Philipp Bücher
Bernard M.E. Moret (Eds.)

LNBI 4175

Algorithms in Bioinformatics

6th International Workshop, WABI 2006
Zurich, Switzerland, September 2006
Proceedings

Springer

Philipp Bücher Bernard M.E. Moret (Eds.)

Algorithms in Bioinformatics

6th International Workshop, WABI 2006
Zurich, Switzerland, September 11-13, 2006
Proceedings

Springer

Series Editors

Sorin Istrail, Brown University, Providence, RI, USA
Pavel Pevzner, University of California, San Diego, CA, USA
Michael Waterman, University of Southern California, Los Angeles, CA, USA

Volume Editors

Philipp Bücher
Ecole Polytechnique Fédérale de Lausanne, Switzerland
E-mail: Philipp.Bucher@isrec.unil.ch

Bernard M.E. Moret
Ecole Polytechnique Fédérale de Lausanne, Switzerland
E-mail: bernard.moret@epfl.ch

Library of Congress Control Number: 2006932026

CR Subject Classification (1998): F.1, F.2.2, E.1, G.1-3, J.3

LNCS Sublibrary: SL 8 – Bioinformatics

ISSN 0302-9743
ISBN-10 3-540-39583-0 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-39583-6 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11851561 06/3142 5 4 3 2 1 0

Lecture Notes in Bioinformatics 4175

Edited by S. Istrail, P. Pevzner, and M. Waterman

Subseries of Lecture Notes in Computer Science

Preface

We are very pleased to present the proceedings of the *Sixth Workshop on Algorithms in Bioinformatics (WABI 2006)*, which took place in Zürich on September 11-13, 2006, under the auspices of the *International Society for Computational Biology (ISCB)*, the *European Association for Theoretical Computer Science (EATCS)*, and the *Eidgenössische Technische Hochschule Zürich (ETHZ)*.

The *Workshop on Algorithms in Bioinformatics* covers research on all aspects of algorithmic work in bioinformatics. The emphasis is on discrete algorithms that address important problems in molecular biology, that are founded on sound models, that are computationally efficient, and that have been implemented and tested in simulations and on real datasets. The goal is to present recent research results, including significant work-in-progress, and to identify and explore directions of future research. Specific topics of interest include, but are not limited to:

- Exact, approximate, and machine-learning algorithms for genomics, sequence analysis, gene and signal recognition, alignment, molecular evolution, population genetics and nucleotide polymorphism, structure determination or prediction, gene expression and gene networks, proteomics, functional genomics, and drug design.
- Methods, software and dataset repositories for the development and testing of such algorithms and their underlying models.
- High-performance approaches to computationally hard problems in bioinformatics, particularly optimization problems.

A major goal of the workshop is to bring together researchers spanning the range from abstract algorithm design to biological dataset analysis, so as to enable a dialogue between application specialists and algorithm designers, mediated by algorithm engineers and high-performance computing specialists. We believe that such a dialogue is necessary for the progress of computational biology, inasmuch as application specialists cannot analyze their datasets without fast and robust algorithms and, conversely, algorithm designers cannot produce useful algorithms without being conversant with the problems faced by biologists.

Part of this mix has been achieved for all six *WABI* events to date by collocating *WABI* with the *European Symposium on Algorithms (ESA)*, along with other occasional conferences or workshops, so as to form the interdisciplinary scientific meeting known as *ALGO*. This year, *ALGO 2006* comprised the *14th European Symposium on Algorithms (ESA 2006)*, the *6th Workshop on Algorithms in Bioinformatics (WABI 2006)*, the *4th Workshop on Approximation and Online Algorithms (WAOA 2006)*, the *2nd International Workshop on Parameterized and Exact Computation (IWPEC 2006)*, and the *6th Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS 2006)*.

We received 100 submissions in response to our call for *WABI 2006* and were able to accept 36 of them, ranging from mathematical tools to experimental

studies of approximation algorithms and reports on significant computational analyses. Numerous biological problems are dealt with, including genetic mapping, sequence alignment and sequence analysis, phylogeny, comparative genomics, and protein structure. This year was the first in which *WABI* also called for machine-learning approaches along with combinatorial optimization, and we are delighted to feature five contributions from this area.

We would like to thank all authors for submitting their work to the workshop and all the presenters and attendees for their participation. We were particularly fortunate in enlisting the help of a very distinguished panel of researchers for our program committee, which undoubtedly accounts for the large number of submissions and the high quality of the presentations. Our heartfelt thanks go to all:

Vincent Berry (U. Montpellier)
Rita Casadio (U. di Bologna)
Phoebe Chen (Deakin U.)
Nadia El-Mabrouk (U. Montréal)
Raffaele Giancarlo (U. di Palermo)
David Gilbert (U. Glasgow)
Roderic Guigo (U. Pompeu Fabra)
Vasant Honavar (Iowa State U.)
Daniel Huson (U. Tübingen)
Jens Lagergren (KTH Stockholm)
C. Randal Linder (U. Texas Austin)
Joao Meidanis (U. Campinas)
Satoru Miyano (Tokyo U.)
Gene W. Myers (HHMI Janelia Farm)
Luay Nakhleh (Rice U.)
Cedric Notredame (CNRS Marseilles)
Sven Rahmann (U. Bielefeld)
Knut Reinert (Freie U. Berlin)
Mikhail Roytberg (Russian Academy of Sciences)
Marie-France Sagot (U. Claude Bernard)
David Sankoff (U. Ottawa)
Joao Setubal (U. Campinas)
Adam Siepel (Cornell U.)
Jijun Tang (U. South Carolina)
Olga Troyanskaya (Princeton U.)
Alfonso Valencia (CNB-CSIC)
Jaak Vilo (Egeen Inc.)
Tandy Warnow (U. Texas Austin)
Lusheng Wang (City U. Hong Kong)
Tiffani Williams (Texas A&M U.)
Louxin Zhang (National U. Singapore)

We were fortunate to attract Ron Shamir, from Tel Aviv University, to address the joint conferences on topics in computational biomedicine, along with other distinguished speakers lecturing in more classical algorithmic areas: Erik Demaine (Massachusetts Institute of Technology), Lisa Fleischer (IBM T.J. Watson Research Labs), László Lovász (Eőtvős Loránd University and Microsoft Research), and Kurt Mehlhorn (Max-Planck-Institute Saarbrücken).

Last but not least, we thank Michael Hoffman and his colleagues Angelika Steger, Emo Welzl, and Peter Widmayer, all at ETHZ, for doing a superb job of organizing the joint conferences.

We hope that you will consider contributing to future *WABI* events, through a submission or by participating in the workshop.

September 2006

Phillip Bücher and Bernard M.E. Moret
WABI'06 Program Co-Chairs

Lecture Notes in Bioinformatics

Vol. 4175: P. Bücher, B.M.E. Moret (Eds.), Algorithms in Bioinformatics. XII, 402 pages. 2006.

Vol. 4146: J.C. Rajapakse, L. Wong, R. Acharya (Eds.), Pattern Recognition in Bioinformatics. XIV, 186 pages. 2006.

Vol. 4115: D.-S. Huang, K. Li, G.W. Irwin (Eds.), Computational Intelligence and Bioinformatics, Part III. XXI, 803 pages. 2006.

Vol. 4075: U. Leser, F. Naumann, B. Eckman (Eds.), Data Integration in the Life Sciences. XI, 298 pages. 2006.

Vol. 4070: C. Priami, X. Hu, Y. Pan, T.Y. Lin (Eds.), Transactions on Computational Systems Biology V. IX, 129 pages. 2006.

Vol. 3939: C. Priami, L. Cardelli, S. Emmott (Eds.), Transactions on Computational Systems Biology IV. VII, 141 pages. 2006.

Vol. 3916: J. Li, Q. Yang, A.-H. Tan (Eds.), Data Mining for Biomedical Applications. VIII, 155 pages. 2006.

Vol. 3909: A. Apostolico, C. Guerra, S. Istrail, P. Pevzner, M. Waterman (Eds.), Research in Computational Molecular Biology. XVII, 612 pages. 2006.

Vol. 3886: E.G. Bremer, J. Hakenberg, E.-H.(S.) Han, D. Berrar, W. Dubitzky (Eds.), Knowledge Discovery in Life Science Literature. XIV, 147 pages. 2006.

Vol. 3745: J.L. Oliveira, V. Maojo, F. Martín-Sánchez, A.S. Pereira (Eds.), Biological and Medical Data Analysis. XII, 422 pages. 2005.

Vol. 3737: C. Priami, E. Merelli, P. Gonzalez, A. Omicini (Eds.), Transactions on Computational Systems Biology III. VII, 169 pages. 2005.

Vol. 3695: M.R. Berthold, R.C. Glen, K. Diederichs, O. Kohlbacher, I. Fischer (Eds.), Computational Life Sciences. XI, 277 pages. 2005.

Vol. 3692: R. Casadio, G. Myers (Eds.), Algorithms in Bioinformatics. X, 436 pages. 2005.

Vol. 3680: C. Priami, A. Zelikovsky (Eds.), Transactions on Computational Systems Biology II. IX, 153 pages. 2005.

Vol. 3678: A. McLysaght, D.H. Huson (Eds.), Comparative Genomics. VIII, 167 pages. 2005.

Vol. 3615: B. Ludäscher, L. Raschid (Eds.), Data Integration in the Life Sciences. XII, 344 pages. 2005.

Vol. 3594: J.C. Setubal, S. Verjovski-Almeida (Eds.), Advances in Bioinformatics and Computational Biology. XIV, 258 pages. 2005.

Vol. 3500: S. Miyano, J. Mesirov, S. Kasif, S. Istrail, P. Pevzner, M. Waterman (Eds.), Research in Computational Molecular Biology. XVII, 632 pages. 2005.

Vol. 3388: J. Lagergren (Ed.), Comparative Genomics. VII, 133 pages. 2005.

Vol. 3380: C. Priami (Ed.), Transactions on Computational Systems Biology I. IX, 111 pages. 2005.

Vol. 3370: A. Konagaya, K. Satou (Eds.), Grid Computing in Life Science. X, 188 pages. 2005.

Vol. 3318: E. Eskin, C. Workman (Eds.), Regulatory Genomics. VII, 115 pages. 2005.

Vol. 3240: I. Jonassen, J. Kim (Eds.), Algorithms in Bioinformatics. IX, 476 pages. 2004.

Vol. 3082: V. Danos, V. Schachter (Eds.), Computational Methods in Systems Biology. IX, 280 pages. 2005.

Vol. 2994: E. Rahm (Ed.), Data Integration in the Life Sciences. X, 221 pages. 2004.

Vol. 2983: S. Istrail, M.S. Waterman, A. Clark (Eds.), Computational Methods for SNPs and Haplotype Inference. IX, 153 pages. 2004.

Vol. 2812: G. Benson, R.D. M. Page (Eds.), Algorithms in Bioinformatics. X, 528 pages. 2003.

Vol. 2666: C. Guerra, S. Istrail (Eds.), Mathematical Methods for Protein Structure Analysis and Design. XI, 157 pages. 2003.

Table of Contents

Measures of Codon Bias in Yeast, the tRNA Pairing Index and Possible DNA Repair Mechanisms

Markus T. Friberg[1], Pedro Gonnet[1], Yves Barral[2],
Nicol N. Schraudolph[3,4], and Gaston H. Gonnet[1]

[1] Institute of Computational Science, ETH Zurich, 8092 Zurich, Switzerland
[2] Institute of Biochemistry, Department of Biology, ETH Zurich, Switzerland
[3] Statistical Machine Learning, National ICT Australia, Canberra ACT 2601, Australia
[4] RSISE, Australian National University, Canberra ACT 0200, Australia

Abstract. Protein translation is a rapid and accurate process, which has been optimized by evolution. Recently, it has been shown that tRNA reusage influences translation speed. We present the tRNA Pairing Index (TPI), a novel index to measure the degree of tRNA reusage in any gene. We describe two variants of the index, how to combine various such indices to a single one and an efficient algorithm for their computation. A statistical analysis of gene expression groups indicate that cell cycle genes have high TPI. This result is independent of other biases like GC content and codon bias. Furthermore, we find an additional unexpected codon bias that seems related to a context sensitive DNA repair.

1 Introduction

Protein translation is a rapid and accurate process, despite the need to discriminate between many possible incoming and competing tRNAs. One can assume that the process has been optimized by evolution. It has been shown that tRNA availability is both a limiting step and a regulatory parameter during translation [1,2]. Recently, through an experiment with synthesized GFP genes, it was shown that tRNA reusage (codon order) influences translation speed in yeast [3]. Here we describe the tRNA Pairing Index (TPI), an index that measures the degree of tRNA reusage in any gene.

By a statistical analysis of the TPI and gene expression, we show that genes that change their expression level rapidly (and thus require the most rapid translation) have a (statistically significant) higher TPI. Specifically, genes involved in cell cycle and DNA damage have a high TPI. These genes are regulated in the most dynamic manner, i.e. they are most rapidly turned on and off in response to intra- or extra-cellular activities.

The TPI distribution over all yeast coding sequences is biased towards positive values, indicating that there is a general tendency of tRNA reusage in the yeast genome.

Codon bias has been extensively studied previously [4,5,6,7,8,9,10]. However, to the best of our knowledge, the problem of measuring tRNA reusage in a gene has not been addressed before. The general analysis of codon autocorrelation suffers from the bias that may be induced by different base frequencies in different parts of the genome. It is known that some parts of the genome are GC-rich while other parts are GC-poor. Such

P. Bücher and B.M.E. Moret (Eds.): WABI 2006, LNBI 4175, pp. 1–11, 2006.

long-stretched biases induce an autocorrelation in the codons, which could be significant. Our first version of the TPI can measure autocorrelation without being affected by this kind of bias.

2 Methods

The TPI is an index which is computed for each protein and measures the autocorrelation (positive or negative) of its codons. Depending on how the background distribution is chosen, it is possible to make TPI completely independent of the frequencies of the amino acids, tRNAs, codons or bases, so that it will not suffer from any of the common sources of bias.

We measure the autocorrelation independently of everything else by analyzing the usage of tRNA in each amino acid of a protein as a combinatorial problem on symbols. For example, suppose that we are considering an amino acid which occurs 7 times in the protein in question and can be translated by two different tRNAs, A and B (e.g. 3 A's and 4 B's). We will extract the tRNAs from our sequence and represent them as a sequence of 7 symbols, e.g. AABABBB.

Highly autocorrelated cases are AAABBBB and BBBBAAA. A highly negatively autocorrelated case is BABABAB. This autocorrelation can be quantified by the number of identical pairs in the sequence or, conversely, by the number of changes C as we read from left to right. Notice that for a sequence of length n, the number of identical pairs plus the number of changes is $n - 1$. The mathematics is completely analogous for the number of pairs or number of changes. We call these breaks in the sequences changes, with the thought that if a tRNA molecule is doing the translation for one particular amino acid, when these breaks happen, this tRNA will have to be changed for another molecule. The first two examples have 1 change each, the last example has 6 changes. The TPI measures how high the actual number of pairs are, or how low C is, compared to all possible permutations of the sequence of tRNAs.

We present two different background distributions: one (TPI_1) based on codon frequencies given by the actual gene/genome under study, i.e. all possible orders considered equally likely (2.1) and another one (TPI_2) based on variable codon frequencies extracted from the entire genome (2.3).

2.1 TPI_1: Constant Codon Frequencies

Computation of the Probability of the Number of Changes. We will now describe the function to compute the probability and cumulative distribution of a given number of changes x.

It is easy to observe that the probability of the number of changes $C(x, n_1, n_2, ..., n_k)$ does not depend on what the symbols are, but rather on how many symbols there are of each kind $(n_1, n_2, ..., n_k)$. C is a (symmetric) function of the number of each kind of different symbols. It is difficult to write a recursion based on C, so instead we will base its computation on another function, called C_r, which does the recursive part of the computation. $C_r(x, n_1, n_2, ..., n_k)$ assumes that we are not at the beginning of the sequence, but rather that the last symbol observed is known (Fig. 1). To identify this known symbol (all symbols are otherwise equivalent), we will make it the first of the

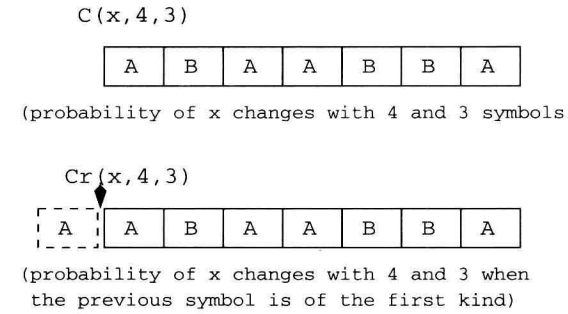

Fig. 1. C and C_r

arguments. Our function C_r assumes that it is called with a symbol of the first class preceding the rest of the symbols (Fig. 1). We explain C_r for $k = 2$ symbols in detail.

$$C_r(x, n_1, n_2) = \begin{cases} 0 & \text{if } n_1 < 0 \text{ or } n_2 < 0 \text{ or } x < 0 \text{ or } x > n_1 + n_2 \\ 1 & \text{if } n_1 = n_2 = 0 \ (x \text{ must be } 0) \\ \frac{1}{n_1+n_2}(n_1 Cr(x, n_1 - 1, n_2) + n_2 Cr(x - 1, n_2 - 1, n_1)) \end{cases} \quad (1)$$

The first symbol is either from the class of n_1 (no change) or from the class of n_2, in which case the preceding symbol now is of the second class and we invert the arguments: $Cr(x - 1, n_2 - 1, n1)$.

The extension of this function to higher k is simple. Supplementary material (http://www.biorecipes.com/TPI/appendix/Cr.M) shows a production quality version of this procedure which takes into account more refined border conditions. $C(x, n_1, n_2)$ can be expressed in two forms in terms of C_r. First, if we allow an arbitrary number of symbols we use

$$C(x, n_1, n_2) = C_r(x + 1, [0, n_1, n_2]) \quad (2)$$

i.e., we create an artificial first symbol (of which we have 0 left) and allow for one more change. Else we can expand based on the first symbol:

$$C(x, n_1, n_2) = \frac{n_1}{n_1 + n_2} C_r(x, n_1 - 1, n_2) + \frac{n_2}{n_1 + n_2} C_r(x, n_2 - 1, n_1) \quad (3)$$

The code for C_r as written above, is exponential. We can use dynamic programming, or we could use something equivalent to *option remember* in Maple [11] to make it polynomial in the product of the n_i.

To estimate how rare a given number of changes is, we need to compute its cumulative distribution. Since the distribution is over the integers, we will take the cumulative distribution which adds one half of the probability at the point.

$$C_{cum}(x, n_1, n_2, ..., n_k) = \sum_{i=0}^{x-1} C(i, n_1, n_2, ..., n_k) + \frac{1}{2} C(x, n_1, n_2, ..., n_k) \quad (4)$$

Our TPI is $1 - 2C_{cum}$, which is more intuitive to use than C_{cum}.

Expected Values and Moments of the Number of Changes. The expected value of the number of changes can be expressed in terms of the symmetric functions S_1 and S_2 on the arguments:

$$\mu_1' = \sum_{i=1}^{\infty} i \times C(i, n_1, n_2, ..., n_k) = \frac{S_1^2 - S_2}{S_1} \quad \text{where} \quad S_i = \sum_{j=1}^{k} n_j^i \tag{5}$$

The derivation of this formula is not trivial in its general form (for an arbitrary k). However if we observe that all the probabilities are sums of binomial coefficients, then we can conclude that the result (expected value or higher moments) must be a polynomial expression divided an appropriate descending factorial. Since all the moments are symmetric in all the arguments, the moments must be functions of the symmetric polynomials derived from the n_i. Hence by symbolic interpolation we can determine all the moments in a much easier (and safer) way. Of interest are the expected value and the variance. This is because we will attempt a normal approximation to the distribution.

$$\mu_2 = \sum_{i=1}^{\infty} (i - \mu_1')^2 C(i, n_1, n_2, ..., n_k) = \frac{S_1 S_2 - S_1^3 - 2S_1 S_3 + S_2 S_1^2 + S_2^2}{S_1^2 (S_1 - 1)} \tag{6}$$

Unfortunately, despite the simplicity of the formulas resulting from this approach, they do not resolve our problem completely. The normal approximation gives a good approximation of the cumulative distribution around the average (for large values of S_1) and very good approximations when $\min(n_i)$ is high. However, it gives poor approximations at the tails when some of the n_i are small, which is an important case.

Computing the Distribution of C in Practice. The recursion in C_r, although simple, swaps its arguments, which makes it almost impossible to handle with the standard techniques. Even dynamic programming becomes very difficult to express. In this section we find a mechanism to rewrite the recursion in a way that the argument order is maintained.

Since the function is totally symmetric in its arguments (and C_r is totally symmetric in its arguments but the first) we can sort the arguments in increasing order guaranteeing a time of $O(n_1 n_2 .... n_k)$. This makes the recursion marginally acceptable for real problems (for yeast $k \leq 4$ and for most other genomes $k \leq 5$). This ordering is partly ruined by the swapping of arguments in the recursion (1). Each recursive call to C_r uses a different argument as second argument.

To resolve this problem we find recursions which (while maybe more complicated) do not jumble the arguments. We can illustrate this by doing the transformation on the simplest recursion, $k = 2$. For further simplicity, we will use the auxiliary function $H(x, n_1, n_2) = C_r(x, n_1, n_2)\binom{n_1 + n_2}{n_1}$. As expected, the recursion on $H(x, n_1, n_2)$ is significantly simpler.

$$H(x, n_1, n_2) = H(x, n_1 - 1, n_2) + H(x - 1, n_2 - 1, n_1) \tag{7}$$

We now apply this formula to the shifted arguments

$$-H(x, n_1, n_2 - 1) = -H(x, n_1 - 1, n_2 - 1) - H(x - 1, n_2 - 2, n_1) \quad (8)$$

$$H(x - 1, n_2 - 1, n_1) = H(x - 1, n_2 - 2, n_1) + H(x - 2, n_1 - 1, n_2 - 1) \quad (9)$$

Adding these three equations results in

$$H(x, n_1, n_2) = H(x, n_1 - 1, n_2) + H(x, n_1, n_2 - 1) - H(x, n_1 - 1, n_2 - 1) + H(x - 2, n_1 - 1, n_2 - 1) \quad (10)$$

Notice that we have managed to obtain a recursion for which all the arguments (n_1, n_2) are in the same order. The new recursion with four terms instead of two is a bit more complicated, but this is an insignificant cost when we observe that in this form it is easy to write a recursive program to compute it. The computation can be done over the space of $n_1 x n_2$ for increasing x, having to keep two copies of the older H.

Transformations for up to $k = 5$ were obtained by doing a Knuth-Bendix style elimination procedure among all shifts of the basic recurrence. This was done in Maple and required some careful and extensive manipulations. Table 1 shows the summary of the results.In the supplementary material, `http://www.biorecipes.com/TPI/appendix/recursions`, we show the recursions for $k = 2$ to $k = 5$. With these recursions it was possible to write a C program that can compute all the TPI values for a genome like yeast in about 6 hours. Previous attempts failed after weeks of computing in very large machines.

Table 1. Recursions

k	terms	eq. used	max shift x	max shift $n_1, n_2, \ldots$
2	4	3	-2	-1
3	12	37	-3	-1
4	32	657	-4	-1
5	80	19125	-5	-1

Analytic Solution for two Symbols. The case with two symbols can be resolved explicitly (unfortunately, we were not able to find closed forms for higher k, and conjecture that no simple forms exist). **Theorem:**

$$H(x, n_1, n_2) = \binom{n_1}{\lfloor \frac{x}{2} \rfloor} \binom{n_2 - 1}{\lfloor \frac{x-1}{2} \rfloor} \quad (11)$$

This is easily proved by plugging the recursion that defines $H(x, n_1, n_2)$ and separating the case when x is even and when x is odd. For example if x is even then $x = 2w$ and the recursion becomes:

$$\binom{n_1}{w}\binom{n_2 - 1}{w - 1} = \binom{n_1 - 1}{w}\binom{n_2 - 1}{w - 1} + \binom{n_2 - 1}{w - 1}\binom{n_1 - 1}{w - 1} \quad (12)$$