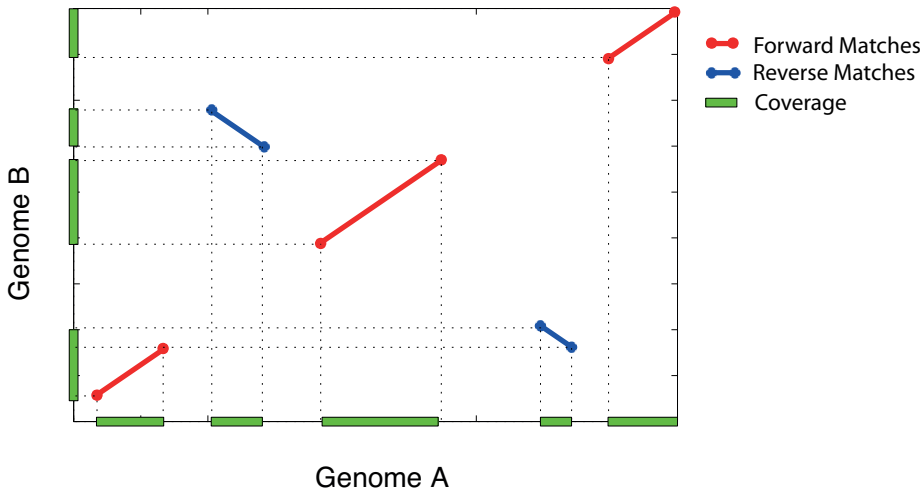

Fig. 1. In axes x and y and in green we show the match projections used by our measures.

that has the following meaning: it is the number of species (or genomes) that the algorithm can place in the maximum agreement subtree. The higher this number the more agreement there is between the two trees being compared.

Table 1. Results for closely related genomes.

Bacterial group	MUMi	NUCMi	Genomes available	Genomes in T_{ref}
Pseudomonadaceae	14	15	18	16
Mycobacterium	14	15	21	16
Xanthomonas	8	8	9	9
Shewanella	8	8	20	9

For the case of closely related genomes we performed four tests, whose results are shown in Table 1. These results show that the NUCMi measure performed better or at the same level as MUMi. The reference trees used were as follows: *Pseudomonadaceae* [10]; *Mycobacterium* [1]; *Xanthomonas* [8]; and *Shewanella* [12].

We perfomed one test for the case of more distant genomes. In this test we used 33 genomes from the gamma and beta-proteobacteria phyla. The reference tree was the one provided by Wu et al. [13]. The results were as follows. The PROMi distance resulted in a MAST value of 23; NUCMi and MUMi resulted in 15 and 13, respectively.

4 Conclusion

We propose two new genome distance measures, called NUCMi and PROMi. We show that the NUCMi measure has performance equal to or better than that of MUMi. For the PROMi measure we have performed one test, showing the unsurprising result that it performs better than NUCMi and MUMi, which are meant to be used for closely related genomes only. We plan to conduct further tests of PROMi.

References

1. Alam, M.T., Merlo, M.E., Takano, E., Breitling, R.: Genome-based phylogenetic analysis of Streptomyces and its relatives. Mol. Phylogenet. Evol. 54, 763–772 (2010)
2. Deloger, M., El Karoui, M., Petit, M.A.: A genomic distance based on MUM indicates discontinuity between most bacterial species and genera. J. Bacteriol. 191, 91–99 (2009)
3. Felsenstein, J.: PHYLIP (Phylogeny Inference Package) version 3.5c. Distributed by the author Department of Genetics, University of Washington, Seattle (1993)
4. Finden, C.R., Gordon, A.D.: Obtaining common pruned trees. Journal of Classification 2, 255–276 (1985), http://dx.doi.org/10.1007/BF01908078, 10.1007/BF01908078
5. Goris, J., Konstantinidis, K.T., Klappenbach, J.A., Coenye, T., Vandamme, P., Tiedje, J.M.: DNA-DNA hybridization values and their relationship to whole-genome sequence similarities. Int. J. Syst. Evol. Microbiol. 57, 81–91 (2007)
6. Konstantinidis, K.T., Tiedje, J.M.: Genomic insights that advance the species definition for prokaryotes. Proc. Natl. Acad. Sci. U.S.A. 102, 2567–2572 (2005)
7. Kurtz, S., Phillippy, A., Delcher, A.L., Smoot, M., Shumway, M., Antonescu, C., Salzberg, S.L.: Versatile and open software for comparing large genomes. Genome Biol. 5(2), R12 (2004)
8. Moreira, L.M., et al.: Novel insights into the genomic basis of citrus canker based on the genome sequences of two strains of Xanthomonas fuscans subsp. aurantifolii. BMC Genomics 11, 238 (2010)
9. Saitou, N., Nei, M.: The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol. Biol. Evol. 4, 406–425 (1987)
10. Setubal, J.C., et al.: Genome sequence of Azotobacter vinelandii, an obligate aerobe specialized to support diverse anaerobic metabolic processes. J. Bacteriol. 191, 4534–4545 (2009)
11. de Vienne, D.M., Giraud, T., Martin, O.C.: A congruence index for testing topological similarity between trees. Bioinformatics 23, 3119–3124 (2007)
12. Williams, K.P., Gillespie, J.J., Sobral, B.W., Nordberg, E.K., Snyder, E.E., Shallom, J.M., Dickerman, A.W.: Phylogeny of gammaproteobacteria. J. Bacteriol. 192, 2305–2314 (May 2010)
13. Wu, D., et al.: A phylogeny-driven genomic encyclopaedia of bacteria and archaea. Nature 462, 1056–1060 (2009)

Using assembly and phylogeny to classify metagenomic sequences*

Nalvo F. Almeida[1], Habib Asseiss[1] and João C. Setubal[2]

[1] College of Computing, Federal University of Mato Grosso do Sul, Brazil
[2] Virginia Bioinformatics Institute, Virginia Tech, USA
nalvo@facom.ufms.br, habibasseiss@gmail.com, setubal@vbi.vt.edu

Abstract. In order to better understand the diversity of an ecological system, one of the goals in metagenomic projects is to assign a taxonomic unit to each read of the sample. Several methods are known, mainly the ones based on sequence similarity and sequence composition. In this preliminary work we present a new method using assembly, sequence similarity and phylogeny. The method has been tested on simulated samples with different error rates and has been compared with other methods.

1 Introduction

Recent advances in large-scale sequencing technologies has provided new possibilities in the understanding of complex microbial environments. By directly extracting DNA samples from these environments at lower cost and without isolating single organisms, it is possible to obtain genetic information from whole microbial communities, leading to new insights into their biological complexity. This is the main idea behind metagenomic projects [6, 9].

One of the main goals of metagenomic studies is to taxonomically classify sequences according to their phylogenetic origin, assigning to each one read an Operational Taxonomic Unit (OTU), or at least a taxonomic group. The more accurate the classification process is, the better is the understanding of the diversity of the metagenomic sample and so is the interpretation of the whole ecological system. Although some authors consider binning the process of only clustering sequences into distinct groups, not necessarily assigning them an exact OTU [2], other ones [6, 7], including us, define binning, or classification, as the process of assigning an OTU to each read.

Classification methods can be divided into two main types: methods based on sequence composition and methods based on sequence similarity. The first ones rely on the fact that different genomes might have different composition signatures, such as GC content, codon usage, oligonucleotide frequences, the presence or absence of specific markers, such as specific 16S rRNAs, or even more complex features, using, for example, variable-order Interpolated Markov Models (IMMs). Phymm [2] uses IMMs by characterizing variable-length oligonucleotides typical of a phylogenetic group and integrating the results from the different lengths.

* This work has been supported by grants Fundect 23/200.194/2009, CNPq 305503/2010-3 and CNPq 474814/2010-6.

Methods based on sequence similarity use comparison programs, such as BLAST [1], to classify a read according to the best hits found in protein or domain databases [4, 5]. Other methods mix both approaches. PhymmBL [2] is an example, where results from Phymm and BLAST are incorporated.

A naive approach to deal with metagenomic reads would be to assemble them and to compare the contigs against known protein databases. However, in metagenomics the usual lengths (100-800 bp) are shorter than those obtained in single genome projects, thus the assembly process is more difficult [10]. Besides, contigs that should be assembled by reads from under represented species in the sample tend to be missed [7]. Another approach, without assembling, as used in Megan [4], is to compare each read against a database of known proteins and to look for the best hits in an also known species tree. Megan applies this method by using BLASTX and the NCBI species tree. Another currently available method is called RAIphy [7], a composition-based algorithm that uses what they call Relative Abundance Index (RAI), a measure of the relative abundance of oligonucleotides in genomic fragments. Each taxon has its RAI model, representing all fragmens in the genomes of that taxon. Each read is then scored against all taxons. The taxon with the highest score is assigned to the read.

In this work we propose a method that uses both assembling and phylogenetic trees (like in the approaches cited in the previous paragraph), as well as similarity programs. Using simulated metagenomics samples containing reads with different error rates, our method could get good results when compared to RAIphy and is much less computationally intensive than Phymm.

2 The method

Instead of assembling the whole set of reads of a sample, our method assembles only groups of reads that agree with each other, according to their hits found by comparison against a protein database. After getting the contigs, when it is possible, these are compared against a protein database and a phylogenetic tree of all best hits is built. Instead of giving as a result a unique species for each read, as made in Phymm [2] or just an internal node of the NCBI tree (representing a higher level taxon), as made in Megan [4], our method outputs a small set of leaves of the tree, representing the closest species of the assembled contig. The rationale behind this approach is that we can get more promising contigs by assembling only reads getting the same protein as a hit, instead of assembling all reads from the sample.

Below we present our algorithm. The input is R, the set of reads, κ, the maximum number of hits to be considered by BLAST programs, and δ, the maximum distance between two leaves in the tree to represent close species.

Step 1. Blastx all reads of R against *nr*. Discard reads with no hits.
Step 2. Create group G_i of all reads having p_i as first hit. Let s_i be its species.
Step 3. For each G_i, if $|G_i| = 1$ (G_i has only read r_i) then go to Step 7, using the set of $k \leq \kappa$ first hits of r_i as input. Otherwise, assembly reads of G_i. Let C_i

be the longest contig found.
Step 4. If $C_i = \emptyset$ then report s_i for each read in G_i and finish. Otherwise go to Step 5.
Step 5. Blastx each contig $C_i \neq \emptyset$ against nr, getting the $k \leq \kappa$ first hits, namely $H_i = \{h_1, h_2, \ldots, h_k\}$.
Step 6. If $|H_i| < 3$ then for each read in G_i report all the species present in $H_i \cup \{p_i\}$. Otherwise go to Step 7.
Step 7. Build the phylogenetic tree T_i for $H_i \cup \{p_i\}$.
Step 8. Let S_i be the set of leaves of T_i whose distance to p_i in the tree is $\leq \delta$. Report species present in S_i for each read in G_i.

3 Results and Discussion

We have tested our method using the same four simulated sets of *454* reads used in [3]. Those reads are on average 450nt long and we know which species they came from. The sets have 98,974 reads each and different error rates (see Table 1). Given a leaf f of the tree built in Step 7, let m_f be the mean of the tree distances between f and all other leaves. A leaf g is *close* to f if the distance between f and g is not greater than m_f. $\delta = m_f$ was used to determine which leaves are close to f in Step 8. Both BLASTX runs (Step 1 and Step 5) have been done with e-value 10^{-5} and taking $\kappa = 10$ best hits. The assembly program used was Velvet [11], and the phylogenetic construction in Step 7 was RAxML [8].

Table 1 shows the number of hits (out of 87,974 reads) obtained by each method, considering the different error rates of the samples. As we can see, our method is comparable to RAIphy. Phymm, although is much more computationally intensive, has a much better performance.

Table 1. Comparison of our method with RAIphy and Phymm.

Set	Error rate	Our method	RAIphy	Phymm
S_1	0.00%	48,902 (49.4%)	56,848 (57.4%)	87,483 (88.4%)
S_2	0.22%	51,770 (52.3%)	52,676 (53.2%)	87,095 (87.9%)
S_3	0.49%	53,119 (53.7%)	52,224 (52.8%)	86,791 (87.7%)
S_4	2.80%	60,218 (60.8%)	44,997 (45.5%)	80,608 (81.4%)

We also could notice that our method is not so sensitive to error rates. This happens because the greater is the error rate, the larger is the number os cases where $|H_i| < 3$ in Step 6, since contigs tend to lose quality. For those cases, the right OTU is given in this step and the algorithm does not continue, leading us to suspect that the tree (Step 8) is not necessary, since Blastx solves most of the cases. This is our very next investigation. Figure 1 shows the behavior of the three steps where the algorithm may output the OTUs.

This is a preliminary study and much more work need to be done. Besides the investigation cited in the previous paragraph, we plan to test our method on other sample sets, and compare it with another methods, using more detailed metrics. Other programs for building the phylogenetic trees need to be tested,

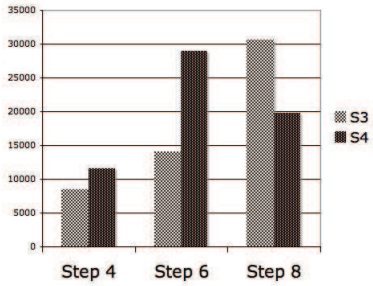

Fig. 1. Number of right OTUs assigned in each of Steps 4, 6 and 8, considering sets S_3 and S_4. Because the lack of quality in contigs when the error rate is larger, the number of Blastx hits is small enough to stop the algorithm in Step 6.

in order to gain efficiency. For assembling, other methods will be tested, in other to minimize the number of missing contigs. Another important observation that deserves further analisys has to do with how our method can help other ones. For example, our method could get the right OTU for 72.7% of the reads (on average, considering the four samples) to which Phymm could not correctly bin. Another job to be done is to make the method portable and available for users.

References

1. S. Altschul and *et al.* Gapped Blast and Psi-Blast: a new generation of protein database search programs. *Nucleic Acid Research*, 25:3389–3402, 1997.
2. A. Brady and S. Salzberg. Phymm and PhymmBL: metagenomic phylogenetic classification with interpolated Markov models. *Nature Methods*, 6(9), 2009.
3. K. Hoff. The effect of sequencing errors on metagenomic gene prediction. *BMC Genomics*, 10(1):520+, Nov. 2009.
4. D. H. Huson, A. F. Auch, J. Qi, and S. C. Schuster. MEGAN analysis of metagenomic data. *Genome research*, 17(3):377–386, March 2007.
5. L. Krause, N. N. Diaz, A. Goesmann, S. Kelley, T. W. Nattkemper, F. Rohwer, R. A. Edwards, and J. Stoye. Phylogenetic classification of short environmental DNA fragments. *Nucleic acids research*, 36(7):2230–2239, April 2008.
6. V. Kunin, A. Copeland, A. Lapidus, K. Mavromatis, and P. Hugenholtz. A Bioinformatician's Guide to Metagenomics. *Micr. Mol. Biol. Rev.*, 72(4):557–578, 2008.
7. O. Nalbantoglu, S. Way, S. Hinrichs, and K. Sayood. RAIphy: Phylogenetic classification of metagenomics samples using iterative refinement of relative abundance index profiles. *BMC Bioinformatics*, 12(1):41+, 2011.
8. A. Stamatakis. Raxml-vi-hpc: Maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. *Bioinformatics*, 22(21):2688–2690, 2006.
9. F. Warnecke and P. Hugenholtz. Building on basic metagenomics with complementary technologies. *Genome Biology*, 8(12):231+, December 2007.
10. J. C. Wooley, A. Godzik, and I. Friedberg. A Primer on Metagenomics. *PLoS Comput Biol*, 6(2):e1000667+, February 2010.
11. D. R. Zerbino and E. Birney. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome research*, 18(5):821–829, May 2008.

On the Performance of Sorting by Transpositions Without Using Cycle Graph

Gustavo Rodrigues Galvão and Zanoni Dias

University of Campinas, Institute of Computing, Brazil
gustavo.galvao@students.ic.unicamp.br
zanoni@ic.unicamp.br

Abstract. We study the problem of sorting by transpositions, which is related to genome rearrangements. Our goal is to determine how good approximation algorithms which do not rely on the cycle graph are when it comes to performance guarantee. We measure their performance on all permutations of up to 13 elements, and then we discuss the possibility of improving their approximation factor. We also correct previous results on the performance for one of these algorithms.

1 Introduction

In genome rearrangements, a transposition is the rearrangement event that switches the location of two contiguous portions of a chromosome. The problem of sorting by transpositions consists in finding the minimum number of transpositions needed to transform one genome into another. This problem is NP-hard [2], and the best approximation algorithms for solving it are based on a complex structure named cycle graph [3]. In an attempt to derive simpler algorithms, some researches posed new approaches to tackle this problem.

Walter, Dias and Meidanis [5] presented a 2.25-approximation algorithm based on a structure named breakpoint diagram. Benoît-Gagné and Hamel [1] developed a 3-approximation algorithm based on permutation codes. In this work we implement both algorithms, analyze their performance in practice, and present results indicating that the performance guarantee of the former may be lowered, but of the latter it probably can not. Furthermore, we show that previous results on the performance of the 2.25-approximation algorithm are incorrect.

The remainder of this paper is organized as follow. In next section, we give the definitions and notation of the paper, and we briefly describe the approximation algorithms. In Sect. 3, we present the experimental results along with a discussion on the approximation factor of the algorithms. The last section concludes the paper.

2 Preliminaries

We represent genomes as permutations, where genes appear as elements. A permutation π is a bijection of $\{1, 2, \ldots, n\}$ onto itself. The group of all permutations

of $\{1, 2, \ldots, n\}$ is denoted by S_n, and we write a permutation $\pi \in S_n$ as $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_n)$.

A transposition is an operation $\rho(i, j, k)$, $1 \leq i < j < k \leq n+1$, that moves blocks of contiguous elements of a permutation $\pi \in S_n$ in such way that $\rho(i, j, k) \cdot (\pi_1\ \ldots\ \pi_{i-1}\ \underline{\pi_i\ \ldots\ \pi_{j-1}}\ \underline{\pi_j\ \ldots\ \pi_{k-1}}\ \pi_k\ \ldots\ \pi_n) = (\pi_1\ \ldots\ \pi_{i-1}\ \underline{\pi_j\ \ldots\ \pi_{k-1}}\ \underline{\pi_i\ \ldots\ \pi_{j-1}}\ \pi_k\ \ldots\ \pi_n)$. The problem of sorting by transpositions consists in finding the minimum number of transpositions that transform a permutation $\pi \in S_n$ into identity permutation $I_n = (1\ 2\ \ldots\ n)$. This number is known as the transposition distance of permutation π and it is denoted by $d(\pi)$.

Given a permutation $\pi \in S_n$, the left and right codes of element π_i, denoted $lc(\pi_i)$ and $rc(\pi_i)$ respectively, are defined as $lc(\pi_i) = |\{\pi_j : \pi_j > \pi_i \text{ and } 1 \leq j \leq i-1\}|$ and $rc(\pi_i) = |\{\pi_j : \pi_j < \pi_i \text{ and } i+1 \leq j \leq n\}|$. The left (resp. right) code of permutation π is then defined as the sequence of lc's (resp. rc's) of its elements, and it is denoted by $lc(\pi)$ (resp. $rc(\pi)$).

Let us call plateau any maximal length sequence of contiguous elements in a number sequence that have the same nonzero value. The number of plateaux in a code c is denoted p(c). We denote by $p(\pi)$ the minimum of $p(lc(\pi))$ and $p(rc(\pi))$. Benoît-Gagné and Hamel [1] showed that it is always possible to remove a plateau of a permutation π, without creating a new one, by applying a transposition. Given that I_n is the only permutation having zero plateaux, an algorithm that removes one plateau at each iteration sorts π with $p(\pi)$ transpositions. Since Benoît-Gagné and Hamel [1] also showed that $1 \leq \frac{p(\pi)}{d(\pi)} \leq 3$, we concluded that such algorithm is a 3-approximation.

Given a permutation $\pi \in S_n$, we will extend it with two elements $\pi_0 = 0$ and $\pi_{n+1} = n+1$ (the extended permutation is still denoted π). We call position i of π a breakpoint if $\pi_{i+1} - \pi_i \neq 1$, $0 \leq i \leq n$. The number of breakpoints of π is denoted by $b(\pi)$. Note that I_n is the only permutation in S_n having zero breakpoints.

By using a structure based on breakpoints, Walter, Dias and Meidanis [5] were able to develop an approximation algorithm that removes at least 4 breakpoints with 3 transpositions. Thus, if the algorithm sorts a permutation π with $a(\pi)$ transpositions, we have $a(\pi) \leq \frac{3}{4}b(\pi)$. Since a transposition can eliminate at most 3 breakpoints, we have $d(\pi) \geq \frac{b(\pi)}{3}$. Then, $\frac{a(\pi)}{d(\pi)} = \frac{9}{4}$ in the worst case, and the algorithm is therefore a 2.25-approximation.

3 Results and Discussion

Both approximation algorithms were implemented and tested by its authors for verifying their performance in practice. One kind of test was to compare $p(\pi)$ (resp. $a(\pi)$) with $d(\pi)$ for every $\pi \in S_n$ in order to obtain the approximation factor (*i.e.* the maximum value of $\frac{p(\pi)}{d(\pi)}$ (resp. $\frac{a(\pi)}{d(\pi)}$)) of the respective approximation algorithm for small permutations. More specifically, Walter, Dias and Meidanis [5] ran this test for $1 \leq n \leq 11$, while Benoît-Gagné and Hamel [1] ran it for $1 \leq n \leq 9$. We ran this kind of test for $1 \leq n \leq 13$ for both algorithms, and

the results are presented in Table 1. The transposition distances were computed using a breadth first search method [4].

Table 1. Approximation factors observed for each approximation algorithm

	n									
Approximation Factor	4	5	6	7	8	9	10	11	12	13
2.25-approximation algorithm	1.00	1.00	1.34	1.34	1.50	1.50	1.50	1.60	1.60	1.60
3-approximation algorithm	1.00	1.50	1.67	2.00	2.00	2.00	2.25	2.25	2.25	2.40

We remark that the results obtained by us for the 2.25-approximation algorithm are different from the results obtained by Walter, Dias and Meidanis [5]. For instance, the maximum value of $\frac{a(\pi)}{d(\pi)}$ they observed for $n = 11$ was $\frac{10}{5}$. But this result can not be right because $a(\pi) \leq 9$ for every $\pi \in S_{11}$. Given a permutation $\pi \in S_n$, it is easy to see that $0 \leq b(\pi) \leq n + 1$. As discussed in Sect. 2, $a(\pi) \leq \frac{3}{4}b(\pi)$, therefore $a(\pi) \leq \frac{3n+3}{4}$, and the claim follows. Since the values we obtained for $a(\pi)$ are consistent with the breakpoint upper bound, we conclude that the approximation factors presented in Table 1 are correct.

Table 2. Permutations π_m of size $3m+1$, $m \in \{5, 6, 7\}$, for which $\frac{p(\pi_m)}{d(\pi_m)} = \frac{3m}{m+1}$. Note that $d(\pi_m) \geq \frac{b(\pi_m)}{3} \geq m + 1$

Permutation	**Transposition Sorting Sequence**
π_5 = (16 9 4 11 6 15 8 2 12 7 5 3 14 13 10 1)	$\rho(5, 9, 12)$, $\rho(1, 7, 10)$, $\rho(3, 9, 14)$, $\rho(5, 10, 17)$, $\rho(6, 10, 14)$, $\rho(1, 7, 11)$
π_6 = (19 11 4 18 6 14 8 13 10 2 15 5 9 7 3 17 16 12 1)	$\rho(4, 12, 17)$, $\rho(7, 12, 16)$, $\rho(6, 9, 15)$, $\rho(1, 5, 11)$, $\rho(3, 8, 20)$, $\rho(4, 13, 17)$, $\rho(1, 5, 13)$
π_7 = (22 13 4 21 6 17 8 16 10 15 12 2 18 5 11 9 7 3 20 19 14 1)	$\rho(4, 14, 20)$, $\rho(8, 13, 19)$, $\rho(7, 10, 18)$, $\rho(6, 9, 17)$, $\rho(1, 5, 11)$, $\rho(3, 8, 23)$, $\rho(4, 16, 20)$, $\rho(1, 5, 15)$

Regarding the 3-approximation algorithm, if we just consider the approximation factors obtained for $n \in \{7, 10, 13\}$, we can observe that they seem to follow the progression $\frac{6}{3}$, $\frac{9}{4}$, $\frac{12}{5}$, ..., $\frac{3k}{k+1}$. We ran further experiments to verify the strength of this assumption, and we found permutations π_m of size $3m + 1$, $m \in \{5, 6, 7\}$, for which $\frac{p(\pi_m)}{d(\pi_m)} = \frac{3m}{m+1}$ (these permutations are presented in Table 2). This result strongly indicates that the approximation factor of the 3-approximation algorithm is tight, contradicting the hypothesis raised by Benoît-Gagné and Hamel [1] that its approximation factor "tends to a number significantly smaller than 3".

The approximation factors observed for the 2.25-approximation algorithm seem to increase in a progression that converges to 2, that is, $\frac{2}{2}$, $\frac{4}{3}$, $\frac{6}{4}$, $\frac{8}{5}$, ...,

$\frac{2k}{k+1}$. This may indicate that a deeper analysis of this algorithm could lead one to prove that it is in fact a 2-approximation.

4 Conclusions

We conducted an experimental investigation on the approximation factors of the 2.25-approximation algorithm presented by Walter, Dias and Meidanis [5] and of the 3-approximation algorithm developed by Benoît-Gagné and Hamel [1] for sorting by transpositions. We verified that their approximation factors seem to increase in a progression that converges, respectively, to 2 and to 3, what suggests that the 2.25-approximation algorithm is in fact a 2-approximation, and that the approximation factor of the 3-approximation algorithm can be as close to 3 as we want it. Furthermore, we corrected the performance results presented by Walter, Dias and Meidanis [5] for the 2.25-approximation algorithm.

For future work, we plan to work on formal proofs for these hypothesis.

Acknowledgments. This work was supported by CNPq (processes 483177/2009-1, 200815/2010-5 and 135212/2010-3).

References

1. Benoît-Gagné, M., Hamel, S.: A new and faster method of sorting by transpositions. In: Proceedings of the 18th Annual Symposium on Combinatorial Pattern Matching (CPM'2007). pp. 131–141. Springer-Verlag, London, Ontario, Canada (2007)
2. Bulteau, L., Fertin, G., Rusu, I.: Sorting by transpositions is difficult. In: Proceedings of the 38th International Colloquium on Automata, Languages and Programming (ICALP'2011). Zürich, Switzerland (2011), to appear
3. Fertin, G., Labarre, A., Rusu, I., Tannier, E., Vialette, S.: Combinatorics of Genome Rearrangements. The MIT Press (2009)
4. Galvão, G.R., Dias, Z.: A flexible framework for computing rearrangement distance of every permutation in the symmetric group. In: Proceedings of the 6th Brazilian Symposium on Bioinformatics (BSB'11). Brasília, Brazil (2011), to appear
5. Walter, M.E.M.T., Dias, Z., Meidanis, J.: A new approach for approximating the transposition distance. In: Proceedings of the Seventh International Symposium on String Processing Information Retrieval (SPIRE'2000). pp. 199–208. IEEE Computer Society, Washington, DC, USA (2000)

Evaluation of genome distance measures using tree-derived distances

Ulisses Dias *, Zanoni Dias, and João C. Setubal

Institute of Computing, University of Campinas
{udias,zanoni}@ic.unicamp.br
Virginia Bioinformatic Institute, Virginia Tech
{setubal,zanoni}@vbi.vt.edu

Abstract. We present a method to evaluate prokaryotic genomic distance measures that uses distances derived from a reference phylogenetic tree. We employ the method to compare three genomic distance measures using as reference a maximum-likelihood tree containing 332 prokaryotic organisms. This methodology avoids the need to create phylogenetic trees to compare genomic distance measures.

Keywords: Whole-Genome Comparison

1 Introduction

In this work we present a method to evaluate prokaryotic genomic distance measures that uses distances derived from a reference phylogenetic tree. In an companion abstract in this conference [2] we present two new whole-genome distance measures, which we call NUCMi and PROMi. They were developed as an improvement over a previous proposed measure called MUMi [1].

In [2] we compared the three genomic distance measures by generating distance-based trees with the neighbor-joining algorithm and comparing the resulting trees with maximum-likelihood reference trees obtained from the literature. This was done for four bacterial groups.

Doing this comparison in this way is unsatisfactory because it involves two steps that are best avoided if possible: the first is to compute a tree using distances and the second is choosing a method to compare trees. In this work we show how to avoid these steps by comparing distances directly with a set of reference distances.

2 Methods

We derive a reference distance matrix from a reference tree. The tree we chose is that presented by Wu *et al.* [5]. In that work Wu and colleagues computed a

* This work was made possible in part by a PhD fellowship from FAPESP to UD (2007/05574-4), a sabbatical fellowship from CNPq to ZD (200815/2010-5) and by project funding from CNPq to ZD (483177/2009-1 and 473867/2010-9).

maximum-likelihood tree for prokaryotes using a concatenated alignment of 31 marker genes. A total of 350 organisms were included in the tree. We obtained an enlarged tree (with 720 organisms) from the authors in NEWICK format. From this tree we computed a distance matrix by simply measuring the distance between any two nodes on the tree as the sum of the branch lengths between each node and the lowest common ancestor of the two nodes. Such a method of constructing a distance matrix ensures that it is additive [4, Section 6.5]. The neighbor-joining algorithm will correctly reconstruct a tree if the input distance matrix is additive [3]. Hence the distance matrix obtained as above is a good proxy for the tree itself.

The distance matrix that we used as reference was trimmed to contain only 332 organisms from 17 different prokaryotic groups (each group is a phylum or a class). This trimming was necessitated to ensure that there was a perfect correspondence between the downloaded genome sequences used to compute distances and the genomes used in building the tree presented by Wu *et al.* [5]. In addition, we wanted each phylum or class represented to contain at least four genomes.

Once the organisms were defined we scanned all corresponding genomes to check for the presence of plasmids. Because plasmids are generally considered accessory elements in genomes, we wanted to base our distance measuring on chromosomes only. We therefore discarded plasmids, and concatenated chromosome sequences, when there was more than one chromosome for a given organism. For each organismal group we did all pairwise comparisons between the genomes in that group using MUMi, NUCMi, and PROMi.

The next issue is how to compare the reference matrix R and the query matrix A (the matrix derived from the distances computed by each of the measures). There are numerical methods that can compare matrices, but they do not seem to be appropriate to our case, because they focus solely on numerical differences between matrix entries. We wanted to have a more qualitative view, and hence have developed our own method. We use the notation $R(X, Y)$ to denote the distance between genome sequences X and Y as given by matrix R; analogously for A. We have adopted the following comparative methodology. Given three genome sequences X, Y and Z, if $A(X, Y) < A(X, Z)$ and $R(X, Y) > R(X, Z)$, or if $A(X, Y) > A(X, Z)$ and $R(X, Y) < R(X, Z)$, then we tabulate this as an *error;* otherwise we tabulate this as a *correct answer*. We score each query matrix A by the number of correct answers divided by the total number of genome sequence triplets.

3 Results

We present results using the prokaryotic groups derived from the reference tree. Our rationale for focusing on these groups is that we want to have an idea of how well the distance measures perform when the genomes under study are not too far apart.

Table 1. Mean value of pairwise distances for each group of organisms. All distance values can vary in the interval $[0, 1]$.

Bacterial group	# Organisms	NUCMi	PROMi	MUMi
actinobacteria	52	0.992	0.844	0.973
alphaproteobacteria	43	0.995	0.852	0.989
bacteroidetes	15	0.999	0.881	0.998
betaproteobacteria	18	0.987	0.761	0.981
chlamydiae	7	0.997	0.839	0.997
chlorobia	5	0.997	0.603	0.994
chloroflexi	5	0.999	0.869	0.997
cyanobacteria	18	0.998	0.734	0.996
deinococcus and thermus	4	0.980	0.637	0.977
deltaproteobacteria	19	0.998	0.885	0.993
epsilonproteobacteria	12	0.998	0.701	0.993
firmicutes	57	0.999	0.871	0.996
fusobacteriales	4	0.995	0.796	0.976
gammaproteobacteria	45	0.998	0.836	0.997
spirochaetales	7	0.995	0.878	0.993
tenericutes	16	0.997	0.866	0.989
thermotogae	5	0.998	0.664	0.994

We present two tables with results. Table 1 presents the mean distances computed for all pairwise comparisons of genomes within each group. The results in this table show that even restricting comparisons to genomes within groups results in NUCMi and MUMi distances that are rather large. This is expected, because these measures make comparisons based on nucleotide sequences. PROMi on the other hand resulted in smaller distance values.

Table 2 shows the scores obtained for each of the three distance measures. PROMi is clearly the best measure among the three. We have also done a comparison using all genomes. In this case PROMi averages 70.52% correct answers, NUCMi 64.37%, and MUMi 57.78%.

The main conclusion we can derive from these results is that PROMi is a better measure for comparing genomes at the phylum or class level, supporting the findings presented in [2]. In fairness to MUMi it must be pointed out that this measure was not developed to compare genomes that are as distant as those we have considered here. But this observation provides a justification for developing a measure such as PROMi.

4 Conclusion

We have presented a methodology to compare genomic distance measures using distances derived from a reference tree. We believe this methodology to have a more solid basis than the one presented in [2].

We want to stress the fact that this extended abstract presents work in progress. We expect to continue the development of genomic distance measures,

Table 2. Percentage of correct answers for each genomic distance measure.

Bacterial group	# Organisms	NUCMi (%)	PROMi (%)	MUMi (%)
actinobacteria	52	76.18	81.15	69.23
alphaproteobacteria	43	72.63	80.60	75.84
bacteroidetes	15	58.11	83.99	75.22
betaproteobacteria	18	65.45	76.14	64.84
chlamydiae	7	70.95	93.33	86.67
chlorobia	5	64.44	86.67	68.89
chloroflexi	5	55.56	86.67	82.22
cyanobacteria	18	62.42	78.60	72.73
deinococcus and thermus	4	80.00	93.33	73.33
deltaproteobacteria	19	71.65	76.98	60.70
epsilonproteobacteria	12	63.78	80.42	57.58
firmicutes	57	71.72	84.17	61.15
fusobacteriales	4	66.67	66.67	33.33
gammaproteobacteria	45	50.68	79.66	65.05
spirochaetales	7	67.14	87.62	61.43
tenericutes	16	80.59	75.42	62.69
thermotogae	5	51.11	95.56	64.44
Mean	19.52	66.42	82.76	66.78

so that the values in Table 2 will become even better. On the other hand, it must also be stressed that our simple genomic distance measures are not meant to be a replacement for careful phylogenetic analysis. They are tools that can be convenient for certain situations, especially those that require an approximation of phylogenetic distance that can be computed substantially faster than by computationally reconstructing phylogenetic trees.

5 Acknowledgments

We thank Jonathan Eisen and Dongying Wu for making their tree in the NEWICK format available to us.

References

1. Deloger, M., El Karoui, M., Petit, M.A.: A genomic distance based on MUM indicates discontinuity between most bacterial species and genera. J. Bacteriol. 191, 91–99 (2009)
2. Dias, U., Dias, Z., Setubal, J.C.: Two new whole-genome distance measures. In: Proceedings of the Brazilian Symposium on Bioinformatics (2011)
3. Saitou, N., Nei, M.: The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol. Biol. Evol. 4, 406–425 (1987)
4. Setubal, J.C., Meidanis, J.: Introduction to Computational Molecular Biology. PWS (1997)
5. Wu, D., et al.: A phylogeny-driven genomic encyclopaedia of bacteria and archaea. Nature 462, 1056–1060 (2009)

Machine Learning Approaches for miRNA Prediction

Ivani de O. N. Lopes[1,2], André C. P. de L. F. de Carvalho[2], and Eliseu Binneck[1]

[1] Empresa Brasileira de Pesquisa Agropecuária, Embrapa Soja,
Caixa Postal 231, Londrina-PR, CEP 86001-970, Brazil
{negrao,binneck}@cnpso.embrapa.br
[2] Instituto de Ciências Matemáticas e de Computação
Avenida Trabalhador São-carlense, 400 - Centro, São Carlos-SP, Brasil
andre@icmc.usp.br

Abstract. An obstacle to predict miRNAs from genomic RNA transcripts by computational approaches is their short size, which harm comparative genomic methods. *Ab initio* methods, which are based on either Machine Learning (ML) algorithms or features from the miRNAs precursors (pre-miRNA), have been shown valuable to find new, as well as the already known, pre-miRNAs. However, different ML algorithms can achieve different performance on the same task beyond the fact that they are highly dependent on the training dataset and the features selected to represent the problem. A prior study showed evidences that, for a given transcript, its negative examples would be useful to create a more specific classifier. Moreover, the search for miRNAs in a given transcript can generate thousands of candidates. Based on these informations, as part of the PhD project of the first author, we propose to investigate the potential of using pre-miRNA candidates, in addition to real and pseudo pre-miRNAs, to construct more accurate classifiers, through semi-supervised algorithms. Afterward, through a deep characterization of both the datasets and the algorithms, we also propose to draw a system that would automatically recommend ML algorithms to predict pre-miRNAs from genomic RNA transcripts.

Keywords: Metalearning, generative models, graph-based models, TSVMs

1 Introduction

MicroRNAs (miRNAs) are a specialized class of small silencing RNAs [8] that was recently recognized as important regulators of gene expression at the post-transcriptional level and have been found to be involved in plant stress responses [6]. Moreover, overexpression or underexpression of some miRNAs are linked to various human diseases like cancer [3].

Computational approaches for miRNA discovery are based on comparative genomics or *ab initio* methods. The former compares candidate sequences against

known miRNAs and the latter uses features from the precursor miRNA (pre-miRNA) sequences and from other hairpin sequences with similar stem-loop features (pseudo pre-miRNAs) to establish new miRNAs. Although comparative genomic-based methods provide important techniques to predict new miRNAs, they are unable to identify novel miRNAs for which there are no known close homologies, either due to the limitation of the data or due to the possible evolution of miRNAs [5]. *Ab initio* methods, on the other hand, have been shown valuable to distinguish real from pseudo pre-miRNA with close homology or not [5].

Current *ab initio* methods use supervised machine learning (ML) algorithms. Since pre-miRNAs are sequences of nucleotides and ML methods deals with features describing the sequences, quantitative features are extracted from both positive (real) and negative(pseudo) sequences. Previously applied ML approaches include: Support Vector Machines (SVMs) [11, 10], random-forest (RF) [5], kernel density estimator [2]. However, a reliable comparison among these classifiers are not possible to be performed, since they were trained by different algorithms, features and training datasets. Also, [12] showed that using negative training examples extracted from the transcript being considered, led to a more specific classifier. Therefore, by this work, we intend to perform a deep characterization of pre-miRNAs datasets of 88 species, stating meta-features useful to describe them. Then, inspired in [12], we propose to investigate semi-supervised learning algorithms as an approach to incorporate additional knowledge from pre-miRNA candidates, to obtain more accurate classifiers. Finally, through meta-learning methods and the characteristics obtained from the datasets and the algorithms adopted, we intend to design a system that would automatically recommend semi-supervised algorithms to predict pre-miRNAs from a specific genomic RNA transcript.

2 Methods and Proposal

Semi-Supervised Learning (SSL) algorithms are an important class of ML algorithms that can benefit from either labeled (real and pseudo pre-miRNAs) and unlabeled objects (candidates). Most classes of SSL algorithms were constructed under certain assumptions about the datasets. Here we are going to explore some of these classes, in an attempt to define more suitable assumptions to the actual domain. Because of its flexibility, generative models offers a good way to benefit from domain specific knowledge. Graph-based models use the graph structure obtained by capturing pairwise similarities between labeled and unlabeled cases. Since the most applied algorithm on *ab initio* tools for miRNAs prediction is SVMs, Transductive SVMs (TSVMs) were included as natural adaptation of the usual ones. However, transduction are also going to be investigated in the context of graph-based models, as in [4]. It is good to reinforce that some algorithms can perform simultaneously transductive and inductive learning.

As described by [1], metalearning is concerned with accumulating experience on the performance of multiple applications of a learning system. The key point

in metalearning is to design approaches to gather knowledge about the learning process, also known as *metaknowledge*. Metaknowledge can be defined as any kind of knowledge that is derived in the course of employing a given learning system [1]. As examples of metaknowledge, [1] cited: sets of learning algorithms that have shown good (a priori) performance on datasets similar to the one under analysis; algorithms to characterize ML algorithms and datasets and metrics available to compute dataset similarity or task relatedness. All these metaknowledges form a dataset called metadata, which is accessed when new datasets arrive to be analyzed. Through approaches of matching and search, the new dataset is contrasted with the metadata. So, based on descriptive measures, an algorithm is recommended for that task. Nevertheless, these processes only intend to guide the specialist on the hard task of choosing the most appropriated ML algorithm for specific tasks.

The domain to be explored involves sets of validated (labeled as pseudo or true miRNAs) pre-miRNAs sequences of species from the same or different kingdoms [7]. Prior applications [11, 10] of ML algorithms to distinguish real from pseudo miRNA showed evidences that their performance can degrade among species, even from the same kingdom. Besides, the different characteristics of pre-miRNAs in animals and plants have justified different approaches to predict them [9]. The first phase of this work is to characterize the datasets adopted and to set the performance of SSL algorithms from different classes on those datasets (formed by candidates, true and pseudo pre-miRNAs). Then, through metalearning methods, we are going to characterize those algorithms as well as the datasets. The product of these phases is the so called metadata. These metadata is going to be exploited by metalearning methods to answer the following questions:

- How to select the labeled training dataset?
- Are pre-miRNA candidates useful to enhance a classifier?
- If the answer for the second question is yes, how many candidates should be added to the learner?
- Given a set of unlabeled sequences, what is a minimum number of labeled sequences to train the classifier?
- What are the best meta-features to define the training dataset?
- What are the most plausible semi-supervised assumption for pre-miRNAs datasets?

After analyzing the previous questions, we hope to propose a system that would automatically recommend semi-supervised algorithms to distinguish real from pseudo miRNAs from specific species.

3 Conclusions

In this work we presented the problem of pre-miRNA findings in genomic transcripts, as our rationale toward to elucidate the learning mechanism by Machine

Learning (ML) tools. It was emphasized that ML algorithms have great potential to train classifiers by using features describing sequences of pseudo and real pre-miRNAs. We introduced the idea of using pre-miRNAs candidates, in addition to validated pre-miRNAs, to enhance the accuracy of a given classifier, through semi-supervised algorithms. By applying metalearning methods to obtain a better characterization of the miRNA datasets and semi-supervised learning applied, we proposed to draw a system to automatically recommend ML algorithms to predict pre-miRNAs from a genomic transcript.

Acknowledgments. The authors are thankful to CNPq and Embrapa Soybean by their financial support. We also thanks to Bruno Feres by his review and comments on the draft of this work.

References

1. Brazdil, P., Giraud-Carrier, C., Soares, C., Vilalta, R.: Combining Base-Learners, pp. 73–77. Springer Publishing Company, Incorporated (Nov 2008)
2. Chang, D., Wang, C.C., Chen, J.W.: Using a kernel density estimation based classifier to predict species-specific microRNA precursors. BMC Bioinformatics 9(Suppl 12) (2008)
3. Chatterjee, S., Grosshans, H.: Active turnover modulates mature microRNA activity in C. elegans. Nature 461(7263), 546–549 (Sep 2009)
4. Culp, M., Michailidis, G.: Graph-based semisupervised learning. IEEE Trans. Pattern Anal. Mach. Intell. 30(1), 174–179 (2008)
5. Jiang, P., Wu, H., Wang, W., Ma, W., Sun, X., Lu, Z.: MiPred: classification of real and pseudo microRNA precursors using random forest prediction model with combined features. Nuc. Ac. Res. 35(suppl 2), W339–W344 (Jul 2007)
6. Khraiwesh, B., Zhu, J.K., Zhu, J.: Role of mirnas and sirnas in biotic and abiotic stress responses of plants. Biochimica et Biophysica Acta (BBA) - Gene Regulatory Mechanisms In Press, Corrected Proof, – (2011)
7. Kozomara, A., Griffiths-Jones, S.: miRBase: integrating microRNA annotation and deep-sequencing data. Nucleic Acids Research 39(suppl 1), D152–D157 (Jan 2011)
8. Lee, R.C., Ambros, V.: An Extensive Class of Small RNAs in Caenorhabditis elegans. Science 294(5543), 862–864 (Oct 2001)
9. Mendes, N.D., Freitas, A.T., Sagot, M.F.F.: Current tools for the identification of miRNA genes and their targets. Nucleic acids research 37(8), 2419–2433 (May 2009)
10. Xuan, P., Guo, M., Liu, X., Huang, Y., Li, W., Huang, Y.: PlantMiRNAPred: efficient classification of real and pseudo plant pre-miRNAs. Bioinformatics 27(10), 1368–1376 (May 2011)
11. Xue, C., Li, F., He, T., Liu, G.P., Li, Y., Zhang, X.: Classification of real and pseudo microRNA precursors using local structure-sequence features and support vector machine. BMC Bioinformatics 6 (2005)
12. Yousef, M., Nebozhyn, M., Shatkay, H., Kanterakis, S., Showe, L.C.C., Showe, M.K.K.: Combining multi-species genomic data for microRNA identification using a Naive Bayes classifier machine learning for identification of microRNA genes. Bioinformatics (Mar 2006)

In silico Identification of Cytokines Sequence Patterns

William F. Porto, Osmar N. Silva, Ludovico Migliolo and Octávio L. Franco

Centro de Análises Proteômicas e Bioquímicas, Pós-Graduação em Ciências Genômicas e Biotecnologia, Universidade Católica de Brasília, 70790-160, Brasília-DF, Brazil
{williamfp7, osmarns, kingvico, ocfranco}@gmail.com

Cytokines play a crucial role in immune system, modulating several processes such as cell recruitment, clone expansion, and communication among the immune cell system [1]. Moreover, these molecules also play an essential role in autoimmune diseases, such as psoriasis, lupus and rheumatoid arthritis. In this view, the study of cytokines may bring novel advances in the treatment of autoimmune diseases, through of design and development novel drugs. This study was carried out in order to identify cytokines sequence patterns, aiming to design novel drugs with minor side effects than glucocorticoids, since these last are known for DNA binding function, while cytokines attach only to cognate receptors.

Sequence patterns were generated by Pratt 2.1 [2] using 153 cytokines sequences from SwissProt. Figure 1 shows the logo for the best pattern. Based on this issue, 10.000 random sequences were generated. These sequences were predicted on ChemoPred 1.0 [1]. Only 242 sequences were negatively predicted. These predictions show that the pattern sequences might act as cytokines, once they have the binding site motif BBXB with three basic residues (B) and any other (X) in 40s loop.

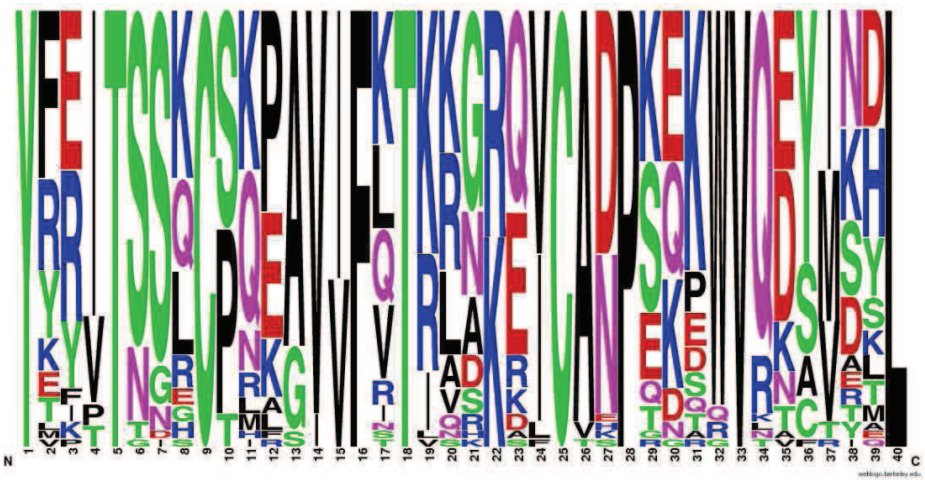

Fig. 1. Frequency logo for the best generated pattern.

In order to investigate the structural conservation of this pattern, two sequences were designed being one hydrophilic and other hydrophobic, once the hydrophobic interactions among the amino acids can change the tertiary structure. Their 3D models

were generated by Modeller 9v8 [3] using the structure PDB 2EOT as template for hydrophilic sequence (35.0 % of identity); and PDB 2VXW for the hydrophobic one (27.5 % of identity). The predicted structure from both sequences was composed of two antiparallel β-sheets and one α-helix (Figure 1). Both structures are possibly stables according to Ramachandran Plot. The predicted structure of hydrophilic sequence shows 91.4% of residues in favored and 8.6% in allowed ones; while the predicted structure of hydrophobic one shows 94.1% of residues in favored ones and 2.9% in allowed ones. Comparisons among the structures of designed and natural sequences were performed by Dali Server [4]. The structural alignments show that the structures are similar; alignments show Z-Sores ranging from 3.2 and 5.9. Figure 2 shows the structures analyzed.

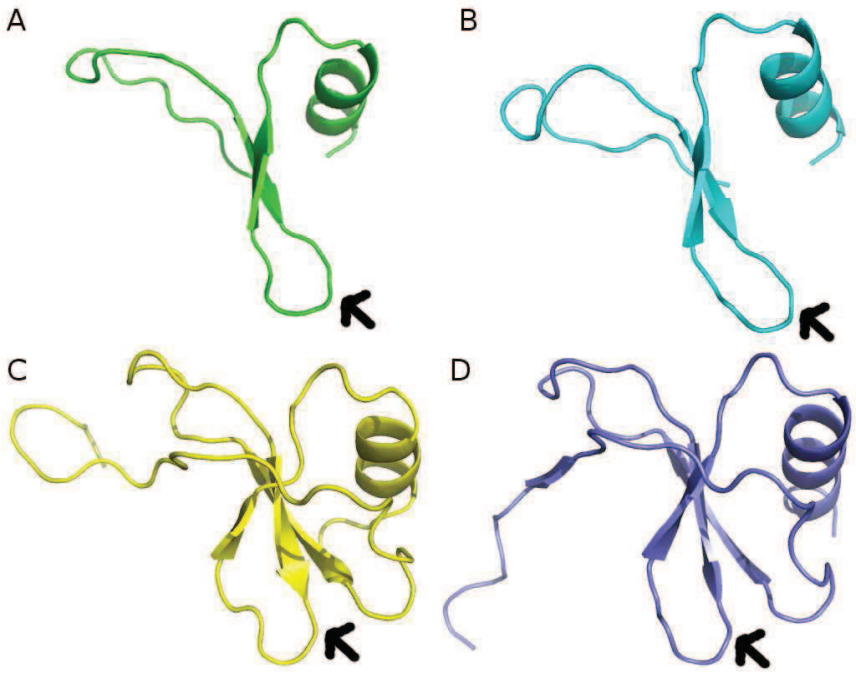

Fig. 2. Structures analyzed on Dali. Arrows indicate the 40s loop. (A) Hydrophilic sequence; (B) Hydrophobic sequence; (C) Human RANTES (PDB: 1HRJ); and (D) 44-AANA-47 RANTES.

The close relationship between the pattern and the cytokine 3D structure, the presence of BBXB motif and the high rates of positive predictions in ChemoPred, the pattern here described may allow the engineering of novel drugs to immune system. In a near future, the heterologous expression of the designed sequences will be done, and their activities evaluated *in vitro* and *in vivo*, in order to develop novel drugs with activity toward auto-immune diseases.

References

1. Lata, S., Raghava, G.P.S.: Prediction and Classification of Chemokines and their Receptors. PEDS 7(22), 441--444 (2009)
2. Jonassen, I., Collins, J. F., Higgins, D.: Finding Flexible Patterns in Unaligned Protein Sequences. Prot. Sci. 4(8), 1587--1595 (1995)
3. Fiser, A., Sali, A.: Modeller: Generation and Refinement of Homology-Based Protein Structure Models. Methods in Enzymology 374, 461--491 (2003)
4. Holm, L., Rosenström, P.: Dali Server: Conservation Mapping in 3D. Nucl. Acids Res. 38, W545--549 (2010)

Design of Novel Primers for Screen Cyclotides Precursors from Violaceae and Rubiaceae Families: Deciphering the Cyclotide Diversity on Brazilian Plants

Renato G. Almeida, William F. Porto, Octávio L. Franco, and Simoni C. Dias

Centro de Análises Proteômicas e Bioquímicas, Pós-Graduação em Ciências Genômicas e Biotecnologia, Universidade Católica de Brasília, 70790-160, Brasília-DF, Brazil
{renatogda, williamfp7, ocfranco, si.camposdias}@gmail.com

Plant cyclotides compose a family of circular proteins with ~29 amino acid residues length. These proteins have been commonly found in Violaceae, Rubiaceae and Curcubitaceae families, and more recently, in Apocynaceae, Poaceae and Fabaceae too [1-2]. Those proteins have a macrocyclic backbone and their main core is composed of a cysteine knot motif composed of six cysteines, forming a clear pattern of disufide bounds (Cys^1-Cys^4, Cys^2-Cys^5 and Cys^3-Cys^6). Furthermore, several biological activities have been reported to cyclotides, such as antimicrobial, antiviral, uterotonic and insecticidal. Due to the large range of activities and stable structure of cyclotides, those peptides seems to be an important part of the host plant defense system, presenting enormous biotechnological potential.

Lately, more than 340 cyclotides sequences have been listed in Cybase [3], despite of the fact that their wider distribution among other plant families remains unclear. These peptides have been identified by several molecular tools, including genomics, transcriptomics, proteomics and bioinformatics from various cultivated or wild plants [4]. The combination of these fields can bring novel insights into the distribution of cyclotides in plant kingdom. In summary, the study here presented reports the design of novel primers to discover novel cyclotide precursors in Violaceae and Rubiaceae plants in order to decipher the cyclotide distribution of Brazilian plants at different biomes.

Starting from 42 complete precursors in Cybase, 32 cDNA sequences were recovered by tBLASTn [5]. Primers were designed based on several rounds of multiple alignments. They were compared to the degenerated codon usage primer for AAFALPA (Figure 1), a conserved motif in the cyclotide precursor ER signal sequence [2].

Fig. 1. Frequency logo for AAFALPA's cDNA.

Comparisons among the primer for AAFALPA and designed primers show that AAFALPA has a major percentage of CG, and show eight wildcards, but it is more specific for 14 precursors (Table 1). On the other hand, designed primers are more

general, taking up to 22 precursors (data not shown) and have, generally, less wildcards (Table 1).

Table 1. Sample of Designed Primers. For Violaceae, three different primers were developed (Violaceae 1, 2 and 3).

Source	Primer	%CG	Sequences	Wildcards
Rubiaceae	CCTGWGCTTGCTTCTW GCAGC	57.14	4	2
Violaceae 1	TBCTSRTYGCHRCCTT TG	38.88	15	6
Violaceae 2	GTGCTCWYTGCMRCHT TYGC	45.00	9	6
Violaceae 3	ATGGAGAGCAACAAGA AGATGC	45.45	3	0
AAFALPA	GCHGCYTTYGCKCTHC CHGCH	52.38	14	7

In a near future is expected, by using RACE and RT-PCR methods, the identification of multiple cDNA clones in Brazilian plants using these novel primers. These cyclotides might be used to develop novel biotechnological tools, such as transgenic plants and/or novel drugs.

References

1. Poth, A.G., Colgrave, M.L., Philip, R., Kerenga, B., Daly, N.L., Anderson, M.A., Craik, D.J.Discovery of Cyclotides in the Fabaceae Plant Family Provides New Insights into the Cyclization, Evolution, and Distribution of Circular Proteins. ACS Chem Biol 6(4), 345--355 (2011)
2. Burman, R., Gruber, C.W., Rizzardi, K., Herrmann, A., Craik, D.J., Gupta, M.P., Göransson, U.: Cyclotide proteins and precursors from the genus Gloeospermum: filling a blank spot in the cyclotide map of Violaceae. Phytochemistry 71, 13--20 (2010)
3. Wang, C.K., Kaas, Q., Chiche, L., Craik, D.J.: CyBase: a database of cyclic protein sequences and structures, with applications in protein discovery and engineering. Nucl Acids Res 36, D206--D210 (2008)
4. Pestana-Calsa, M.C., Ribeiro, I.L., Calsa, T. Jr.: Bioinformatics-coupled molecular approaches for unravelling potential antimicrobial peptides coding genes in Brazilian native and crop plant species. Curr Protein Pept Sci, 11(3), 199--209 (2010)
5. Altschul, S.F., Madden, T.L., Schäffer, A.A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.J.: Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucl Acids Res 25(17), 3389--3402 (1997)

In silico Interactions Studies of a Cyclotide Peptide Docked with DPC Micelle

Migliolo, L.[1], Pinto, M.F.S.[1]; Fensterseifer , I.C.M.[1]; Silva, O.N.[1]; Porto, W.F.[1]; Amaro, D.[1]; Arboleda, J.[2]; Colgrave, M.L.[3]; Craik, D.J.[3]; Magalhães, B.S.[1]; Dias, S.C.[1]; Franco, O.L.[1*]

[1] Centro de Análises Proteômicas e Bioquímicas, Pós-Graduação em Ciências Genômicas e Biotecnologia, Universidade Católica de Brasília, 70790-160, Brasília, DF, Brazil

[2] Embrapa Recursos Genéticos e Biotecnologia, PqEB, Final W5 Norte, 70770-900, Brasília, DF, Brazil

[3] Institute for Molecular Bioscience, University of Queensland, Brisbane, Queensland-Australia.

[*]Corresponding author: ocfranco@gmail.com

Plant cyclotides are a family of circular cysteine-rich peptides with ~29 amino acid residues mainly found in Poaceae, Violaceae, Curcubitaceae and Rubiaceae family [1]. This report aims to study the molecular interaction of the cyclotide peptide Pr-br1, previously purified from *Palicourea rigida* (Rubiaceae) leaves, with DPC (mimetic erythrocytes membranes) micelles relaxed by molecular dynamics.

The peptide showed *in vitro* hemolytic activity (40 %) toward fresh human erythrocytes from healthy donors in a concentration of 40 + 0.02 μM. PSI-BLAST [2] was used in order to find best templates for homology modeling for tridimensional model construction. An alignment was realized between the template pdb 2eri (a Rubiaceae cyclotide peptide) showing 77 % of identity when compared with the Pr-br1 primary sequence [3]. Fifty theoretical tridimensional models were constructed by Modeller v.9.8 [4]. The final model was evaluated by using PROSA II and PROCHECK [5, 6]. In addition, RMSD was calculated by overlap of Cα traces and backbones onto the template structure through the program 3DSS [7]. The protein structures were visualized and analyzed on Delano Scientific's PyMOL. Molecular docking HEX v.6.1 [8] program was used to elucidate possible modes of interaction of peptide with DPC micelle consisting of 65 DPC lipids, after relaxation of 1100 ps. A list of 100 complexes of peptide-DPC was generated in order to rank and evaluate according spatial restraint, salt and hydrogen bounds formation as well hydrophobic interaction.

Tridimensional peptide structure showed two antiparallel β-sheets, a 3_{10} helix and six conserved cysteine residues involved in three disulfide bonds between six loops (Figure 1A). The complex results indicate that Pr-br1 interact with the micelle although of hydrophobic interactions, as observed in loop 2 (Phe^{7} and Iso^{8}) and 3 (Iso^{11}, Ser^{13} and Leu^{14}). These loops present hydrophobic ratio of 83 % (loop 2) and 71 % (loop 3) and distances among 2.5 at 3.5 Å (Figure 1B).

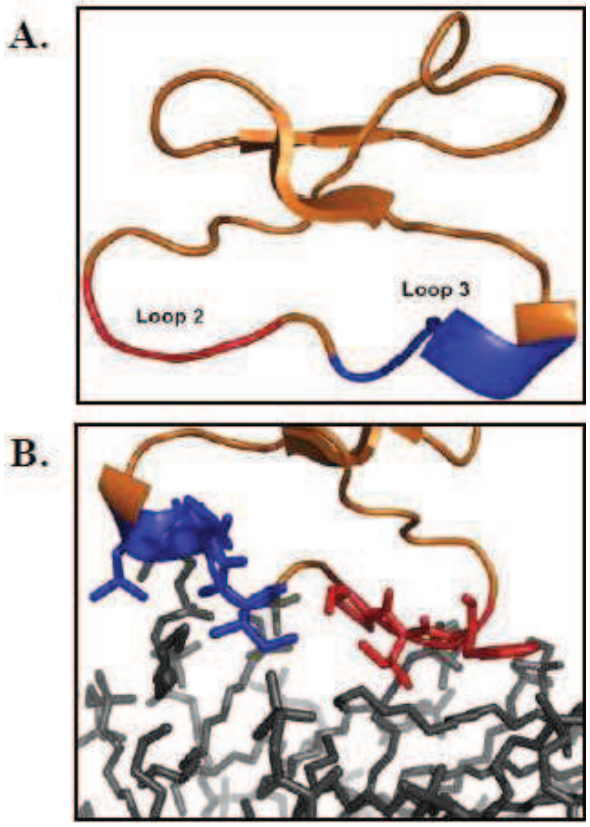

Fig. 1. Tridimensional model of Pr-br1 and our interaction with a micelle consisting of 65 DPC lipids. In blue is represented red loop 2 and loop 3.

In conclusion the work present a cyclic hemolytic peptide in lower concentration and interaction here described with neutral DPC seems to be driven by loops in accordance of data previously described in literature.

References

1. Jennings, C., West, J., Waine, C., Craik, D., Anderson, M.: Biosynthesis and insecticidal properties of plant cyclotides: The cyclic knotted proteins from *Oldenlandia affinis* Proc Natl Acad Sci U S A 98 (19), 10614--10619 (2001)
2. Altschul, S.F., Madden, T.L., Schäffer, A.A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.J.: Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucl Acids Res 25(17), 3389--3402 (1997)
3. Koltay, A., Daly, N.L., Gustafson, K.R., Craik, D.J.: Structure of Circulin B and Implications for Antimicrobial Activity of the Cyclotides. Int J Pept Res Ther 11, 99--106 (2005)

4. Fiser, A., Sali, A.: Modeller: Generation and Refinement of Homology-Based Protein Structure Models. Methods in Enzymology 374, 461--491 (2003)
5. Wiederstein, M., Sippl, M.J.: ProSA-web: interactive web service for the recognition of errors in three-dimensional structures of proteins. Nucl Acids Res 35, W407--W410 (2007)
6. Laskowski, R.A., MacArthur, M.W., Moss, D.S., Thornton, J.M.: PROCHECK: a program to check the stereochemical quality of protein structures. J Appl Cryst 26, 283--291 (1993)
7. Sumathi, K., Ananthalakshmi, P., Roshan, M.N., Sekar, K.: 3dSS: 3D structural superposition. Nucl Acids Res 34, W128--132 (2006)
8. Ritchie DW, Venkatraman V.: Ultra-fast FFT protein docking on graphics processors. Bioinformatics 26(19), 2398--2405 (2010)

Author Index