German Conference on Bioinformatics 2013

GCB'13, September 10–13, 2013, Göttingen, Germany

Edited by

Tim Beißbarth
Martin Kollmar
Andreas Leha
Burkhard Morgenstern
Anne-Kathrin Schultz
Stephan Waack
Edgar Wingender

OASIcs – Vol. 34 – GCB'13 www.dagstuhl.de/oasics

Editors

Tim Beißbarth
Department of Medical Statistics
University Medical Center Göttingen
Tim.Beissbarth@med.uni-goettingen.de

Martin Kollmar
NMR Based Structural Biology
MPI for Biophysical Chemistry, Göttingen
mako@nmr.mpibpc.mpg.de

Andreas Leha
Department of Medical Statistics
University Medical Center Göttingen
andreas.leha@med.uni-goettingen.de

Burkhard Morgenstern
Department of Bioinformatics (IMG)
University of Göttingen
bmorgen@gwdg.de

Anne-Kathrin Schultz
Department of Bioinformatics (IMG)
University of Göttingen
aschult2@gwdg.de

Stephan Waack
Institute of Computer Science
University of Göttingen
waack@informatik.uni-goettingen.de

Edgar Wingender
Institute of Bioinformatics
University Medical Center Göttingen
edgar.wingender@bioinf.med.uni-goettingen.de

ACM Classification 1998
J.3 Life and Medical Sciences

ISBN 978-3-939897-59-0

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at http://www.dagstuhl.de/dagpub/978-3-939897-59-0.

Publication date
September, 2013

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at http://dnb.d-nb.de.

License
This work is licensed under a Creative Commons Attribution 3.0 Unported license (CC-BY 3.0): http://creativecommons.org/licenses/by/3.0/legalcode.

In brief, this license authorizes each and everybody to share (to copy, distribute and transmit) the work under the following conditions, without impairing or restricting the authors' moral rights:

- Attribution: The work must be attributed to its authors.

The copyright is retained by the corresponding authors.

Digital Object Identifier: 10.4230/OASIcs.GCB.2013.i

ISBN 978-3-939897-59-0 **ISSN 2190-6807** **http://www.dagstuhl.de/oasics**

OASIcs – OpenAccess Series in Informatics

OASIcs aims at a suitable publication venue to publish peer-reviewed collections of papers emerging from a scientific event. OASIcs volumes are published according to the principle of Open Access, i.e., they are available online and free of charge.

Editorial Board

- Daniel Cremers (TU München, Germany)
- Barbara Hammer (Universität Bielefeld, Germany)
- Marc Langheinrich (Università della Svizzera Italiana – Lugano, Switzerland)
- Dorothea Wagner (*Editor-in-Chief*, Karlsruher Institut für Technologie, Germany)

ISSN 2190-6807

www.dagstuhl.de/oasics

Contents

On the estimation of metabolic profiles in metagenomics
Kathrin Petra Aßhauer and Peter Meinicke 1

On Weighting Schemes for Gene Order Analysis
Matthias Bernt, Nicolas Wieseke, and Martin Middendorf 14

Alignment-free sequence comparison with spaced k-mers
Marcus Boden, Martin Schöneich, Sebastian Horwege, Sebastian Lindner, Chris Leimeister, and Burkhard Morgenstern 24

PanCake: A Data Structure for Pangenomes
Corinna Ernst and Sven Rahmann 35

Reconstructing Consensus Bayesian Network Structures with Application to Learning Molecular Interaction Networks
Holger Fröhlich and Gunnar W. Klau 46

Efficient Interpretation of Tandem Mass Tags in Top-Down Proteomics
Anna Katharina Hildebrandt, Ernst Althaus, Hans-Peter Lenhof, Chien-Wen Hung, Andreas Tholey, and Andreas Hildebrandt 56

GEDEVO: An Evolutionary Graph Edit Distance Algorithm for Biological Network Alignment
Rashid Ibragimov, Maximilian Malek, Jiong Guo, and Jan Baumbach 68

Dinucleotide distance histograms for fast detection of rRNA in metatranscriptomic sequences
Heiner Klingenberg, Robin Martinjak, Frank Oliver Glöckner, Rolf Daniel, Thomas Lingner, and Peter Meinicke 80

Utilization of ordinal response structures in classification with high-dimensional expression data
Andreas Leha, Klaus Jung, and Tim Beißbarth 90

Extended Sunflower Hidden Markov Models for the recognition of homotypic *cis*-regulatory modules
Ioana M. Lemnian, Ralf Eggeling, and Ivo Grosse 101

Avoiding Ambiguity and Assessing Uniqueness in Minisatellite Alignment
Benedikt Löwes and Robert Giegerich 110

Aligning Flowgrams to DNA Sequences
Marcel Martin and Sven Rahmann 125

German Conference on Bioinformatics 2013 (GCB'13).
Editors: T. Beißbarth, M. Kollmar, A. Leha, B. Morgenstern, A.-K. Schultz, S. Waack, E. Wingender
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Preface

This proceedings volume contains original research papers presented at the German Conference on Bioinformatics 2013 (GCB'13) held at Georg-August-University, Göttingen, Germany, September 11–13, 2013.

The GCB is an annual, international conference devoted to all areas of bioinformatics. Recent meetings attracted a multinational audience with 250 – 300 participants each year.

GCB'13 is organized by the bioinformatics groups at *Göttingen Research Campus* in cooperation with the *German Society for Chemical Engineering and Biotechnology (DECHEMA)*, the *Society for Biochemistry and Molecular Biology (GBM)* and the *Special Interest Group on Informatics in Biology* of the German *Society of Computer Science (GI)*.

Five internationally renowned speakers agreed to give keynote talks at GCB'13: Manfred Eigen, Gene Myers, Erwin Neher, Terry Speed and Sarah Teichmann. Four satellite workshops were held on 10 September 2013 on *Statistical Methods in Bioinformatics, Computational Methods for Metagenomics and Meta-Omics, Alignment-Free Sequence Comparison* and *Methods for Integrated Analysis of Multi-Level Datasets*.

Submissions to GCB'13 were possible as *Regular Papers*, *i.e.* original research papers, *Highlight Papers*, usually reporting on work published during the last year, or poster abstracts.

Overall, we received 26 submissions for *Regular Papers* and 19 Submissions for *Highlight Papers*. After a careful reviewing procedure and discussions in the *Program Committee*, 12 out of the 26 *Regular* submissions and 8 out of the 19 *Highlight* submission were selected for oral presentation at the conference. This proceedings volume contains revised versions of the 12 selected *Regular Papers*.

We would like to thank all authors, members of the *Program Committee* and subreviewers as well as the members of the local *Organizing Committee* and the support team for their work. In particular, we are indebted to Dr. Anne-Kathrin Schultz for doing most of the organization work for GCB'13. We thank Andreas Leha for organizing the production of this proceedings volume and Britta Leinemann for administrative support.

Göttingen, September 2013
Burkhard Morgenstern and Edgar Wingender

German Conference on Bioinformatics 2013 (GCB'13).
Editors: T. Beißbarth, M. Kollmar, A. Leha, B. Morgenstern, A.-K. Schultz, S. Waack, E. Wingender

OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Program Committee

Program Chairs

Burkhard Morgenstern
Edgar Wingender

Program Committee

Mario Albrecht
Rolf Backofen
Jan Baumbach
Michael Beckstette
Niko Beerenwinkel
Tim Beissbarth
Sebastian Böcker
Erich Bornberg-Bauer
Thomas Dandekar
Andreas Dress
Mareike Fischer
Dmitrij Frishman
Holger Froehlich
Georg Fuellen
Robert Giegerich
Ivo Grosse
Volker Heun
Andreas Hildebrandt
Daniel Huson
Christoph Kaleta
Gunnar W. Klau
Ina Koch
Oliver Kohlbacher
Martin Kollmar
Antje Krause
Stefan Kurtz
Thomas Lengauer
Hans-Peter Lenhof
Thomas Lingner
Manja Marz
Alice Mchardy
Peter Meinicke
Irmtraud Meyer
Axel Mosig
Eugene Myers
Steffen Neumann
Kay Nieselt
Sven Rahmann
Matthias Rarey
Knut Reinert
Uwe Scholz
Dietmar Schomburg
Falk Schreiber
Michael Schroeder
Stefan Schuster
Torsten Schwede
Joachim Selbig
Rainer Spang
Peter Stadler
Mario Stanke
Jens Stoye
Martin Vingron
Arndt Von Haeseler
Stephan Waack
Thomas Werner
Ralf Zimmer

Additional Referees

Volker Helms
Dirk Willrodt
Alexander Kel
Michaela Bayerlova
Michael Love
Frank Kramer
Juliane Siebourg-Polster
Anja Hartmann
Reinhard Guthke
Walton White
Kousik Kundu
Christian Colmsee
Martin Engler
Sascha Steinbiss
Dragos Sorescu
Jonathan Goeke
Patrick Trampert
Sascha Winter
Anne Hildebrandt
Eva Grafahrend-Belau
Christoph Kaleta
Jochen Singer
Tobias Petri
Anne-Christin Hauschild

German Conference on Bioinformatics 2013 (GCB'13).
Editors: T. Beißbarth, M. Kollmar, A. Leha, B. Morgenstern, A.-K. Schultz, S. Waack, E. Wingender
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Supporters and Sponsors

Supporting Scientific Institutions

DECHEMA Gesellschaft für Chemische Technik
und Biotechnologie e.V.
http://www.dechema.de

GBM Gesellschaft für Biochemie und
Molekularbiologie e.V.
http://www.gbm-online.de

Fachgruppe "Informatik in den Biowissenschaften"
der GI
http://www.cebitec.uni-bielefeld.de/groups/fg402

Max-Planck-Institute for Biophysical Chemistry
http://www.mpibpc.mpg.de

University of Göttingen
http://www.uni-goettingen.de

University Medical Center Göttingen
http://www.med.uni-goettingen.de

GWDG: IT in der Wissenschaft
http://www.gwdg.de

German Conference on Bioinformatics 2013 (GCB'13).
Editors: T. Beißbarth, M. Kollmar, A. Leha, B. Morgenstern, A.-K. Schultz, S. Waack, E. Wingender
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Sponsors and Donors

geneXplain: From genes to drugs
`http://genexplain.com`

KWS: Saatgutspezialisten für Landwirte
`http://www.kws.de`

Speise- & Schankwirtschaft Bullerjahn
`http://www.bullerjahn.info`

MoBiTec: Innovative Tools for Molecular and Cell Biology
`http://www.mobitec.com`

Index of Authors

A
Althaus, Ernst 56
Aßhauer, Kathrin 1

B
Baumbach, Jan 68
Beißbarth, Tim 90
Bernt, Matthias 14
Boden, Marcus 24

D
Daniel, Rolf 80

E
Eggeling, Ralf 101
Ernst, Corinna 35

F
Fröhlich, Holger 46

G
Giegerich, Robert 110
Glöckner, Frank 80
Grosse, Ivo 101
Guo, Jiong 68

H
Hildebrandt, Andreas 56
Hildebrandt, Anna 56
Horwege, Sebastian 24
Hung, Chien-Wen 56

I
Ibragimov, Rashid 68

J
Jung, Klaus 90

K
Klau, Gunnar 46
Klingenberg, Heiner 80

L
Löwes, Benedikt 110
Leha, Andreas 90
Leimeister, Chris 24
Lemnian, Ioana 101
Lenhof, Hans-Peter 56
Lindner, Sebastian 24
Lingner, Thomas 80

M
Malek, Maximilian 68
Martin, Marcel 125
Martinjak, Robin 80
Meinicke, Peter 1, 80
Middendorf, Martin 14
Morgenstern, Burkhard 24

R
Rahmann, Sven 35, 125

S
Schöneich, Martin 24

T
Tholey, Andreas 56

W
Wieseke, Nicolas 14

German Conference on Bioinformatics 2013 (GCB'13).
Editors: T. Beißbarth, M. Kollmar, A. Leha, B. Morgenstern, A.-K. Schultz, S. Waack, E. Wingender
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

On the estimation of metabolic profiles in metagenomics*

Kathrin Petra Aßhauer and Peter Meinicke†

Department of Bioinformatics, Institute for Microbiology and Genetics
University of Göttingen
37077 Göttingen, Germany
peter@gobics.de

Abstract

Metagenomics enables the characterization of the specific metabolic potential of a microbial community. The common approach towards a quantitative representation of this potential is to count the number of metagenomic sequence fragments that can be assigned to metabolic pathways by means of predicted gene functions. The resulting pathway abundances make up the metabolic profile of the metagenome and several different schemes for computing these profiles have been used. So far, none of the existing approaches actually estimates the proportion of sequences that can be assigned to a particular pathway. In most publications of metagenomic studies, the utilized abundance scores lack a clear statistical meaning and usually cannot be compared across different studies. Here, we introduce a mixture model-based approach to the estimation of pathway abundances that provides a basis for statistical interpretation and fast computation of metabolic profiles. Using the KEGG database our results on a large-scale analysis of data from the Human Microbiome Project show a good representation of metabolic differences between different body sites. Further, the results indicate that our mixture model even provides a better representation than the dedicated HUMAnN tool which has been developed for metabolic analysis of human microbiome data.

1998 ACM Subject Classification J.3 Life and Medical Sciences – Biology and genetics

Keywords and phrases metagenomics, metabolic profiling, taxonomic profiling, abundance estimation, mixture modeling

Digital Object Identifier 10.4230/OASIcs.GCB.2013.1

1 Introduction

In metagenomics a central task is to characterize the metabolic potential of a microbial community. The metabolic profile of a metagenome quantifies the amount of genetic material that can be attributed to metabolic pathways. The abundance of a pathway is usually estimated by the number of sequences that can be mapped to gene families with functional roles within that pathway. Several heuristics exist to compute a corresponding estimate. Using for instance the KEGG database, an abundance may be estimated by counting all BLAST best hit matches to KEGG Orthologs which are annotated for the particular pathway (see e.g. [4]). There are two major difficulties with this classical approach of metabolic profiling: First, the computational effort for the identification of homologs can become burdensome. Usually the BLASTX tool is required, which takes a considerable amount of

* This work was partially supported by the Deutsche Forschungsgemeinschaft (ME3138 "Compositional descriptors for large scale comparative metagenome analysis").
† corresponding author


© Kathrin Petra Aßhauer and Peter Meinicke;
licensed under Creative Commons License CC-BY
German Conference on Bioinformatics 2013 (GCB'13).
Editors: T. Beißbarth, M. Kollmar, A. Leha, B. Morgenstern, A.-K. Schultz, S. Waack, E. Wingender; pp. 1–13

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

CPU-time even for a moderately sized data set. Second, the usual counting scheme lacks a probabilistic model that would provide a clear statistical interpretation of the resulting quantities. To our knowledge, none of the existing heuristics actually yields an estimate of the fraction of sequence material that can be mapped to a particular pathway. Depending on the particular method the existing tools merely provide different kinds of abundance scores [14, 12, 1, 5, 4]. Although these scores may be used for comparative analysis as well, they do not provide a strictly probabilistic description of metagenomic sequence data. Therefore, the comparison and combination with other methods or models is at least problematic. We address both problems, the algorithmic and statistical efficiency within a metabolic mixture model in terms of a mixture of pathways (MoP). This model is capable to provide both, a sound statistical basis and a fast estimation of pathway abundances. Our results on a large-scale analysis of data from the Human Microbiome Project (HMP) show the utility of our method for fast model-based estimation of pathway abundances. Further, the results for the mixture-based metabolic profiles indicate a better separation between body sites than for the profiles of the HUMAnN tool which has particularly been developed for analyis of HMP data.

2 Material

2.1 Human Microbiome Project (HMP)

Within the scope of the Human Microbiome Project (HMP) [3] an extensive collection of samples from healthy individuals from diverse human body sites was established allowing an insight into the functions of the healthy human microbiome. More than thousand HMP data sets are recorded in HMP's Data Acquisition and Coordination Center (DACC) Project Catalog (http://www.hmpdacc.org/resources/data_browser.php) providing a comprehensive data basis for large-scale comparative studies investigating the associations of the human microbiome in healthy and diseased states.

From the HMP-DACC website we assessed the available metadata for the metagenomic samples (http://www.hmpdacc-resources.org/hmp_catalog/main.cgi) and the metabolic reconstruction data (http://www.hmpdacc.org/HMMRC/). The metabolic reconstruction data is obtained through the HMP Unified Metabolic Analysis Network (HUMAnN) pipeline [1]. HUMAnN performs functional and metabolic profiling directly from high-throughput metagenomic short sequence reads. The pipeline starts with a similarity search against a functional sequence database including the KEGG Orthologs (Release 54) using an accelerated translated BLAST implementation. Subsequently, the output is used for a series of gene- and pathway-level quantification, noise reduction, and smoothing steps resulting in the identification of present/absent pathways and modules together with their relative abundances. From the available metabolic reconstruction data, we used the "KEGG pathway abundance values – Summary file" (as of February 2013).

For our mixture modeling approach we used the reduced data samples of the HMP as describes in [10]. For comparability, the available samples and pathway abundances of HUMAnN and our mixture modeling approach were reduced to a subset of samples and pathways available in both methods. The final dataset includes 680 data samples from 14 specific body subsites, which can be grouped into five major body sites.

2.2 KEGG database

For the metabolic mixture modeling approach introduced here, we use the Kyoto Encyclopedia of Genes and Genomes (KEGG) database as reference knowledge base for estimating the pathway abundances of metagenomic samples [9, 8]. KEGG integrates a variety of information and provides links from gene catalogs to higher-level systematic functions of the organisms enabling biological interpretation of genomes and high-throughput datasets.

An essential part of the database with respect to metabolic profiling are the KEGG Orthologs (KO) that consist of gene groups with specific functions directly linked to known pathways in the KEGG Pathway database. Further, the KEGG Orthology is structured as a hierarchy of four flat levels: top, second, third level, and leaf nodes. While the leaf nodes represent the KEGG Orthologous groups, the third level represents the KEGG Pathways, which can be further summarized in higher level pathway classes (top and second level).

For the mixture modeling the required data reference was extracted from the KEGG database (Release 64.0).

2.3 MarVis

The MarVis-Suite (**Mar**ker **Vis**ualization) [7, 6], a toolbox originally developed for the analysis of metabolomic data, was used for filtering, clustering, and visualization of the pathway abundances. For exploration of complex pattern variation within the samples of the different body sites/subsites we used the MarVis-Cluster interface which permits high-level visualization and cluster analysis based on a one-dimensional self-organizing map (1D-SOM). The MarVis-Filter software was used for the identification of pathways overrepresented in the gastrointestinal tract samples compared to the other body subsites.

3 Methods

3.1 Taxonomic mixture modeling

The mixture model based Taxy approach provides a fast and direct estimation of taxonomic abundances in metagenomes. Taxy-Oligo [13] and Taxy-Pro [10] do not perform a taxonomic classification of sequencing reads but instead apply a mixture model to approximate the overall metagenome distribution of oligonucleotides and protein domain hits, respectively. The discrete distribution of oligonucleotides/protein domains is modeled by a mixture of organism-specific profiles as obtained from sequenced reference genomes. Because of the computational efficiency of the taxonomic mixture model approach, both methods were able to perform a large-scale analysis of sequence data from the HMP without using a computer cluster or special hardware. All reference profiles were obtained from the bacterial and archaeal genomes available in the KEGG database (Release 64.0). These genomes were also used for pre-computing the organism-specific pathway abundances for the metabolic profiling of metagenomes. For Taxy-Pro, all protein domain profiles according to the Pfam database [2] were obtained from the CoMet web server [11].

3.2 Metabolic mixture modeling

For metabolic profiling, we assume that the genomic sequence material to some degree can be explained by a mixture of pathways. The mixture approach accounts for the fact that in most cases a putative gene function as observed in a sequence fragment provides evidence for more than one metabolic pathway. The statistical representation of this ambiguity of the

function-to-pathway mapping was the main motivation for the development of the following model. With M pathways P_i the probability to observe a function F encoded in sequences under this model is:

$$\tilde{p}(F) = \sum_{i=1}^{M} p(P_i)p(F|P_i) \tag{1}$$

The tilde indicates that $\tilde{p}(F)$ only is an approximation of the functional profile $p(F)$ because not every function can be explained in terms of metabolic pathways. The prior pathway probabilities $p(P_i)$ denote the overall sequence-based abundance of functions associated with pathway P_i and correspond to the mixture weights of the model. These weights are the central model parameters, which can directly be used and interpreted in terms of the relative abundances of a metabolic profile. The conditional probability $p(F|P_i)$ denotes the i-th pathway-specific distribution over N possible gene functions F_j. The annotation in current databases, such as KEGG, can be represented by some $M \times N$ assignment matrix $\mathbf{A}$ with binary entries $A_{ij} = 1$ denoting that function j is associated with pathway i. From that assignment it follows that all functions not associated with pathway i must attain a zero conditional probability. Just from the annotation, we cannot draw any conclusions about the other probabilities. Without further knowledge the only reasonable assumption is that the $p(F|P_i)$ are proportional to the corresponding overall function probabilities, i.e.

$$\forall i, j : \; p(F_j|P_i) \propto A_{ij}p(F_j). \tag{2}$$

This constraint implies that the ratio between any two non-zero function probabilities in a pathway is equal for all pathways these two functions are associated with and must equal the global ratio of the corresponding probabilities of the functional profile $p(F)$. With the N estimates $\hat{p}(F_j)$ of the specific function probabilities of the profile as derived from the observed frequencies, e.g. from BLAST hit counts, we have the following estimator of the conditional probabilities:

$$\hat{p}(F_j|P_i) = \frac{A_{ij}\hat{p}(F_j)}{\sum_{k=1}^{N} A_{ik}\hat{p}(F_k)}. \tag{3}$$

Now let us consider the assignment probability

$$p(P|F_j) = \frac{p(P)p(F_j|P)}{\sum_{i=1}^{M} p(P_i)p(F_j|P_i)} \tag{4}$$

which denotes the responsibility of a pathway for a given function F_j, i.e. the contribution of a pathway to the explanation of that function. We assume that this probability is equal for all pathways the function is associated with. Without further knowledge, just with the underlying pathway annotation, there is no reason to prefer a particular pathway for the explanation of an observed function. This implies the following additional constraint:

$$\forall i, j, k : \; A_{kj}p(P_i|F_j) = A_{ij}p(P_k|F_j). \tag{5}$$

For a function F_j that is annotated in two pathways P_i and P_k we can obtain the ratio of the corresponding pathway abundance estimators using the former three equations (3), (4) and (5):

$$\frac{\hat{p}(P_i)}{\hat{p}(P_k)} = \frac{\hat{p}(F_j|P_k)}{\hat{p}(F_j|P_i)} = \frac{\sum_{s=1}^{N} A_{is}\hat{p}(F_s)}{\sum_{t=1}^{N} A_{kt}\hat{p}(F_t)}. \tag{6}$$

From the above proportionality, we finally obtain the estimator of the pathway probabilities:

$$\hat{p}(P_i) = \frac{\sum_{j=1}^{N} A_{ij}\hat{p}(F_j)}{\sum_{k=1}^{M}\sum_{l=1}^{N} A_{kl}\hat{p}(F_l)}. \tag{7}$$

Using matrix vector algebra we can compute the whole metabolic profile vector $\mathbf{p}$ with entries $\hat{p}(P_i)$ from the functional profile vector $\mathbf{f}$ with entries $\hat{p}(F_j)$ by

$$\mathbf{p} = \frac{\mathbf{Af}}{\mathbf{1}^T\mathbf{Af}} \tag{8}$$

where $\mathbf{1}$ is an M-vector of ones. In an application of the above mixture model most time will be spent for the computation of the functional profile which usually requires a costly BLASTX matching of metagenomic reads against a database of functionally annotated protein sequences, such as the KEGG Orthologues. However, with our formulation in terms of a statistical model we are able to provide a shortcut that utilizes the combination with another model to obtain a hierarchical mixture of pathways. Assume that we have the functional profiles of K reference organisms as columns in an $N \times K$ matrix $\mathbf{F}$ and we have estimated the relative abundances of the reference organisms in a taxonomic profile vector $\mathbf{t}$. Then we can approximate the functional profile of the metagenome by a linear combination of reference profiles $\mathbf{Ft}$. In Taxy-Pro [10] we use this mixture model in combination with Pfam functional profiles to estimate the taxonomic abundances in a metagenome. Here, we propose a combination with K pre-computed KEGG reference profiles to predict the functional profile of a metagenome from its taxonomic profile which may be obtained by some fast method such as the oligonucleotide-based Taxy tool [13]. The estimator of the metabolic profile is then

$$\mathbf{p} = \frac{\mathbf{AFt}}{\mathbf{1}^T\mathbf{AFt}}. \tag{9}$$

Note that also the matrix product $\mathbf{AF}$ can be pre-computed to obtain K organism-specific metabolic profiles which are then just combined by the taxonomic weights $\mathbf{t}$ of a metagenome to obtain its metabolic profile. In principle, this gives rise to a nested model where a mixture of pathways is first used for each reference organism to estimate its metabolic profile. This step has only to be performed once for each organism and therefore even a costly BLASTX analysis may be used for the "offline" training of the organism-specific models. When applied to metagenomic data a mixture of the utilized reference organisms has to be estimated by some taxonomic profiling method. In order to combine the two models the second step requires a profiling method that actually estimates the abundances in terms of the amount of sequence material that can be attributed to a particular organism. For example, this requirement is automatically fulfilled when using Taxy-Oligo [13] or Taxy-Pro [10], which we both included in the evaluation of our approach, as described above. For an application of the MoP model, it is important to check whether the metagenome composition can actually be approximated by a mixture of known reference organisms. If the reference is completely insufficient for a description of the metagenome composition, the mixture approach in general would become inadequate. Therefore, it is desirable, that the taxonomic profiling method gives us an indication of the fidelity of the abundance estimates. Both, Taxy-Oligo and Taxy-Pro provide a specific error measure to assess the adequacy of the underlying model. In this case, the fraction of oligonucleotides unexplained (FOU) and the fraction of domain-hits unexplained (FDU) should be inspected when using Taxy-Oligo and Taxy-Pro, respectively.

3.2.1 Workflows

For the evaluation of our model, we implemented the direct application of the metabolic mixture model as well as the nested model.

The direct application of the mixture model starts with a BLASTX analysis where the metagenomic reads are mapped against a reference database consisting of KO amino acid sequences of bacterial or archaeal origin. By default BLAST hits with E-value $\leq 10^{-2}$ were considered to be significant. The functional profile vector $\mathbf{f}$ is obtained by counting the KO-specific BLAST hits using a fractional increment of $1/K$ if K different KOs simulataneously show significant hits for a particular sequence. Note that due to the computational expense of BLASTX on metagenomes, we restricted the correlation analyis (see section 4.1) to a subset of six HMP data samples from different body subsites (SRS013825, SRS016752, SRS022621, SRS024265, SRS024428, and SRS055401). For the computation of the assignment matrix $\mathbf{A}$ the association of KOs with KEGG Pathways was extracted from the database and transformed into a binary matrix. Finally, the mixture model was applied as described above using the functional KO profile vector $\mathbf{f}$ and matrix $\mathbf{A}$ as input.

For the nested model, we first pre-computed the organism-specific metabolic profiles from reference genomes using all bacterial and archaeal KEGG Genomes. The KEGG Genomes were downloaded and subsequently fragmented in overlapping reads of length 400 bp with 200 bp overlap simulating a two-fold coverage of the genomes as previously described in [10]. For each reference organism, first the functional profile vector is calculated and then the metabolic profile is estimated applying the steps as described for the direct mixture approach. By combining the weights $\mathbf{t}$ of a metagenome with the pre-computed organism-specific metabolic profiles the metabolic profile of a metagenome can be obtained in an efficient manner. Note that a BLASTX/KO analysis of the metagenome is not required in this case. For the estimation of the taxonomic profile $\mathbf{t}$, we were using both, Taxy-Oligo and Taxy-Pro. According to the utilized taxonomic profiling method we denote our metabolic mixture model MoP-Oligo and MoP-Pro, respectively.

4 Results

To validate the metabolic mixture model on a well-studied dataset, we analyzed metagenomic sequences from the Human Microbiome Project (HMP) [3]. Originally, the metabolic profiles of the HMP data have been investigated by means of the HMP Unified Metabolic Analysis Network (HUMAnN) pipeline [1]. In the following, we use the metabolic profiles of HUMAnN for comparison with the abundance estimates that we obtained from our mixture of pathways model.

4.1 Correlation analysis

To study the similarity of metabolic profiles across different methods we computed the Pearson and Spearman (rank) correlation coefficients of the pathway abundance estimates. First, we evaluated the fast approximation scheme using pre-computed reference profiles based on Taxy-Pro taxonomic profiles (MoP-Pro). The resulting metabolic profiles were compared with the direct application of the mixture model to KO frequencies, which were obtained from a more time consuming BLASTX analysis. For each data sample, the correlation of the pathway abundances on two different pathway hierarchy levels (second and third level) was calculated.

The means and standard deviations of all data examples of the Pearson and Spearman

correlation coefficients are shown in Table 1. The results show a very high correlation of the approximation-based and the directly obtained abundances. By reducing the number of pathways from 340 to 38 according to the third and second pathway hierarchy levels an increase of the correlation from 0.9558 to 0.9804 and 0.9491 to 0.9842 could be observed for the Taxy-Pro-based approximation. These results indicate that the approximative approach is very close to the direct approach and therefore provides a computationally attractive alternative to the BLAST-based estimation.

Table 1 Correlation analysis based on the metabolic abundances obtained by applying the Taxy-Pro-based approximation and the direct mixture approach. The correlation is calculated according to Spearman and Pearson and at the third and second pathway hierarchy level.

	Pearson	Spearman
Third level	0.9557 (± 0.0409)	0.9491 (± 0.0124)
Second level	0.9803 (± 0.0150)	0.9842 (± 0.0110)

The correlations are similarly high for the even faster Taxy-Oligo variant (MoP-Oligo) with results shown in Table 2.

Table 2 Correlation analysis based on the metabolic abundances obtained by applying the Taxy-Oligo-based approximation and the direct mixture approach. The correlation is calculated according to Spearman and Pearson and at the third and second pathway hierarchy level.

	Pearson	Spearman
Third level	0.9575 (± 0.0409)	0.9466 (± 0.0105)
Second level	0.9796 (± 0.0138)	0.9813 (± 0.0087)

In contrast to the high similarity of results between different variants of the mixture approach the correlation between the mixture-based pathway abundances and the HUMAnN-based profiles is comparatively low with a Pearson correlation of 0.5290 as shown in Table 3. However, the correlation is increasing when considering the second pathway level or when using the Spearman rank correlation. A maximum rank correlation of 0.9080 indicates that the coarse shape of metabolic profiles is still rather similar between different approaches. Note that the correlation with HUMAnN profiles was averaged over all 680 HMP samples.

Table 3 Correlation analysis based on the metabolic abundances obtained by applying HUMAnN and the TaxyPro-based mixture model. The correlation is calculated according to Spearman and Pearson and at the third and second pathway hierarchy level.

	Pearson	Spearman
Third level	0.5290 (± 0.0206)	0.7588 (± 0.0242)
Second level	0.7884 (± 0.0308)	0.9080 (± 0.0135)

4.2 Nearest neighbor classification

To assess the quality of the estimated metabolic profiles we first investigated whether the body site (subsite) classification of HMP samples can be reproduced by the corresponding pathway abundances. For that purpose, we evaluated the predictive power of metabolic profiles by some nearest neighbor classification scheme using different profile distance measures. We utilized a leave-one-out cross validation, measuring the classification rate for Euclidean distance, City block metric and Shannon-Jensen divergence on profiles.

The results for body sites and subsites as shown in Table 4 reveal that the nearest neighbor classification rate is rather high and varies between 0.9735 and 0.9897 for the five body sites and between 0.8853 and 0.9235 for the 14 body subsites. For both classification problems, HUMAnN shows the highest prediction accuracy irrespective of the distance measure used. However, the two mixture variants are always very close with a maximum difference of 2.94% for the Euclidean distance on body subsite level between HUMAnN and MoP-Oligo.

Table 4 Nearest neighbor classification performing a leave-one-out cross validation with the Euclidean distance, City block metric and Shannon-Jensen divergence as distance measure for the approaches HUMAnN, MoP-Pro, and MoP-Oligo.

	Body site			Body subsite		
	Euclidean	City block	Jensen-Shannon	Euclidean	City block	Jensen-Shannon
HUMAnN	0.9838	0.9897	0.9868	0.9147	0.9235	0.9132
MoP-Pro	0.9794	0.9809	0.9779	0.9103	0.9103	0.9059
MoP-Oligo	0.9735	0.9750	0.9779	0.8853	0.8956	0.9132

4.3 Clustering performance

For a more comprehensive analysis of profile distances, we compared the body site (subsite) classification of samples with a profile-based clustering of the data. For clustering, we used a standard hierarchical approach with average linkage, also known as UPGMA. In this context, we evaluated the same three distance measures as for the nearest neighbor classification experiment. The quality of the cluster partitioning was assessed by the Jaccard coefficient, measuring the overlap of the resulting clusters with the HMP body site (subsite) groups.

The results obtained through the application of HUMAnN, MoP-Pro, and MoP-Oligo are presented in Table 5 which shows a large variation of the clustering performance.

Table 5 Cluster partitioning quality in terms of the Jaccard coefficient based on Euclidean distance, City block metric and Shannon-Jensen divergence for metabolic profiles of HUMAnN, MoP-Pro, and MoP-Oligo

	Body site			Body subsite		
	Euclidean	City block	Jensen-Shannon	Euclidean	City block	Jensen-Shannon
HUMAnN	0.4335	0.4342	0.4325	0.2361	0.3715	0.2344
MoP-Pro	0.6958	0.8817	0.6971	0.4791	0.4603	0.4801
MoP-Oligo	0.6577	0.7251	0.6382	0.3671	0.3008	0.3939

The Jaccard coefficient varied between 0.4325 and 0.8817 at body site level and between 0.2344 and 0.4801 at body subsite level. The partitioning of the MoP-Pro approach always showed the highest values on body site and subsite level. For both levels, the clustering performance of the MoP-Oligo approach is superior to HUMAnN except for the City block metric at body subsite level.

4.4 1D-SOM clustering and visualization

In order to study the overall variation of pathway abundance patterns over the whole range of HMP samples, we analyzed the estimated metabolic profiles with the MarVis tool. A one-dimensional self-organizing map (1D-SOM) was created using MarVis-Cluster (see Figure 1) to obtain a set of ordered prototypes well-suitable for visualization of profile variations.

Here, we utilized the second pathway level where we reduced the profiles to include just the top 10 pathways with the highest variance over all samples. Taking the union of the top 10 MoP and HUMAnN pathways we achieved a total of 13 profile dimensions that we used for 1D-SOM clustering with 14 prototypes and a unit 2-norm scaling of profile vectors. The resulting visualization indicates that most of the body sites are separated into distinct clusters (Figure 1). For the MoP profiles three major groups of clusters can be identified: gastrointestinal tract (GI tract, left side), urogenital tract (UG tract, right side), and an intermediate set of clusters from airways, oral and skin sites. Furthermore there are some interesting gradients (left to right) that show a decreasing relative abundance for *Amino Acid Metabolism*, *Carbohydrate Metabolism*, and *Signal Transduction* pathways and an increasing abundance for *Membrane Transport*, *Nucleotide Metabolism*, *Replication and Repair*, and *Translation* pathways. In contrast, the 1D-SOM based on the HUMAnN pathway profiles shows a distinct picture of the overall variation. The different body sites are not as clearly separated as for the MoP-based SOM and the overall abundance gradients of selected pathways are not as prominent as for the MoP results. The visible gradients (left to right) that show a decreasing abundance include the *Amino Acid Metabolism* and *Metabolism of Cofactors and Vitamins* pathways while an increasing abundance can be observed for *Metabolism of Other Amino Acids*, *Replication and Repair*, and *Translation* pathways.

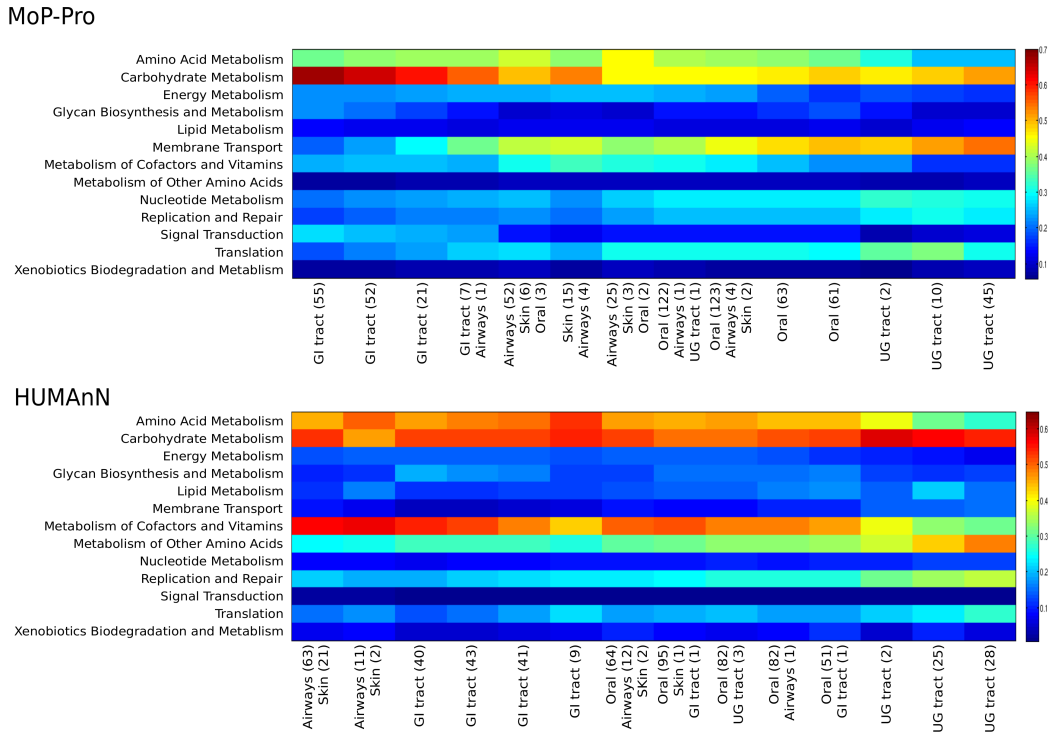


Figure 1 1D-SOM created with MarVis-Cluster at second pathway hierarchy level for MoP-Pro (upper) and HUMAnN (lower) profiles (GI tract – gastrointestinal tract; UG tract – urogenital tract). The numbers (in brackets) indicate the number of profiles (samples) assigned to the corresponding prototype (cluster) above.

4.5 Significant pathways

For a specific analysis of the metabolic profiles in terms of statistically significant differences in pathway abundances between different body sites we compared the gastrointestinal (GI) tract samples with all other HMP samples. To identify overrepresented pathways for the GI body site we applied an ANOVA with Holm-Bonferroni (FWER) correction on pathway abundances of the second level pathway hierarchy, filtered for pathways with an FWER below 0.05, and ranked the remaining pathways according to their fold-change in terms of the corresponding overrepresentation factor on mean abundances. In Table 6 the significant pathways of MoP-Pro and HUMAnN with a calculated fold-change larger than 1 are listed.

Table 6 MarVis-Filter analysis for the identification of overrepresented pathways in the gastrointestinal tract samples in comparison to all other body subsites. All second level pathways obtained through the application of the MoP-Pro and HUMAnN approach with a fold-change larger than 1 are listed.

MoP-Pro

Pathway (second level)	Fold-Change
Transport and Catabolism	2.00
Signal Transduction	1.81
Digestive System	1.79
Biosynthesis of Other Secondary Metabolites	1.53
Nervous System	1.48
Carbohydrate Metabolism	1.23
Glycan Biosynthesis and Metabolism	1.15
Endocrine System	1.13
Immune System	1.12

HUMAnN

Pathway (second level)	Fold-Change
Digestive System	2.04
Endocrine System	1.12
Glycan Biosynthesis and Metabolism	1.07
Amino Acid Metabolism	1.06
Biosynthesis of Other Secondary Metabolites	1.06
Energy Metabolism	1.01

HUManN and MoP-Pro identified pathways associated with the *Digestive System*, *Endocrine System*, *Biosynthesis of Other Secondary Metabolites*, *Glycan Biosynthesis and Metabolism* to be overrepresented in GI tract samples. For all these pathways, except for the *Digestive System*, the MoP-Pro fold-change was higher than the corresponding factor of HUManN. Exclusively for the HUMAnN approach, pathways associated with *Amino Acid Metabolism* and *Energy Metabolism* are found to be slightly overrepesented. Furthermore, through the application of the MoP-Pro we detected five additional pathways to be overrepresented: *Transport and Catabolism*, *Signal Transduction*, *Nervous System*, *Carbohydrate Metabolism*, and *Immune System*. These additional pathways are possibly related to a mutually beneficial relationship between the gut microbiota and the host, maintaining a normal mucosal immune function and nutrient absorption. Furthermore, the overrepresentation of pathways associated with the nervous system may provide an indication for the bidirectional brain-gut interactions which have an important role in the modulation of gastrointestinal functions and possibly support the hypothesis of a communication pathway between the microbiota and the host's central nervous system [15].

4.6 Runtime

To get an overview of the computational cost of the different variants of the mixture modeling, we measured the approximate runtime averaged over the six selected HMP data samples (average size ~200 MB) used for the correlation analysis. For the selected data sets the

mean runtime ranges from minutes to months. The longest CPU times were required by the direct application of the mixture model due to the costly similarity searches against the KO database. On a computer with four CPUs (2.4 GHz) BLASTX searches and calculation of the metabolic profile took approximately 58 days. The fastest method was MoP-Oligo with about half a minute, followed by the MoP-Pro method with about one minute runtime in total. Once the taxonomic profile is estimated, using either MoP-Oligo or MoP-Pro, the resulting matrix vector multiplication for obtaining the metabolic profile of a metagenome can be done within a second.

5 Discussion

We presented a novel metabolic profiling approach for metagenomics, which is based on a mixture of pathways (MoP) model for estimation of pathway abundances. To overcome computationally intense homology searches, we implemented a shortcut to estimate the metabolic profile of a metagenome. Here, we link the taxonomic profile of the metagenome to a set of pre-computed metabolic reference profiles. The combination of the taxonomic abundance estimates, obtained through the fast methods Taxy-Oligo and Taxy-Pro, and the metabolic reference profiles, based on the KEGG database, achieves an unrivaled speed of the metabolic profiling approach.

We are aware of the difficulties in the evaluation that arise when trying to assess the quality of the resulting metabolic profiles. Therefore we restricted our evaluation to the large-scale data from the Human Microbiome Project (HMP) and to the comparison with the observations and findings for this data obtained through the HUMAnN approach. In this setup we tried to provide several views on metabolic profiles considering different aspects of quality: Our correlation analysis has shown that the pathway abundances obtained through our statistical model are slightly different when compared to the HUMAnN abundance predictions. However, we demonstrated through the nearest neighbor classification that our model based approach is at least comparable to the HUMAnN approach when considering the prediction of body sites and subsites. Considering the cluster performance analysis, our approach even outperforms the HUMAnN pipeline in most cases. Furthermore, our case study on statistically overrepresented pathways in the gastrointestinal tract provides additional insight in comparison with the results of the dedicated HUMAnN approach.

To our knowledge, the MoP approach for the first time provides a potentially unbiased estimator of the fraction of sequences that can be attributed to a particular pathway. In addition, our model-based combination with taxonomic abundance estimators also provides the fastest way to estimate the metabolic profile of a metagenome. We intend to make the method accessible via an easy-to-use interface by integration into the CoMet web server [11] (http://comet.gobics.de).

Acknowledgements We would like to thank Heiner Klingenberg, Thomas Lingner, Alexander Kaever, and Manuel Landesfeind for fruitful discussions.

References

1 Sahar Abubucker, Nicola Segata, Johannes Goll, Alyxandria M. Schubert, Jacques Izard, Brandi L. Cantarel, Beltran Rodriguez-Mueller, Jeremy Zucker, Mathangi Thiagarajan, Bernard Henrissat, Owen White, Scott T. Kelley, Barbara Methé, Patrick D. Schloss, Dirk Gevers, Makedonka Mitreva, and Curtis Huttenhower. Metabolic Reconstruction

for Metagenomic Data and Its Application to the Human Microbiome. *PLoS Comput Biol*, 8(6):e1002358, 06 2012.

2 Robert D. Finn, Jaina Mistry, John Tate, Penny Coggill, Andreas Heger, Joanne E. Pollington, O. Luke Gavin, Prasad Gunasekaran, Goran Ceric, Kristoffer Forslund, Liisa Holm, Erik L. L. Sonnhammer, Sean R. Eddy, and Alex Bateman. The Pfam protein families database. *Nucleic Acids Research*, 38(suppl 1):D211–D222, 2010.

3 The NIH HMP Working Group, Jane Peterson, Susan Garges, Maria Giovanni, Pamela McInnes, Lu Wang, Jeffery A. Schloss, Vivien Bonazzi, Jean E. McEwen, Kris A. Wetterstrand, Carolyn Deal, Carl C. Baker, Valentina Di Francesco, T. Kevin Howcroft, Robert W. Karp, R. Dwayne Lunsford, Christopher R. Wellington, Tsegahiwot Belachew, Michael Wright, Christina Giblin, Hagit David, Melody Mills, Rachelle Salomon, Christopher Mullins, Beena Akolkar, Lisa Begg, Cindy Davis, Lindsey Grandison, Michael Humble, Jag Khalsa, A. Roger Little, Hannah Peavy, Carol Pontzer, Matthew Portnoy, Michael H. Sayre, Pamela Starke-Reed, Samir Zakhari, Jennifer Read, Bracie Watson, and Mark Guyer. The NIH Human Microbiome Project. *Genome Research*, 19(12):2317–2323, 2009.

4 Daniel H. Huson, Suparna Mitra, Hans-Joachim Ruscheweyh, Nico Weber, and Stephan C. Schuster. Integrative analysis of environmental sequences using MEGAN4. *Genome Research*, 21(9):1552–1560, 2011.

5 Dazhi Jiao, Yuzhen Ye, and Haixu Tang. Probabilistic Inference of Biochemical Reactions in Microbial Communities from Metagenomic Sequences. *PLoS Comput Biol*, 9(3):e1002981, 03 2013.

6 Alexander Kaever, Manuel Landesfeind, Mareike Possienke, Kirstin Feussner, Ivo Feussner, and Peter Meinicke. MarVis-Filter: ranking, filtering, adduct and isotope correction of mass spectrometry data. *BioMed Research International*, 2012, 2012.

7 Alexander Kaever, Thomas Lingner, Kirstin Feussner, Cornelia Gobel, Ivo Feussner, and Peter Meinicke. MarVis: a tool for clustering and visualization of metabolic biomarkers. *BMC Bioinformatics*, 10(1):92, 2009.

8 Minoru Kanehisa and Susumu Goto. KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Research*, 28(1):27–30, 2000.

9 Minoru Kanehisa, Susumu Goto, Yoko Sato, Miho Furumichi, and Mao Tanabe. KEGG for integration and interpretation of large-scale molecular data sets. *Nucleic Acids Research*, 40(D1):D109–D114, 2012.

10 Heiner Klingenberg, Kathrin Petra Aßhauer, Thomas Lingner, and Peter Meinicke. Protein signature-based estimation of metagenomic abundances including all domains of life and viruses. *Bioinformatics*, 2013.

11 Thomas Lingner, Kathrin Petra Aßhauer, Fabian Schreiber, and Peter Meinicke. CoMet - a web server for comparative functional profiling of metagenomes. *Nucleic Acids Research*, 39(suppl 2):W518–W523, 2011.

12 Victor M. Markowitz, I-Min A. Chen, Ken Chu, Ernest Szeto, Krishna Palaniappan, Yuri Grechkin, Anna Ratner, Biju Jacob, Amrita Pati, Marcel Huntemann, Konstantinos Liolios, Ioanna Pagani, Iain Anderson, Konstantinos Mavromatis, Natalia N. Ivanova, and Nikos C. Kyrpides. IMG/M: the integrated metagenome data management and comparative analysis system. *Nucleic Acids Research*, 40(D1):D123–D129, 2012.

13 Peter Meinicke, Kathrin Petra Aßhauer, and Thomas Lingner. Mixture models for analysis of the taxonomic composition of metagenomes. *Bioinformatics*, 27(12):1618–1624, 2011.

14 Folker Meyer, Daniel Paarmann, Mark D'Souza, Robert Olson, Elizabeth M Glass, Michael Kubal, Tobias Paczian, A Rodriguez, Rick Stevens, Andreas Wilke, et al. The metagenomics RAST server - a public resource for the automatic phylogenetic and functional analysis of metagenomes. *BMC Bioinformatics*, 9(1):386, 2008.

15 Sang H Rhee, Charalabos Pothoulakis, and Emeran A Mayer. Principles and clinical implications of the brain–gut–enteric microbiota axis. *Nature Reviews Gastroenterology and Hepatology*, 6(5):306–314, 2009.

On Weighting Schemes for Gene Order Analysis

Matthias Bernt, Nicolas Wieseke, and Martin Middendorf

Parallel Computing and Complex Systems Group, Institute of Computer Science
University Leipzig, Germany
{bernt,wieseke,middendorf}@informatik.uni-leipzig.de

Abstract

Gene order analysis aims at extracting phylogenetic information from the comparison of the order and orientation of the genes on the genomes of different species. This can be achieved by computing parsimonious rearrangement scenarios, i.e. to determine a sequence of rearrangements events that transforms one given gene order into another such that the sum of weights of the included rearrangement events is minimal. In this sequence only certain types of rearrangements, given by the rearrangement model, are admissible and weights are assigned with respect to the rearrangement type. The choice of a suitable rearrangement model and corresponding weights for the included rearrangement types is important for the meaningful reconstruction. So far the analysis of weighting schemes for gene order analysis has not been considered sufficiently. In this paper weighting schemes for gene order analysis are considered for two rearrangement models: 1) inversions, transpositions, and inverse transpositions; 2) inversions, block interchanges, and inverse transpositions. For both rearrangement models we determined properties of the weighting functions that exclude certain types of rearrangements from parsimonious rearrangement scenarios.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, J.3 Life and Medical Sciences

Keywords and phrases Gene order analysis, maximum parsimony, weighting

Digital Object Identifier 10.4230/OASIcs.GCB.2013.14

1 Introduction

The order of the genes on the chromosomes has changed during evolution by different types of rearrangement operations. For unichromosomal genomes, like most bacterial and mitochondrial genomes, inversions, transpositions, inverse transpositions, and tandem duplication random loss operations modified the order and/or orientation of the genes. In addition deletions, duplications, and horizontal transfer changed the gene content. Multichromosomal genomes, e.g. nuclear genomes, have been subject to additional interchromosomal rearrangement operations (e.g. fission, fusion, translocation, and chromosome duplication).

Gene order data has become an important source of phylogenetic information over the last two decades [14, 22]. The phylogenetic information contained in gene orders can be extracted with methods based on the maximum maximum parsimony principle, i.e. an explanation for given gene order data is sought that uses a minimal number of rearrangement operations (but see also [1]). For a pair of gene orders such an explanation is given by a shortest sequence of rearrangements that transforms one of the given gene orders into the other. If more than two gene orders are given a phylogenetic tree with the given gene orders at the leaves and a minimum number of rearrangements along the edges of the tree, such that a gene order at the root is transformed into the leaf gene orders, serves as an explanation of the data.

Algorithms for pairwise gene order analysis have been studied extensively for separate rearrangement operations. Efficient algorithms for the case that only inversions are considered


© Matthias Bernt, Nicolas Wieseke, and Martin Middendorf;
licensed under Creative Commons License CC-BY
German Conference on Bioinformatics 2013 (GCB'13).
Editors: T. Beißbarth, M. Kollmar, A. Leha, B. Morgenstern, A.-K. Schultz, S. Waack, E. Wingender; pp. 14–23

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

are known [15]. In contrast the problem is NP-hard if only transpositions are allowed [9]. But, in order to get reliable reconstructions all rearrangement operations that played a role during evolution need to be considered. Ideally, different weights should be used for the different types of rearrangements reflecting their importance in gene order evolution.

There are only a few algorithms for weighted gene order analysis considering more than one type of rearrangement operation. `DERANGE` [8, 21] is one of the first algorithms for gene order analysis. It is a 3-approximation algorithm that considers inversions, transpositions, and inverse transpositions. The algorithm explores possible rearrangement scenarios greedily by minimizing the number of breakpoints. Both, the possibility to weight the different types of rearrangements and to weight the operations by the number of affected genes, have been implemented. It was shown empirically that the reconstructions obtained with `DERANGE` are strongly influenced by the choice of the weights [8], see also [12]. Several improved approximation algorithms for this set of rearrangement operations have been introduced [4, 13, 16, 17]. `DERANGE` and the other algorithms assume the same weight for the two types of transpositions. The algorithm from [4] allows a weight of 1 for inversions and a weight in $\in [1:2]$ for transpositions. Other algorithms allow only fixed weights (1:2 [13] or 1:1 [16, 17]). With `CREx` an efficient heuristic for unweighted rearrangement analysis is available that incorporates tandem duplication random loss as a fourth type of rearrangement operation [6]. Also for multiple genome rearrangement analysis a heuristic, that is based on [4], is available that incorporates weights [3].

Block interchange is a generalization of the transposition operation. Since pairwise gene order analysis for this operation can be solved efficiently [10] this operation has become an interesting alternative for transpositions. It is an integral part (with inversions and translocations) of the double cut and join frame work [23] that became a highly active research area in the last years. The `SPRING` software [19] and the approach described in [20] allow for reconstructing pairwise rearrangement scenarios based on inversions and block interchanges using a corresponding weight ratio of 1:2. Approximation algorithms for other weighting schemes have been devised in [18]. By allowing for the additional operations tandem duplication and deletion the heuristic presented in [2] can also compute rearrangement scenarios consisting of inversions and block interchanges for gene orders with unequal gene content using a weight of 1 for inversions and 2 for each of the other operations.

All approaches mentioned above assign a (usually two times) larger weight to transpositions or block interchanges than to inversions. This is justified by the larger number of inversions than transpositions that can be observed for several biological data sets. But this is not the case for all data sets, e.g. for metazoan mitogenomes inversions seem to account for only a small proportion of the rearrangements [7].

Besides the possibility to weight rearrangements by the type of the rearrangement also weighting by the length of the affected segments (e.g. [5, 8, 12]), the types of the affected genes, or by other factors that determine the likelihood of a rearrangement (e.g. transcript structures) might be incorporated. Certain constraints as an extreme case can be introduced by allowing for infinite weights which excludes certain types of rearrangements. `CREx` for instance forbids rearrangement that destroy conserved gene clusters [6].

Weighting schemes for genome rearrangement analysis have not been considered in sufficient detail in the literature [14]. Here we analyze weighting schemes for two rearrangement models: (1) Inversions, transpositions, and inverse transpositions (Section 3); (2) Inversions, block interchanges, an inverse transpositions (Section 4). For both cases we derive properties of weighting schemes that exclude one/more of the rearrangement operations from any parsimonious reconstruction of genome rearrangement evolution.

2 Basic Definitions

In the context of this work a gene order is regarded as a *signed permutation* $\pi = (\pi(1), \ldots, \pi(n))$, which is a permutation of the elements $\{1, \ldots, n\}$ where each element has an additional orientation, denoted by a "+" or "−" sign. Each element of a gene order represents one genetic marker, e.g. a gene, and the sign its strandedness. If not stated otherwise, we assume a signed permutation to be directed, i.e. $\pi \neq -\pi = (-\pi(n), \ldots, -\pi(1))$. An *interval* X of a permutation π is a non-empty subset of (unsigned) elements $X \subseteq \{1, \ldots, n\}$ which are consecutive with respect to π, i.e. $\exists i, j \in [1:n] : X = \{|\pi(k)| : i \leq k \leq j\}$. $\mathfrak{I}(\pi)$ gives the set of all possible intervals of a permutation π. A *rearrangement* ρ is an operation applied to a signed permutation π that changes the position and/or orientation of some of the elements resulting in a new signed permutation denoted as $\pi \circ \rho$. For two rearrangements ρ and ρ', with $\rho \neq \rho'$, and a gene order π it holds that $\pi \circ \rho \neq \pi \circ \rho'$. Let $\mathfrak{R}$ be the set of all $n!2^n$ different rearrangement operations.

Let $w : \mathfrak{R} \to \mathbb{R}_{>0}$ be a weighting function for rearrangement operations. A classification into rearrangement *types* $\mathfrak{T} = \{T_1, \ldots, T_k, T_\epsilon\}$ is a partition of $\mathfrak{R}$ into distinct sets of rearrangements with all $\rho \in T_j$ having the same weight $w(\rho) = w_{T_j}$. The set T_ϵ refers to rearrangements with a weight of $w_{T_\epsilon} = \infty$ and is used for rearrangement operations that are not regarded. The set of valid rearrangement types is denoted as $\mathfrak{T}_{|T_\epsilon} = \mathfrak{T} \setminus T_\epsilon$. A *rearrangement scenario* for a permutation π is a sequence of rearrangements $S = (\rho_1, \ldots, \rho_l)$ such that $\pi \circ \rho_1 \circ \ldots \circ \rho_l = \iota$ and $\forall : i \in [1:l] : \rho_i \notin T_\epsilon$. The weight of a scenario S is given by $w(S) = \sum_{i=1}^{|S|} w(\rho_i)$. A scenario for π with minimal weight is called *parsimonious*. If not stated otherwise, we consider normalized weights for the admissible rearrangement types, i.e. the weight for one type of rearrangement operation is divided by the sum of the weights of all allowed rearrangement types.

Here we consider the rearrangement operations inversions (I), transpositions (T), inverse transpositions (iT), and block interchanges (BI). In Section 3 the set of valid rearrangement types $\mathfrak{T}_{|T_\epsilon} = \{I, T, iT\}$, with weights w_I, w_T, and w_{iT}, respectively, is considered. In Section 4 block interchange with weight w_{BI} is considered instead of transpositions. Each rearrangement is specified by the intervals it affects. An *inversion* ρ_I on a signed permutation π is defined by the interval $A \in \mathfrak{I}(\pi)$, where in $\pi \circ \rho_I$ the order of the elements from A is reversed and the orientation is switched. A *transposition* ρ_T is defined by two disjoint and consecutive intervals $A, B \in \mathfrak{I}(\pi)$, i.e. $A \cup B \in \mathfrak{I}(\pi), A \cap B = \emptyset$. By a transposition the position of the two intervals is switched in $\pi \circ \rho_T$. Analogous to a transposition an *inverse transposition* ρ_{iT} is also defined by two disjoint and consecutive intervals A and B. It is a combination of transposition and inversion, where after the transposition of A and B an additional inversion of A is performed. A *block interchange* ρ_{BI} is a generalization of the transposition in the way that the two intervals A and B do not have to be consecutive. There might be an interval X in between A and B such that in $\pi \circ \rho_{BI}$ A and B switch their positions with respect to X. In the following we denote the rearrangements $\rho_I, \rho_T, \rho_{iT}$, and ρ_{BI} together with the intervals they affect by $I(A), T(A,B), iT(A,B)$, and $BI(A,B)$, respectively. It holds that $T(A,B) = T(B,A)$ and $BI(A,B) = BI(B,A)$.

3 Inversions, transpositions, and inverse transpositions

First we consider the rearrangement model consisting of the following three types of rearrangements: inversions, transpositions, and inverse transpositions. There are several possibilities to mimic a single rearrangement, i.e. achieve the same effects, by combinations of rearrangements of other types. In the following we derive the different possibilities for

replacing one rearrangement of a certain type by a smallest number of rearrangements of the other type(s).

▶ **Lemma 1.** *For any rearrangement scenario the following replacements are possible within the* $\mathfrak{T}_{|T_\epsilon} = \{I, T, iT\}$ *model:*

1. *A transposition* $T(A, B)$ *can be replaced by each of the following sets of rearrangements*
   - **a.** *three inversions* $I(A)$*,* $I(B)$*, and* $I(A \cup B)$*,*
   - **b.** *one inverse transposition* $iT(A, B)$ *and one inversion* $I(A)$*, or*
   - **c.** *two inverse transpositions when the genome has at least three genes. Then at least one of the following cases holds:*

     Case i. There exists an interval X *such that* $B \cup X$ *is an interval and* $X \cap A = \emptyset = X \cap B$*:* $iT(A, B \cup X)$ *and* $iT(A, X)$*.*

     Case ii. There exists an interval X *such that* $X \cup A$ *is an interval and* $X \cap A = \emptyset = X \cap B$*:* $iT(B, X \cup A)$ *and* $iT(B, X)$

     Case iii. There exists a bipartition of A *into intervals* A_1 *and* A_2 *(i.e.* $|A| \geq 2$*) such that* $A_2 \cup B$ *is an interval:* $iT(B, A_2)$ *and* $iT(B, A_1)$

     Case iv. There is a bipartition of B *into intervals* B_1 *and* B_2 *(i.e.* $|B| \geq 2$*) such that* $B_1 \cup A$ *is an interval:* $iT(A, B_1)$ *and* $iT(A, B_2)$*.*

4. *An inverse transposition* $iT(A, B)$ *can be replaced by each of the following sets of rearrangements:*
   - **a.** *one transposition* $T(A, B)$ *and one inversion* $I(A)$ *or*
   - **b.** *two inversions* $I(A \cup B)$ *and* $I(B)$*.*

3. *An inversion* $I(A)$ *can be replaced by each of the following sets of rearrangements:*
   - **a.** *inverse transposition(s) and one transposition according to the following cases:*

     Case i. When at least one gene is not included in A*, i.e. there exists an interval* X *with* $X \cap A = \emptyset$ *and* $X \cup A$ *is an interval:* $iT(A, X)$ *and* $T(A, X)$*.*

     Case ii. A *includes the whole gene order and can be partitioned into two intervals* A_1 *and* A_2 *(i.e.* $|A| \geq 2$*):* $iT(A_2, A_1)$*,* $iT(A_1, A_2)$*,* $T(A_1, A_2)$*.*
   - **b.** *three inverse transpositions when the gene order has at least three genes. Then at least one of the following cases holds:*

     Case i. There exists a partition of A *into three intervals* A_1*,* A_2*, and* A_3 *(i.e.* $|A| \geq 3$*) such that* $A_1 \cup A_2$ *and* $A_2 \cup A_3$ *are intervals:* $iT(A_1, A_2)$*,* $iT(A_3, A_1)$*, and* $iT(A_2, A_3)$*.*

     Case ii. There exists a bipartition of A *into two intervals* A_1 *and* A_2 *(i.e.* $|A| \geq 2$*) and there exists an interval* X *with* $A \cap X = \emptyset$ *such that* $A_2 \cup X$ *is an interval:* $iT(A_2 \cup X, A_1)$*,* $iT(X, A_2)$*, and* $iT(A_1, X)$*.*

     Case iii. There exists an interval X *that can be partitioned into intervals* X_1 *and* X_2 *(i.e.* $|X| \geq 2$*) with* $A \cap X = \emptyset$ *such that* AX_1 *is an interval:* $iT(A, X_1)$*,* $iT(X_1, A \cup X_2)$*, and* $iT(X_1, X_2)$*.*

     Case iv. There exist two disjoint intervals X_1 *and* X_2 *with* $X_1 \cap A = \emptyset = X_2 \cap A$ *such that* $X_1 \cup A$ *and* $A \cup X_2$ *are intervals:* $iT(A, X_2)$*,* $iT(X_2, X_1)$*, and* $iT(X_2, X_1 \cup A)$*.*

▶ **Lemma 2.** *Lemma 1 lists all possibilities (with respect to number and type of the rearrangement operations) to replace a single rearrangement of a certain type* $\in \{T, iT, I\}$ *by a smallest number of rearrangements of one or two of the other types of rearrangements* $\in \{T, iT, I\}$*.*

Proof. By definition of the rearrangement operations it is not possible to replace a single operation of one type by any single operation of another type. Observe also that a transposition $T(A, B)$ cannot be replaced by any number of inverse transpositions when the genome has only two genes, i.e. A and B contain only a single gene and there exists no other gene

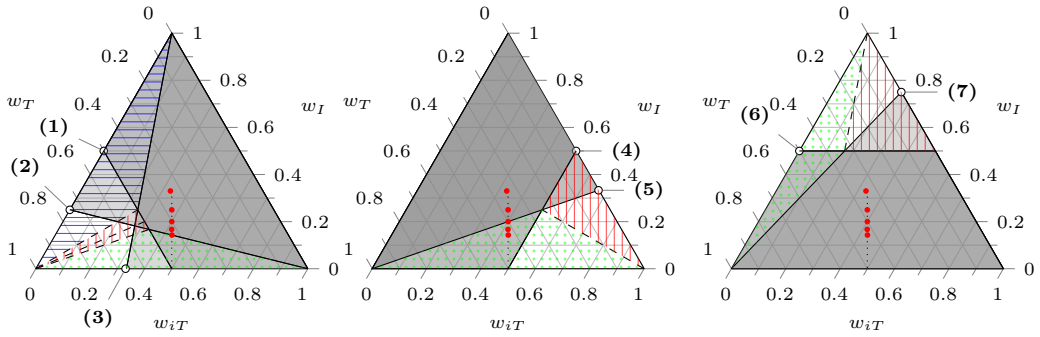


Figure 1 Barycentric plot showing the weighting schemes where transpositions (left), inverse transpositions (middle), and inversions (right) need to be considered; shaded areas indicate for each of the inequalities the valid weighting schemes; darker shading indicates the area where all inequalities hold; the limiting cases, i.e. equality, is annotated by the corresponding equation number given in bold text; dashed lines give the demarcation line between each pair of the alternatives, colored dashed or dotted areas indicate which of the alternatives needs to be considered: blue horizontal lines 2iT, red vertical lines iT+ I, green dots 3I (left) red vertical lines I+T, green dots 2I (middle) red vertical lines T+iT, green dots 3iT (right); the dotted line and red dots indicate the weighting schemes considered in [8]; note that the borders of the plot, i.e. weights of 0, are excluded.

in the genome. Moreover, it is easy to see that a transposition cannot be replaced by two inversions. Hence, the lemma follows with respect to the replacement of transpositions.

Now observe, that any combination of transpositions can neither replace one inversion nor one inverse transposition. This is because both an inversion and an inverse transposition change the sign of at least one gene but a transposition cannot change the sign of a gene. It can also be seen that an inversion cannot be replaced by inverse transpositions when the genome has only one or two genes. It is also not hard to see that an inversion cannot be replaced by two or less inverse transpositions. An inversion of the whole genome cannot be replaced by one inverse transposition plus any number of transpositions. Hence, it follows that for this case at least two inverse transposition plus one transposition are necessary. When the genome has only one gene it is not possible to replace an inversion by any number of inverse transpositions plus transpositions. Thus, the lemma holds also with respect to the replacement of inverse transpositions and inversions. ◀

In the following we will only consider gene orders consisting of at least three elements. Furthermore we exclude the case of the inversion of the complete gene order, i.e. Case 3.b.ii. The seven replacement possibilities that are listed in Lemma 1 imply certain properties that a weighting function for the different types of rearrangement operations has to satisfy in order to make the corresponding rearrangement operation possible for a parsimonious scenario. These properties can be formulated in the form of inequalities between the different weights. A graphical representation of the inequalities and their consequences is given in Fig. 1. The seven inequalities implied by Lemma 1 are:

$$w_T \leq w_I + w_{iT} \quad (1) \qquad w_{iT} \leq w_I + w_T \quad (4) \qquad w_I \leq w_T + w_{iT} \quad (6)$$

$$w_T \leq 3w_I \quad (2) \qquad w_{iT} \leq 2w_I \quad (5)$$

$$w_T \leq 2w_{iT} \quad (3) \qquad\qquad\qquad\qquad w_I \leq 3w_{iT} \quad (7)$$

Each of these inequalities decides if a single rearrangement of a certain type or an alternative more complex, i.e. longer, rearrangement scenario of the other types of rearrangements is parsimonious. The respective single rearrangement operation can occur in a parsimonious scenario only if all of the corresponding inequalities are satisfied, i.e. (1) to (3) for transpositions, (4) and (5) for inverse transpositions, and (6) and (7) for inversions (unless other restrictions exclude one of the corresponding replacement scenarios). If one of these inequalities is violated the corresponding alternative is more parsimonious. For example, when inequality (4) is not satisfied an inverse transposition cannot occur in any parsimonious scenario (unless other restrictions exclude an inversion or a transposition).

In case that a rearrangement operation of a certain type cannot occur in a parsimonious scenario not all of the replacements that are listed in Lemma 1 might be possible in a parsimonious scenario. In the following this aspect is discussed in more detail.

There are three alternatives for a transposition: $iT + I$, $3I$, and $2iT$ with associated weights: $w_{iT} + w_I$, $3w_I$, and $2w_{iT}$. Thus, one can decide between the three alternatives by comparing their weights.

- $iT+I$ needs to be considered only if $w_{iT}+w_I \leq 3w_I$ ($\Leftrightarrow w_{iT} \leq 2w_I$) and $w_{iT}+w_I \leq 2w_{iT}$ ($\Leftrightarrow w_I \leq w_{iT}$).
- $3I$ is a feasible alternative only if $3w_I \leq w_{iT} + w_I$ ($\Leftrightarrow w_{iT} \geq 2w_I$) and $3w_I \leq 2w_{iT}$.
- $2iT$ needs to be considered as alternative only if $2w_{iT} \leq w_{iT} + w_I$ ($\Leftrightarrow w_{iT} \leq w_I$) and $2w_{iT} \leq 3w_I$.

This set of inequalities "partitions" the set of all weighting schemes where transpositions are not parsimonious between the three alternatives (in case of equal weights two or three of the alternatives might be possible). The different sets of the partition are shown as differently patterned areas in Fig. 1.

For the weighting schemes where inverse transpositions are not parsimonious there are the two replacements $2I$ and $I + T$. The former is possible only if $2w_I \leq w_I + w_T$ ($\Leftrightarrow w_I \leq w_T$) holds. Similarly the alternatives for the case that an inversion is not parsimonious, i.e. $T+iT$ and $3iT$, can be chosen on the basis of the comparison $w_T + w_{iT} \leq 3w_{iT}$ ($\Leftrightarrow w_T \leq 2w_{iT}$). As presented in Fig. 1 the remaining weighting schemes are partitioned between the alternatives. It can be readily verified that for each alternative scenario the corresponding necessary rearrangement operations are itself not excluded by any of the other inequalities.

Weighting schemes for the rearrangement model with transpositions, inverse transpositions and inversions as rearrangement operations and the implications of the choice of the weights on the reconstructions that can be obtained with a greedy heuristic that was called `DERANGE` have been discussed in [8]. The weights that have been analyzed in greater detail assumed a fixed weight for inversions and equal weights for transpositions and inverse transpositions that are at least as large as the weight for inversions. In particular, the following (unnormalized) weights have been considered: $w_I = 1$ and weights for transpositions and inverse transpositions that are "somewhat less to somewhat more than" 2, or more exactly $w_T = w_{iT} \in [1 : 3]$. In terms of normalized weights the corresponding set of weights corresponds to a (half open) line in the barycentric plot between the center point of the plot at $w_I = w_T = w_{iT} = 1/3$ and the middle point of the bottom line (i.e. $w_I = 0$) but excluding the middle point itself. This line and five selected points (corresponding to unnormalized inversion weight of 1 and (inverse) transposition weights $1, 1.5, 2, 2.5$, and 3) of it are shown in Fig. 1.

For the considered weighting schemes a "phase transition" for the length of the reconstructions, i.e. the number of rearrangements, made with heuristic `DERANGE` was observed at approximately $w_T = w_{iT} = 2$ [8]. The corresponding strong increase of the reconstruction lengths was observed for random data as well as real, i.e. mitochondrial and bacterial, gene

orders. This effect was attributed to the greedy nature of the algorithm that tries to find a move x, with weight w_x removing B_x breakpoints, minimizing $w_x - B_x$ and the observation that B_x is nearly always $1:2$ for inversions vs. (inverse) transpositions (and not the optimal case $2:3$). This leads to the preference of (inverse) transpositions for $w_{iT} = w_T < 2$ and of inversions for $w_T > 2$ inversions. Since more inversions are necessary to remove all breakpoints the rearrangement scenarios are longer in the latter case.

In the light of the analysis presented here we can add a further explanation for the observed "phase transition". Exactly for the unnormalized (inverse) transposition weight of 2 an inverse transposition has equal weight as the alternative consisting of two inversions. For a weight larger than two inverse transpositions cannot be in a parsimonious rearrangement scenario but they must be replaced by two inversions, i.e. twice the number of rearrangements. The other way round inverse transpositions cannot be replaced by this alternative for weights smaller than two. Based on the empirical results Blanchette et al. [8] suggested to that an (inverse) transposition weight of "slightly greater than 2 may be an appropriate value". Our analysis shows that this is not maintainable for any (optimal/suboptimal) solution, since in such a weighting inverse transpositions are excluded as they need to be replaced by the more parsimonious alternative consisting of two inversions. Another "phase transition" should occur for the reconstructions made with `DERANGE` (and must occur for optimal reconstruction) for unweighted (inverse) transposition weights > 3 which makes inversions the only type of rearrangements that can occur in parsimonious rearrangement scenarios.

4 Inversion, inverse transposition, and block interchange

In this section we study the rearrangement model consisting of inversions, inverse transpositions, and block interchanges, i.e. $\mathfrak{T}_{|T_\epsilon} = \{I, BI, iT\}$. It is assumed here that transpositions are a special case of block interchanges. A block interchange $BI(A, B)$ is called *proper* when there exists an interval $X \neq \emptyset$ such that $X \cap A = \emptyset = X \cap B$ and $A \cup X$ and $X \cup B$ form intervals. X is called the *intermediate interval*.

It is clear that for any rearrangement scenario all the replacements that are listed in Lemma 1 also hold for the rearrangement model $\mathfrak{T}_{|T_\epsilon} = \{I, BI, iT\}$ when a transposition $T(A, B)$ is exchanged by a (non-proper) block interchange $BI(A, B)$. In addition the replacements listed in Lemma 3 are relevant for the $\mathfrak{T}_{|T_\epsilon} = \{I, BI, iT\}$ model.

▶ **Lemma 3.** *For any rearrangement scenario the following replacements are possible within the $\mathfrak{T}_{|T_\epsilon} = \{I, BI, iT\}$ model:*

1. *A proper block interchange $BI(A, B)$ with intermediate interval X can be replaced by each of the following sets of rearrangements*
   - **a.** *three inversions: $I(A \cup X)$, $I(A \cup B)$, $I(B \cup X)$,*
   - **b.** *three inverse transpositions: $iT(A \cup X, B)$, $iT(X, A)$, $iT(A, X)$*
   - **c.** *one inverse transposition and two inversions: $iT(A \cup X, B)$, $I(X)$, $I(A)$, or*
   - **d.** *two inverse transpositions and one inversion: $iT(X, B)$, $iT(A, B)$, $I(A \cup X)$.*

▶ **Lemma 4.** *Lemmas 1 and 3 list all possibilities (wrt. number and type of rearrangement operations) to replace a single rearrangement of a certain type $\in \{BI, iT, I\}$ by a smallest number of rearrangements of one or two of the other types of rearrangements $\in \{BI, iT, I\}$.*

Proof. It is clear that for all replacements listed in Lemma 1 the use of a proper block interchange instead of transpositions can not lead to a shorter replacement. Hence, by Lemma 2 it follows that the result holds for all replacements listed in Lemma 1. It remains to consider replacements for a proper block interchange $BI(A, B)$ as considered in Lemma 3.

Since inversions and inverse transpositions change the sign of at least one gene it is clear that $BI(A, B)$ can neither be replaced by a single inversion nor by a single inverse transposition. Assume that it is possible to replace $BI(A, B)$ by two rearrangements from $\{I, iT\}$. Then the sign changes that are made by the first of the two operations has to be reversed by the second operation (and no other sign changes can be made by the second operation). Hence, the interval with the sign changes has to be the same for both operations. Then it is not possible that both operations are inversions (since then one inversion simply reverses the effect of the other inversion). It can also not be the case that one or both rearrangements are inverse transpositions. This is due to the fact that the interval which is inverted is the same for both operations which implies that their effect is equal to the effect of one transposition. ◀

Interestingly block interchanges and transpositions can be replaced with the same number of inversions, i.e. three, but a larger number of rearrangements is necessary if inverse transpositions or mixed rearrangement types are involved. Another difference of block interchanges to transpositions, as discussed in Section 3, is that there are two alternatives consisting of inversions and inverse transpositions.

The replacements given above are captured by a set of inequalities that need to be satisfied if a certain type of rearrangements can be part of a parsimonious rearrangement scenario. Since the replacements for inverse transpositions and inversions are the same as for $\mathfrak{T}_{|T_\epsilon} = \{I, T, iT\}$ also the corresponding inequalities and properties of the weighting schemes are the same when replacing T by BI. Thus, in the following only the case of the block interchange is discussed. Equations (8) to (11) describe the relations of the weights that render block interchanges impossible if one of them is violated. A visual representation is shown in Fig. 2. Note that, a transposition (as a special block interchange) can be replaced by two (instead of three) inverse transpositions and one inversion and inverse transposition (instead of two for one of the rearrangement types). These replacements would induce tighter bounds on the weights for block interchanges. But since these replacements are not possible for proper block interchanges these tighter bounds cannot be applied in general.

$$w_{BI} \leq w_I + 2w_{iT} \quad (8) \qquad w_{BI} \leq 3w_I \quad (10)$$

$$w_{BI} \leq 2w_I + w_{iT} \quad (9) \qquad w_{BI} \leq 3w_{iT} \quad (11)$$

For weighting schemes where block interchanges are not possible one or more of the replacement scenarios must be employed. For each of the six pairs of replacements the parsimonious replacement is determined by comparing w_I and w_{iT}. Weighing schemes where these two weights are equal are indicated by a dashed line in Fig. 2. Weighing schemes on this line are a "no man's land" where all four replacements have equal weight. Above this line the replacement by three inverse transpositions and below this line the replacement by three inversions is parsimonious. Interestingly the replacement consisting of inversions and inverse transpositions is not parsimonious in these areas but only on the "no man's land". For $w_I = 0.2, w_{BI} = 0.6, w_{iT} = 0.2$ where all lines in the plot intersect, block interchanges and all of its replacements that are listed in Lemma 3 have equal weights.

5 Conclusion and Discussion

In this paper weighting schemes for two rearrangement models have been analyzed formally: 1) inversions, transpositions, and inverse transpositions; 2) inversions, block interchanges, and inverse transpositions. These rearrangement models are important for the analysis of unichromosomal genomes with equal gene content. For both models inequalities have been

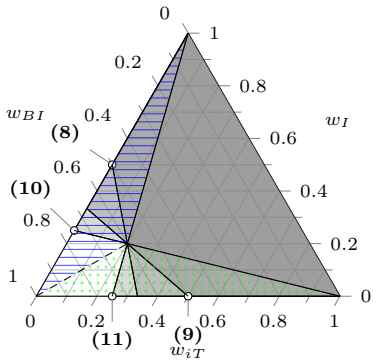


Figure 2 Barycentric plot showing the weighting schemes where block interchanges need to be considered; shaded areas indicate for each of the inequalities the valid weighting schemes; darker shading indicates the area where all inequalities hold; the limiting cases, i.e. equality, is annotated by the corresponding equation number given in bold text; the dashed line gives the demarcation line between the alternative 3I (green dots) and 3iT (blue horizontal lines).

derived that describe weighting schemes for which certain rearrangement types are excluded from parsimonious scenarios. This has been done by analyzing the possibilities to achieve the effects of one rearrangement type by rearrangements of one or more other type(s).

The choice of appropriate weights is an open problem. But, if estimates for the frequency of the different rearrangement operation are available, e.g. from large scale pairwise comparisons [7], it seems to be intuitive to use weights that are inversely (e.g. reciprocal or antiproportional) related to the frequencies. In fact, this is often done to justify chosen weights, e.g. [8]. But then, our results imply hard bounds for the reconstructibility of genome rearrangements by parsimony. If, for instance, inversions are more than three times as frequent as (inverse) transpositions the corresponding reciprocal (unnormalized) weighting scheme ($w_I = 1$ and $w_T, w_{iT} > 3$) forbids an exact reconstruction by parsimony since transpositions and inverse transpositions can not be included in any optimal solution for weighted genome rearrangement problems. These hard bounds might be loosened by using other inverse functional relations of frequency and weight, e.g. by adjusting the factor in case of antiproportionality. Introducing constraints enforcing certain proportions of the frequencies of rearrangement types are another option.

Considering rearrangement models for the case of undirected gene orders (i.e. $\pi = -\pi$), distinguishing between transpositions and proper block interchanges, or incorporating multichromosomal rearrangements (e.g. [11, 23]) is future work.

References

1 Z. Adam, M. Turmel, C. Lemieux, and D. Sankoff. Common intervals and symmetric difference in a model-free phylogenomics, with an application to streptophyte evolution. *J Comput Biol*, 14(4):436–445, 2007.

2 M Bader. Sorting by reversals, block interchanges, tandem duplications, and deletions. *BMC Bioinformatics*, 10(Suppl 1):S9, 2009.

3 M. Bader, M. Abouelhoda, and E. Ohlebusch. A fast algorithm for the multiple genome rearrangement problem with weighted reversals and transpositions. *BMC Bioinformatics*, 9:516, 2008.

4 M. Bader and E. Ohlebusch. Sorting by weighted reversals, transpositions, and inverted transpositions. *J Comput Biol*, 14(5):615–636, 2007.

5 M. A. Bender, D. Ge, S. He, H. Hu, R. Y. Pinter, S. Skiena, and F. Swidan. Improved bounds on sorting by length-weighted reversals. *J Comput Syst Sci*, 74(5):744–774, 2008.

6 M. Bernt, D. Merkle, K. Ramsch, G. Fritzsch, M. Perseke, D. Bernhard, M. Schlegel, P. F. Stadler, and M. Middendorf. CREx: inferring genomic rearrangements based on common intervals. *Bioinformatics*, 23(21):2957–2958, 2007.

7 M. Bernt and M. Middendorf. A method for computing an inventory of metazoan mitochondrial gene order rearrangements. *BMC Bioinformatics*, 12(Suppl 9):S6, 2011.

8 M. Blanchette, T. Kunisawa, and D. Sankoff. Parametric genome rearrangement. *Gene*, 172(1):11–17, 1996.

9 L. Bulteau, G. Fertin, and I. Rusu. Sorting by transpositions is difficult. In *ICALP*, number 6755 in LNCS, pages 654–665, 2011.

10 D. A. Christie. Sorting permutations by block-interchanges. *Inform Process Lett*, 60(4):165–169, 1996.

11 Z. Dias and J. Meidanis. Genome rearrangements distance by fusion, fission, and transposition is easy. *SPIRE'2001*, pages 250–253, 2001.

12 N. Eriksen. Combinatorics of genome rearrangements and phylogeny, 2001.

13 N. Eriksen. $(1 + \epsilon)$-approximation of sorting by reversals and transpositions. *Theor Comput Sci*, 289(1):517–529, 2002.

14 G. Fertin, A. Labarre, I. Rusu, E. Tannier, and S. Vialette. *Combinatorics of Genome Rearrangements*. MIT Press, 2009.

15 Sridhar Hannenhalli and Pavel A. Pevzner. Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. *JACM*, 46(1):1–27, 1999.

16 T. Hartman and R. Sharan. A 1.5-approximation algorithm for sorting by transpositions and transreversals. *J Comput Syst Sci*, 70(3):300–320, 2005.

17 G-H. Lin and G. Xue. Signed genome rearrangement by reversals and transpositions: Models and approximations. In *COCOON*, volume 1627 of *LNCS*, pages 71–80. 1999.

18 Y. Lin, C-Y. Lin, and C. Lin. Sorting by reversals and block-interchanges with various weight assignments. *BMC Bioinformatics*, 10(1):398, 2009.

19 Y-C. Lin, C-L. Lu, Y-C. Liu, and C-Y. Tang. Spring: a tool for the analysis of genome rearrangement using reversals and block-interchanges. *Nucleic Acids Res*, 34(suppl 2):W696–W699, 2006.

20 C. Mira and J. Meidanis. Sorting by block-interchanges and signed reversals. In *ITNG*, pages 670–676, 2007.

21 D. Sankoff. Edit distance for genome comparison based on non-local operations. In *CPM*, volume 644 of *LNCS*, pages 121–135. Springer, 1992.

22 G. A. Watterson, W. J. Ewens, T. E. Hall, and A. Morgan. The chromosome inversion problem. *J Theor Biol*, 99(1):1–7, 1982.

23 S. Yancopoulos, O. Attie, and R. Friedberg. Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics*, 21(16):3340–3346, 2005.

Alignment-free sequence comparison with spaced k-mers

Marcus Boden, Martin Schöneich, Sebastian Horwege, Sebastian Lindner, Chris Leimeister, and Burkhard Morgenstern*

University of Göttingen, Institute of Microbiology and Genetics, Department of Bioinformatics, Goldschmidtstr. 1, 37073 Göttingen, Germany
bmorgen@gwdg.de

Abstract

Alignment-free methods are increasingly used for genome analysis and phylogeny reconstruction since they circumvent various difficulties of traditional approaches that rely on multiple sequence alignments. In particular, they are much faster than alignment-based methods. Most alignment-free approaches work by analyzing the k-mer composition of sequences. In this paper, we propose to use 'spaced k-mers', *i.e.* patterns of deterministic and 'don't care' positions instead of contiguous k-mers. Using simulated and real-world sequence data, we demonstrate that this approach produces better phylogenetic trees than alignment-free methods that rely on contiguous k-mers. In addition, distances calculated with spaced k-mers appear to be statistically more stable than distances based on contiguous k-mers.

1998 ACM Subject Classification J.3 Life and Medical Sciences

Keywords and phrases Alignment-free sequence comparison, phylogeny reconstruction

Digital Object Identifier 10.4230/OASIcs.GCB.2013.24

1 Introduction

Traditional methods for comparative sequence analysis and phylogeny reconstruction are based on pairwise and multiple sequence alignment, see *e.g.* [14, 29] for an overview. During the last years, however, a large number of *alignment-free* methods have been proposed for sequence comparison, see [38] for a review. The main advantage of these methods is that they are much faster than alignment-based approaches. While aligning two sequences takes time proportional to the product of the sequence lengths, most alignment-free approaches work in *linear* time.

Consequently, alignment-free methods are increasingly used for genome comparison, in particular for genome-based phylogeny reconstruction [17, 10, 24]. In addition to being faster, alignment-free approaches circumvent some well-known problems such as finding ortholog genes [32] or aligning large genomic sequences [4]. Another advantage of alignment-free genome comparison is that these approaches can work with unassembled reads [34] and are not sensitive to genome rearrangements. Alignment-free methods have also been used for database searching [40] and to construct *guide trees* as a prerequisite for progressive multiple sequence alignment [22, 11, 3]. Here, alignment-free sequence comparison could crucially speed-up progressive multiple alignment, since the run time for alignment-based phylogeny reconstruction becomes prohibitive if the number of input sequences exceeds a few hundred or so.

* corresponding author

© M. Boden, M.Schöneich, S. Horwege, S. Lindner, C. Leimeister, and B. Morgenstern;
licensed under Creative Commons License CC-BY

German Conference on Bioinformatics 2013 (GCB'13).
Editors: T. Beißbarth, M. Kollmar, A. Leha, B. Morgenstern, A.-K. Schultz, S. Waack, E. Wingender; pp. 24–34

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Most alignment-free methods that have been proposed so far rely on some sort of k-mer statistics. That is, for a fixed integer k, they consider the (relative) frequencies of all possible k-mers for each of the input sequences and then define some distance measure based on these frequency vectors [20, 7, 39, 15]. Standard distance-based methods such as *UPGMA* [33] or *NeighbourJoining* [31] can then be used to calculate phylogenetic trees from the resulting distance matrices. Other approaches consider the local *context* of each sequence position in terms of overlapping k-mers [8]. Some alignment-free methods do not rely on a fixed k but allow for matches of variable length [37, 19, 9]. However, these methods are still based, in one way or the other, on *contiguous* exact matches.

Exact pattern matching is used in many areas of biological sequence comparison, see for example [16]. A traditional application of k-mer comparison in bioinformatics is *database searching*. Fast alignment programs such as *FASTA* [27] and *BLAST* [1] originally relied on identifying word matches of a fixed length. Such word matches, that are referred to as *seeds*, can be rapidly found in an initial phase of the algorithm, in a second phase these 'seeds' are then extended into both directions by slower but more accurate methods for local sequence alignment. The size of seeds is a trade-off between *sensitivity* and *speed*: short words are more sensitive, since more matches are found where local alignments are triggered and evaluated. This increases, however, the running time of these programs, since the local alignment step is the most time-consuming part of the algorithm. Longer word lengths lead to an increase in speed, but result in lower sensitivity.

In a pioneering paper, Ma *et al.* proposed to use *spaced seeds* instead of *contiguous* word matches as the first step in database searching [28]. That is, they proposed to use fixed patterns of *match* and *don't care* positions and to search for word pairs matching at the pre-defined *match* positions, with possible mismatches at the *don't care* positions. Their approach is implemented in the program *PatternHunter* [25]. The main advantage of *spaced seeds* is that hits at different positions are statistically less correlated with each other than contiguous word matches are. Also *spaced seeds* are better able to identify homolog sequence regions in the presence of mismatches. Ma *et al.* showed that for database searching, *spaced seeds* are superior to contiguous word matches in terms of *sensitivity* and *speed*. For this reason, the original contiguous *seeds* have been largely replaced by *spaced seeds* in rapid database search programs. Similarly, Burkhardt and Kärkkäinen [6] used *gapped q-grams* instead of contiguous $q-$grams (q-mers) in a filtering step for the well-studied k-differences problem.

In this paper, we propose to use *spaced k-mers*, *i.e.* k-mers with *don't care* characters at fixed, pre-defined positions, as a basis for alignment-free sequence comparison. Note that this approach is quite different form the above mentioned spaced-seeds approach to database searching: instead of using spaced k-mers to trigger *local alignments* for homology searching, we want to estimate the *global* degree of similarity between sequences by comparing their spaced k-mer composition. To do so, we use a generic distance measure on DNA and protein sequences based on their spaced k-mer frequencies. We use these distances to construct phylogenetic trees for simulated and real-world sequences, and we compare these results with trees constructed by the same method, but with *contiguous* k-mers that are traditionally used by alignment-free methods for sequence comparison. Our study shows that, for phylogeny reconstruction, *spaced k-mers* often outperform *contiguous* k-mers. In addition, we found that the *variance* of distances values calculated from spaced k-mers is lower than the variance calculated with contiguous k-mers.

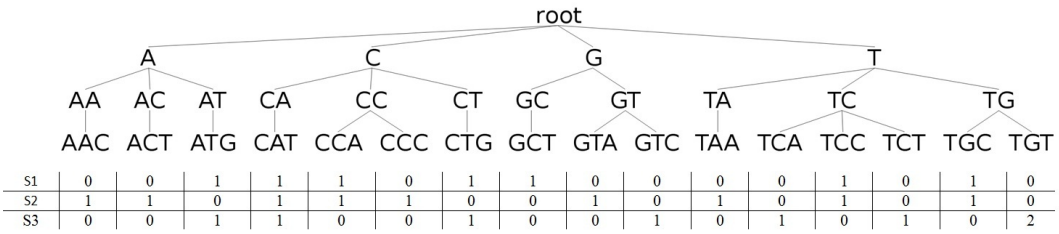


Figure 1 Tree representing the frequencies of *spaced k-mers* ($k = 3$) for an underlying pattern $P = X0X0X$ in a set of three sequences $S1 = ATTCGCCATTG$, $S2 = GTTCACACCATT$, $S3 = GTTCCATTGGTT$. For example, the spaced 3-mer corresponding to the word TGT and pattern P occurs twice in sequence S_3 and does not occur in sequences S_1 and S_2.

2 Calculating trees with spaced k-mers

As usual, for an alphabet $\mathcal{A}$ and $l \in \mathbb{N}$, $\mathcal{A}^l$ denotes the set of all sequences over $\mathcal{A}$ with length l. $S[i]$ denotes the i-th character of a sequence S. In our study, the alphabet $\mathcal{A}$ represents the set of nucleotides or amino acids, respectively. In analogy to the terminology introduced by Ma *et al.*, we define a *spaced k-mer* as a sequence $P \in \{0, X\}^l$, *i.e.* a sequence of '0' and 'X' characters (the underlying *pattern*), such that there are exactly k positions i in P with $P[i] = 1$, together with a finite sequence $w \in \mathcal{A}^k$ (the underlying *word*) with $l, k \in \mathbb{N}$ and $k < l$. In addition, we require that $P[1] = P[l] = X$ holds, *i.e.* the first and the last characters in P must be 'X'. The 'X' positions in the pattern P denote *match* positions while the '0' positions are the *don't care* positions. We call l the *length* of the spaced k-mer and k its *weight*. (One could also include the case $k = l$, but in order to distinguish 'spaced k-mers' from *words* or *k-mers* in the usual sense, we require k to be smaller than l.) We use the notation *length* and *weight* for the underlying pattern P accordingly.

Let α be a spaced k-mer with pattern P, word w, weight k and length l such that $1 = p_1, \ldots, p_k = l$ denote the positions of the 'X' characters in P. We say that α occurs in a sequence S at position i if $S[i + p_j - 1] = w[j]$ for all $1 \leq j \leq k$. For example, the spaced k-mer α consisting of the pattern $P = XX00X$ and the word $w = AGT$ occurs in the sequence $S = GGAGCTTCAGGATCC$ at positions 3 and 9.

In order to define a *distance function* on a set of sequences $S_1, \ldots, S_N$ over $\mathcal{A}$, we consider a fixed pattern P with length l and weight k. We then calculate for each sequence S_i the *relative frequencies* of all possible spaced k-mers that involve our pattern P – relative to the sequence length –, and we represent each sequence S_i by the $|\mathcal{A}|^k$-dimensional vector of the relative frequencies of the *spaced* k-mers with respect to the pattern P - similarly as sequences are represented as vectors of (relative) k-mer frequencies in standard alignment-free approaches.

The spaced k-mer frequencies of the input sequences $S_1, \ldots, S_N$ can be conveniently stored in a tree, as for the usual (contiguous) k-mers, see Figure 1. It is straight forward to calculate the spaced k-mer composition of a sequence of length n in $O(n \times k)$ time with a 'naive' algorithm. For *contiguous* k-mers, this can be reduced to $O(n)$ time, *e.g.* using a *rolling hash* approach [21]. This approach can be easily generalized to spaced k-mers by first considering (contiguous) l-mers and then correcting for the *don't-care* positions. This way, the spaced-k-mer frequencies can be calculated in $O(n \times d)$ where $d = l - k$ is the number of *don't-care* positions in the pattern P.

Once the spaced k-mer compositions are calculated for all input sequences, we proceed

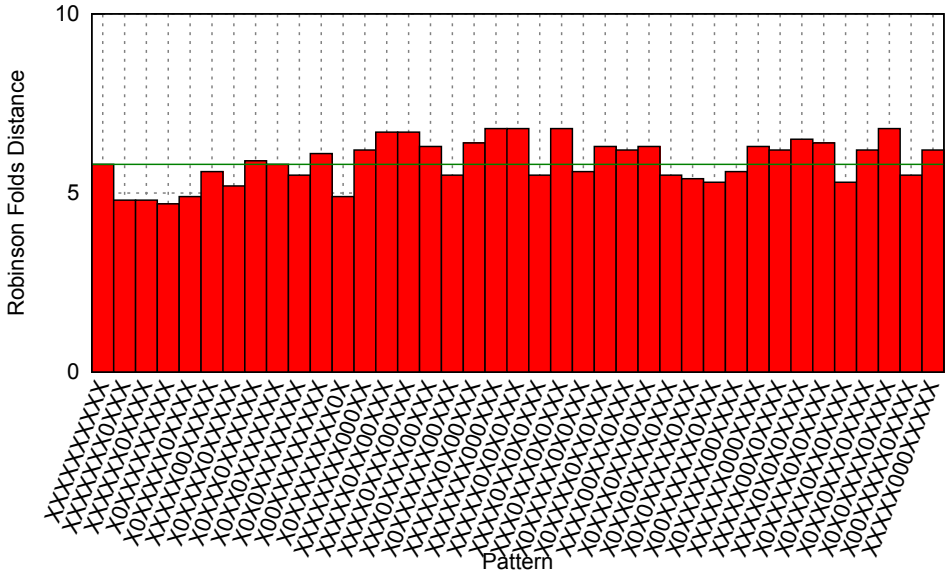


Figure 2 Performance of different patterns of weight $w = 10$ on simulated DNA sequences. We used *Rose* [35] to create 40 sequence sets, each containing 100 simulated DNA sequences of length 20,000. Tree topologies calculated with spaced and contiguous 10-mers were compared to the respective reference trees from *Rose* using the *Robinson-Foulds* metric. The horizontal green line is the RF distance obtained with the contiguous 10-mer.

as other alignment-free methods: we define the *distance* between sequences S_i and S_j as the distances of the corresponding frequency vectors. In the present study, we used the *Jensen-Shannon* distance [26] as this distance measure led to better results than alternative distances. Finally, we construct unrooted trees from the calculated distance matrix using the *Neighbour-Joining* program [31] from the *PHYLIP* package [13].

3 Benchmark data

To evaluate the k-mer and pattern-based methods, we used four different categories of benchmark data, consisting of simulated and real-world sets of DNA and protein sequences. For each sequence set, we used a *reference* tree that we consider to be reliable. We evaluated the methods under consideration by comparing the trees that they produce to the respective reference trees in our benchmark sequence sets.

To generate simulated sequences, we used the program *Rose* [35]. *Rose* mimics molecular evolution by producing a set of sequences along an evolutionary tree, starting with a common ancestral sequence. Mutations are randomly incorporated according to a predefined stochastic model of molecular evolution. As a result, one obtains a set of sequences with known evolutionary history that can be used to benchmark methods for phylogeny reconstruction. The parameter *relatedness* determines the *average* evolutionary distance, measured in *PAM* units, between the sequences produced by *Rose*.

For DNA sequence comparison, we created 40 sets of sequences, each of which containing 20 sequences of length 20,000 using *Rose* with a *relatedness* value of 70. In addition, to evaluate the *variance* of the distances defined with contiguous and spaced k-mers, we created pairs of DNA sequences with *Rose* using different values for *relatedness* (see below for details).

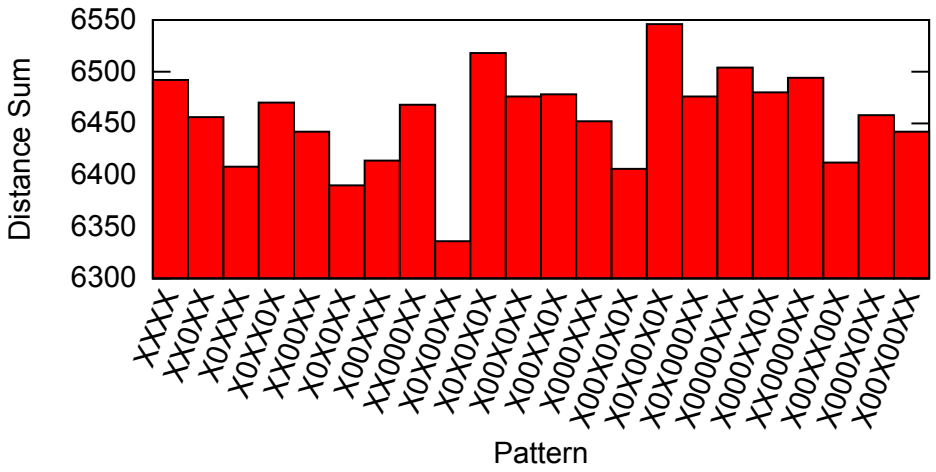


Figure 3 Performance of different patterns of weight $w = 4$ on BAliBASE. The total sum of Robinson-Foulds distances over the BAliBASE is shown for spaced k-mers with weight = 4 and length between 4 and 8.

As real-world DNA sequences, we used a set of 27 primate mitochondrial genomes that has already been used by Haubold *et al.* [18] as benchmark data for alignment-free sequence comparison.

To benchmark the various phylogeny-reconstruction methods on protein sequences, we simulated sets of 100 protein sequences with *Rose*, each sequence with a length of 300. Here, we used the *Rose* default values, together with *relatedness* values of 200, 350, 450 and 550. As real-world protein sequences, we used the *BAliBASE* benchmark database, a standard benchmark database for multiple alignment [2]. Since BAliBASE contains no information about the underlying phylogenetic trees, we applied *Maxium Likelihood* [12] approach to the reference multiple alignments from BAliBASE, and we used the resulting trees as the reference trees in our program evaluation.

To evaluate the various alignment-free methods described in the previous section and to compare them to the classical alignment-based approach, we compared the tree topologies generated by these methods to the topologies of the respective reference trees using the *Robinson-Foulds (RF)* metric [30].

4 Test results

To each category of benchmark data, we first applied the above outlined approach using *contiguous* k-mers with various values for k. This way, we identified for each category of benchmark sequences the k-mer length that gives the best result regarding the RF distances to the respective reference trees. For the this value of k, we then generated patterns with weight k and with varying lengths. Since the number of possible patterns grows rapidly with k, we randomly selected patterns for those data sets where the optimal k was too large to test all possible patterns exhaustively.

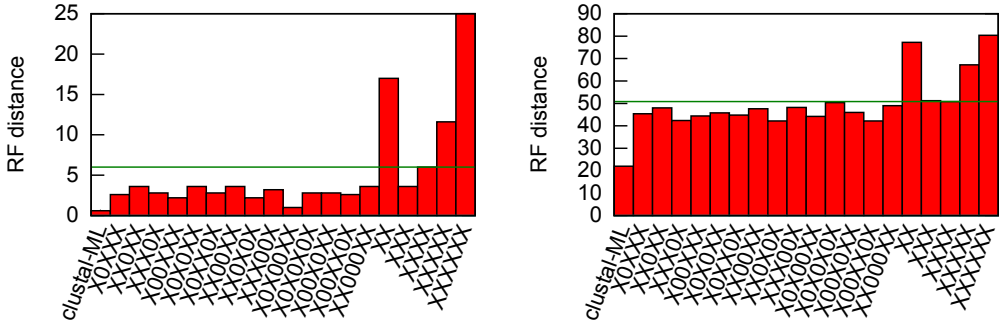


Figure 4 Performance of different approaches on protein sequences, simulated with *Rose* using *relatedness* values of 200 (left) and 350 (right). *Clustal W* and *Maximum Likelihood*

4.1 Genomic sequences

For the simulated DNA sequences of length 20,000, we found that k-mers with $k = 10$ gave the best results, with an average RF distance of 5 to the respective reference trees. Thus, we created patterns with a *weight* of 10; we varied their lengths between 11 and 13. That is, each pattern contained 10 'match' positions and between one and three 'don't care' positions. Overall, we used 32 different patterns of weight $w = 10$, namely 3 patterns of length 11, 8 patterns of length 12 and 21 patterns of length 13; the patterns were randomly selected. The test results with these patterns are shown in Figure 2. *All* patterns of length 11 and most patterns of length 12 produced better results than the corresponding approach with the contiguous 10-mer.

For the primate mitochondrial genomes, we obtained the best results with k-mers of length $k = 8$, so we generated patterns P with a weight of 8 and with varying lengths. In contrast to the simulated DNA sequences, our pattern-defined spaced 8-mers performed slightly worse than the contiguous 8-mer. The RF distance to the reference tree was 4 for the contiguous 8-mer, while the average RF distance for our spaced 8-mers was 4.68, with a range between 2 and 10.

4.2 Protein sequences

On our simulated and real-world protein sequences, contiguous k-mers with $k = 4$ produced the best results, with the exception of simulated *Rose* sequences with *relatedness* of 200 where $k = 3$ performed better, see Figures 3,4, 5. For protein sequences, we therefore generated patterns with weight $k = 4$ and varying lengths.

As shown in Figure 3, 4 and 5, *all* test runs with spaced 4-mers of length $l \leq 7$ gave better results than the corresponding test runs with contiguous 4-mers – with the notable exception of the *periodic* pattern $P = X0X0X0X$ which performed similar to the contiguous k-mers or even worse. On the simulated sequence sets with a *relatedness* value of 550, however, the difference was less clear. On these distantly related protein sequences, contiguous and spaced 4-mers performed almost equally.

We also applied the classical approach to phylogeny reconstruction to our sequence sets by calculating multiple alignments with *Clustal W* [36] and then applying the *Maximum Likelihood (ML)* software from the *PHYLIP* package [13]. The results of these test runs for the simulated protein sequences are inluded in Figures 4 and 5. On *BAliBASE*, the total RF

distance between the *Clustal/ML* trees and the reference trees was 4,478. On all sets of real and simulated protein sequences, this classical approach produced better results than our alignment-free methods, except for the simulated protein sequences with *relatedness* of 550 where both approaches performed comparably.

4.3 Variance of sequence distances calculated with spaced and contiguous k-mers

Finally, we investigated the *variance* of the distances that we used for the various alignment-free methods in this study. To do so, we simulated pairs of DNA sequences with pre-defined evolutionary distances with *Rose*, using *relatedness* values between 2 and 51. For each *relatedness* value we created 200 pairs of sequences of length 2,500 each. We then calculated the distance for each sequence pair as described in section 2 with a value of $k = 5$ which performed best on DNA sequences of this length. We measured the *variance* of these distance values for each value of *relatedness*, for all 200 sequence pairs and for each 5-mer. It turned out that for the contiguous pattern $XXXXX$ and for the periodic pattern $X0X0X0X0X$, the variance was considerably higher than for the non-periodic patterns. The results are summarized in Figure 6.

5 Discussion

The k-mer composition of DNA and protein sequences is frequently used to analyse evolutionary relationships and to construct phylogenetic trees. A certain disadvantage of this approach is the fact that k-mer occurrences at different sequence positions are far from independent from each other. For this reason, some authors corrected the k-mer statistics of sequences for the dependency of overlapping k-mer matches, *e.g.* Göke *et al.* [15]. For the same reason, k-mer matches have been replaced in homology searching by so-called *spaced seeds* where non-periodic patterns of 'match' and 'don't care' positions are used instead of contiguous word matches [28]. Motivated by this approach, we propose to use *spaced k-mers* instead of the traditionally used *contiguous k-mers* to define distances between sequences and to construct phylogenetic trees. While, under an *i.i.d.* Bernoulli model, the *expected* number of occurrences of a *spaced* k-mer in a random sequence is approximately the same as for a *contiguous* k-mer, occurrences of a spaced k-mers at different sequence positions are less dependent, provided that a non-periodic underlying pattern P is used.

To compare *spaced* and *contiguous* k-mers, we implemented a generic approach to phylogeny reconstruction based on the (spaced) k-mer composition of sequences and evaluated the resulting trees on various types of benchmark data. Figures 3, 4 and 5 show that distance matrices based on *spaced* 4-mer frequencies in protein sequences led to consistently better phylogenetic trees than the same approach with *contiguous* 4-mers. This is similar for simulated DNA sequences as shown in Figure 2, although here improvements could only be achieved with shorter patterns containing only up to two *don't care* positions. If the number of don't care positions - and thus the length of the pattern - was further increased, the results deteriorated. This is probably due to the frequency of insertions and deletions in *Rose* which make longer gap-free matches in 'homologous' regions less likely. Not surprisingly, the conventional approach for phylogeny reconstruction using *maximum likelihood* and *multiple alignments* performed better than the alignment-free approaches that we tested. Nevertheless, for very distantly related simulated protein sequences, the performance of k-mer and alignment-based phylogeny methods converged.

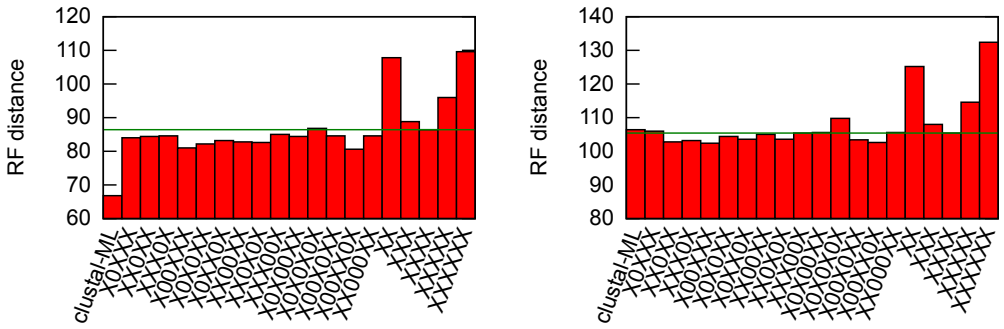


Figure 5 Performance of different approaches on protein sequences, simulated with *Rose* using *relatedness* values of 450 (left) and 550 (right).

As mentioned, a main advantage of spaced k-mers is that matches at different sequence positions are statistically less dependent on each other than matches of contiguous words are - as long as the underlying pattern is non-periodic (or 'irregular'). Distance measures using spaced k-mers can therefore be expected to be more stable than distances based on contiguous words. Figure 6 shows, for example, that the statistical variance of distances based on the contiguous pattern $XXXXX$ and the periodic ('regular') pattern $X0X0X0X0X$ is higher than for non-periodic patterns with the same number of 'match' positions.

Correspondingly, for the real-world and simulated protein sequences, *'non-regular'* patterns for which matches at different sequence positions have less overlap, often performed better than 'regular' patterns where matches at different positions are statistically more dependent. For *BAliBASE* and for the *Rose* sequence sets of relatedness 200 and 350, for example, the 'irregular' pattern $X0X00XX$ performed clearly better than the more 'regular' pattern $XX000XX$. Spaced 4-mers performed always better than contiguous words on these sequence sets – except for the periodic pattern $X0X0X0X$. Spaced k-mers with this periodic pattern often performed similar or even worse than the contiguous 4-mer.

A crucial question in our approach is how to select good patterns P. One approach would be to minimize the dependency of spaced k-mer matches at different sequence positions. First test runs indicate that the summed correlation coefficients for matches at different positions may be an indicator of how good a pattern is at distinguishing random similarities from true homologies.

The statistical properties of *spaced seeds* used in database searching have been studied extensively during the last ten years, and efforts have been made to identify optimal spaced seeds, see for example [23, 5]. Note, however, that these questions are quite different from the questions that are relevant in our approach. In database searching, one is interested in the probability of finding (at least) one hit between sequences with a certain degree of similarity, to trigger a local alignment. This probability determines the *sensitivity* of a spaced seed and has to be balanced against the number of random hits that slow down the program. By contrast, with our spaced k-mer approach, we want to study the *global* degree of similarity between two sequences by comparing their (spaced) k-mer compositions. Here, we are interested in the expected number of matching spaced k-mers and its *variance* in homologous vs. unrelated sequences. We are planning to study the statistical behaviour of spaced and contiguous k-mer matches in more detail to identify optimal patterns for alignment-free sequence comparison and phylogeny reconstruction.

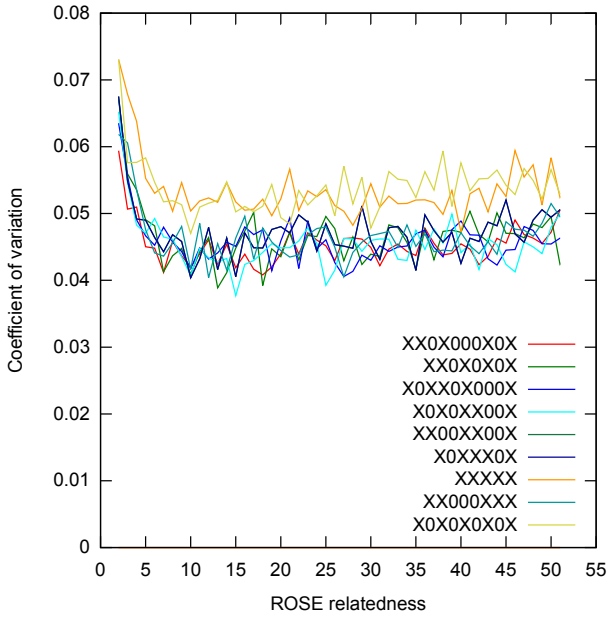


Figure 6 Variation coefficients for the distance values calculated with various *spaced* and *contiguous* 5-mers. We used *Rose* to simulate pairs of DNA sequences with different values of *relatedness*, *i.e.* evolutionary distances. For each value of *relatedness* between 2 and 51, we we generated 200 sequence pairs of length 2,500 and estimated the pairwise distances with the different k-mer based approaches. For each approach, the graphic shows the *variation coefficient* for the resulting 200 distance values.

Acknowledgements We want to thank Thomas Lingner, Anja Sturm, Anirban Mukhopadhyay and Susana Vinga for useful comments and discussions.

References

1 Stephen F. Altschul, Warren Gish, Webb Miller, Eugene M. Myers, and David J. Lipman. Basic local alignment search tool. *J. Mol. Biol.*, 215:403–410, 1990.

2 Anne Bahr, Julie Dawn Thompson, Jean-Claude Thierry, and Olivier Poch. BAliBASE (Benchmark Alignment dataBASE): enhancements for repeats, transmembrane sequences and circular permutations. *Nuc. Acids Research*, 29:323–326, 2001.

3 Gordon Blackshields, Fabian Sievers, Weifeng Shi, Andreas Wilm, and Desmond Higgins. Sequence embedding for fast construction of guide trees for multiple sequence alignment. *Algorithms for Molecular Biology*, 5:21, 2010.

4 Michael Brudno, Alexander Poliakov, Simon Minovitsky, Igor Ratnere, and Inna Dubchak. Multiple whole genome alignments and novel biomedical applications at the vista portal. *Nucleic Acids Research*, 35(Web-Server-Issue):669–674, 2007.

5 Jeremy Buhler, Uri Keich, and Yanni Sun. Designing seeds for similarity search in genomic dna. *J. Comput. Syst. Sci.*, 70:342–363, 2005.

6 Stefan Burkhardt and Juha Kärkkäinen. Better filtering with gapped q-grams. *Fundam. Inf.*, 56:51–70, 2003.

7 Benny Chor, David Horn, Yaron Levy, Nick Goldman, and Tim Massingham. Genomic DNA k-mer spectra: models and modalities. *Genome Biology*, 10, 2009.

8 Gilles Didier. Caractérisation des n-écritures et application à'l étude des suites de complexité ultimement n + cst. *Theor. Comp. Sci*, 215:31–49, 1999.

9 Gilles Didier, Eduardo Corel, Ivan Laprevotte, Alex Grossmann, and Claudine Landés-Devauchelle. Variable length local decoding and alignment-free sequence comparison. *Theoretical Computer Science*, 462:1 – 11, 2012.

10 Gilles Didier, Laurent Debomy, Maude Pupin, Ming Zhang, Alexander Grossmann, Claudine Devauchelle, and Ivan Laprevotte. Comparing sequences without using alignments: application to HIV/SIV subtyping. *BMC Bioinformatics*, 8:1, 2007.

11 Robert C. Edgar. MUSCLE: Multiple sequence alignment with high score accuracy and high throughput. *Nuc. Acids. Res.*, 32:1792–1797, 2004.

12 Joseph Felsenstein. Evolutionary trees from DNA sequences:a maximum likelihood approach. *Journal of Molecular Evolution*, 17:368–376, 1981.

13 Joseph Felsenstein. PHYLIP - Phylogeny Inference Package (Version 3.2). *Cladistics*, 5:164–166, 1989.

14 Joseph Felsenstein. *Inferring Phylogenies*. Sinauer Associates, Sunderland, MA, USA, 2003.

15 Jonathan Göke, Marcel H. Schulz, Julia Lasserre, and Martin Vingron. Estimation of pairwise sequence similarity of mammalian enhancers with word neighbourhood counts. *Bioinformatics*, 28(5):656–663, 2012.

16 Dan Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, Cambridge, UK, 1997.

17 Klas Hatje and Martin Kollmar. A phylogenetic analysis of the brassicales clade based on an alignment-free sequence comparison method. *Front Plant Sci*, 3:192, 2012.

18 Bernhard Haubold, Peter Pfaffelhuber, Mirjana Domazet-Loso, and Thomas Wiehe. Estimating mutation distances from unaligned genomes. *Journal of Computational Biology*, 16(10):1487–1500, 2009.

19 Bernhard Haubold, Nora Pierstorff, Friedrich Möller, and Thomas Wiehe. Genome comparison without alignment using shortest unique substrings. *BMC Bioinformatics*, 6:123, 2005.

20 Michael Höhl, Isidore Rigoutsos, and Mark A. Ragan. Pattern-based phylogenetic distance estimation and tree reconstruction. *Evolutionary Bioinformatics Online*, 2:359–375, 2006.

21 Richard M. Karp and Michael O. Rabin. Efficient randomized pattern-matching algorithms. *IBM Journal of Research and Development*, 31(2):249–260, 1987.

22 Kazutaka Katoh, Kazuharu Misawa, Kei-Ichi Kuma, and Takashi Miyata. MAFFT: a novel method for rapid multiple sequence alignment based on fast fourier transform. *Nuc. Acids Research*, 30:3059 – 3066, 2002.

23 Uri Keich, Ming Li, Bin Ma, and John Tromp. On spaced seeds for similarity search. *Discrete Applied Mathematics*, 138:253 – 263, 2004.

24 Pandurang Kolekar, Mohan Kale, and Urmila Kulkarni-Kale. Alignment-free distance measure based on return time distribution for sequence analysis: Applications to clustering, molecular phylogeny and subtyping. *Molecular Phylogenetics and Evolution*, 65(2):510 – 522, 2012.

25 Ming Li, Bin Ma, Derek Kisman, and John Tromp. PatternHunter II: Highly sensitive and fast homology search. *Genome Informatics*, 14:164–175, 2003.

26 Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37:145–151, 1991.

27 David P Lipman and William R Pearson. Rapid and sensitive protein similarity searches. *Science*, 227(4693):1435–1441, 1985.

28 Bin Ma, John Tromp, and Ming Li. PatternHunter: faster and more sensitive homology search. *Bioinformatics*, 18:440–445, 2002.

29 David A. Morrison. Multiple sequence alignment for phylogenetic purposes. *Australian Systematic Botany*, 19:479–539, 2006.

30 DF Robinson and LR Foulds. Comparison of phylogenetic trees. *Mathematical Biosciences*, 53:131–147, 1981.

31 Naruya Saitou and Masatoshi Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4:406–425, 1987.

32 Fabian Schreiber, Kerstin Pick, Dirk Erpenbeck, Gert Wörheide, and Burkhard Morgenstern. Orthoselect: a protocol for selecting orthologous groups in phylogenomics. *BMC Bioinformatics*, 10:219, 2009.

33 Robert R Sokal and Charles D Michener. A statistical method for evaluating systematic relationships. *University of Kansas Science Bulletin*, 38:1409–1438, 1958.

34 Kai Song, Jie Ren, Zhiyuan Zhai, Xuemei Liu, Minghua Deng, and Fengzhu Sun. Alignment-free sequence comparison based on next generation sequencing reads: extended abstract. In *Proceedings of the 16th Annual international conference on Research in Computational Molecular Biology*, RECOMB'12, pages 272–285, Berlin, Heidelberg, 2012. Springer-Verlag.

35 Jens Stoye, Dirk Evers, and Folker Meyer. Rose: Generating sequence families. *Bioinformatics*, 14:157–163, 1998.

36 Julie D. Thompson, Desmond G. Higgins, and Toby J. Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22:4673–4680, 1994.

37 Igor Ulitsky, David Burstein, Tamir Tuller, and Benny Chor. The average common substring approach to phylogenomic reconstruction. *Journal of Computational Biology*, 13:336–350, 2006.

38 Susana Vinga and Jonas Almeida. Alignment-free sequence comparison - a review. *Bioinformatics*, 19(4):513–523, 2003.

39 Susana Vinga, Alexandra M. Carvalho, Alexandre P. Francisco, Luís M. S. Russo, and Jonas S. Almeida. Pattern matching through chaos game representation: bridging numerical and discrete data structures for biological sequence analysis. *Algorithms for Molecular Biology*, 7:10, 2012.

40 John W. Wilbur and David J. Lipman. Rapid similarity searches of nucleic acid and protein data banks. *Proc. Natl. Acad. Sci. USA*, 80:726–730, 1983.

PanCake: A Data Structure for Pangenomes

Corinna Ernst and Sven Rahmann

Genome Informatics, Institute of Human Genetics, Faculty of Medicine
University of Duisburg-Essen, Essen, Germany
{corinna.ernst,sven.rahmann}@uni-due.de

Abstract

We present a pangenome data structure ("PanCake") for sets of related genomes, based on bundling similar sequence regions into shared features, which are derived from genome-wide pairwise sequence alignments. We discuss the design of the data structure, basic operations on it and methods to predict core genomes and singleton regions. In contrast to many other pangenome analysis tools, like EDGAR or PGAT, PanCake is independent of gene annotations. Nevertheless, comparison of identified core and singleton regions shows good agreements. The PanCake data structure requires significantly less space than the sum of individual sequence files.

1998 ACM Subject Classification E.2 Data Storage Representations, J.3 Life and Medical Sciences

Keywords and phrases pangenome, data structure, core genome, comparative genomics

Digital Object Identifier 10.4230/OASIcs.GCB.2013.35

1 Introduction

With an increasing amount of available sequence data, biological research shifts towards the exploration of the global gene repertoire of related species, called the pangenome, instead of single genomic sequences. The term "pangenome" was introduced in 2005 in a study that compared eight strains of *Streptococcus agalactiae* [13], identified roughly 1800 genes shared by all strains and defined them as the "core genome". The core genome is assumed to consist mainly of genes regulating essential life processes, and hence being indispensable to cell survival. Genes only present in a subset of genomes were called "dispensable" [13]. Analysis of the pangenome of a set of related prokaryotic strains yields insight in the delineation of species and can be taken as a basis for taxonomic classification [10]. Genes identified as specific to a single genome, called "singletons", act as candidates accountable for strain-specific characteristics like virulence or synthesis of certain metabolites [9, 14].

Several tools for the analysis of pangenomes exist. Many of them require pre-computed information stored in databases [2, 4, 5]. Consequently these tools are applicable only for a set of provided strains. Furthermore, most applications rely on the availability of gene annotations [2, 15], which may not be on hand in all cases, or incomplete, or erroneous [11, 12].

The pangenome concept can be extended to the level of pure genomic sequences without annotations. Information about pairwise local sequence similarities can be obtained from alignment tools like `BLAST` [1] or `nucmer` [7]. Based on pairwise similarities, Mancheron et al. [9] introduced an approach for the identification of regions shared by all input sequences, which was subsequently improved by Jahn et al. [6]. Regions that appear similar in all compared sequences are expected to be part of the core genome, while putative singletons lie in areas not aligned to any of the other genomes. Core and singleton identification based on pairwise alignments is independent from annotaions, is able to handle gene duplication events and can even serve as a resource for annotation refinement [9].


© Corinna Ernst and Sven Rahmann;
licensed under Creative Commons License CC-BY
German Conference on Bioinformatics 2013 (GCB'13).
Editors: T. Beißbarth, M. Kollmar, A. Leha, B. Morgenstern, A.-K. Schultz, S. Waack, E. Wingender; pp. 35–45

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

We present a novel approach for the analysis of pangenomes based on pooling related genomic subsequences into common objects, which we call *shared features*. Pangenome-related information of the genomes (such as core regions) can be derived directly from shared features. Additionally, storage requirements are reduced in comparison to pure sequences, even without using explicit compression, but by storing similar sequences via sequences of edit operations with respect to a common reference [3, 8].

In Section 2 we introduce the PanCake data structure, especially shared features and feature instances. In Section 3 we explain our approach of decoding sequences as edit operations with respect to a given reference. In Section 4 we show how the data structure is built iteratively from pairwise alignments between the included genome sequences. In Section 5 we explain how the data structure is used to identify core and singleton regions. Section 6 briefly describes the PanCake software. In Section 7 we report results on strains from three different prokaryotic genera by comparing our findings with those of pangenome analysis tools PGAT [4] and EDGAR [2]. A discussion with outlook concludes the paper.

2 The PanCake Data Structure

In our model, a *pangenome* P consists of $n_g \geq 1$ genomes. Each genome consists of one or several chromosomes, such that the pangenome consists in total of $n_c \geq n_g$ chromosomes. The sequence of chromosome C between positions p and q, inclusive, is written as $C[p : q]$.

The centerpiece of our approach is the bundling of similar subsequences from diverse genomes into a common object, which we call a *shared feature*.

▶ **Definition 1** (pangenome). A *pangenome* consists of a set of genomes, a set of chromosomes, a mapping of chromosomes to genomes, and a set of shared features (Definition 2).

Each shared feature consists of one reference sequence r and a non-empty set of so-called *feature instances*. An example of a shared feature with four feature instances is shown in Figure 1 (left). Each feature instance represents a single genomic interval on a chromosome.

▶ **Definition 2** (shared feature). A *shared feature* is a pair $(r, \mathcal{F})$, consisting of a DNA reference sequence r and a set $\mathcal{F}$ of feature instances (Definition 3).

▶ **Definition 3** (feature instance). A *feature instance* $F = (C, start, stop, S, e, b, prev, next)$ consists of a chromosome identifier C with start and stop positions $start \leq stop$ on C. Further, S references the shared feature F is organized in, e is the sequence of edit operations which have to be applied to S's reference sequence r to obtain the feature instance's sequence, and b is a *direction bit* that takes the values *forward* or *reverse*. If the direction bit is *forward*, application of e to the reference sequence results in the chromosome sequence $C[start : stop]$ directly, otherwise in its reverse complement. The feature instances belonging to the same chromosome are organized as a doubly linked list, with *prev* pointing to the previous (upstream) feature instance ending at position $start - 1$, and *next* pointing to the next (downstream) one starting at position $stop + 1$. (At the chromosome telomers, these take a special null value.)

Any chromosome C can be reconstructed by iterating over linked feature instances, starting from the feature instance covering the chromosome's start and stopping at its end. Linearly iterating over a linked list to access a specific chromosome position can be slow, so we use an index that maps each Δ-th position of a chromosome to the feature instance covering it, for navigation within the data structure. Currently, we use $\Delta = 10\,000$.

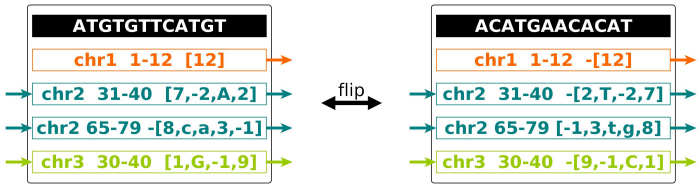


Figure 1 A shared feature before and after reverse-complementing (flipping) with reference sequence `ATGTGTTCATGT` or rev. complement `ACATGAACACAT` and four feature instances (Definition 3). The direction bit is shown as a minus sign in front of the list of edit operations if it is *reverse*. Colored arrows represent links to next and from previous feature instances. The feature instance covering chr1 is not linked to an upstream feature instance because it covers the chromosome's start.

If a shared feature contains only a single feature instance (as all do initially, when no sequence similarities have been detected and processed), we dispense with the overhead of shared features and store such an instance as an *unaligned feature instance*, which is a tuple $(C, start, stop, s, prev, next)$, whereby definitions of C, $start$, $stop$, $prev$ and $next$ are the same as in Definition 3 and s directly spells the covered $C[start : stop]$ (instead of representing it indirectly via a reference r of a shared feature S and edit operations e and a direction bit b).

3 Sequence Encoding by Edit Operations

As explained in Definition 3, the chromosomal sequence s of a feature instance is encoded through a reference sequence r (of the enclosing shared feature) and edit operations e plus a direction bit b. In Section 3.1 we explain how e is derived from a pairwise alignment of r and s. However, if (e.g. for incorporation of similarity information) a feature instance has to be moved from one shared feature to another, its edit operations have to be adapted (rebased) to a new reference sequence without explicit knowledge of the corresponding pairwise alignment. Our rebasing approach is discussed in Section 3.2.

3.1 Deriving Edit Operations from Pairwise Alignments

Edit operations describe the alignment of a feature instance's sequence to the reference sequence of its enclosing shared feature by using the standard operations (match, substitution, insertion, deletion) on single characters. They can be encoded efficiently as byte sequences, as each represented DNA sequence is assumed to be closely similar to the reference. Therefore one can store bytes with the most significant bit deciding wheter a number or character is stored. If a number is stored, the second most significant bit determines the sign. A positive number indicates consecutive matches; a negative number indicates consecutive deletions. If a character is stored, we store its ASCII code in seven bits: Substitutions are encoded by the uppercase IUPAC symbol of the substituted nucleic acid, insertions are encoded by lower-case symbols. In this paper, we show edit operations as lists of numbers and IUPAC symbols, prepending them by a minus sign if and only if the direction bit is *reverse*. It is straightforward to convert between a sequence of edit operations and an alignment, as edit operations are simply a compact encoding of the alignment, given the original sequence. We summarize both e and the direction bit b as an edit transformation $t = (e, b)$ and write $r \overset{t}{\mapsto} s$. Such a transformation is equivalent to a pairwise alignment of r and s, or of r and the reverse complement of s.

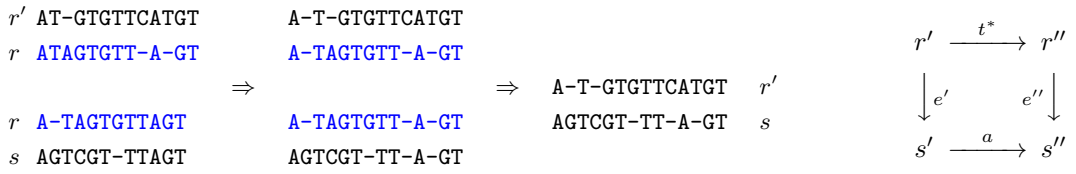


Figure 2 Left: Rebasing s on a new reference r' from previous reference r when a transformation (alignment) between r' and r is known (left): Gaps in both alignments are expanded to their union, such that the gapped representation of the old reference r becomes equal in both alignments (middle) This directly results in an alignment between r' and s by forgetting r (right). Right: Commutative diagram showing how to find a transformation t^* between two reference sequences $r^* = r'$ and r'', when transformations $r' \overset{e'}{\mapsto} s'$, $s' \overset{a}{\mapsto} s''$ and $r'' \overset{e''}{\mapsto} s''$ are given: $t^* = e' \cdot a \cdot e''^{-1}$.

3.2 Rebasing Edit Operations on a Different Reference Sequence

If we have to rebase a feature instance's sequence s on a different reference sequence r', we need to find a transformation t' such that $r' \overset{t'}{\mapsto} s$. Of course, we could compute an optimal alignment between r' and s (or its reverse complement) from scratch. However, this would be time-consuming, and we can assume that we already have an alignment (or transformation u) between r' and original reference r. Algebraically, if we have $r' \overset{u}{\mapsto} r$ and $r \overset{t}{\mapsto} s$, we can compose them to $r' \overset{u}{\mapsto} r \overset{t}{\mapsto} s$, and we will write $t' = u \cdot t$. We will also use the multiplicative notation if a transformation is applied to a sequence, i.e. $r' \cdot t' = r' \cdot u \cdot t = s$. This notation is completely analogous that of linear algebra, multiplying a row vector r' with several (size-compatible) matrices u, t, obtaining a new row vector s. Of course, the operations have nothing in common with matrix multiplication; we simply borrow the notational convenience.

The transformation t' corresponds to a pairwise alignment between r' and s, but not necessarily an optimal one, even if transformations t and u are optimal. The process of composing edit transformations can be understood in terms of pairwise alignements, as explained in Figure 2. In some cases, the resulting alignment or transformation can be simplified directly: In the alignment, columns of gap aligned to gap are removed. Insertions directly followed by deletions (or vice versa) can be converted to substitutions.

4 Building the PanCake Data Structure

Initially, each of the n_c chromosomes in a PanCake data structure is represented by its own shared feature and feature instance. During an iterative building process, feature instances are bundled into shared features on the basis of pairwise alignments computed by external tools. Section 4.1 describes how similarity information arising from a single pairwise alignment is integrated into the data structure. In summary, integration occurs in three steps, namely division (Section 4.2), (conditional) flipping (Section 4.3), and merging (Section 4.4) of shared features.

4.1 Including a Pairwise Alignment into the PanCake Data Structure

Independently of using `BLAST` [1] or `nucmer` [7] for computation, PanCake represents a pairwise alignment as follows.

▶ **Definition 4** (PanCake pairwise alignment)**.** PanCake describes a pairwise alignment $A = (A_1, A_2)$ between two chromosomal intervals by two 5-tuples $A_i = (C_i, start_i, stop_i, b_i, s_i)$ with $i \in \{1, 2\}$. Here $C_i[start_i : stop_i]$ defines the i-th sequence by specifying its chromosome,

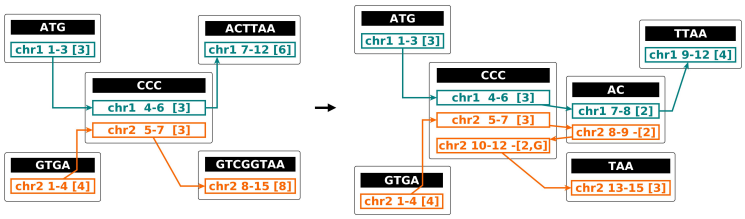


Figure 3 Inclusion of pairwise alignment A into an initial PanCake data structure containing two artificial chromosmomes of length 12bp and 15bp and consisiting of six feature instances organized in five shared features. Let $A = ((\text{chr1}, 4, 8, \textit{forward}, \texttt{CCCAC}), (\text{chr2}, 8, 12, \textit{reverse}, \texttt{CCGAC}))$. As the aligned subsequence chr1[4:8] spans more than one feature instance initially, A is divided into $A' = ((\text{chr1}, 4, 6, \textit{forward}, \texttt{CCC}), (\text{chr2}, 10, 12, \textit{reverse}, \texttt{CCG}))$ and $A'' = ((\text{chr1}, 7, 8, \textit{forward}, \texttt{AC}), (\text{chr2}, 8, 9, \textit{reverse}, \texttt{AC}))$. Then, A' and A'' are integrated separately.

start and stop position ($C_1 = C_2$ is possible). If b_i is *forward*, that sequence is aligned, otherwise its reverse complement. The rows of the alignment (sequences with gap characters) are given by s_1, s_2.

To incorporate the information of a pairwise alignment $A = ((C_1, \mathit{start}_1, \mathit{stop}_1, b_1, s_1), (C_2, \mathit{start}_2, \mathit{stop}_2, b_2, s_2))$ into the data structure, we proceed as follows. We find the (at most four different) feature instances F_1^{start}, F_1^{stop}, F_2^{start} and F_2^{stop}, covering positions $C_1[\mathit{start}_1]$, $C_1[\mathit{stop}_1]$, $C_2[\mathit{start}_2]$ and $C_2[\mathit{stop}_2]$, respectively. We divide F_1^{start} at $C_1[\mathit{start}_1]$ according to Section 4.2 (unless the feature instance ends at that position anyway) and do so analogously for the other feature instances and corresponding positions.

If, thereafter, $C_1[\mathit{start}_1 : \mathit{stop}_1]$ or $C_2[\mathit{start}_2 : \mathit{stop}_2]$ spans more than a single feature instance, we divide the alignment A into several disjoint alignments such that for each resulting subalignment $A' = ((C'_1, \mathit{start}'_1, \mathit{stop}'_1, b'_1, s'_1), (C'_2, \mathit{start}'_2, \mathit{stop}'_2, b'_2, s'_2))$, chromosmal region $C'_1[\mathit{start}'_1 : \mathit{stop}'_1]$ is covered entirely by a feature instance F_1 and chromosomal region $C'_2[\mathit{start}'_2 : \mathit{stop}'_2]$ is covered entirely by F_2. We then merge the shared features containing F_1 and F_2 into a single one according to Section 4.4. Depending on the direction bits of the alignment, one shared feature may have to be flipped before (Section 4.3). If division results in a subalignment A' with either s'_1 or s'_2 consisting exclusively of gaps, then this subalignment is discarded, and no merge is performed. An example is shown in Figure 3.

The resulting data structure depends on the order in which the alignments are processed.

4.2 Dividing a Shared Feature and its Feature Instances

Dividing a feature instance into two disjoint parts implies the division of its containing shared feature and hence all other contained feature instances, too.

Given a feature instance $F = (C, \mathit{start}, \mathit{stop}, S, e, b, \mathit{prev}, \mathit{next})$ and a cutting index c with $1 \le c \le \mathit{stop} - \mathit{start} + 1$, the task is to divide F at distance c from the start or stop position, depending on the direction bit b. This results in F being divided into two new feature instances F' and F'' and corresponding new shared features S' and S''.

Precisely, if b is *forward*, this results in two new feature instances

$$\begin{aligned} F' &= (C, \ \mathit{start}, \ \mathit{start} + c - 1, \ S', \ e', \ \mathit{forward}, \ \mathit{prev}, \ F''), \\ F'' &= (C, \ \mathit{start} + c, \ \mathit{stop}, \ S'', \ e', \ \mathit{forward}, \ F', \ \mathit{next}). \end{aligned}$$

Otherwise, if b is *reverse*, this results in

$$\begin{aligned} F' &= (C, \ \mathit{start}, \ \mathit{stop} - c, \ S', \ e', \ \mathit{reverse}, \ \mathit{prev}, \ F''), \\ F'' &= (C, \ \mathit{stop} - c + 1, \ \mathit{stop}, \ S'', \ e', \ \mathit{reverse}, \ F', \ \mathit{next}). \end{aligned}$$

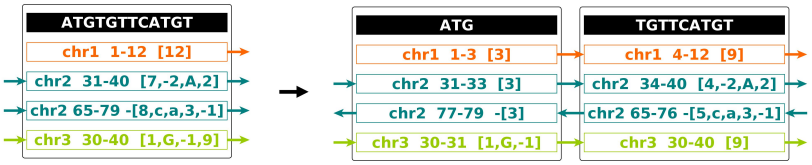


Figure 4 Division of the orange feature instance $F = (\text{chr}1, 1, 12, S, [12], \textit{forward}, \textit{prev}, \textit{next})$ at cutting index $c = 3$. Computing the cutting indexes for the reference r and all other feature instances $\tilde{F} \in S$ results in $c_r = 3$ and $c_{\tilde{F}} = 3$ except instance $(\text{chr}3, 30, 40, \dots)$, where $c_{\tilde{F}} = 2$. The newly formed shared features are *concatenated shared features*, cf. Section 5.

Note that *all* feature instances of the corresponding shared feature S have to be divided as well to maintain the data structure. An example is given in Figure 4.

To divide the containing shared feature S, we must compute the position c_r at which the reference r of S has to be divided, such that the references of S', S'' are $r' = r[1 : c_r]$ and $r'' = r[(c_r + 1) : |r|]$. Computation of c_r from c and implicit sequence s of F proceeds by counting positions in the implicit aligment represented by edit transformation $r \overset{(e,b)}{\mapsto} s$ until the length of the processed part of s becomes $\geq \mathsf{c}$.

Once c_r is known, divided shared features S', S'' are initialized with the prefix and suffix of the reference sequence, respectively. Their feature instances are included successively while iterating over all feature instances in S. For each feature instance $\tilde{F} \in \mathcal{F} \setminus F$, splitting position $\tilde{c}$ is determined (analogously to computation of c_r) in order to compute new edit operation lists $\tilde{e}', \tilde{e}''$ and adapt the start and stop positions of the newly formed feature instances. Start and stop positions of the divided feature instances $\tilde{F}'$ and $\tilde{F}''$, as well as their links to next and previous feature instances, depend on $\tilde{F}$'s direction bit $\tilde{b}$.

During division, two special cases may arise. First, empty feature instances $\tilde{F}'$ or $\tilde{F}''$ with edit operations consisting of only deletions may occur. Such feature instances are deleted entirely and links from previous and next instances adjusted accordingly. Second, an empty reference sequence $r_{S'}$ or $r_{S''}$ may occur if the beginning or end of F's edit operation list e consists exclusively of insertions. Then any feature instance is chosen (e.g., randomly) whose decoded sequence provides the new reference. The edit operations of the remaining feature instances are then rebased on the new reference (Section 3.2).

Dividing a shared feature and its feature instances requires updating the navigation index if for any chromosome, a position divisible by Δ belonged to the divided S.

4.3 Flipping a Shared Feature

Flipping a shared feature $S = (r, \mathcal{F})$ works as follows. After reverse-complementing the reference sequence r, new edit operations are computed for each feature instance $F \in \mathcal{F}$ by reversing the sequence, reverse-complementing symbols referring to substitutions or insertions, and finally flipping F's direction bit. By flipping, none of the chromosome sequences change, but only their representation by a reference sequence and edit operations. Applying flipping twice results in the original representation.

4.4 Merging Shared Features

When two shared features $S' = (r', \mathcal{F}')$ and $S'' = (r'', \mathcal{F}'')$ have similar reference sequences $r' \approx r''$ (e.g., as evidenced by finding a good alignment between feature instances of $S' \neq S''$), it is beneficial to merge S' and S'' into a new combined shared feature $S^* = (r^*, \mathcal{F}' \cup \mathcal{F}'')$

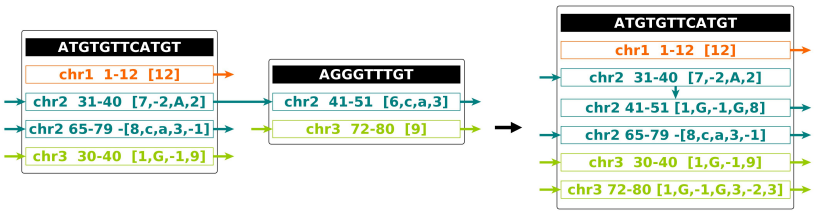


Figure 5 Merging two shared features, assuming that an alignment between chr2, positions 41–51, and chr3, positions 30–40, has been found.

containing all feature instances from both S' and S''. Assume without loss of generality that S' is larger, i.e., $|\mathcal{F}'| \geq |\mathcal{F}''|$. Then we pick $r^* := r'$ as reference and move each $F' \in \mathcal{F}'$ into $\mathcal{F}^*$ unmodified. We have to base each feature instance of S'' on $r^* = r'$. This works as explained in Section 3.2, but we need to know a transformation t^* such that $r' \overset{t^*}{\mapsto} r''$. In a typical case, we do not have such a transformation available directly, but first need to compute it from a given alignment (e.g., found by BLAST) between two feature instances F' and F''. Say the rows of the pairwise alignment are given by s' and s'', corresponding to an edit transformation a. We also know edit transformations $r' \overset{e'}{\mapsto} s'$ and $r'' \overset{e''}{\mapsto} s''$ stored in the feature instances. Now $t^* = e' \cdot a \cdot e''^{-1}$, as shown by the commutative diagram in Figure 2 (Right). Here e''^{-1} denotes the inverse edit transformation, replacing insertions by deletions and vice versa and swapping goal and target of substitutions. An example of the result of merging two shared features is provided in Figure 5.

Merging two shared features requires updating the navigation index for those positions divisible by Δ contained in the smaller S''.

5 Applications: Core and Singleton Identification

Once a PanCake representation of several genomes is built and stored, there are several applications; the two most important of which are the identification of singletons (here meaning genomic regions not shared with any other genome) and of the core genome (here referring to genomic regions shared with every other genome).

Singleton identification is straightforward. By definition, the set of singleton regions of a genome G consists of all *unaligned* feature instances belonging to chromosomes of G (see Section 2) or being part of a shared feature containing exclusively feature instances originating from G. They can be easily enumerated by iterating over all feature instances of chromosomes of G.

The core genome can be identified by considering core features, defined as follows.

▶ **Definition 5** (core feature). A $\mathcal{G}$-*core feature* for a set $\mathcal{G}$ of genomes is a shared feature that contains at least one feature instance from each genome $G \in \mathcal{G}$.

Identification and enumeration of core features is straightforward by using the map of contained chromosomes to genomes. If we only need to know which positions in a genome belong to the core, we are done now. However, we additionally want to list all (maximal) core regions whose definition corresponds to maximum common intervals (MCIs) by Mancheron et al. [9] or maximum overlapping intervals (MOIs) by Jahn et al. [6]. We first need the notion of concatenated shared features.

▶ **Definition 6** (concatenated shared features). An ordered pair (S', S'') of shared features is *concatenated*, if they have the same number of feature instances and for all feature instances

F' in S' with direction bit *forward*, there exists a feature instance F'' in S'' with $next' = F''$ and direction bit *forward*, and for all F' in S' with direction bit *reverse*, there exists F'' in S'' with $next'' = F'$ and direction bit *reverse*. (S', S'') is also called concatenated if any combination of flipping results in concatenation.

Concatenated shared features can arise from dividing a shared feature (Section 4.2; Figure 4). The concept is iteratively extended to more than two shared features. We now define what it means that a sequence of consecutive feature instances from the same chromosome forms a *core region part* and then define a *core region* as a maximal core region part.

▶ **Definition 7** (core region part)**.** Let $\mathcal{G}$ be a set of genomes. Let F_i, $1 \leq i \leq n$ be consecutive feature instances (meaning that $stop_i + 1 = start_{i+1}$ for all i) on a chromosome C from genome $G \in \mathcal{G}$. Let $S_i = (r_i, \mathcal{F}_i)$ be the shared feature containing F_i.

The F_i, $1 \leq i \leq n$ (and the induced interval $C[start_1, stop_n]$) is a *core region part of* G with respect to $\mathcal{G}$, if for all $1 \leq i \leq n$, there exists a subset of feature instances, $\hat{\mathcal{F}}_i \subseteq \mathcal{F}_i$, such that: (a) the reduced shared features $\hat{S}_i := (r_i, \hat{\mathcal{F}}_i)$ are concatenated shared features, and (b) for each genome $G' \in \mathcal{G}$, there exists at least one feature instance in $\hat{\mathcal{F}}_i$ covering a chromosome from G'.

Every core feature gives rise to a core region part for each of its contained feature instances, but a core region part can be longer than a single feature instance. Our interest lies in connecting these parts to a (full) core region which cannot be extended further.

▶ **Definition 8** (core region)**.** A core region part (F_i), $1 \leq i \leq n$, of C or G is a *core region* of C or G if it is maximal in the sense that both upstream elongation (by prepending $prev_1$) and downstream elongation (by appending $next_n$) do not yield core region parts.

Identification of core regions of a chromosome C with respect to a genome set $\mathcal{G}$ occurs in two steps. First, starting from each C-covering feature instance F contained in a core feature with respect to $\mathcal{G}$, we search for the longest core region part of C starting with F. In a second step, whenever identified core region parts share stop positions on the reference genome, shorter ones are removed from output.

In practice, there exist short unaligned feature instances (e.g., 1–5 bp) between otherwise concatendated shared features, breaking the concatenation property. As this results in unnecessary cutting of long core regions, we have implemented a gap-tolerant version of the above method that ignores intermediate feature instances shorter than a user-defined length.

6 The PanCake Software

The PanCake software is written in Python 3.2 and available from `https://bitbucket.org/CorinnaErnst/pancake` under the MIT license. It has a command-line interface with several subcommands, allowing to add chromosomes from `.fasta` files, to specify a genome for each chromosome, to add alignments, to compute core and singleton regions, and to output selected subsequences of the contained chromosomes. Intermediate representations of the data structure are serialized into a text-based file format (PanCake `.pan` format), which is manipulated by these subcommands. Even though unoptimized, significant savings against the `.fasta` files are visible (cf. Table 1). Core genome computation outputs a `.bed` file with intervals for each chromosome covering the core regions, and optionally one `.fasta` file per core region, containing their unaligned sequences, so they can be optimally aligned and inspected with standard tools.

Table 1 PanCake statistics on three genera (see text). Computation time refers to a single core on an Intel Core i7-2600 CPU at 3.40GHz with 8 GB RAM.

Genus (number of strains)	genome size [Mbp]	FASTA size [MB]	number of alignments	PanCake size [MB]	comp time
Pseudomonas (3)	19.4	19.7	1405	7.7 (39%)	20 sec
Yersinia (8)	38.0	38.5	324925	8.9 (23%)	16.5 h
Burkholderia (10)	65.3	66.2	147344	22.0 (33%)	10.8 h

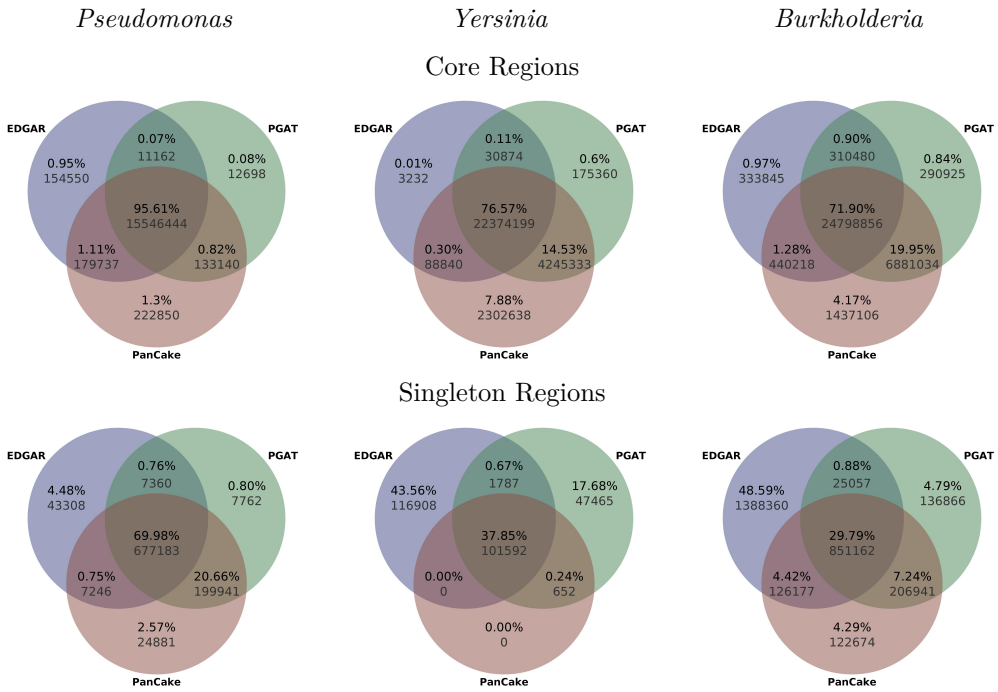


Figure 6 Overlap of core and singleton regions as identified by EDGAR, PGAT and PanCake on the datasets of Table 1.

7 Results

We compare the results of PanCake against those of two other comparative genome analysis tools: EDGAR [2] and PGAT [4]. Both approaches identify core genes and singletons by comparing pre-annotated coding sequences only, while PanCake does whole-genome comparisons. As EDGAR and PGAT are web-based database applications, analysis is limited to the sets of provided pre-processed strains, at least in open access mode. PGAT provides 8 bacterial genera, from which we chose *Pseudomonas*, *Yersinia* and *Burkholderia*; we exclude strains marked as draft assembly and strains with chromosomes or plasmids which are absent in EDGAR's open access mode. This results in the following sets of strains:

Pseudomonas (3 strains): *P. aeruginosa PAO1*, *P. aer. UCBPP-PA14*, *P. aer. LESB58*.

Yersinia (8 strains): *Y. pestis Angola*, *Y. pestis Antiqua*, *Y. pestis KIM 10*, *Y. pestis Microtus 91001*, *Y. pestis Nepal 516*, *Y. pestis Pestoides F*, *Y. pestis Z176003*, *Y. pestis CO92*.

Burkholderia (10 strains): *B. pseudomallei 1026b*, *B. pseudomallei 1106a*, *B. pseudomallei 1710b*, *B. pseudomallei 668*, *B. pseudomallei K96243*, *B. mallei ATCC 23344*, *B. mallei NCTC 10229*, *B. mallei NCTC 10247*, *B. mallei SAVP1*, *B. thailandensis E264*.

With PanCake, we build a data structure for each genus, using all pairwise alignments computed by `nucmer` [7] with option `--maxmatch` (use all anchor matches regardless of their uniqueness), removing obivously redundant alignments. Some statistics are given in Table 1.

For each genus we compute the core genome and singletons, according to each tool's definitions and default parameters and recommendations, using the EDGAR web applications and PGAT's 'Prescence and Abscence Tool' with option 'consider pseudogenes as present'. As EDGAR acts exclusively on annotated genes, we only evaluate on such positions.

The results (Figure 6) show that core regions identified by the three tools are in good agreement. For *Yersinia* and *Burkholderia*, the amounts of identical core positions identified by PanCake and PGAT are higher than in combinations including EDGAR. This may be explained by EDGAR's approach of determining orthologs stringently by bidirectional best BLAST hits [2], resulting in fewer predicted core regions and more predicted singleton regions. In contrast, PGAT allows genes to be related to various similar regions in other genomes; so PGAT's results show significantly better agreement with the results of PanCake than with EDGAR. Concerning the singleton regions, only small amounts from 4.29% down to 0% of the genomic positions identified by PanCake do not agree with one of the other tools.

8 Discussion and Conclusion

We presented a data structure and software implementation (PanCake) for pangenomes. It is based on pooling similar genomic subsequences, as evidenced by pairwise alignments, into shared features. We discussed basic operations on the data structure (flipping, re-basing, division, merging) and how to iteratively build it from alignments. We also presented a method to identify the core genome and singletons from the data structure. Comparison with PGAT and EDGAR shows good agreement with PGAT, while EDGAR uses a more stringent approach to identify orthologs (instead of "similar regions"). PanCake is not restricted to annotated genes, and the data structure can be built iteratively from available (un-annotated) FASTA files and stored persistently.

In the future, we aim to reduce computation times (the current version uses un-optimized pure Python code) and storage requirements by optionally using a more efficient binary format to store the data structure. Already, the PanCake file size is only 40% to 25% of the sum of FASTA file sizes. At the moment, the resulting representation of the data structure depends on the order in which the alignments are processed (and on the quality of the alignments themselves). We are working on a refactoring operation that will provide a better representation of each shared feature (say, using a median reference sequence with short edit operation lists). New classes of shared features, such as one representing variable-length repeats, are also of interest, as well as avoiding occasional artifacts of short feature instances that result from division when an alignment does not end at, but close to, the border of a shared feature. To determine core regions faster, efficient algorithms [6, 9] could be implemented.

The current state of PanCake is a proof of concept. We plan to substantially broaden PanCake's applications by including additional features, such as better support for other alignment tools, optional inclusion of annotation data, taxonomic analysis and creation of synteny plots. A typical future query might be: Output all regions (in any genome) that are similar to the *metH* gene in any *Y. pestis* strain.

Acknowledgements We thank Johannes Köster and Marcel Martin for helpful discussions and sharing their Python knowledge.

References

1 S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

2 J. Blom, S. P. Albaum, D. Doppmeier, A. Pühler, F.-J. Vorhölter, M. Zakrzewski, and A. Goesmann. EDGAR: a software framework for the comparative analysis of prokaryotic genomes. *BMC Bioinformatics*, 10:154, 2009.

3 M. C. Brandon, D. C. Wallace, and P. Baldi. Data structures and compression algorithms for genomic sequence data. *Bioinformatics*, 25(14):1731–1738, 2009.

4 M. J. Brittnacher, C. Fong, H. S. Hayden, et al. PGAT: a multistrain analysis resource for microbial genomes. *Bioinformatics*, 27(17):2429–2430, 2011.

5 T. Davidsen, E. Beck, A. Ganapathy, R. Montgomery, N. Zafar, Q. Yang, R. Madupu, P. Goetz, K. Galinsky, O. White, and G. Sutton. The comprehensive microbial resource. *Nucleic Acids Research*, 38(Database issue):D340–345, 2010.

6 K. Jahn, H. Sudek, and J. Stoye. Multiple genome comparison based on overlap regions of pairwise local alignments. *BMC Bioinformatics*, 13(Suppl 19):S7, 2012.

7 S. Kurtz, A. Phillippy, A. L. Delcher, M. Smoot, M. Shumway, C. Antonescu, and S. L. Salzberg. Versatile and open software for comparing large genomes. *Genome Biology*, 5(2):R12, 2004.

8 P.-R. Loh, M. Baym, and B. Berger. Compressive genomics. *Nature Biotechnology*, 30(7):627–630, 2012.

9 A. Mancheron, R. Uricaru, and E. Rivals. An alternative approach to multiple genome comparison. *Nucleic Acids Research*, 39(15):e101, 2011.

10 D. Medini, C. Donati, H. Tettelin, V. Masignani, and R. Rappuoli. The microbial pan-genome. *Current Opinion in Genetics & Development*, 15(6):589–594, 2005.

11 M. S. Poptsova and J. P. Gogarten. Using comparative genome analysis to identify problems in annotated microbial genomes. *Microbiology*, 156(7):1909–1917, 2010.

12 A. M. Schnoes, S. D. Brown, I. Dodevski, and P. C. Babbitt. Annotation error in public databases: Misannotation of molecular function in enzyme superfamilies. *PLoS Computational Biology*, 5(12), 2009.

13 H. Tettelin, V. Masignani, M. J. Cieslewicz, et al. Genome analysis of multiple pathogenic isolates of streptococcus agalactiae: implications for the microbial "pan-genome". *Proc. Natl. Acad. Sci.*, 102(39):13950–13955, 2005.

14 E. Trost, J. Blom, S. C. Soares, et al. Pangenomic study of corynebacterium diphtheriae that provides insights into the genomic diversity of pathogenic isolates from cases of classical diphtheria, endocarditis, and pneumonia. *Journal of Bacteriology*, 194(12):3199–3215, 2012.

15 Y. Zhao, J. Wu, J. Yang, S. Sun, J. Xiao, and J. Yu. PGAP: pan-genomes analysis pipeline. *Bioinformatics*, 28(3):416–418, 2012.

Reconstructing Consensus Bayesian Network Structures with Application to Learning Molecular Interaction Networks

Holger Fröhlich[1] and Gunnar W. Klau[2]

1 **University of Bonn, Bonn-Aachen International Center for IT, Algorithmic Bioinformatics**
Dahlmannstr. 2, 53113 Bonn, Germany
frohlich@bit.uni-bonn.de

2 **Centrum Wiskunde & Informatica (CWI), Life Sciences**
Science Park 123, 1098 XG Amsterdam, The Netherlands
gunnar.klau@cwi.nl

Abstract

Bayesian Networks are an established computational approach for data driven network inference. However, experimental data is limited in its availability and corrupted by noise. This leads to an unavoidable uncertainty about the correct network structure. Thus sampling or bootstrap based strategies are applied to obtain edge frequencies. In a more general sense edge frequencies can also result from integrating networks learned on different datasets or via different inference algorithms. Subsequently one typically wants to derive a biological interpretation from the results in terms of a consensus network. We here propose a log odds based edge score on the basis of the expected false positive rate and thus avoid the selection of a subjective edge frequency cutoff. Computing a score optimal consensus network in our new model amounts to solving the maximum weight acyclic subdigraph problem. We use a branch-and-cut algorithm based on integer linear programming for this task. Our empirical studies on simulated and real data demonstrate a consistently improved network reconstruction accuracy compared to two threshold based strategies.

1998 ACM Subject Classification J.3 Life and Medical Sciences, I.5 Pattern Recognition, G.1.6 Optimization, G.2.2 Graph Theory, G.3 Probability and Statistics

Keywords and phrases Bayesian Networks, Network Reverse Engineering, Minimum Feedback Arc Set, Maximum Acyclic Subgraph, Molecular Interaction Networks

Digital Object Identifier 10.4230/OASIcs.GCB.2013.46

1 Introduction

Reverse engineering of biological networks is key to the understanding of biological systems. The exact knowledge of interdependencies between genes and proteins not only provides deep insight into the functionality of a cell, but is crucial for the identification of drug targets for various diseases. Apart from literature driven approaches, a wide range of methods from statistics and machine learning for estimating regulatory networks from experimental data has been proposed. Examples thereof are static and dynamic Bayesian Networks (BNs) [5, 16, 18, 25, 14, 7], correlation and mutual information based algorithms, such as ARACNE [11, 27], and Gaussian Graphical Models (GGMs) [19, 13, 17, 22], see [12] for a review. In this paper our focus lies on Bayesian Networks as an established probabilistic graphical modeling approach for network reverse engineering.

© Holger Fröhlich and Gunnar W. Klau;
licensed under Creative Commons License CC-BY
German Conference on Bioinformatics 2013 (GCB'13).
Editors: T. Beißbarth, M. Kollmar, A. Leha, B. Morgenstern, A.-K. Schultz, S. Waack, E. Wingender; pp. 46–55

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

When learning BN structures from experimental data the uncertainty about individual network structures has to be taken into account. Each edge can only be inferred up to a certain probability. In a Bayesian sense this amounts to compute posterior probabilities, which can be approximated via Markov Chain Monte Carlo (MCMC) sampling techniques. From a frequentist point of view, bootstrapping [4] provides a possibility to assess edge confidences. Both approaches result in a matrix of relative frequencies for each edge, which then can be thresholded appropriately to decide which interactions are supported by the data with high confidence and which not. Similarly, a matrix of relative edge frequencies can result from integrating networks inferred on different datasets [8] or via different inference algorithms [10].

In any of these cases the choice of the threshold above which edges are considered to be relevant is typically highly subjective and can influence the subsequent biological interpretation significantly. In addition, a complicating but most often ignored issue is that applying a cutoff typically leads to network structures that are inconsistent with the original model assumptions for BNs. That means that the obtained network graph is no longer acyclic (more specifically: a completed partially directed acyclic graph). As a consequence this may result in a graph that is principally impossible to model via BNs. This can in turn lead to a reduced network reconstruction accuracy, because it only makes sense to choose BNs as a modeling approach, if indeed the true biological network can be assumed to be acyclic.

Our contribution in this paper is two-fold: First, we propose a method to avoid the selection of an arbitrary frequency cutoff by estimating the proportion of false positive edges. This results in an edge-wise score, which is essentially an adjusted log odds ratio. Second, we show that computing a score optimal consensus network structure in this new model amounts to solving the maximum weight acyclic subdigraph problem. We propose a branch-and-cut algorithm based on integer linear programming (ILP) for this task. Our simulation results indicate that we enhance the network reconstruction accuracy significantly in comparison to a threshold guided as well as the Consensus Bayesian Network algorithm by Steele and Tucker [21], which is not guaranteed to yield a valid completed partially directed acyclic graph (CPDAG) structure. The utility of our approach is further demonstrated by applications on learning the yeast heat-shock response network [24] as well as the yeast Raf signaling pathway structure [18]. Notably, our proposed approach is fast enough to run on a standard desktop computer within a few seconds.

2 Problem Definition

Given a set $\mathcal{G} = \{G_1, G_2, \ldots, G_\ell\}$ of ℓ Bayesian network structures on the same set of nodes V. For example, each $G_k \in \mathcal{G}$ may be inferred on a different dataset, a different data sub-set or via a different learning algorithm. We propose to find a consensus CPDAG $G^* = (V, E^*)$, which explains a maximal fraction of the observed edges. A CPDAG is defined as the set of all Markov equivalent directed acyclic graph (DAG) structures and can be represented conveniently by a partially directed graph (see [15] for an excellent description). It can be computed from a DAG by making all edges undirected that are neither part of a v-structure (i.e., a motif of the form $A \to C \leftarrow B$) nor by reversal introduce any new v-structure.

Let $F = (f_{ij}), i, j = 1, \ldots, n$ with $n = |V|$ be a matrix of observed relative edge frequencies. Then we define G^* as the CPDAG that characterizes the equivalence class of the maximum weight DAG $D^* = (V, A^*)$, where $A^* \subset V \times V$ and the weight of an edge $(i, j) \in V \times V$ is given by a suitable weight function derived from the frequencies. In other words, the task is to find the maximum weight acyclic subdigraph D^* of a complete digraph with edge weights

$w(F)$. A natural weight function is, for example, $w(ij) = f_{ij} - \theta$, for $1 \leq i, j \leq n$, where θ is a suitable frequency threshold. In the next section, we propose a more sophisticated weight function based on the expected false positive rate of edges.

Once D^* is computed, the CPDAG that characterizes the corresponding equivalence class can be calculated via the algorithm of Chickering [3]. It has to be emphasized that Bayesian Network structures can principally only be resolved up to these equivalence classes.

3 Expected FPR Adjustment

In the absence of any gold standard network the selection of a frequency cutoff θ becomes a practically difficult problem. We here suggest a rational approach based on the expected false positive rate of edges. The underlying idea is that relative edge frequencies $\{f_{ij}\}$ can be assumed to be drawn from a mixture of two beta distributions, namely $\text{Beta}(\alpha, 1)$, if in reality the edge exists, and $\text{Beta}(1, \beta)$, if in reality there is no such biological interaction. Correspondingly we have:

$$f_{ij} \sim \pi_0 \text{Beta}(1, \beta) + (1 - \pi_0)\text{Beta}(\alpha, 1) \tag{1}$$

The parameters of this mixture model can be estimated conveniently via an expectation maximization (EM) algorithm. We can now compute the posterior log odds ratio for each edge (i, j):

$$r_{ij} = \log \frac{\text{Beta}(f_{ij}, \alpha, 1)(1 - \pi_0)}{\text{Beta}(f_{ij}, 1, \beta)\pi_0} \tag{2}$$

If we predicted every edge with a positive log odds ratio, then the overall expected fraction of false positive edges would be given by the area under the $\text{Beta}(1, \beta)$ distribution, which is below the $\text{Beta}(\alpha, 1)$ distribution, see Figure 1 left. However, instead of choosing a log odds ratio cutoff of 0, we could also take any other one. In particular we can select a cutoff τ such that $\int_\tau^1 \text{Beta}(x, 1, \beta)dx = q$, where q is a prescribed false positive rate (here: 10%). This is equivalent to setting $w(ij) = r_{ij} - \tau$, for $1 \leq i, j \leq n$, which is an adjusted log odds ratio score.

4 Finding Score Optimal Network Structures

Finding the maximum weight acyclic subgraph is a well-known NP-hard problem. It is equivalent to the minimum weight feedback arc set problem, which was one of the 21 problems for which Karp showed hardness in his famous work [9]. We model the problem as an integer linear programming (ILP) formulation and solve it using branch-and-cut.

Given matrix W containing the weights of the complete digraph on n nodes, our goal is to find the maximum weight subgraph $D = (V, A^*)$ such that A^* contains no directed cycles. We therefore introduce binary variables $x \in \{0, 1\}^{|V \times V|}$ with the interpretation $x_a = 1$ for $a \in A^*$ and $x_a = 0$ otherwise. The ILP is as follows:

$$\begin{aligned} \max \quad & \sum_{a \in V \times V} w_a x_a && && (3) \\ \text{subject to} \quad & \sum_{a \in C} x_a \leq |C| - 1 && \text{for all directed cycles } C \\ & x_a \in \{0, 1\} && \text{for all } a \in V \times V \end{aligned}$$

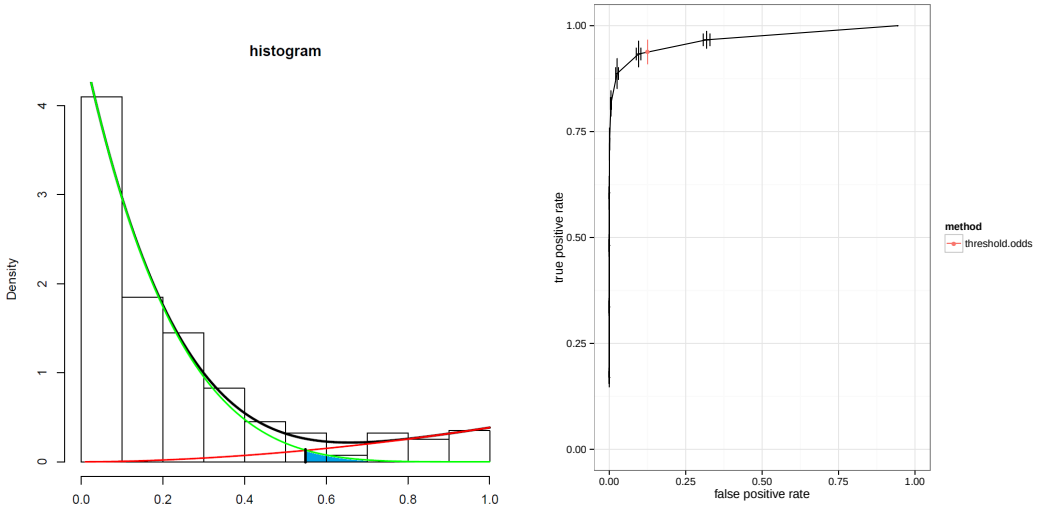


Figure 1 Left: Example of a fitted two component beta mixture model. Green = null distribution (edge does not exist in reality); red = alternative distribution (edge exists in reality); black = mixture. The shaded region indicates the expected false positive rate at a log odds ratio cutoff of 0. **Right:** ROC curve. True positive (TPR) and false positive rates (FPR) averaged over repeated data drawings for 20 networks. The plot shows the median TPR and FPR, respectively as well as the median absolute deviation (MAD) (depicted as error bars) for these 20 networks. Orange: point selected by the expected false discovery rate based method.

Since there is an exponential number of directed cycles in the complete digraph, we resort to a cutting plane approach to solve the linear programming relaxation of (3), which is the ILP without integrality constraints. This means, we initially leave out the directed cycle constraints and iteratively solve the following separation problem: Given a fractional solution $\bar{x}$ of an intermediate linear program, find a violated directed cycle inequality, that is, a directed cycle C with $\sum_{a \in C} \bar{x}_a > |C| - 1$ or state that no such violated inequality exists. This problem can be solved in polynomial time by defining $\bar{y} = 1 - \bar{x}$ and by computing shortest paths with respect to $\bar{y}$ between all pairs of nodes j and i. It is easy to see that a violated inequality is found if and only if the $\bar{y}$-weight of such a path plus $\bar{y}_{ij}$ is less than one. In this case, we add the inequality $\sum_{a \in C} x_a \leq |C| - 1$ and iterate. If no such cycle exists, we have solved the LP relaxation.

As we can solve the separation problem in polynomial time, we can also solve the full LP relaxation of (3) in polynomial time [6]. We use the LP bound within a branch-and-bound algorithm to obtain an optimal solution for the integer linear program and thus a maximum weight DAG in our input data. Subsequently, we compute the corresponding CPDAG with the algorithm of Chickering [3].

5 Results

5.1 Simulating KEGG Signaling Sub-Pathways

In order to test our approach we conducted simulation studies with sub-networks of KEGG signaling pathways with 10, 20, 40 and 60 nodes. We generated 20 directed, acyclic networks for each number of nodes. To obtain our ground truth networks we parsed XML files of all KEGG signaling pathways and converted them into graphs via the R-package KEGGgraph

[26]. Then we randomly picked one of these graphs and performed a random walk starting from a randomly selected core node. The random walk was stopped once the predefined number of distinct nodes had been visited. In case that the resulting sub-network between visited nodes contained cycles we repeated the whole procedure until a DAG structure had been found.

5.1.1 Network and Edge Frequency Sampling

We now simulated edge frequency matrices for the simulated KEGG DAGs by drawing for each existing edge from a Beta$(\alpha, 1)$ and for each non-existing edge from a Beta$(1, \beta)$ distribution for varying parameters α and β, see below. We made sure that $f_{ij} + f_{ji} \leq 1$ for each existing edge (i, j). This is necessary, because an undirected edge $\{i, j\}$ can never appear with a frequency of more than 100%. We repeated the simulation procedure five times for each network. For each frequency matrix we reconstructed a consensus Bayesian Network and calculated the corresponding CPDAG. We measured the performance in terms of balanced accuracy (BAC), i.e., average of sensitivity and specificity, by comparing the inferred CPDAG to the CPDAG of the original network. We conducted the comparison on the level of CPDAGs, because Bayesian Network structures can only be resolved up to these equivalence classes.

5.1.2 Effect of Edge Scoring Scheme

We exemplify the effect of our adjusted log odds scoring scheme compared to a varying cutoff applied to raw edge frequencies. This is done for a five times repeated simulation with $n = 20$ network nodes and $\alpha = 2$, $\beta = 10$. By varying the edge frequency threshold we obtain a ROC curve, which depicts true positive against false positive rates for network reconstruction (Figure 1 right). Our proposed edge scoring method corresponds to picking a point at the prescribed FPR of $\sim 10\%$ in the ROC curve. This demonstrates that our adjusted log odds scoring scheme can be used as an objective criterion to select a suitable edge frequency threshold.

5.1.3 Dependency on Number of Network Nodes

We went on to investigate the dependency of our proposed exact consensus Bayesian Network method (*exact*) on the number of network nodes for fixed $\alpha = 2$, $\beta = 10$. This was done in comparison to two other methods: (i) simple thresholding of the adjusted log odds ratio (*threshold*) at 0 and (ii) the consensus Bayesian Network algorithm proposed by Steele and Tucker (*CBN*) [21], which was also applied to adjusted log odds ratios here in order to have a fair comparison. Despite its name the CBN method does *not* guarantee to obtain a valid CPDAG structure, since it simply assigns each edge a direction based on a majority vote. The *threshold* method is equivalent to applying an edge frequency cutoff such that the expected FPR is 10%.

We observed a better network reconstruction performance of our *exact* method compared to both competing methods for all number of nodes. To assess the statistical significance of the observed differences we conducted one-sided paired Wilcoxon signed rank tests (with FDR multiple testing correction [1]). This indicated a highly significant result in all cases (Figure 2). Interestingly, significance levels increased with more network nodes. This may be explained by the larger space of CPDAG structures being consistent with the observed edge frequencies. The larger this space the more likely it is that a simple threshold based strategy (or variation thereof) fails. Hence, there is an increasing advantage of our *exact* approach.

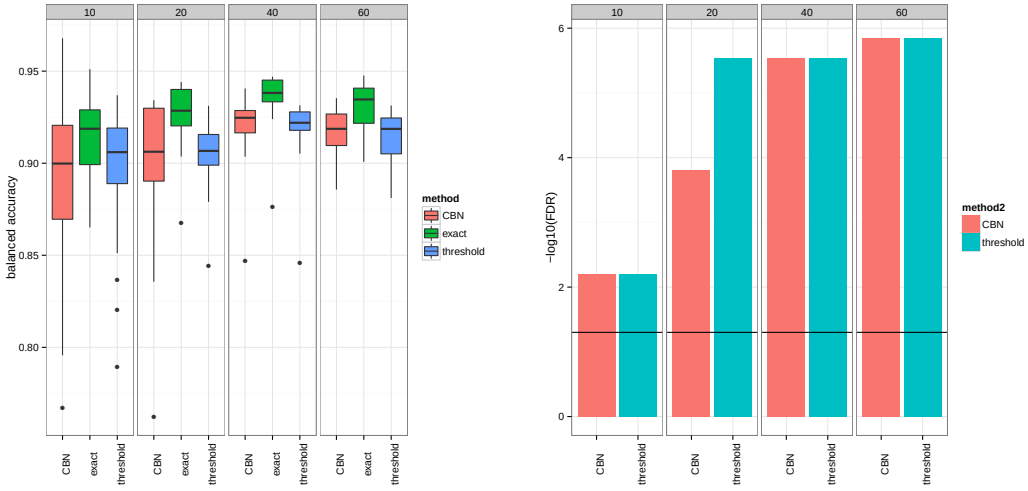


Figure 2 Left: Balanced accuracies (BACs) for network reconstruction with different consensus methods in dependency on number of network nodes. **Right:** Significance level ($-\log_{10}$ FDR) of *exact* versus *threshold* and *CBN*.

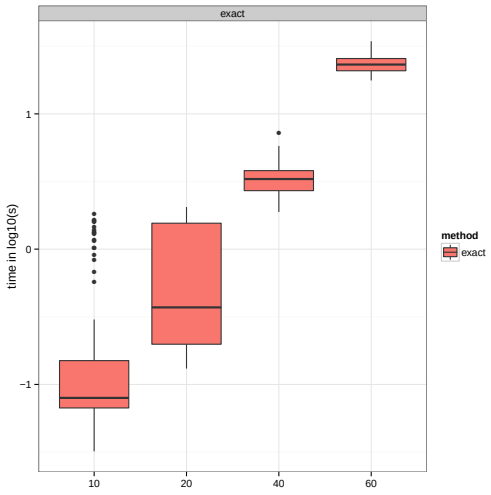


Figure 3 Empirical run time behavior for our *exact* method ($\log_{10} s$). Times were measured on an Intel Xeon 8 core machine with 96GB RAM.

Notably, even for networks with $n = 60$ nodes with our proposed method a consensus Bayesian Network that is provably optimal with respect to our scoring function could be found below 30 CPU s (Figure 3) on a standard desktop computer.

5.1.4 Dependency on Distribution Parameters

We repeated our above described simulation with different combinations of α and β for $n = 20$ nodes. This showed that in all but the extreme case ($\alpha = 2$, $\beta = 2$) a highly significant improvement of our method compared to *threshold* and *CBN* could be achieved (Figure 4). Please note that in the case ($\alpha = 2, \beta = 2$) the two beta distributions are very flat and

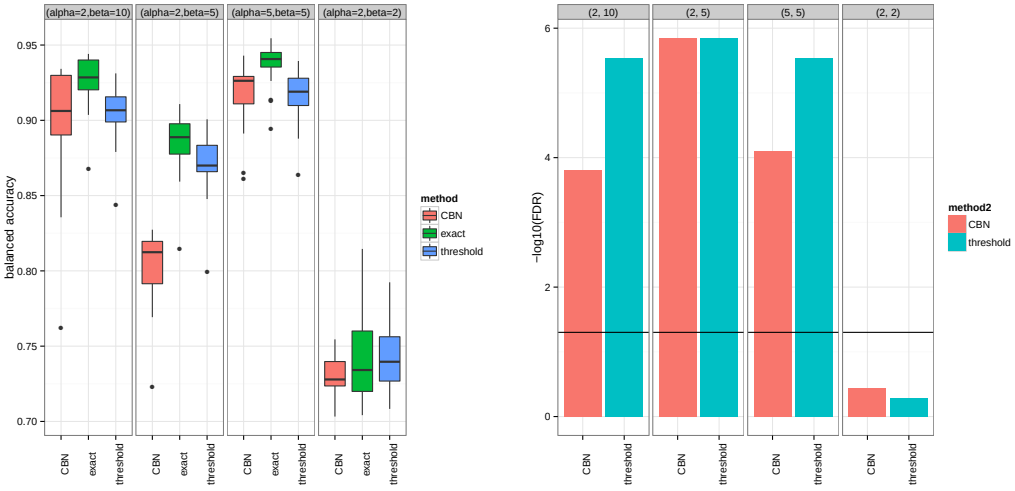


Figure 4 Left: Balanced accuracies (BACs) for network reconstruction with different consensus methods in dependency on different distribution parameter settings. **Right:** Significance level ($-\log_{10}$ FDR) of *exact* versus *threshold* and *CBN*.

extremely overlapping, making a reliable distinction of likely existing and non-existing edges highly difficult.

5.2 Yeast Heat-Shock Network

We applied all tested methods to a network of nine transcription factors (TFs) related to heat-shock response in yeast (Figure 5 left). The network was taken from [21]. We collected four microarray datasets (GSE3406, GSE3316, GSE40073, GSE40817) consisting of gene expression measurements after heat shock for varying conditions (GSE40817), time points (GSE3406, GSE3316, GSE40073) and strains or strain mutations (GSE3406, GSE40073). After k-NN imputation of missing values [23] quantile normalization was applied to each dataset [2]. We then used a quantile based discretization of each dataset. As variables in the Bayesian Network structure, which we wanted to learn from these datasets, we considered—besides the nine TFs—three additional variables, namely condition, time and strain. This was done, because these factors could potentially influence TF expression. The additional three variables were only allowed to have ingoing, but no outgoing edges. We then applied Bayesian Network learning using a greedy hill climber, as implemented in the R-package *bnlearn* [20]. This was done using a non-parametric bootstrap with 10,000 replicates. Accordingly we arrived at an edge frequency matrix, which we used to calculate a consensus network structure.

We compared the CPDAG of the consensus network on each of the four datasets against the CPDAG of the gold standard network from [21] and computed the balanced accuracies, see Figure 5 right. This demonstrates that our *exact* approach achieved the best performances on all four datasets. Interestingly, the performance of all methods showed a high variability across datasets. Whereas on dataset GSE3316 a balanced accuracy of 82% with our method could be achieved, on GSE3406 this was only 55%. The reason is probably the high number of different conditions, time and strain combinations coupled with a comparably low number of replicates in GSE3406.

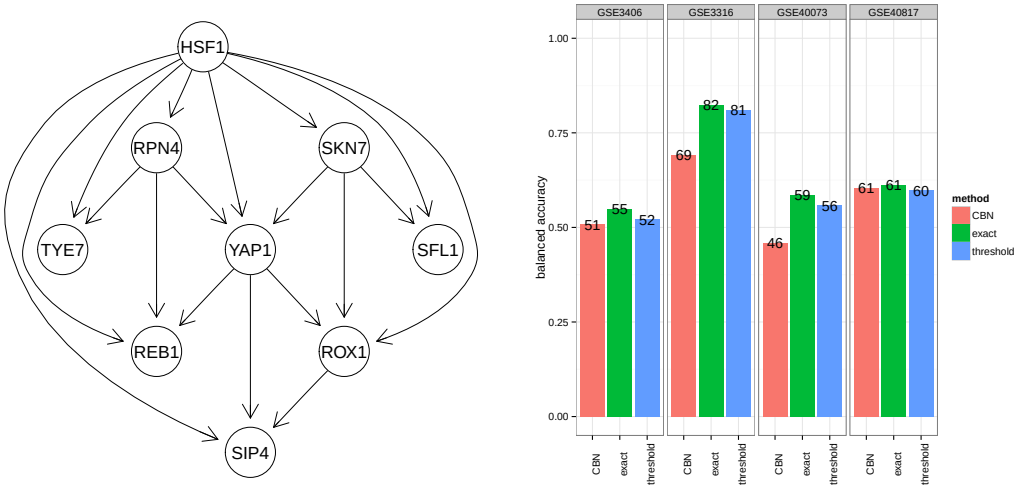


Figure 5 Left: Gold standard heat-shock network [21]. **Right:** Balanced accuracies for network reconstruction using a 10,000 times bootstrapped greedy hill climber and different consensus network approaches.

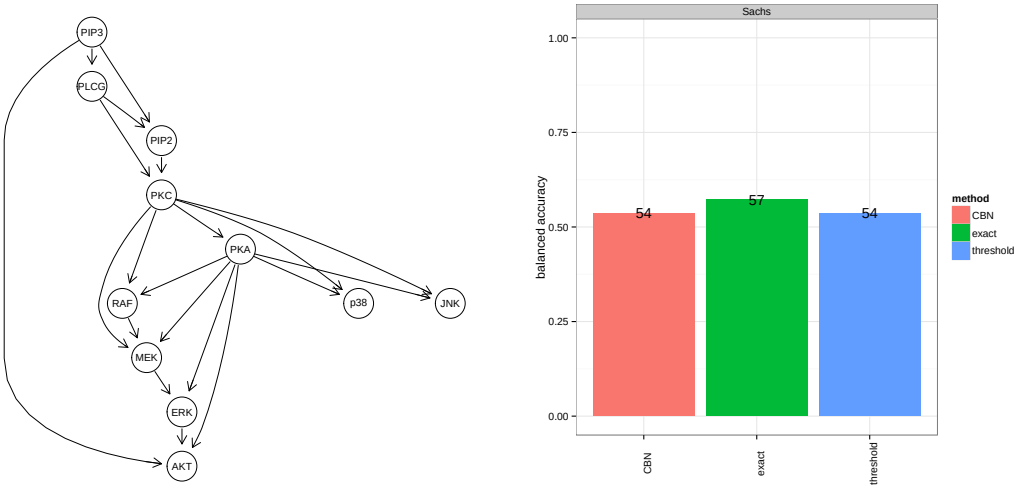


Figure 6 Left: Gold standard Raf signaling pathway [18] **Right:** Balanced accuracies for network reconstruction using a 10,000 times bootstrapped greedy hill climber and different consensus network approaches.

5.3 Yeast Raf Signaling Pathway

In addition we applied all tested methods to inferring parts of the yeast Raf signaling pathway based on the dataset by Sachs et al. [18], see Figure 6 left. The dataset contains measurements of 11 proteins under 14 different treatment conditions. The same technique for Bayesian Network inference as for the yeast heat-shock network with the same number of bootstraps was applied. Comparison of our *exact* consensus method against the *CBN* and *threshold* approaches yielded an improvement of 3% in terms of balanced accuracy (Figure 6 right).

6 Conclusions

In reverse engineering of biological networks edge frequency matrices can result from aggregating bootstrap or MCMC results, combining outputs of different inference algorithms [10] or integrating experimental datasets [8]. In order to interpret these meta results one usually has to come up with a way to compute a consensus network. So far most authors do this by applying a defined threshold to the edge frequencies, which is (i) highly subjective and (ii) leads to consensus models inconsistent with the model assumptions. Here we propose an automated threshold selection based on the expected false positive rate. This yields an adjusted log odds ratio as an edge-wise score. Based on this score we showed that computing provably score optimal consensus Bayesian Network structures amounts to solving the maximum weighted directed subgraph problem. We proposed a branch-and-cut algorithm based on an integer linear programming formulation for this task.

Our simulation studies as well as our results on two yeast networks showed that our new approach consistently improves network reconstruction accuracy. Our simulations showed that the expected gain increases with the number of network nodes. At this point we should emphasize that both of our tested yeast networks were comparably small. Higher significant results can thus be expected for larger datasets. Although the computation time of our proposed method scales exponentially with the number of network nodes we did not observe any practical limitation by this fact. Even with 60 nodes networks our algorithm took only a few seconds on a standard desktop computer. This should specifically be seen in relation to the high computation time needed for bootstrapping or sampling based network inference.

References

1 Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *J. Royal Statist. Soc., Series B*, 57:289 – 300, 1995.

2 B. M. Bolstad, Irizarry R. A., M. Astrand, and T. P. Speed. A comparison of normalization methods for high density oligonucleotide array data based on bias and variance. *Bioinformatics*, 19:185–193, 2003.

3 D. M. Chickering. Learning equivalence classes of Bayesian network structures. *Journal of Machine Learning Research*, 2:445–498, 2002.

4 A.C. Davison and D.V. Hinkley. *Bootstrap Methods and Their Application*. Cambridge University Press, Cambdrige, UK, 1997.

5 N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using Bayesian networks to analyze expression data. *J Comput Biol*, 7(3-4):601–620, 2000.

6 M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer, second corrected edition, 1993.

7 M. Grzegorczyk and D. Husmeier. Improvements in the reconstruction of time-varying gene regulatory networks: dynamic programming and regularization by information sharing among genes. *Bioinformatics*, 27(5):693–699, Mar 2011.

8 C. Huttenhower, K. T. Mutungu, N. Indik, W. Yang, M. Schroeder, J. J. Forman, O. G. Troyanskaya, and H. A. Coller. Detailing regulatory networks through large scale data integration. *Bioinformatics*, Oct 2009.

9 R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations, Proc. Sympos. IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y., New York: Plenum*, pages 85–103, 1972.

10 D. Marbach, J. C. Costello, R. Küffner, N. M. Vega, R. J. Prill, D. M. Camacho, K. R. Allison, The Dream Consortium, M. Kellis, J. J. Collins, and G. Stolovitzky. Wisdom of crowds for robust gene network inference. *Nature Methods*, 9(8), 2012.

11 A. A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, R. Dalla Favera, and A. Califano. ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics*, 7 Suppl 1:S7, 2006.

12 Florian Markowetz and Rainer Spang. Inferring cellular networks–a review. *BMC Bioinformatics*, 8 Suppl 6:S5, 2007.

13 N. Meinshausen and P. Bühlmann. High-dimensional graphs and variable selection with the Lasso. *Annals of Statistics*, 34(3):1436–1462, 2006.

14 S. Mukherjee and T. P. Speed. Network inference using informative priors. *Proceedings of the National Academy of Sciences*, 105(38):14313–14318, 2008.

15 R. Neapolitan. *Learning Bayesian Networks*. Pearson Prentice Hall, Upper Saddle River, NJ 07458, 2004.

16 D. Pe'er, A. Regev, G. Elidan, and N. Friedman. Inferring subnetworks from perturbed expression profiles. *Bioinformatics*, 17(Suppl 1):S215 – S224, 2001.

17 V. Pihur, S. Datta, and S. Datta. Reconstruction of genetic association networks from microarray data: a partial least squares approach. *Bioinformatics*, 24(4):561–568, Feb 2008.

18 K. Sachs, O. Perez, D. Pe'er, D. Lauffenburger, and G. Nolan. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 208(5721):523 – 529, 2005.

19 J. Schäfer and K. Strimmer. An empirical Bayes approach to inferring large-scale gene association networks. *Bioinformatics*, 21(6):754–764, Mar 2005.

20 M. Scutari. Learning Bayesian networks with the bnlearn R package. *Journal of Statistical Software*, 35(3):1–22, 2010.

21 E. Steele and A. Tucker. Consensus and meta-analysis regulatory networks for combining multiple microarray gene expression datasets. *J Biomed Inform*, 41(6):914–926, Dec 2008.

22 A. Tenenhaus, V. Guillemot, X. Gidrol, and V. Frouin. Gene association networks from microarray data using a regularized estimation of partial correlation based on PLS regression. *IEEE/ACM Trans Comput Biol Bioinform*, 7(2):251–262, 2010.

23 O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman. Missing value estimation methods for DNA microarrays. *Bioinformatics*, 17(6):520–525, Jun 2001.

24 Y. Wang, T. Joshi, X.-S. Zhang, D. Xu, and L. Chen. Inferring gene regulatory networks from multiple microarray datasets. *Bioinformatics*, 22(19):2413–2420, Oct 2006.

25 A. V. Werhli and D. Husmeier. Reconstructing gene regulatory networks with Bayesian networks by combining expression data with multiple sources of prior knowledge. *Stat Appl Genet Mol Biol*, 6:Article15, 2007.

26 S. Zhang, H. Chen, K. Liu, and Z. Sun. Inferring protein function by domain context similarities in protein-protein interaction networks. *BMC Bioinformatics*, 10:395, January 2009.

27 P. Zoppoli, S. Morganella, and M. Ceccarelli. TimeDelay-ARACNE: Reverse engineering of gene networks from time-course data by an information theoretic approach. *BMC Bioinformatics*, 11:154, 2010.

Efficient Interpretation of Tandem Mass Tags in Top-Down Proteomics

Anna Katharina Hildebrandt[1], Ernst Althaus[2], Hans-Peter Lenhof[1], Chien-Wen Hung[3], Andreas Tholey[3], and Andreas Hildebrandt[2]

1 **Center for Bioinformatics, Saarland University, 66041 Saarbrücken, Germany**
{anna.hildebrandt, lenhof} @ bioinf.uni-sb.de
2 **Johannes-Gutenberg-University Mainz, 55128 Mainz, Germany**
{andreas.hildebrandt, ernst.althaus} @ uni-mainz.de
3 **Institut für Experimentelle Medizin, Kiel University, 24105 Kiel, Germany**
{cw.hung, a.tholey} @ iem.uni-kiel.de

Abstract

Mass spectrometry is the major analytical tool for the identification and quantification of proteins in biological samples. In so-called top-down proteomics, separation and mass spectrometric analysis is performed at the level of intact proteins, without preparatory digestion steps. It has been shown that the tandem mass tag (TMT) labeling technology, which is often used for quantification based on digested proteins (bottom-up studies), can be applied in top-down proteomics as well. This, however, leads to a complex interpretation problem, where we need to annotate measured peaks with their respective generating protein, the number of charges, and the a priori unknown number of TMT-groups attached to this protein. In this work, we give an algorithm for the efficient enumeration of all valid annotations that fulfill available experimental constraints. Applying the algorithm to real-world data, we show that the annotation problem can indeed be efficiently solved. However, our experiments also demonstrate that reliable annotation in complex mixtures requires at least partial sequence information and high mass accuracy and resolution to go beyond the proof-of-concept stage.

1998 ACM Subject Classification J.3 Biology and genetics, G.1.6 Optimization

Keywords and phrases Mass spectrometry, TMT labeling, Top-down Proteomics

Digital Object Identifier 10.4230/OASIcs.GCB.2013.56

1 Introduction

The two major goals of proteomics are to identify and quantify proteins present in a given sample. Today, the most important analytical technique for this purpose is mass spectrometry (MS). Typical protein mixtures are highly complex: proteomes contain hundreds to several hundreds of thousands of proteins and protein forms. Often, many components of the sample will have similar masses, leading to overlapping signals in the spectrum that are hard to disentangle. Hence, the proteins usually have to be separated prior to MS with respect to a property that is not strongly correlated with the mass; a powerful technique for this purpose is liquid chromatography (LC).

The separated samples are then injected into a mass spectrometer, leading to a series of mass spectrometric runs, each applied on the sample content eluting from the chromatographic column at a specific retention time (RT) interval. In the mass spectrometer, the molecules are then ionized by the attachment of z protons (simultaneously delivering z positive charges), and accelerated in an electric field. Since the reaction of the peptide to the field depends on


© A. Hildebrandt, E. Althaus, H.-P. Lenhof, C.-W. Hung, A. Tholey, and A. Hildebrandt;
licensed under Creative Commons License CC-BY
German Conference on Bioinformatics 2013 (GCB'13).
Editors: T. Beißbarth, M. Kollmar, A. Leha, B. Morgenstern, A.-K. Schultz, S. Waack, E. Wingender; pp. 56–67

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

the ratio $\frac{m}{z}$ of peptide mass over its acquired charge, this quantity can now be measured. In realistic spectra, the informative parts of the mass spectrum (the 'signal' content) have to be identified and separated from parasitics, such as high-frequency noise or low-frequency baseline terms [5]. The resulting parts of the signal that are believed to arise from a molecule of interest are known as *peaks*. MS signals are usually formed by groups of peaks, representing the sum of all isotopes contained in the molecule. Consequently, if the spectrometer records a peak for a molecule with mass m_i at $\frac{m_i}{z}$, we can typically expect to find a peak also at $\frac{m_i+m_p}{z}$, where m_p is the mass of a single proton. If the spectral resolution allows to separate and identify at least two successive isotopic peaks, the molecular charge can be inferred.

With this information, we can now try to identify the molecular content of the sample. In proteome analysis, we are typically given a database with the amino acid sequences of potentially occurring proteins. In the general case, this database might be comprised of all proteins contained in, e.g., Uniprot [2] for the species of interest. The masses found in the experiment are then used as a query against the database. But unfortunately, the molecular mass alone is often not sufficiently characteristic for the molecule. Trivially, all sequence permutations of a given protein lead to the same mass and, hence, cannot be distinguished from this information alone. Even mutations of the sequence often lead to mass differences that are too small to be recognizable.

In *tandem mass spectrometry*, or *MS/MS*, this problem is solved by fragmentation of those parts of the sample that were identified to be potentially relevant. Since proteins preferably break at well-defined positions along their backbone, a large enough sample of the fragmentation space (induced, e.g., through molecular collisions) will lead to pairs of corresponding masses from which at least partial sequence information can be derived. Since this information characterizes the molecule much better MS/MS is typically required today for reliable identification.

For molecules as large as proteins, many steps of this procedure become very challenging. For instance, the MS/MS spectra of proteins are much harder to interpret than those of smaller molecules, and their isotopic patters are much more complex. Thus, proteins are often first digested into peptides with the help of specific proteases. The query database is then virtually digested: for a given protein sequence, the resulting peptides can be easily inferred from the protein's primary sequence, since the restriction enzymes cut the sequence at specific cleavage sites[1]. From the identification of the peptides found in the mass spectrum, we try to infer the proteins that contained those peptides. To this end, different scoring schemes [3, 10] based on different statistical models can be used to generate p-values for the occurrence of the proteins in the database.

Such a setup, with its digestion of proteins into peptides, which are then identified and used as evidence for their containing proteins, is known as *shotgun-* or *bottom-up* proteomics. The major advantage of this technology is that peptides are much simpler to separate by LC and can be measured with higher mass accuracy and sensitivity in mass spectrometry. The MS/MS spectra of peptides are easier to interpret, even though in many cases a large percentage of them cannot be annotated successfully. Thus, even though bottom-up proteomics is a very sensitive method, many of the peptides are missed in practice. Also, some of the digestion peptides for a given protein might not ionize sufficiently well to allow their detection, or they might be too small or too large for the given experimental setup. This often leads to non-optimal coverage of the protein sequence by peptides in the

[1] Since the protease might miss potential cuts, it is customary to generate all peptide combinations up to a fixed number of missed cleavage sites.

digest: even though the protein has been identified through several of its digestion peptides, large parts of its sequence might not be represented in the results. In summary, the increased complexity of the samples – out of a single protein, multiple peptides are generated – imposes challenges even though each individual component is more easily identified.

These drawbacks are avoided by so-called *top-down* proteomics studies, where no digestion of the protein into peptides is performed. Instead, the sample is separated via LC at the level of intact proteins, at which also the MS experiment is performed. Hence, the information belonging to a single protein is not distributed over many peptides, which allows direct distinction of protein isoforms or of post-translationally modified forms from their non-modified counterparts. However, the separation of intact proteins is not as straightforward as that of peptides, and the detection limits of proteins in MS are strongly elevated with increasing protein size. The situation is complicated further, as from the MS/MS spectra of intact proteins, only limited information about N- and C-terminal parts of the protein sequence can be derived, hampering unambiguous identification. Nevertheless, the clear advantages for downstream analysis due to an automatically full sequence coverage of the proteins make top-down proteomics an increasingly popular alternative to bottom-up studies.

An even greater challenge than the identification of proteins is their accurate quantification. Several strategies for relative as well as for absolute quantification have been proposed [1]. In addition to label-free approaches, methods for quantification in MS mode using stable isotope labeling quantification have been developed, where the molecules of interest are modified with chemical groups that allow for an accurate quantification. Another approach is the use of isobaric labeling strategies, where the samples are labeled with reagents which consist of three major groups: (i) a reactive group that allows covalent attachment of the reagent to the peptide, in particular the N-termini and epsilon-amino groups of Lysine residues; (ii) a reporter group and (iii) a balancer group. These reagents can now be formed in four or eight (iTRAQ [4]) or six (TMT [13]) different flavors: for TMT, for instance, a reporter group in the first flavor has a mass of 126 Da, the corresponding balancer of 103 Da, yielding a total mass of the label of 229 Da. The reporter of the second flavor has a mass of 127 Da, the balancer of 102 Da, again yielding the same total mass of 229 Da. As the different flavors share the same molecular properties and only differ in the isotope composition, they also appear at the same retention times. Hence, after labeling of different biological samples with these reagents (one flavor per sample), the labeled samples can be combined, treated by LC, and analyzed by MS. In MS mode, equivalent peptides from different biological samples will have the same m/z-values, as the reagents were isobaric. But upon fragmentation of the peptides in MS/MS experiments, the reporter groups are liberated, yielding signals of the corresponding reporter ions at 126, 127, ..., 131 Da. The intensities of these reporter ion signals deliver a direct readout of the relative quantities of the peptides in the different biological samples. Isobaric labeling strategies have been originally developed for quantification studies in bottom-up approaches, i.e., at the level of peptides. But a recent pilot study [6] has shown that tandem mass tag labelling can be applied in a top-down setting as well. At this stage, the method is still restricted to the quantification and simultaneous MS/MS-based identification of relatively small proteins up to ≈ 35 kDa. A severe bottleneck is the interpretation of MS- and MS/MS-spectra of such isobarically labeled intact proteins, which will be the focus of this work.

Three different effects render this a challenging task: (i) The degree of labeling may differ and is unknown for an a-priori unknown protein. This is caused by incomplete labeling or unwanted non-specific labeling of residues in proteins. Consequently, a theoretical protein with 15 Lysine residues can lead to a mixture containing protein species with 12 to about

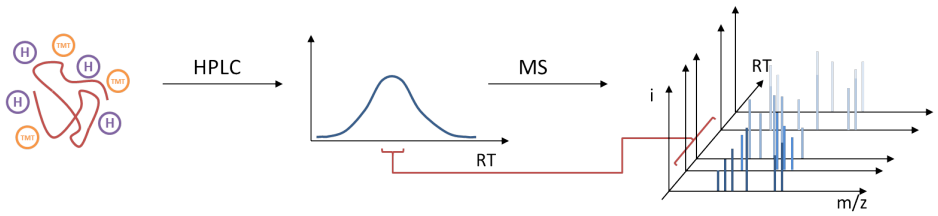


Figure 1 The experimental setup of top-down tandem mass tag proteomics [6]

18 isobaric labels attached. (ii) Each of the proteins features a complex isotopic pattern as outlined above. (iii) In the ionization process (electrospray) used for the analyis of intact proteins, species with different charge states are formed by attachment of different numbers of protons. Thus, a single theoretical protein with a mass of 15 kDa may have 10, 11, ..., 20 attached protons, leading to peak groups at $\frac{15}{10}$ kDa, $\frac{15}{11}$ kDa, ..., $\frac{15}{20}$ kDa. This number of attached protons cannot be predicted, in particular not for unknown proteins in unknown proteomes. But it can be deduced from the mass differences of the isotopes in the peak group according to the relation given above.

These factors lead to a difficult interpretation problem, where we want to analyze for each peak which proteins could have generated it, and compute the corresponding number of charges[2] and TMT marker groups. In this work, we will present an efficient algorithm for this annotation task and will apply it to real-world experimental data. Using our algorithm, we will further demonstrate that the information contained in the experiment is insufficiently specific, resulting in false-positive annotations that match the given masses. We will then discuss how to integrate further experimental insight into the algorithm at moderate computational cost that can weed out many false positives.

2 Methods

2.1 Experimental setup

For generation of MS and MS/MS datasets, a mixture of six known model proteins was labeled with the TMT-6-plex reagent. It has to be noted that several of the six model proteins contained impurities, thus finally ten different proteins were present in the test mixture (see Tab. 1). The proteins were separated via ion paring reversed phase chromatography using monolithic columns and analysed in a Thermo Orbitrap Velos mass spectrometer equipped with ETD. The mass spectrometer was operated in the data-dependent mode to switch automatically between Full-MS (scan 1), HCD-MS^2 (scan 2), and ETD-MS^2 (scan 3). After a Full-MS scan acquired in the ion trap MS, the most abundant protein ion (top 1) was selected for an HCD-MS^2 scan and an ETD-MS^2 scan. Full details of the experimental procedure were described in [6]. A schematic sketch of the approach is shown in Fig. 1.

[2] Please note that, in principle, other ionization types than addition of protons can occur, and can indeed be handled by our method. For reasons of simplicity, these will not be considered in the current manuscript.

Table 1 The ten protein mix and the results of the manual annotation [6]. Note that in the manual annotation, all Lysine residues were assumed to carry a TMT group. The mix consists of six known model proteins labeled with the TMT-6-plex reagent and four impurities.
* : C-terminal seq. of ovalbumin - ASVSEEFRADHPFLFCIKHIATNAVLFFGRCVSP
#: undefinable in manual annotation due to the poor MS quality derived from post-translational modifications (e.g. phosphorylation, glycosylation etc)

Protein Name	Theo. MW (kDa)	No. Of Cys/Lys	Theo. MW with TMT (kDa)	Observed m/z	Charge State	Calc. MW (Da)
Cytochrome C	12.3	2C/19K	16.8	1045.7	16	16714.3
(Equine)				1115.2	15	16713.2
				1194.8	14	16713.0
Myoglobin	16.9	0C/19K	21.5	1436.6	15	21533.9
(Equine)				1539.1	14	21533.3
				1657.4	13	21533.1
Carbonic Anhydrase	29.1	0C/18K	33.2	949.7	35	33205.6
(Bovine)				977.7	34	33208.9
				1007.2	33	33205.3
Carb. Anhydrase Impurity 1	8.5	0C/7K	10.2	1017.9	10	10168.9
Ubiquitin				1130.8	9	10168.1
partial seq.				1272.0	8	10168.2
Carb. Anhydrase Impurity 2	15.5	3C/10K	17.9	1278.4	14	17882.8
Superoxide Dismutase				1376.7	13	17883.7
				1491.2	12	17882.2
Ovalbumin *(Gallus)*	42.8	6C/20K	47.7	UD#		
Ovalbumin Impurity 1	20.1	18C/13K	24.3	UD#		
Ovomucoid						
Ovalbumin Impurity 2	3.8	2C/1K	4.3	877.5	5	4382.5
C-terminal ovalbumin*				1096.5	4	4381.9
				1461.9	3	4382.7
BSA *(Bovine)*	66.6	34C/60K	82.3	UD#		
Apo-transferrin *(Bovine)*	77.7	38C/64K	94.5	UD#		

2.2 Formal problem formulation: The TMT annotation problem

In this section, we will introduce the formal definition of the annotation problem posed by top-down TMT labelling. Informally, we want to query a database of known protein masses (e.g., the whole proteome of the organisms contained in the sample) against the peaks detected in the experiment. To this end, we want to decide for every protein in the database and for every peak, whether this protein could have led to the peak's observed mass-over-charge ratio through a feasible combination of base protein mass, TMT attachments, and charges (protons). To formally formulate the problem, we first need a few definitions.

Let m_T denote the mass of the TMT marker group ($m_T \approx 229.162932$ Da), and m_p the mass of a single proton. By $DB := \{m_i | i = 1, \dots n_{DB}\}$, we denote the database we want to query, where m_i is the monoisotopic mass of the i-th protein in DB. We assume that the spectrum has been pre-processed to yield a set of mass spectrometric peaks $\mathcal{S} := \{p_j | j = 1, \dots n_{\mathcal{S}}\}$.

Let us further assume that one of the populations in the sample was given by the i-th protein, to which β_i TMT-groups and α_i excess protons have been attached, with $\beta_i, \alpha_i \in \mathbb{N}^+$. This protein will have a charge of $z = \alpha_i$, measured in units of elementary charge, and a total mass-over-charge ratio of

$$m_{z,i}\left(\alpha_i, \beta_i\right) := \frac{m_i + \alpha_i m_p + \beta_i m_T}{\alpha_i}$$

This relation between protein, ionization state, and TMT assignment is not unique: one protein species may acquire different ionization states as well as different numbers of attached TMT groups. However, in practice, not all values of α_i and β_i are possible: the amount of charges and of TMT groups that a given protein can acquire falls within limited ranges, i.e.,

$$\alpha_i \in \{\alpha_i^{min}, \dots \alpha_i^{max}\} \quad \text{and} \quad \beta_i \in \{\beta_i^{min}, \dots \beta_i^{max}\}$$

In the following we describe how to efficiently reduce the parameter search space. Since the TMT markers attach to Lysine-residues, it is natural to choose $\beta_i^{min}, \beta_i^{max}$ accordingly, and hence limit the number of TMT-attachments to a 2x window, i.e.:

$$\beta_i \in \{\max(0, \#\text{LYS} - x), \dots \#\text{LYS} + x\} \text{ for } x \in \mathbb{N}^+$$

We now want to annotate all *measured* peaks $p_j \in \mathcal{S}$ with all *predicted* peaks $\hat{p_i}$ due to feasible protein/TMT/proton combinations within a given accuracy threshold. As the accuracy of mass spectra is typically dependent on the mass-over-charge ratio, it is customary to use relative measures of error. Thus, for every measured peak p_j we want to determine all feasible predicted peaks $\hat{p_i}$ with

$$\frac{|\hat{p_i} - p_j|}{p_j} \leq \epsilon$$

To solve this problem, we will compute for every protein m_i in DB and for every peak p_j all feasible values of $\alpha_{i,j}$ and $\beta_{i,j}$, such that the assumption of protein i with $\alpha_{i,j}$ attached protons and $\beta_{i,j}$ attached TMT markers explains peak p_j within the given relative accuracy threshold ϵ, which gives the following combinatorial problem:

$$\begin{aligned}
&\forall i \in \{1, \dots, n_{DB}\}: \\
&\quad \forall j \in \{1, \dots, n_S\}: \\
&\qquad \text{find } \alpha_{i,j} \in \{\alpha_{i,j}^{min}, \dots, \alpha_{i,j}^{max}\}, \\
&\qquad\quad \beta_{i,j} \in \{\max(0, \#\text{LYS} - x), \dots \#\text{LYS} + x\}: \quad |m_{z,i}(\alpha_{i,j}, \beta_{i,j}) - p_j| \quad \leq \quad \epsilon \cdot p_j
\end{aligned}$$

Obviously, not all combinations of $\alpha_{i,j}, \beta_{i,j}$ will lead to a valid annotation. In the following we will determine reasonable boundaries for $\alpha_{i,j}$, given $\beta_{i,j}$:

$$\begin{aligned}
& & |m_{z,i}(\alpha_{i,j}, \beta_{i,j}) - p_j| & \leq & \epsilon \cdot p_j \\
\Leftrightarrow & & \left|\frac{m_i + \alpha_{i,j} m_p + \beta_{i,j} m_T}{\alpha_{i,j}} - p_j\right| & \leq & \epsilon \cdot p_j \\
\overset{\alpha_{i,j} > 0}{\Longleftrightarrow} & & |m_i + \alpha_{i,j} m_p + \beta_{i,j} m_T - \alpha_{i,j} p_j| & \leq & \alpha_{i,j} \cdot \epsilon \cdot p_j
\end{aligned}$$

In practice, the allowed window $2x$ for the parameter $\beta_{i,j}$ is quite small (in our experiments, we used $x = 3$). We can thus easily test all allowed values for $\beta_{i,j}$. For any such fixed but

arbitrary $\beta_{i,j}$, we find

$$\begin{array}{lrcrl} & |m_i + \beta_{i,j} m_T + \alpha_{i,j}(m_p - p_j)| & \leq & \alpha_{i,j} \cdot \epsilon \cdot p_j & \\ \Leftrightarrow & (m_i + \beta_{i,j} m_T + \alpha_{i,j}(m_p - p_j)) & \leq & \alpha_{i,j} \cdot \epsilon \cdot p_j & \wedge \\ & (m_i + \beta_{i,j} m_T + \alpha_{i,j}(m_p - p_j)) & \geq & -\alpha_{i,j} \cdot \epsilon \cdot p_j & \\ \Leftrightarrow & \alpha_{i,j}(m_p - p_j - \epsilon \cdot p_j) & \leq & -(m_i + \beta_{i,j} m_T) & \wedge \\ & \alpha_{i,j}(m_p - p_j + \epsilon \cdot p_j) & \geq & -(m_i + \beta_{i,j} m_T) & \end{array}$$

Remembering that m_p denotes the mass of a single proton, and p_j a mass-to-charge ratio in a feasible range for proteins, we see that $m_p \ll p_j$. In addition, the accuracy threshold ϵ is small in practice – on the order of tens or hundreds of parts per million (ppm) – so that we also find $\epsilon \cdot p_j \ll p_j$. Indeed, we can safely assume that $m_p + \epsilon \cdot p_j \ll p_j$ and, hence,

$$(m_p - p_j - \epsilon \cdot p_j) < 0 \quad \wedge \quad (m_p - p_j + \epsilon \cdot p_j) < 0$$

We thus find:

$$\alpha_{i,j} \geq -\frac{m_i + \beta_{i,j} m_T}{m_p - p_j - \epsilon \cdot p_j} \qquad \wedge \qquad \alpha_{i,j} < -\frac{m_i + \beta_{i,j} m_T}{m_p - p_j + \epsilon \cdot p_j}$$

We can thus restrict $\alpha_{i,j}$ as a function of the fixed but arbitrary $\beta_{i,j}$. For simplicity of notation, we introduce $a_{i,j} := m_i + \beta_{i,j} m_T > 0$ and $b_j := p_j - m_p > 0$, which yields

$$\begin{array}{lrcrcrl} & \alpha_{i,j} & \geq & \frac{-a_{i,j}}{-b_j - \epsilon \cdot p_j} & = & \frac{a_{i,j}}{b_j + \epsilon \cdot p_j} & > \quad 0 \\ \wedge & \alpha_{i,j} & < & \frac{-a_{i,j}}{\epsilon \cdot p_j - b_j} & = & \frac{a_{i,j}}{b_j - \epsilon \cdot p_j} & > \quad 0 \end{array}$$

so that we finally obtain

$$\alpha_{i,j}^{min} := \left\lceil \frac{a_{i,j}}{b_j + \epsilon \cdot p_j} \right\rceil, \qquad \alpha_{i,j}^{max} := \left\lfloor \frac{a_{i,j}}{b_j - \epsilon \cdot p_j} \right\rfloor$$

Thus, we only have to consider $\alpha_{i,j} \in \{\alpha_{i,j}^{min}, \ldots, \alpha_{i,j}^{max}\}$. Each of these values will lead to a valid explanation of the peak, i.e., the triple $< m_i, \alpha_{i,j}, \beta_{i,j} >$ yields an m/z-value that deviates from p_j by less than ϵ. We can thus trivially enumerate all valid annotations.

2.3 Results of the procedure

We implemented the scheme described above in OpenMS [9]. To test the correctness, efficiency, and utility of our approach, we applied our implementation to the experimental data set used in [6]. The parameters used in our study were chosen to conform with experience gathered by our experimental partners. For the limits on the number of attachable TMT groups, we used

$$\beta_{i,j}^{min} := (\#LYS + 1) - 2, \quad \beta_{i,j}^{max} := (\#LYS + 1) + 3$$

where the +1 accounts for attachment at the N-terminus. The accuracy threshold was varied to study its influence on the number of generated solutions. Typical values are in the order of tens to hundreds parts per million, i.e., $\epsilon \approx 10^{-5}$ to 10^{-4}. While the minimal and maximal number of charges that can explain a given peak have been computed as a function of $\beta_{i,j}$ above, we also enforce global limits on them. Obviously, we require $\alpha_{i,j} > 0$. In accordance with experimental insight, we also introduce an upper boundary: $\alpha_{i,j} < 40$.

Table 2 Results on the 10-protein mix from [6]. ϵ is given in ppm.

ϵ	annotated peaks	valid annotations	identified proteins
10	510	546	8
20	992	1092	8
100	5090	7665	8
300	12614	30408	8

In [6], the proteins were first subjected to certain chemical modifications, as is commonly the case in proteomics. To account for these modifications in our study, we performed a virtual carboxyaminomethylation. This modification changes the mass of Cysteine-residues by $\delta_m^{\text{carb}} \approx +57.0214$ Da. We assume this modification to be fully effective ("fixed"), and hence arrive at a mass difference of $\#\text{CYS} \cdot \delta_m^{\text{carb}}$ for every protein in the reference database. In addition, every initiator Methionine might be removed, with an acetylation of the new N-terminus, yielding a mass difference of $\delta_m^{\text{acet}} \approx -89.0299$ Da. To account for this variable modification, we add a modified and an unmodified variant for each protein to the reference database.

With these preparations, we first attempted to recreate the results described in [6]. To this end, the reference database was set to contain the 10 proteins known to be present in the sample (for details, see [6]). The spectrum was processed using the OpenMS Wavelet-based peak picker [8, 7, 12], leading to 16,042 peaks. The runtime of our algorithm did not change significantly with varying values of ϵ, and was ≈ 0.55 seconds in each case. Tab. 2 describes the results: 8 out of 10 proteins were consistently found, but many more valid annotations were found than previously expected.

The method as described above uses total mass as a descriptor for a protein, which is known to be insufficiently specific in the general case. Still, the large number of valid annotations came as a surprise for two reasons: first, since the query database is small, only very few proteins were expected to fit a given peak (for larger data bases, partial sequence information derived from MS/MS experiments can help in filtering false positive protein identifications). Second, as a result of the time-consuming manual annotation process used previously, it was suspected that only a small number of TMT/charge-combinations of a given protein would explain any given peak, if it could be explained at all.

In practice, though, the mass accuracy achieved in the ion trap MS is insufficient to rule out many of the possible explanations. For low mass accuracy, several of the peaks of our spectrum could be reasonably explained by multiple variants. For the determination of intact protein masses, the ion trap was used [6]; here the mass accuracies achievable for intact proteins are at best greater than 10 ppm for very small proteins and can reach the Da range for larger ones, yielding very large ppm values[3]. Accordingly, we also performed our experiments with large mass deviations. However, for deviations as large as $\epsilon = 100$ ppm, we could often not even distinguish between acetylated and non-acetylated versions of the protein. We thus decided to adapt the algorithm for improved specificity.

[3] These large mass deviations also prevent to distinguish individual peaks in the isotopic pattern and, hence, a simple determination of the charge. This also prevents application of most feature detection procedures used in bottom-up proteomics.

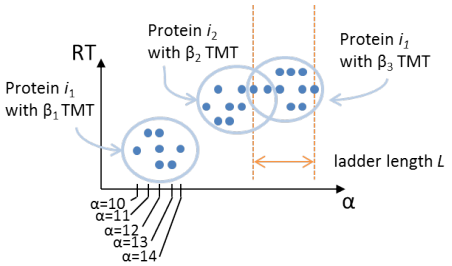


Figure 2 Potential charge ladders of annotated neighbouring peaks (blue nodes). These can be used for filtering significant hits.

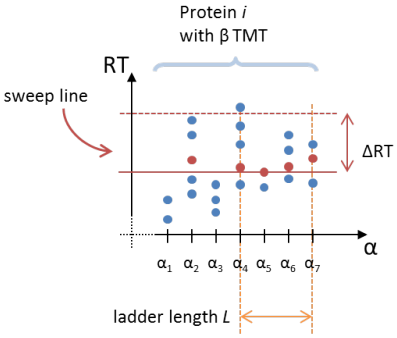


Figure 3 Sweep line (red nodes) traversing all charge variants (i, α_k, β) of protein i having β TMT-assignments.

2.4 Refined problem formulation

As by design every annotation computed by our algorithm is valid, i.e., falls within a chemically reasonable range of TMT- and proton-number, ruling out explanations will require either improved mass accuracy to allow for reducing the threshold ϵ, or the use of additional information. One restriction that can be obtained without any further experimental effort stems from what we call the *charge-ladder assumption*.

Assume that the sample contains a species of protein i with $\alpha_{i,j}$ attached charges and $\beta_{i,j}$ TMT markers. According to experimental experience, it is then very likely that the sample contains the same protein with the same number of TMT markers, but with a charge that is smaller or greater by one. We thus call two annotations $< m_i, \alpha_{i,j}, \beta_{i,j} >$ and $< m_i, \alpha_{i,k}, \beta_{i,k} >$ *neighbours*, iff $|\alpha_{i,j} - \alpha_{i,k}| = 1$. Consequently, we should expect that if we can explain a peak with a given annotation, we can also explain other peaks in the spectrum by its neighbours. Indeed, a manual annotator would reject an explanation if it would not be supported by a gap-less chain of neighbouring annotations. Detection of such chains, however, is complicated by the fact that the corresponding peaks may not necessarily co-elute and, hence, occur at different retention times. But since the physico-chemical change of the neighbouring protein species is small, we expect the neighbouring peaks to be located within a certain retention time interval.

To assign an annotation with a high probability of occurrence, we thus now demand the existence of a *charge ladder* (c.f. Fig. 2) of a minimal length, i.e., a chain of neighbouring explanations that all occur in an RT-window of finite, specified length ΔRT.

Considering the annotations individually as in the last section, we often find many valid explanations for any given peak. However, only one of these annotations will typically be the "true" solution, while the others occur by chance. The idea behind our refined procedure is then that false positive explanations will have a significantly smaller probability of supporting long charge ladders than the true positives.

The assignment of peak annotations into charge ladders can be achieved efficiently using a sweep algorithm [11]: for every protein i in the database, we iterate over all TMT-assignments $\beta_{i,j}$ that lead to a valid explanation. For each of those $(i, \beta_{i,j})$-pairs, we then determine a set of potential charge ladders, i.e., maximal sets of neighbouring assignments $\{(i, \alpha_{i,j} + k, \beta_{i,j}) | k = 0, \ldots, K\}$, disregarding the difference of their retention times (this can be done efficiently by sorting the set with respect to $\alpha_{i,j}$). Please note that, usually, many of the $(i, \alpha_{i,j} + k, \beta_{i,j})$ values will occur at different retention times.

Table 3 Results on the 10-protein mix from [6] when including the charge ladder filter with varying accuracy threshold ϵ, RT window ΔRT, and ladder length L.

ϵ	ΔRT	L	annotated peaks	valid annotations	found ladders	found proteins
10	10	2/3/4	52/10/0	54/10/0	24/3/0	6/3/0
10	20	2/3/4	71/15/0	77/15/0	30/4/0	7/3/0
10	50	2/3/4	83/18/0	91/18/0	31/5/0	7/3/0
20	10	2/3/4	253/83/9	269/87/9	117/28/2	7/5/1
20	20	2/3/4	343/149/44	371/161/56	151/52/10	7/6/2
20	50	2/3/4	408/186/84	444/206/100	151/50/17	7/7/3
100	10	2/3/4	3788/2823/2095	5657/4207/3128	1715/1034/640	8/8/7
100	20	2/3/4	4278/3674/3121	6405/5511/4701	1396/1033/774	8/8/7
100	50	2/3/4	4426/3955/3459	6621/5914/5181	792/571/434	8/8/7

In the next step, we consider each potential ladder individually to extract valid ladders within the given RT interval. For each $\alpha_{i,j}$-value in the current potential ladder, we store the RT values of all peaks that were explained by this annotation in a sorted list. Our sweep line starts at the annotation with lowest[4] RT value, regardless of its charge, and will progress in order of increasing RT value. In each step, we then try to extend valid ladders to the left and right, starting from the annotation currently touched by the sweep line, which will form the lower boundary of the RT interval ΔRT. The ladder can be iteratively extended if a neighbouring annotation within this RT interval exists. Since annotations are sorted with respect to RT, and since the sweep line always rests at the annotation with currently lowest RT value, it suffices to check the lowest-RT remaining (i.e., above the sweep line) annotation for each charge state. Thus, in each step, only one comparison per charge state is necessary. If a consecutive list of a user-defined minimal length L has been detected, all annotations in that list are marked as part of a charge ladder. Finally, the current annotation is removed from the list and the sweep line progresses to the next annotation. A snapshot at one step of the algorithm is depicted in Fig. 3.

For pre-sorted input, this algorithm obviously requires $\mathcal{O}(n \cdot L)$ operations, where n is the total number of annotations, since every annotation can be part of only one potential ladder and is touched at most once by the sweep line, and since every check requires up to L comparisons. Combined with the sorting steps, we arrive at a total runtime of $\mathcal{O}(n \cdot L + n \log(n))$.

3 Results of the refined procedure

We implemented the refined algorithm as a TOPP [9] tool, which is fully integrated into the OpenMS framework. The results of the refined procedure on the ten-protein mix are shown in Tab. 3. These demonstrate that charge-ladders indeed provide a strong filter: with increasing ladder length, the amount of remaining annotations drops particularly strongly for presumed low mass accuracy, and the amount of explanations per annotated peak becomes significantly smaller.

[4] In the degenerate case, where the minimum is not unique, annotations are visited from lowest to highest charge state.

In general, however, the problem setting is still highly ambiguous, despite the charge-ladder constraint. To further demonstrate this fact, we again applied our procedure on the data set of [6], but now with a much larger reference database to query: all proteins contained in UniProtKB/Swiss-Prot [2] that fall into a similar mass range as the ones known to be contained in the sample. The resulting data set consists of $3,990,159$ proteins, including the 10 true positives. Treatment of variable modifications doubles the database size to $7,980,318$ proteins. The computational efficiency of our method is demonstrated by the fact that the query of $16,042$ peaks against this huge database terminated in 488 minutes on a single core of a standard desktop PC (please note that the method can be trivially parallelized with nearly linear speed-up by splitting query database). However, the specificity on this data set is very low. Of the $7,980,318$ proteins in the database, $6,566,123$ have not only been annotated successfully, but also as part of stable charge ladders of length at least 3 in an RT-window of 20 seconds and with a maximal mass deviation of 20 ppm. If we choose more restrictive parameter values, some of the false-positive identifications indeed vanish (with $\epsilon = 10$ ppm the number drops to $1,112,502$), but so do the true positive ones.

4 Conclusion and Outlook

Top-down proteomics is a promising alternative to the popular shotgun approaches that are commonly applied. Its deficiencies in sensitivity are often made up for by avoiding coverage problems as they are common in bottom-up settings. Unfortunately, many of the established solutions for identification and quantification in the bottom-up domain cannot be simply transferred to the top-down case. This work was concerned with one such solution – the use of TMT-labelling for quantification purposes. In [6], the analytical background to apply TMT-markers to intact proteins was established, but resulted in a challenging annotation problem. In the pilot study, annotation was performed in a time-consuming manual fashion which can neither be performed on a high-throughput basis, nor generalized to the case of large reference databases.

Here, we have shown how the same annotation problem can be solved efficiently on a computer. Application of our method on the original data set has shown that the manual annotation was valid, but was only one of a variety of equally probable explanations. Only prior knowledge about the outcome allowed the annotator to select the "true" solution intuitively, which he validated by using further experimental constraints. We then proceeded to derive a refined algorithm for improved specificity without the need for further experimental effort. Through the use of so-called charge-ladders, we can exclude at least some of the noise present in the annotations, i.e., explanations that were only valid by chance without further supporting information in the spectrum. The resulting annotation problem seems to become significantly more complex, but can be solved efficiently using the sweep-paradigm.

Application to the data set has shown that the method is indeed efficient enough to be applied to real-world data sets. But whether it is specific enough to be applied routinely is still an open question. If at least partial sequence information is known from MS/MS experiments, the use of charge-ladders suppresses most false positive TMT/charge-variants. If ambiguity persists, all valid annotations will be returned to the user, who will then have to decide whether one of them can be trusted. Without such constraints, the amount of false-positives is clearly too large to be used routinely, even for moderate sample complexity. To go beyond the proof-of-concept stage in the general case will thus require at least an improved mass accuracy, but possibly also the use of other kinds of experimental constraints. This will be the focus of our future work.

Funding AH acknowledges financial support from the Intel Visual Computing Institute (IVCI) of Saarland University and the 'Schwerpunkt Rechnergestützte Forschungsmethoden in den Naturwissenschaften' of Johannes-Gutenberg University Mainz, AH and HPL financial support from DFG (BIZ4:1-4). AT received funding from the DFG Cluster of Excellence 'Inflammation at Interfaces'.

References

1 Marcus Bantscheff, Simone Lemeer, Mikhail M. Savitski, and Bernhard Kuster. Quantitative mass spectrometry in proteomics: critical review update from 2007 to the present. *Analytical and Bioanalytical Chemistry*, 404(4):939–965, 2012.

2 The UniProt Consortium. Reorganizing the protein space at the Universal Protein Resource (UniProt). *Nucleic Acids Research*, 40(D1):D71–D75, 2012.

3 Jimmy K. Eng, Ashley L. McCormack, and John R. Yates. An approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database. *Journal of the American Society for Mass Spectrometry*, 5(11):976–989, 1994.

4 Ross et. al. Multiplexed Protein Quantitation in Saccharomyces cerevisiae Using Amine-reactive Isobaric Tagging Reagents. *Molecular & Cellular Proteomics*, 3(12):1154–1169, 2004.

5 Simon J. Hubbard and Andrew R. Jones. *Proteome Bioinformatics*. Methods in Molecular Biology. Humana Press, 2010.

6 Chien-Wen Hung and Andreas Tholey. Tandem Mass Tag Protein Labeling for Top-Down Identification and Quantification. *Analytical Chemistry*, 84(1):161–170, 2012.

7 Rene Hussong and Andreas Hildebrandt. *Signal Processing in Proteomics*, chapter 11, pages 145–161. Methods of Molecular Biology. Humana Press, vol. 604 edition, 2009.

8 Rene Hussong, Andreas Tholey, and Andreas Hildebrandt. Efficient Analysis of Mass Spectrometry Data Using the Isotope Wavelet. In *COMPLIFE 2007: The Third International Symposium on Computational Life Science*, pages 139–49. American Institute of Physics (AIP) Proceedings Volume 940, 2007.

9 Oliver Kohlbacher, Knut Reinert, Clemens Gröpl, Eva Lange, Nico Pfeifer, Ole Schulz-Trieglaff, and Marc Sturm. TOPP - the OpenMS proteomics pipeline. *Bioinformatics*, 23(2):e191–e197, 2007.

10 David N. Perkins, Darryl J. C. Pappin, David M. Creasy, and John S. Cottrell. Probability-based protein identification by searching sequence databases using mass spectrometry data. *Electrophoresis*, 20(18):3551–3567, 1999.

11 Michael Ian Shamos and Dan Hoey. Geometric intersection problems. In *Foundations of Computer Science, 1976., 17th Annual Symposium on*, pages 208–215, 1976.

12 Martin Slawski, Rene Hussong, Andreas Tholey, Thomas Jakoby, Barbara Gregorius, Andreas Hildebrandt, and Matthias Hein. Isotope pattern deconvolution for peptide mass spectrometry by non-negative least squares/least absolute deviation template matching. *BMC Bioinformatics*, 13(1):291, 2012.

13 Andrew Thompson, Jürgen Schäfer, Karsten Kuhn, Stefan Kienle, Josef Schwarz, Günter Schmidt, Thomas Neumann, and Christian Hamon. Tandem Mass Tags: A Novel Quantification Strategy for Comparative Analysis of Complex Protein Mixtures by MS/MS. *Analytical Chemistry*, 75(8):1895–1904, 2003.

GEDEVO: An Evolutionary Graph Edit Distance Algorithm for Biological Network Alignment

Rashid Ibragimov[1], Maximilian Malek[1], Jiong Guo[2], and Jan Baumbach[1,3]

1 **Computational Systems Biology Group, Max-Planck-Institut für Informatik Saarbrücken 66123, Germany**
{ribragim,mmalek}@mpi-inf.mpg.de
2 **Universität des Saarlandes Campus E 1.4, Saarbrücken 66123, Germany**
jguo@mmci.uni-saarland.de
3 **Computational Biology group, University of Southern Denmark Campusvej 5, 5230 Odense M, Denmark**
jan.baumbach@imada.sdu.dk

Abstract

Introduction: With the so-called OMICS technology the scientific community has generated huge amounts of data that allow us to reconstruct the interplay of all kinds of biological entities. The emerging interaction networks are usually modeled as graphs with thousands of nodes and tens of thousands of edges between them. In addition to sequence alignment, the comparison of biological networks has proven great potential to infer the biological function of proteins and genes. However, the corresponding network alignment problem is computationally hard and theoretically intractable for real world instances.

Results: We therefore developed GEDEVO, a novel tool for efficient graph comparison dedicated to real-world size biological networks. Underlying our approach is the so-called Graph Edit Distance (GED) model, where one graph is to be transferred into another one, with a minimal number of (or more general: minimal costs for) edge insertions and deletions. We present a novel evolutionary algorithm aiming to minimize the GED, and we compare our implementation against state of the art tools: SPINAL, GHOST, C-GRAAL, and MI-GRAAL. On a set of protein-protein interaction networks from different organisms we demonstrate that GEDEVO outperforms the current methods. It thus refines the previously suggested alignments based on topological information only.

Conclusion: With GEDEVO, we account for the constantly exploding number and size of available biological networks. The software as well as all used data sets are publicly available at http://gedevo.mpi-inf.mpg.de.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Network Alignment, Graph Edit Distance, Evolutionary Algorithm, Protein-Protein Interactions

Digital Object Identifier 10.4230/OASIcs.GCB.2013.68

1 Introduction

We have finally arrived in the post-genome era. At the web site of the National Center for Biotechnology Information (NCBI) we find registered sequencing projects for >1,500 eukaryotes, >8,500 prokaryotes and >3,000 viruses with >8,000,000 gene sequences in total [26]. However, the genes' function is often unclear and most-widely deduced from similarities to the


© Rashid Ibragimov, Maximilian Malek, Jiong Guo, and Jan Baumbach;
licensed under Creative Commons License CC-BY
German Conference on Bioinformatics 2013 (GCB'13).
Editors: T. Beißbarth, M. Kollmar, A. Leha, B. Morgenstern, A.-K. Schultz, S. Waack, E. Wingender; pp. 68–79

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Table 1 The highest edge correctnesses (EC) achieved by different tools for aligning two pairs of networks, adopted from [14] and extended by the results of SPINAL and GHOST. Note that GHOST did not terminate for *yeast2* vs. *human1*. Note that GEDEVO obtained better results (refer to Table 3). Table 2 summarizes all data sets.

	IsoRank [29]	GRAAL [13]	H-GRAAL [17]	MI-GRAAL [14]	C-GRAAL [16]	SPINAL [1]	GHOST [20]
yeast2 vs. *human1*	3.89	11.72	10.92	23.26	22.55	19.33	-
Meso vs. *Syne*	5.33	11.25	4.59	41.79	26.02	25.86	41.98

sequences of genes with known functions. Consequently, we still lack fundamental knowledge about crucial genetic programs, the interplay of genes and their products (the proteins), their biochemical regulations and their evolutionary appearance. We know very little about how cells, organs and tissues regulate survival, reproduction, differentiation or movement in response to changing internal and external conditions [24]. Many problems in understanding these issues are concerned with biological networks that model the interplay of all kinds of biological entities [4]. Most widely known are transcriptional gene regulatory networks and protein-protein interaction (PPI) networks. More than 16 million protein-protein interactions available through PSICQUIC [3] may serve as an example for the ongoing "data explosion".

One of the major computational challenges in systems biology is biological Network Alignment [11], which aims to find a node-to-node mapping between two or more biological networks, optimizing a certain quality criterion. A quality criterion of a mapping usually reflects topological aspects and biological aspects, such as the number of shared interactions induced by a mapping of the nodes from two networks or a similarity of the biological sequences underlying the nodes. Comparing biological networks, particularly protein-protein interaction (PPI) networks, from different organisms has proven very useful for inferring biological function, besides relying on DNA sequence similarity alone [27, 16].

Biological Network Alignment was recently addressed by several tools. IsoRank [29] integrates the nodes' neighborhoods with sequence information and models the alignment as an eigenvalue problem. C-GRAAL [16], SPINAL [1], GHOST [20], and MI-GRAAL [14] use a similar seed-and-extend approach. While C-GRAAL greedily builds a neighborhood-dependent mapping, SPINAL and MI-GRAAL model (and solve) a weighted bipartite graph problem. IsoRank was shown to be be outperformed by MI-GRAAL [14]. On real PPI networks, SPINAL as well as the GRAAL collection, proved to perform best and to offer biologically meaningful alignments. A brief comparison previously used in [14] is given in Table 1. Instead of replicating the conclusion from the above cited papers that Network Alignment offers biological insights, we concentrate on the methodological problem that the existing tools possess.

All approaches struggle to provide high-quality results on the huge, yet constantly increasing, biological networks that we are confronted with nowadays (Table 2). As we will demonstrate, the existing software cannot cope well with such big networks. This becomes most evident when we see them fail on aligning a network to itself, which should result in 100% node mapping accuracy.

In this paper we present GEDEVO, a novel method for PPI Network Alignment. GEDEVO is an evolutionary algorithm that uses the Graph Edit Distance as optimization model for finding the best alignments. For evaluation, we use a set of high quality PPI networks, including the same networks previously used for C-GRAAL [16] and MI-GRAAL [14] for comparison with existing tools (see Table 1). We will demonstrate that GEDEVO performs comparable or better than recent tools, being at the same time fast and flexible. An

implementation of GEDEVO as well as all used data sets are publicly available under `http://gedevo.mpi-inf.mpg.de`.

2 Methods

2.1 Problem definition

Consider a pair of PPI networks modeled as two unlabeled unweighted graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ and a one-to-one mapping f between nodes V_1 and V_2. We define the Graph Edit Distance (GED) between G_1 and G_2 induced by mapping f as follows: $\mathrm{GED}_f(G_1, G_2) = |\{(u, v) \in E_1 : (f(u), f(v)) \notin E_2\} \cup \{(u', v') \in E_2 : (f^{-1}(u'), f^{-1}(v')) \notin E_1\}|$. By definition, $\mathrm{GED}_f(G_1, G_2)$ counts inserted or deleted edges induced by the mapping f, and it can easily be extended to reflect node/edge dissimilarities or any other related information (e.g. protein sequence similarity). Here, for Network Alignment, we aim to find a mapping f that minimizes the $\mathrm{GED}_f(G_1, G_2)$. Graph Edit Distance is a general model for the Graph Matching problem and defined as the minimal amount of modifications required in graph G_1 to make it isomorphic to graph G_2 (see for example [6] for more details).

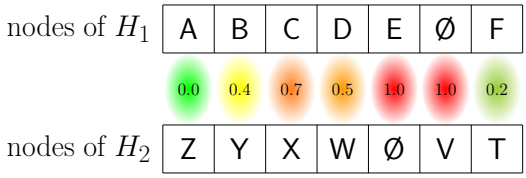
In previous work, the quality of the mapping f of most biological network aligners is assessed by using the number of shared interactions, defined as $|\{(u, v) \in E_1 : (f(u), f(v)) \in E_2\}|$ or the number of conserved interactions, defined as $|\{(u, v) \in E_1 : \mathrm{dist}(v, f(v)) < \Delta, \mathrm{dist}(u, f(u)) < \Delta, (f(u), f(v)) \in E_2\}|$, where $\mathrm{dist}(x, y)$ is a dissimilarity between $x \in V_1$ and $y \in V_2$ (such as BLAST E-value), and Δ is the node dissimilarity threshold. This corresponds to the intuition that the closer two species in the evolutionary tree are, the higher the number of conserved interaction partners they share. Incorporating external biological information, such as sequence similarity, can relax the Network Alignment problem to some extent by significantly reducing the search space by pre-defining preferable sets of nodes to be mapped. Although GEDEVO can include such external information, a "good" method should be able to determine an optimal mapping, by maximizing the number of shared interactions and thus utilizing the graph structure alone. For this reason and to assure comparability between the existing biological Network Alignment tools we focus on topological criteria only in this paper.

2.2 Evolutionary algorithm for Graph Edit Distance

Evolutionary Algorithms (EAs) are nature inspired heuristics, which are widely used to tackle many NP-hard problems (see for example [8]). The key idea behind EAs is mimicking the rule "survival-of-the-fittest" on a population of different individuals. Note that in [15] an EA was suggested for the Graph Matching problem. However, the major hindrance for efficient EA-based combinatorial optimization remained unsolved: the generation of new individuals. In the following we briefly introduce a general scheme of an EA and describe how we modified it with partial adoptions from [15] to Network Alignment.

In an EA an individual represents a solution for the problem, i.e. a mapping of nodes between two networks (see Figure 1). The state of an individual determines how well the individual fits to the requirements of the environment it populates. New individuals result from inheriting parts of solutions from its parents; the better an individual fits the requirements, the higher are its chances to survive and to pass its solution to future generations. Mutations of the solutions exposed to a new individual are another way of mimicking nature in EAs. The requirements of the environment are related to the fitness function, for which we utilize the above defined GED. Starting with generating a (quasi)

Figure 1 A mapping between networks H_1 (nodes A, B, C, D, E, and F) and H_2 (nodes T, V, W, X, Y, and Z) with arbitrary pair scores (for illustration only). In this mapping, node A fits perfectly to node Z, node C corresponds quite poorly to node X, node E is deleted and node V is inserted and both have worst pair scores. One major principle behind GEDOVO's generation of new individuals is to swap pairs of nodes with bad pair scores.



random initial population, an EA repeats the following three steps (individual evaluation, offspring generation, survival function application) until a termination criterion is met.

2.2.1 Initial Population Generation and Evaluation of an Individual

An individual represents a mapping f. Individuals in the initial population are created with random permutations. However, initialization in a more sophisticated manner, which, as a consequence, will require more time, may reduce the convergence time of the algorithm. Here, we may use protein sequence similarities, acquired by BLAST [2], for instance.

Evaluating individuals, for every pair $u \in V_1$ and $v \in V_2$ with $v = f(u)$, we define a *pair score* that reflects how well node u corresponds to node v given a mapping f:

$$\text{pairScore}_f(u, v) = (\text{pairGED}_f(u, v) + \text{grlets}(u, v))/2$$

where $\text{pairGED}_f(u, v)$ is the relative number of deleted and inserted edges induced by mapping node u to node v given mapping f, and $\text{grlets}(u, v)$ is the graphlet degree signature distance introduced in [18]. The graphlet signature distance (GSD) can be interpreted as the difference in neighboring topologies (within distance 4) of two nodes. GSD is computed from two graphlet degree vectors (GDVs); a GDV of a node counts graphlet orbits, which are topologically distinct induced subgraphs (with up to five nodes) the node touches. Note that although computing GDVs for a network with $|V|$ nodes requires $O(|V|^5)$ time it is still practically feasible for graphs as sparse as PPI networks. In addition, GDVs can be precomputed for each network and stored with the graph itself on the hard disk.

The pairScore is mainly used as local optimization guideline. The Graph Edit Distance (GED) is the final, global fitness score of an individual that is to be optimized. GEDEVO exploits the graphlet degree signature distance (GSD) only to accelerate convergence of the algorithm. It is not bound to it; and other external data, such as sequence similarities, may be introduced as additional (weighted or unweighted) terms to the pairScore formula. Computing $\text{pairGED}_f(u, v)$ requires not more than $O(d_1 + d_2) = O(d)$ time, where d_1, d_2 are the maximal node degrees in G_1 and G_2 and $d := \max(d_1, d_2)$. Given precomputed GSDs, a look-up for $\text{grlets}(u, v)$ needs constant time. Thus, computing pair scores for a mapping takes $O(n \cdot (d + s)) = O(n \cdot d)$, where $n = |V_1| + |V_2|$, and s is the number of precomputed terms on the right side of the $\text{pairScore}_f(u, v)$ formula.

The score of an individual together with its *health*, a non-increasing function of the number of iterations and GED of the individual, defines its fitness. The introduction of health allows keeping individuals with a "bad" GED for a number of iterations instead of simply discarding them immediately. This introduces some divergence and contributes to avoiding local optima.

2.2.2 Offspring generation

To generate new individuals we combine a set of different operations to balance between a reasonably high population diversity to avoid local optima and a high and fast convergence towards optimal solutions. The operations are as follows:

- *Random generation* creates an individual by relating it to a mapping based on a random permutation; it requires $O(n)$ time.
- In *PMX-like mutation* we adopt the idea of partially-mapped crossover (PMX), initially introduced in [10]. We partition a mapping into two sets of pairs, low scores and high scores, by using the average over all pair scores in the mapping as a threshold. Afterwards, the high scoring pairs are swapped randomly. To avoid local minima, however, we also swap low scoring pairs with a low probability (of 1% for GEDEVO and PPI networks). PMX-like mutation evaluates each pair by using the pairScore, which requires at most $O(n \cdot d)$ time.
- A so-called *crossover* results in an individual that in the first place preserves pairs with low pair scores from two or more parents. Ties are resolved randomly. Crossover is similar to the previous operation with a term responsible for sorting n pairs from a constant number of parents $p \leq 8$, which results in $O(p \cdot (n \cdot d) + p \cdot n \cdot \log(p \cdot n)) = O(n \cdot (\log n + d))$ time.
- With *directed mutations* we swap of a number $r \leq 20$ randomly chosen "bad" pairs in the mapping of an individual. At the end, the one swap that induces the best score is kept. One swap requires recomputing two pair scores. Thus, the running time of the operation is bound by $O(r \cdot 2 \cdot n \cdot (d + s)) = O(n \cdot d)$.

These operations are GEDEVO's strategies to find and keep "good" pairs while a "bad" pair is swapped more often with another "bad" pair, in this way improving the final score of the mapping. Over a number of iterations, many individuals are exposed to these operations by GEDEVO to traverses the search space and optimizes the final score.

2.2.3 Termination

No practical exact algorithm for the Graph Edit Distance computation on large graphs exists. Consequently, it is hard to theoretically estimate the number of necessary iterations until a "good" solution can be achieved. Convergence time mainly depends on the population size as well as on the input graphs' topological properties. Our implementation of GEDEVO can be set to execute (1) a specified number of iterations, (2) a pre-specified running time, or (3) a fixed number of iterations of no significant changes in the mapping scores of the best individuals (such that convergence was probably reached).

The total theoretical running time of GEDEVO is based on the run times of the individual steps. The evaluation step is performed in $O(N \cdot n \cdot d)$, where N is the population size. The offspring generation step requires $O(N \cdot (n + n \cdot d + n \cdot (\log n + d) + n \cdot d)) = O(N \cdot n \cdot (\log n + d))$ time. The selection step sorts the individuals from the older and new generations in $O(2 \cdot N \cdot \log(2 \cdot N))$ time. Given that GEDEVO runs I iterations, its total running time sums up to $O(I \cdot N \cdot n \cdot (\log n + d))$.

3 Data

For the evaluation of GEDEVO with existing tools we used several PPI networks (see Table 2). The following six networks were previously used for evaluating C-GRAAL and

Table 2 Summary of PPI networks used for evaluations.

Short name	Species	Source	Proteins	Interactions
cjejuni	*Campylobacter jejuni*	[19]	1,095	2,988
Meso	*Mesorhizobium loti*	[28]	1,803	3,094
Syne	*Synechocystis sp.(PCC6803)*	[25]	1,908	3,102
ecoli_fi	*Escherichia coli*	[21]	1,941	3,989
yeast2	*Saccharomyces cerevisiae*	[7]	2,390	16,127
SC	*Saccharomyces cerevisiae*	[31]	5,152	24,847
HS	*Homo Sapiens*	[31]	5,878	14,015
DM	*Drosophila Melanogaster*	[31]	7,533	22,477
ulitsky	*Homo Sapiens*	[30]	7,384	23,462
human1	*Homo Sapiens*	[23]	9,141	41,456
hprd	*Homo Sapiens*	[22]	9,672	3,7047

MI-GRAAL. The two bacterial networks *cjejuni* and *ecoli_fi* are well-studied high-confidence networks: The first network resulted from high-throughput yeast two-hybrid screens; the second network was constructed using experimental and computational data (see [21]). The *Syne* network was obtained through a modified high-throughput yeast two-hybrid assay and covers around half (52%) of the total protein coding genes; similarly for network *Meso* that involves 24% of the protein coding genes. The high-confidence network *human1* was created by combining data from multiple sources including HPRD [22]. The network from [7] is based on (post-processed) data from high throughput experiments.

In addition, we obtained the networks *DM*, *SC*, and *HS* from the DIP database, which contains experimentally determined and manually curated protein interactions. The *hprd* network is a PPI network obtained from the Human Protein Reference Database (HPRD), which is a repository storing high-quality manually curated human interaction data. The human interactome network *ulitsky* is a compilation of protein-protein interactions, based mostly on small-scale experiments, from several interaction databases, including the HPRD database. Refer to Table 2 for a summary and citations.

4 Result and Discussions

Here, we evaluate GEDEVO against the four tools GHOST, SPINAL, C-GRAAL and MI-GRAAL, which form the current state of the art and have been shown to outperform other existing tools [1, 14, 16].

All tools were executed on a 64 bit Linux 2.6.32 kernel, running on an Intel Xeon CPU W3550 @ 3.07GHz and 12 GB RAM. SPINAL is deterministic and was thus executed only one time for each pair of the input networks, while MI-GRAAL, GHOST, C-GRAAL and GEDEVO, as randomized algorithms, we executed 10 times for each pair. The execution of all tools was interrupted after 24 hours of runtime without termination. MI-GRAAL and C-GRAAL, similarly to GHOST, require graphlet degree signatures as preliminary node similarity measures, which were precomputed and used as input (precomputation time not taken into account for evaluation). The termination criterion for GEDEVO was set to stop after 3,000 iterations of no significant improvement of the GED score amongst the best solutions (individuals).

The performance of all methods, similarly to [14] and [16], is assessed with the so-called

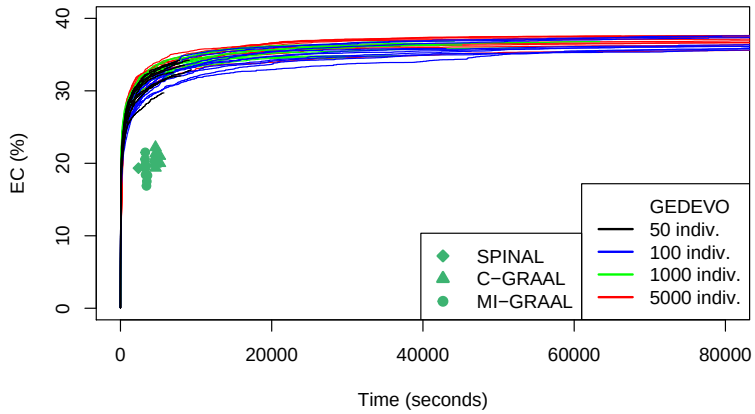


Figure 2 The influence of the population size to the performance of GEDEVO aligning *yeast2* vs. *human1* in comparison to SPINAL, C-GRAAL and MI-GRAAL. Each line/symbol represents one run.

Edge Correctness, which is particularly useful when comparing many networks with different numbers of nodes and edges. Defined as $\text{EC} = \frac{\#\text{sharedInteractions}}{\min(|E_1|,|E_2|)} \times 100$ [%], its highest value is 100% and occurs if one input network is isomorphic (or sub-isomorphic) to the other.

Note that GEDEVO internally utilizes the Graph Edit Distance for optimization, not the EC. This makes GEDEVO more applicable to general graph comparison problems outside computational biology. However, if we set the costs for node deletions/insertions/substitutions and edge substitution to zero but only the cost for edge deletions/insertions to one, the EC will be related to GED as $\text{EC} = (|E_1| + |E_2| - \text{GED})/(2 \cdot \min(|E_1|, |E_2|)) \times 100$ [%]. This allows us to compare GEDEVO to existing approaches on protein-protein interaction Network Alignment based on the EC criterion, as in previous work [14, 16], which is particularly useful for networks where differences between $|V_1|$ and $|V_2|$ are common (as in PPI networks from different organisms).

In Figure 2 we depict the influence of the population size to the progression of the EC (convergence). Runs with 50 individuals (black line) converged earlier to the final solution, still providing quite high values of EC. With larger population sizes the runs obtained slightly better alignments with higher EC and reached them slightly faster. This indicates that GEDEVO is quite robust to different population sizes, given that they are reasonably large. In the remaining (below described) evaluations we used 500 individuals per run.

We executed GEDEVO and the four competing tools on multiple pairs of networks from Table 2. The resulting edge correctnesses and the according run times for all tools are depicted in Figure 3. Since SPINAL, C-GRAAL and MI-GRAAL do not provide intermediate results, the final values are shown point-wise (diamonds, triangles, and circles); for GEDEVO the progression of EC is depicted with lines. The plot illustrates that GEDEVO can provide a "good" solution comparably fast. A summary of the maximal EC values from the plot is given in Table 3. Unsuccessful runs (no termination after 24 hours) of SPINAL and MI-GRAAL are marked with an "x"). Note: Since GHOST only terminated for the alignment of the two small networks *Meso* and *Syne* (with best EC: 41.98, runtime: 140 sec) we did not add it to Table 2 and Figure 3.

The networks *human1*, *hprd* and *ulitsky* are all human PPI networks, therefore the EC scores for GEDEVO are comparably high. The results for aligning *ulitsky* with *human1* or *hprd* are rather "poor" since *ulitsky* is a compilation of data from different databases. The known overlap of *ulitsky* with *hprd* is only 649 nodes and 15,305 interactions, from which GEDEVO aligned 11,800.

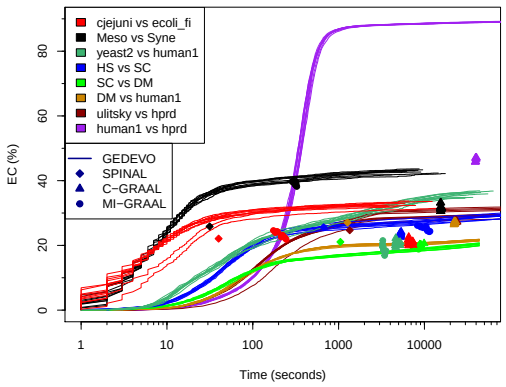


Figure 3 Convergence and Edge Correctness vs. run time for aligning different PPI networks, 10 runs for each tool. Each line/symbol represents one run.

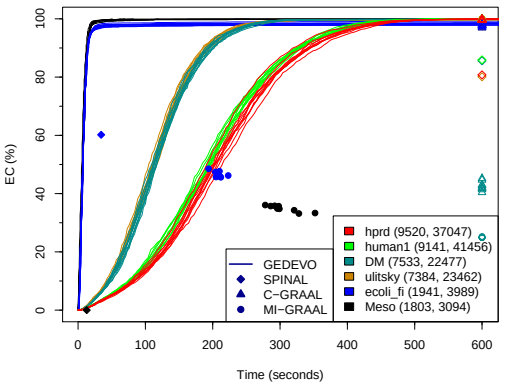


Figure 4 Aligning a network against itself should result in an Edge Correctness of 100%, which is achieved with GEDEVO and C-GRAAL in most cases (see text). Each line/symbol represents one run. Unfilled symbols (at the right side of the plot) mean that it took more than 10 minutes to achieve the corresponding EC values.

To further investigate the robustness of the four methods on networks where we definitely know the correct solution, we aligned some PPI networks against themselves. Naturally, this should result in an EC of 100%. In Figure 4, we plot the EC vs. run time for the following data sets: *Meso*, *ecoli_fi*, *ulitsky*, *DM*, and *human1*. Note that GHOST only terminated for the self-alignment of the two smallest networks *Meso* (with best EC: 100%, runtime: 197 sec) and *ecoli_fi* (with best EC: 100%, runtime: 173 sec). We also downloaded and tested Natalie 2.0 [9] on our servers. It terminated with memory faults for all network pairs but the two smallest ones: For *cjejuni* vs. *ecoli_fi* (runtime: 7 hours) and self-alignment of *Meso* (runtime: 11 hours) the tool resulted with edge correctnesses of 97.64% and 20.38% respectively. Hence, we did not include GHOST and Natalie 2.0 with Figure 4. Further note that a set of methods exists that restrict alignment candidates to a set of pre-mapped nodes (limited search space), see for example [12] and [5]. GEDEVO can be restricted to such pre-mappings (e.g. with BLAST as preprocessing) but it does not rely on it.

In conclusion, GEDEVO, in contrast to the other approaches, was able to achieve the expected 100% EC in all cases, often even faster than the existing tools. C-GRAAL reached around 97-98% of EC in most cases, but required up to 11 hours for the biggest networks (*hprd*, *human*), for which GEDEVO needed only approx. 10 minutes.

To sum up, in almost all cases, GEDEVO outperformed SPINAL, GHOST, C-GRAAL and MI-GRAAL in terms of quality and run time. Moreover, GEDEVO in contrast to the other methods was able to recognize the high similarity (*human1* vs. *hprd*) and composition (*ulitsky* vs. *hprd*) between the human PPI networks using topological information only. In addition, we wish to emphasize that GEDEVO provides intermediate results that allow for a manual termination of the software at earlier iterations when a high EC score (or a corresponding low Graph Edit Distance solution) has been found and convergence seems to be reached.

Table 3 The highest achieved Edge Correctness (EC) quality scores for alignments of different PPI networks from Figure 3.

Network 1	Network 2	EC (%)			
		GEDEVO	MI-GRAAL	C-GRAAL	SPINAL
cjejuni	*ecoli_fi*	**33.70**	*24.60*	22.56	22.09
Meso	*Syne*	**43.60**	*39.88*	33.19	25.86
yeast2	*human1*	**38.14**	21.38	*22.20*	19.33
HS	*SC*	**30.40**	*26.15*	24.15	25.59
SC	*DM*	*20.79*	17.73	20.59	**21.07**
DM	*human1*	21.88	x	**27.36**	*27.04*
ulitsky	*hprd*	**32.00**	x	*27.56*	24.68
human1	*hprd*	**89.37**	x	*47.07*	x

5 Conclusion

We presented GEDEVO, a novel Network Alignment algorithm, and evaluated it on protein-protein interaction networks. GEDEVO uses an evolutionary algorithm to heuristically approximate the Graph Edit Distance optimization problem. On a wide range of real PPI networks our approach outperforms state-of-the-art methods in terms of speed and quality, and provides intermediate alignment results on the fly. GEDEVO is robust and not limited to PPI networks (unlabeled, undirected, and unweighted graphs), but flexible enough to be applicable to other types of networks as well, biological and non-biological.

In the future we will speed-up the convergence of our algorithm by improving the population initialization by complementing the random permutations in this step with an assignment function that depends, for instance, on the degree differences between candidate node pairs (the less the difference, the higher the chance to contribute to low GED and high EC in the final solution). While in this paper, the quality measures were derived from purely topological alignments, in the future we will experiment with integrating additional external node-to-node scoring functions, such as BLAST.

GEDEVO as well as all used data sets are publicly available at `http://gedevo.mpi-inf.mpg.de`.

Acknowledgements We wish to thank Nataša Pržulj from Imperial College London for interesting discussions and her help with the data sets used in her C-GRAAL and MI-GRAAL papers. RI is grateful for financial support from the International Max Planck Research School (IMPRS). All authors with to thank the DFG Cluster of Excellence for Multimodal Computing and Interaction (MMCI) for support. JB is also grateful for funding from the Villum Foundation.

References

1 Ahmet E. Aladag and Cesim Erten. SPINAL: scalable protein interaction network alignment. *Bioinformatics*, 29(7):917–924, 2013.

2 Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, October 1990.

3 Bruno Aranda, Hagen Blankenburg, Samuel Kerrien, Fiona S L Brinkman, Arnaud Ceol, Emilie Chautard, Jose M Dana, Javier De Las Rivas, Marine Dumousseau, Eugenia Ga-

leota, Anna Gaulton, Johannes Goll, Robert E W Hancock, Ruth Isserlin, Rafael C Jimenez, Jules Kerssemakers, Jyoti Khadake, David J Lynn, Magali Michaut, Gavin O'Kelly, Keiichiro Ono, Sandra Orchard, Carlos Prieto, Sabry Razick, Olga Rigina, Lukasz Salwinski, Milan Simonovic, Sameer Velankar, Andrew Winter, Guanming Wu, Gary D Bader, Gianni Cesareni, Ian M Donaldson, David Eisenberg, Gerard J Kleywegt, John Overington, Sylvie Ricard-Blum, Mike Tyers, Mario Albrecht, and Henning Hermjakob. PSICQUIC and PSISCORE: accessing and scoring molecular interactions. *Nature Methods*, 8(7):528–9, Jul 2011.

4 Jan Baumbach. On the power and limits of evolutionary conservation–unraveling bacterial gene regulatory networks. *Nucleic Acids Res*, 38(22):7877–84, Dec 2010.

5 Mohsen Bayati, David F. Gleich, Amin Saberi, and Ying Wang. Message-passing algorithms for sparse network alignment. *ACM Trans. Knowl. Discov. Data*, 7(1):3:1–3:31, March 2013.

6 Horst Bunke and Kaspar Riesen. *Graph Edit Distance – Optimal and Suboptimal Algorithms with Applications*, pages 113–143. Wiley-VCH Verlag GmbH & Co. KGaA, 2009.

7 Sean R. Collins, Patrick Kemmeren, Xue-Chu Zhao, Jack F. Greenblatt, Forrest Spencer, Frank C. P. Holstege, Jonathan S. Weissman, and Nevan J. Krogan. Toward a comprehensive atlas of the physical interactome of saccharomyces cerevisiae. *Molecular and Cellular Proteomics*, 6(3):439–450, 2007.

8 Agoston E. Eiben and J. E. Smith. *Introduction to evolutionary computing*. Natural Computing Series. Springer, December 2010.

9 Mohammed El-Kebir, Jaap Heringa, and GunnarW. Klau. Lagrangian relaxation applied to sparse global network alignment. In Marco Loog, Lodewyk Wessels, MarcelJ.T. Reinders, and Dick Ridder, editors, *Pattern Recognition in Bioinformatics*, volume 7036 of *Lecture Notes in Computer Science*, pages 225–236. Springer Berlin Heidelberg, 2011.

10 David E. Goldberg and Robert Jr. Linge. Alleles, loci, and the traveling salesman problem. In John J. Grefenstette, editor, *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*. Lawrence Erlbaum Associates, Publishers, 1985.

11 Allison P Heath and Lydia E Kavraki. Computational challenges in systems biology. *Computer Science Review*, 3(1):1–17, 2009.

12 Gunnar Klau. A new graph-based method for pairwise global network alignment. *BMC Bioinformatics*, 10(Suppl 1):S59, 2009.

13 O. Kuchaiev, T. Milenković, V. Memišević, W. Hayes, and N. Pržulj. Topological network alignment uncovers biological function and phylogeny. *Journal of the Royal Society Interface*, 7(50):1341–1354, 2010.

14 Oleksii Kuchaiev and Nataša Pržulj. Integrative Network Alignment Reveals Large Regions of Global Network Similarity in Yeast and Human. *Bioinformatics*, 27(10):1390–1396, March 2011.

15 Cheng-Wen Liu, Kuo-Chin Fan, Jorng-Tzong Horng, and Yuan-Kai Wang. Solving weighted graph matching problem by modified microgenetic algorithm. In *Systems, Man and Cybernetics, 1995. Intelligent Systems for the 21st Century., IEEE International Conference on*, volume 1, pages 638–643. IEEE, 1995.

16 Vesna Memišević and Nataša Pržulj. C-GRAAL: Common-neighbors-based global GRAph ALignment of biological networks. *Integr. Biol.*, 4:734–743, 2012.

17 T. Milenković, W.L. Ng, W. Hayes, and N. Pržulj. Optimal network alignment with graphlet degree vectors. *Cancer informatics*, 9:121, 2010.

18 Tijana Milenković and Nataša Pržulj. Uncovering biological network function via graphlet degree signatures. *Cancer Informatics*, 6:0–0, 04 2008.

19 Jodi Parrish, Jingkai Yu, Guozhen Liu, Julie Hines, Jason Chan, Bernie Mangiola, Huamei Zhang, Svetlana Pacifico, Farshad Fotouhi, Victor DiRita, Trey Ideker, Phillip Andrews,

and Russell Finley. A proteome-wide protein interaction map for campylobacter jejuni. *Genome Biology*, 8(7):R130, 2007.

20 Rob Patro and Carl Kingsford. Global network alignment using multiscale spectral signatures. *Bioinformatics*, 28(23):3105–3114, 2012.

21 Jose M. Peregrin-Alvarez, Xuejian Xiong, Chong Su, and John Parkinson. The modular organization of protein interactions in *Escherichia coli*. *PLoS Comput Biol*, 5(10):e1000523, 10 2009.

22 Keshava T.S. Prasad, Renu Goel, Kumaran Kandasamy, Shivakumar Keerthikumar, Sameer Kumar, Suresh Mathivanan, Deepthi Telikicherla, Rajesh Raju, Beema Shafreen, Abhilash Venugopal, Lavanya Balakrishnan, Arivusudar Marimuthu, Sutopa Banerjee, Devi S. Somanathan, Aimy Sebastian, Sandhya Rani, Somak Ray, Harrys C.J. Kishore, Sashi Kanth, Mukhtar Ahmed, Manoj K. Kashyap, Riaz Mohmood, Y.L. Ramachandra, V. Krishna, Abdul B. Rahiman, Sujatha Mohan, Prathibha Ranganathan, Subhashri Ramabadran, Raghothama Chaerkady, and Akhilesh Pandey. Human Protein Reference Database–2009 update. *Nucleic acids research*, 37(Database issue):D767–D772, January 2009.

23 Predrag Radivojac, Kang Peng, Wyatt T. Clark, Brandon J. Peters, Amrita Mohan, Sean M. Boyle, and Sean D. Mooney. An integrated approach to inferring gene-disease associations in humans. *Proteins: Structure, Function, and Bioinformatics*, 72(3):1030–1037, 2008.

24 Richard Röttger, Ulrich Rückert, Jan Taubert, and Jan Baumbach. How little do we actually know? On the size of gene regulatory networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 9:1293–1300, 2012.

25 Shusei Sato, Yoshikazu Shimoda, Akiko Muraki, Mitsuyo Kohara, Yasukazu Nakamura, and Satoshi Tabata. A large-scale protein-protein interaction analysis in synechocystis sp. pcc6803. *DNA Research*, 14(5):207–216, 2007.

26 Eric W Sayers, Tanya Barrett, Dennis A Benson, Evan Bolton, Stephen H Bryant, Kathi Canese, Vyacheslav Chetvernin, Deanna M Church, Michael DiCuccio, Scott Federhen, Michael Feolo, Ian M Fingerman, Lewis Y Geer, Wolfgang Helmberg, Yuri Kapustin, David Landsman, David J Lipman, Zhiyong Lu, Thomas L Madden, Tom Madej, Donna R Maglott, Aron Marchler-Bauer, Vadim Miller, Ilene Mizrachi, James Ostell, Anna Panchenko, Lon Phan, Kim D Pruitt, Gregory D Schuler, Edwin Sequeira, Stephen T Sherry, Martin Shumway, Karl Sirotkin, Douglas Slotta, Alexandre Souvorov, Grigory Starchenko, Tatiana A Tatusova, Lukas Wagner, Yanli Wang, W John Wilbur, Eugene Yaschenko, and Jian Ye. Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res*, 39(Database issue):D38–51, Jan 2011.

27 Roded Sharan, Silpa Suthram, Ryan M Kelley, Tanja Kuhn, Scott McCuine, Peter Uetz, Taylor Sittler, Richard M Karp, and Trey Ideker. Conserved patterns of protein interaction in multiple species. *PNAS*, 102(6):1974–1979, Feb 2005.

28 Yoshikazu Shimoda, Sayaka Shinpo, Mitsuyo Kohara, Yasukazu Nakamura, Satoshi Tabata, and Shusei Sato. A large scale analysis of protein-protein interactions in the nitrogen-fixing bacterium mesorhizobium loti. *DNA Research*, 15(1):13–23, 2008.

29 Rohit Singh, Jinbo Xu, and Bonnie Berger. Pairwise global alignment of protein interaction networks by matching neighborhood topology. In *Proc. of the 11th annual international conference on Research in computational molecular biology*, RECOMB'07, pages 16–31, Berlin, Heidelberg, 2007. Springer-Verlag.

30 Igor Ulitsky, Richard M. Karp, and Ron Shamir. Detecting Disease-Specific Dysregulated Pathways Via Analysis of Clinical Expression Profiles Research in Computational Molecular Biology. In Martin Vingron and Limsoon Wong, editors, *Research in Computational Molecular Biology*, volume 4955 of *Lecture Notes in Computer Science*, chapter 30, pages 347–359. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2008.

31 Ioannis Xenarios, Lukasz Salwínski, Xiaoqun Joyce, Patrick Higney, Sul-Min M. Kim, and David Eisenberg. DIP, the Database of Interacting Proteins: a research tool for studying cellular networks of protein interactions. *Nucleic acids research*, 30(1):303–305, January 2002.

Dinucleotide distance histograms for fast detection of rRNA in metatranscriptomic sequences

Heiner Klingenberg[1], Robin Martinjak[1], Frank Oliver Glöckner[2,3], Rolf Daniel[4], Thomas Lingner[1], and Peter Meinicke*[1]

1 **Department of Bioinformatics, University of Göttingen**
Goldschmidtstr.1, Göttingen, Germany
{heiner,robin,thomas,peter}@gobics.de
2 **Max Planck Institute for Marine Microbiology**
Celsiusstrasse 1, 28359 Bremen, Germany
3 **Jacobs University Bremen gGmbH**
Campus Ring 1, 28759 Bremen, Germany
fog@mpi-bremen.de
4 **Department of Genomic and Applied Microbiology, University of Göttingen**
Goldschmidtstr.1, Göttingen, Germany
rdaniel@gwdg.de

Abstract

With the advent of metatranscriptomics it has now become possible to study the dynamics of microbial communities. The analysis of environmental RNA-Seq data implies several challenges for the development of efficient tools in bioinformatics. One of the first steps in the computational analysis of metatranscriptomic sequencing reads requires the separation of rRNA and mRNA fragments to ensure that only protein coding sequences are actually used in a subsequent functional analysis. In the context of the rRNA filtering task it is desirable to have a broad spectrum of different methods in order to find a suitable trade-off between speed and accuracy for a particular dataset. We introduce a machine learning approach for the detection of rRNA in metatranscriptomic sequencing reads that is based on support vector machines in combination with dinucleotide distance histograms for feature representation. The results show that our SVM-based approach is at least one order of magnitude faster than any of the existing tools with only a slight degradation of the detection performance when compared to state-of-the-art alignment-based methods.

1998 ACM Subject Classification J.3 Biology and genetics

Keywords and phrases Metatranscriptomics, metagenomics, rRNA detection, distance histograms

Digital Object Identifier 10.4230/OASIcs.GCB.2013.80

1 Introduction

Metatranscriptomics has become an essential tool for the investigation of gene expression in microbial communities [6, 16, 7, 2, 13]. Compared to metagenomics, metatranscriptomics provides a dynamic picture of the adaptation of organisms to changing environmental conditions. Depending on the particular protocol for extraction and sequencing of environmental

* corresponding author

© H. Klingenberg, R. Martinjak, F.O. Glöckner, R. Daniel, T. Lingner, and P. Meinicke;
licensed under Creative Commons License CC-BY

German Conference on Bioinformatics 2013 (GCB'13).
Editors: T. Beißbarth, M. Kollmar, A. Leha, B. Morgenstern, A.-K. Schultz, S. Waack, E. Wingender; pp. 80–89

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

RNA, a substantial amount of the resulting sequences actually correspond to ribosomal RNA (rRNA) that cannot be used for the analysis of gene expression levels. Therefore an important first step in the analysis of RNA-Seq data from metatranscriptomic experiments is to filter out the fraction of sequencing reads with significant similarity to known rRNA genes. After that the remaining messenger RNA (mRNA) reads are usually analyzed in terms of possible gene functions based on sequence similarity to known proteins from, for instance, the Pfam [17] or KEGG [10] databases. Without a prior rRNA filtering the risk is high to obtain a large number of false positive protein matches. For example, in a previous release of the Pfam database, due to misannotation, several families have been composed of spurious ORFs on the reverse strand of rRNA and therefore systematically accounted for false protein matches in metatranscriptomic RNA-Seq data [22]. Besides a time-consuming BLASTN [1] search against a comprehensive rRNA database, several recent tools can be used which all provide a computationally faster rRNA detection. The accelerating techniques in these tools include Hidden Markov Models [9, 12], the Burrows-Wheeler transformation [21] and TRIE-structures in combination with a fast bitvector matching [11]. We here propose a machine learning approach using a feature space based on oligomer distance histograms which have originally been introduced for remote homology detection in protein sequence analysis [14]. For rRNA detection we have implemented a specific feature extraction that counts the occurrences of all dinucleotide pairs over a range of possible distances (spacers) between them. Our results indicate that dinucleotide distance histograms provide a suitable representation of rRNA sequences and that SVMs are well-suitable for fast detection of rRNA in metatranscriptomic datasets.

2 Materials & Methods

Our approach for rRNA detection in metatranscriptomic datasets is based on an RNA-specific adaption of the oligomer distance histogram (ODH) representation for biological sequences [14] and Support Vector Machines (SVM) for discrimination between rRNA and non-rRNA sequence fragments. After training of SVM classifiers using reference datasets for 16S/23S-rRNA and non-rRNA examples, we evaluate the performance of our method and several state-of-the-art rRNA detection approaches on simulated and real-world metatranscriptomics datasets. In the following sections we describe in detail the utilized datasets and the modified ODH feature space and we outline the methods used for performance comparison. The C source code for ODH-based rRNA detection is available from the authors.

2.1 Datasets

Reference datasets

To construct a reference dataset for training and test of SVM classifier models, we obtained all available rRNA gene sequences from the SILVA database [18] and separated them according to their phylogenetic origin (Archaea/Bacteria) and type (16S/23S). 3' and 5' sequence overhangs were removed by trimming the sequences according to their relevant rRNA region using the ARB software package [15]. To reduce the redundancy of the dataset and avoid overlaps between training and test data, we clustered the resulting sequence sets with USEARCH (version 6.0.307) [4] using a sequence identity threshold of 95%. The final bacterial/archaeal rRNA reference datasets contained 80,832/4,217 16S-rRNA sequences and 1,930/137 23S-rRNA sequences, respectively.

For evaluation of different methods we partitioned the reference datasets into training

and test sets containing 80% and 20% of the sequences, respectively. Here, we attempted to create sequence sets with similar sequence variability (for details see Appendix). By this means we obtained 64,665 (16,167) full length training (test) 16S-rRNA sequences for Bacteria and 3,373 (844) sequences for Archaea. The respective 23S datasets consist of 1,544 (386) bacterial and 109 (28) archaeal sequences.

For our discriminative learning approach we also require the training and test datasets to contain negative sequence examples, i.e. suitable non-rRNA sequences. For this purpose we masked 1,705/121 completely sequenced bacterial/archaeal genomes with respect to known rRNA and non-coding regions. For each rRNA reference dataset we extracted fragments from the remaining sequence material to yield a negative dataset of identical size and sequence length distribution.

Simulated metatranscriptome dataset

In order to evaluate the rRNA detection performance of different methods, we generated a simulated metatranscriptome dataset with known rRNA and non-rRNA labels. For this purpose we applied MetaSim[19] to our positive and negative test sequences to produce rRNA and non-rRNA sequence reads, respectively. Here, the MetaSim default parameters for the Roche 454 sequencer model (including a relatively high error rate of 5%) were used and the average read length was set to 250 bp. Multiple (forward/reverse) reads were generated for each reference sequence until at least 80% of the original sequence was covered. Sequence fragments that had an overlap of more than 150 bp with another read were removed. In total, our simulated dataset consists of 214,270/10,535 16S/23S-rRNA and 952,215 non-rRNA reads, respectively.

Real-world metatranscriptome datasets

In [12] two metatranscriptome datasets were used for comparative evaluation of the rRNA detection performance of the rRNASelector and Meta-RNA methods. The datasets ('Tidal salt marsh' and 'Mushroom Spring') revealed a remarkably high predicted fraction (54% and 89%, respectively) of rRNA-related sequences. The Mushroom Spring dataset (SRR106861) consists of 113,128 sequences (≈30 Mbp) and an average read length of ≈267 bp, while the Tidal salt marsh dataset (SRR013513) comprises 238,250 sequences (≈62 Mbp) and an average read length of ≈259 bp. Both samples have been sequenced using the Roche 454 FLX Titanium platform. We downloaded the two datasets from the National Center for Biotechnology Information (NCBI) Sequence Read Archive (SRA[1]), converted the sequences into the FASTA format and applied PRINSEQ [20] to the sequence sets to remove replicates and sequences exceeding 5 undefined bases. After application of PRINSEQ, 196,512 sequences (≈50 Mbp) and 91,589 sequences (≈24 Mbp) remained for testing in the reduced SRR013513 and SRR106861 dataset, respectively.

2.2 Dinucleotide distance histogram feature space

In [14], oligomer distance histograms (ODH) have been introduced as a vector space representation method for protein sequences. In the ODH feature space each sequence is represented as a numerical vector in which each dimension indicates the number of occurrences of a particular oligomer (k-mer) pair at a particular distance in the sequence. For a specific

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

sequence analysis problem the ODH feature space consists of the required dimensions to consider all possible oligomer pairs for all distances (including the 'zero' distance $D = 0$) up to the longest observable distance between two oligomers.

To apply ODHs to the rRNA detection problem (in a computationally efficient manner), we here fix the length k of the oligomers to $k = 2$ (dinucleotides) and introduce an upper limit D_{max} for the distance between two oligomers. In contrast to [14] we here omit oligomer distances that reflect an overlap of two oligomers, i.e. the distances $D = 0$ and $D = 1$, and instead refer to an inventory of 'spacers' from $\mathcal{D}_0$ (i.e. $D = k$) to $\mathcal{D}_{max}$ ($D_{max}+k$). Therefore, our dinucleotide distance histogram feature space consists of $16^2 * \mathcal{D}_{max}$ dimensions. The maximum spacer value $\mathcal{D}_{max}$ constitutes a so-called hyperparameter whose optimal value has to be determined by evaluation.

Discriminative classifier training

In order to distinguish between rRNA and non-rRNA sequence reads in metatranscriptomics datasets, we trained discriminative linear classifier models using Support Vector Machines (SVM) in combination with dinucleotide distance histogram feature space representatives of the reference dataset sequences. Here, we aggregated bacterial and archaeal sequences to obtain one 16S- and one 23S-rRNA classifier, respectively. For SVM training we used the LIBLINEAR implementation [5] with default parameters for the slack variables ($C = 1$) and termination tolerance ($\epsilon = 0.1$). Because the LIBLINEAR toolbox does not provide an option to account for imbalanced numbers of positive and negative training data, we 'oversampled' the positive examples to yield the same amount of rRNA and non-rRNA sequences while retaining the diversity of non-rRNA sequences. As a consequence, we used up to 30 duplicates (archaeal 23S-rRNA) of an rRNA example for model training.

2.3 Experimental setup

For performance evaluation we compared our approach to different state-of-the-art methods for rRNA detection. Besides the riboPicker [21] and SortMeRNA [11] methods, we used HMMER3 [3] as a representative method for HMM-based detection approaches such as MetaRNA [9] or rRNASelector [12]. The riboPicker detection software is based on a pre-computed rRNA database[2] which does not allow the convenient removal of particular sequences. To avoid an overlap of training and test sequences, we refrained from using riboPicker for evaluation on the simulated metatranscriptome dataset and instead performed a BLASTN homology search. Here, we used the rRNA/non-rRNA assignment of the best BLAST hit up to an E-value threshold of $1e^{-3}$ for read classification. For the simulated data, the reference models/databases for all methods were built using only the training sequences. Here, separate HMMER3 models for Archaea and Bacteria were trained. For evaluation on the real-world metatranscriptome dataset we used all reference dataset sequences for training of HMMER3 models and SVM classifiers and we utilized the default databases for riboPicker and SortMeRNA.

3 Results

In order to evaluate our distance histogram-based rRNA detection approach, we investigated the prediction performance on simulated and real-world metatranscriptome data. First, we

[2] `http://edwards.sdsu.edu/ribopicker/rrnadb/rrnadb_2012-01-17.tar.gz`

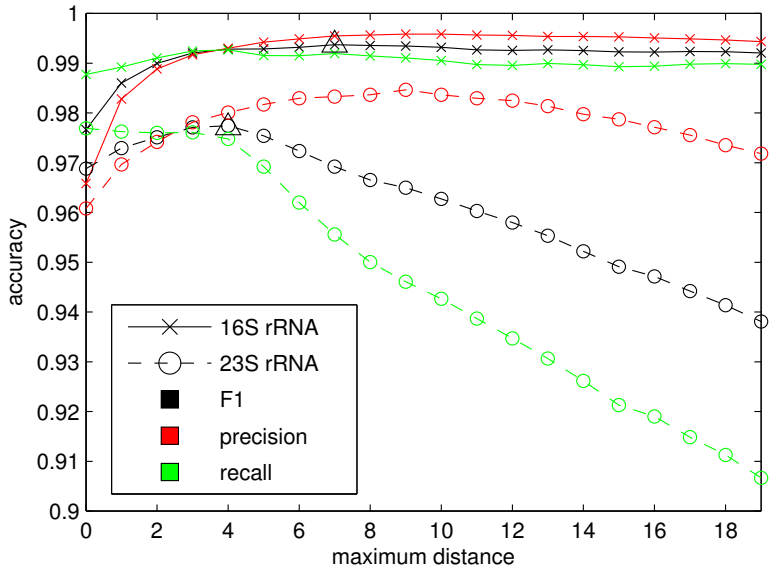


Figure 1 Dependency of the 16S-/23S-rRNA detection performance on the maximum distance parameter. Maximum F_1 performance values are indicated by triangles.

determined optimal values for the maximum distance parameter of our method for 16S- and 23S-rRNA classifiers, respectively. Then we compared the rRNA detection performance and the runtime of our approach to those of state-of-the-art methods.

Selection of optimal values for the maximum distance parameter

Our distance histogram-based feature space for nucleotide sequences (DDH) requires the definition of a maximum distance (spacer) $\mathcal{D}_{max}$ between two dimers (see also section 2.2). While small values for $\mathcal{D}_{max}$ lead to a memory-efficient feature space representation of DNA/RNA sequences, higher values allow to model conserved long-range correlations between particular residues in the sequence. To determine optimal values for the maximum distance parameter, we performed a 5-fold cross-validation on simulated 16S- and 23S-rRNA training datasets using different values for $\mathcal{D}_{max} = [0, .., 19]$. To yield meaningful performance measure values, we balanced the number of positive and negative test examples by oversampling the rRNA example sequences analogously to the classifier training procedure (see section 2.2).

Figure 1 shows the dependency of the rRNA detection performance on the maximum distance value in terms of precision ($\frac{\#TP}{\#TP+\#FP}$) and recall ($\frac{\#TP}{\#TP+\#FN}$) curves. While the recall values already start to decrease for medium values of $\mathcal{D}_{max}$, the precision increases until a plateau is reached. The F_1 measure, which combines precision and recall, shows different specific local performance maxima for 16S ($\mathcal{D}_{max} = 7$) and 23S ($\mathcal{D}_{max} = 4$) data. In the following, we use these optimal values for all evaluations.

Performance on simulated data

Our simulated dataset allows to evaluate the rRNA detection performance of different methods independently for 16S- and 23S-rRNA sequence fragments based on a known classification of the reads. Table 1 shows the performance values of four different methods for our simulated 16S- and 23S-rRNA datasets, respectively. For 16S-rRNA data, the HMMER3 method

Table 1 rRNA detection performance on simulated metatranscriptome data for different methods. All values represent percentages.

		DDH	HMMER3	BLASTN	SortMeRNA	# reads
16s	recall	98.79	99.94	99.06	99.90	428540
	precision	99.60	100.0	100.0	100.0	
	F_1	99.19	99.97	99.53	99.95	
23s	recall	97.17	99.28	91.40	98.92	21070
	precision	98.00	100.0	100.0	100.0	
	F_1	97.58	99.63	95.51	99.46	

achieves almost perfect classification of the reads, closely followed by SortMeRNA. BLASTN and our DDH approach show a slightly lower detection performance in terms of the F_1 measure due to a higher fraction of overlooked 16S-rRNA sequences. Remarkably, HMMER3, SortMeRNA and BLASTN only classify very few non-rRNA reads as ribosomal RNA and thus yield a (rounded) precision of 100%.

For the simulated 23S-rRNA data HMMER3 and SortMeRNA also achieve very high detection performance values. The precision and recall values of the DDH method slightly decrease as compared to the 16S dataset, which has presumably to be attributed to the much smaller number of training examples for the 23S dataset. The detection performance of BLASTN in terms of the recall value substantially decreases for 23S-rRNA reads due to a considerably reduced number of significant hits to the database sequences. Additional experiments with longer sequence reads (400bp) and a lower simulated read error rate (2.5%) indicated that the detection performance of all methods increases, with BLASTN showing the biggest improvement (data not shown).

3.1 Performance on metatranscriptome data

In contrast to the simulated datasets, the classification of reads from real-world metatranscriptome data is not known and thus no ground truth exists. Because the HMMER3 method outperformed the other approaches on the simulated data, we first measured the overlap with the other methods in terms of a hypothetical ground truth on the real-world datasets provided by HMMER3 predictions. Table 2 shows the results of the overlap analysis in terms of hypothetical recall, precision and F_1 estimates for different methods on the two real-world metatranscriptome datasets. For the Mushroom spring dataset, SortMeRNA achieved the highest agreement with the HMMER3 prediction followed by riboPicker and our distance histogram approach. While the overlap recall ranged from 89% to almost 100%, the overlap precision of all methods was very high. This can be attributed to the high fraction of predicted rRNA (≈89%) in this dataset (see also section 2.1). In contrast, the Tidal salt marsh dataset showed a substantially lower predicted fraction of rRNA reads (≈54%). Here, the precision value for riboPicker and our DDH approach decreased to ∼92%, while the SortMeRNA method still showed a high value (99%). However, because of a substantially diminished recall value for SortMeRNA on this dataset, riboPicker exhibited the highest agreement with the HMMER3 predictions in terms of the combined F_1 measure.

Figure 2 shows Venn diagrams representing the overlap of predicted rRNA fractions as obtained from the three abovementioned methods without HMMER3. For the Mushroom Spring datasets the three methods agreed on ≈86% of the sequence fragments, while no method exclusively classified more than 1% of the reads as rRNA. The Venn diagram associated with

Table 2 rRNA detection overlap of different methods with HMMER3 predictions on real-world metatranscriptome data. All values represent percentages.

		riboPicker	SortMeRNA	DDH
Mushroom Spring	recall	97.60	99.78	89.39
	precision	99.75	99.78	98.93
	F_1	98.66	99.78	93.92
Tidal salt marsh	recall	97.54	81.81	82.72
	precision	91.57	99.00	92.80
	F_1	94.46	89.60	87.47

the Tidal marsh dataset shows a substantially smaller consensus of the method predictions. Here, riboPicker and our DDH approach filtered ≈ 15% and 5% of the reads exclusively. The classification overlap between the DDH method and riboPicker/SortMeRNA was ≈7.3%/0.9%, respectively.

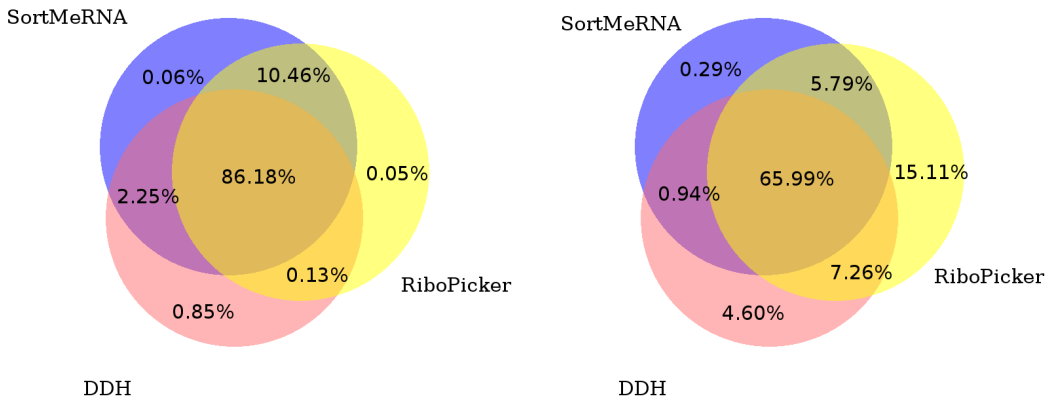


Figure 2 Venn diagrams showing the overlap of rRNA classification results for different methods on real-world metatranscriptome data. Left-hand side: Mushroom spring dataset, right-hand side: Tidal salt marsh dataset.

In comparison with our performance analysis on simulated data the results on the real-world datasets indicate a much larger disagreement of different methods than expected. This discrepancy, in principle, could already be seen in the evaluation of the SortMeRNA tool [11]. On one hand this indicates that the simulation setup that we used in a similar way like other researchers have done before, is too simple to capture the complexity of real metatranscriptomic data. On the other hand this shows the difficulty of measuring the rRNA detection performance in general and we have to admit that the assumption of a putative best method (HMMER3) is possibly not appropriate to tackle this problem.

3.1.1 Runtimes

Current next-generation sequencing methods yield a large number of sequencing reads with rapidly growing sizes of the resulting datasets. Therefore, the speed of an rRNA detection

Table 3 Runtimes of different methods on real-world metatranscriptome datasets.

dataset	method	time in sec
Mushroom Spring	DDH	5.7
	SortMeRNA	81.2
	riboPicker	363
	HMM	3016
Tidal salt marsh	DDH	12.1
	SortMeRNA	156
	riboPicker	591
	HMM	4487

method is an important aspect for the timely downstream analysis of the functionally relevant metatranscriptome data. To compare the runtimes of different detection methods, we performed all classification analyses for the real-world metatranscriptome datasets in single core mode on an Intel(R) Core(TM) i5-3470 CPU @ 3.20GHz with 32GB RAM. Here, the runtimes for HMMER3 and our DDH approach are aggregated over all (16S, 23S, Archaea, Bacteria) model/classifier evaluations.

As shown in table 3, our distance histogram approach is ≈13 times faster than the second-fastest method, SortMeRNA. The speed-up factor of the DDH method over riboPicker and HMMER3 ranged between ≈47 to 64 and 370 to 530, respectively. These numbers indicate the suitability of our approach for fast rRNA detection in very large datasets.

4 Discussion

We introduced a machine learning approach to the detection of ribosomal RNA in metatranscriptomic sequences. The utilized feature space is composed of frequencies of spacer lengths between pairs of dinucleotides. The corresponding dinculeotide distance histograms (DDH) provide a natural representation of mismatches and therefore can cope with a relatively high rate of sequencing errors. In our experiments we found that a maximum distance of approximately 10 nt, i.e. a feature space with at most 2500 dimensions, is sufficient to provide a good discrimination of rRNA from coding regions. In comparison with alignment-based methods the DDH implies a position independent analysis and therefore neglects some conserved position information that is present in rRNA sequences. As a consequence, our results indicate a slightly lower detection sensitivity and specificity. The advantage over the existing methods is the computational speed, which is more than 10 times higher than for the previously fastest method (SortMeRNA). In future work we will address some of the limitations that result from our current training setup where we utilize the LIBLINEAR SVM implementation. With this library we have to keep all training vectors in memory and therefore the number of examples is restricted to about 150,000 DDH feature vectors for 32GB RAM. Using regularized least squares training (see e.g. [8]), we will be able to substantially increase training sets and therefore more realistic training examples may be obtained from a large number of simulated sequencing reads. In particular, we expect a better representation of 23S-rRNA and a better coverage of the negative examples in terms of a more comprehensive sampling of coding regions.

Acknowledgements This work was supported by the Deutsche Forschungsgemeinschaft [grant numbers Me3138 (“Computational models for metatranscriptome analysis”) and Li2050 (“Machine learning methods for functional characterization of the peroxisome”)].

A Construction of training and test sets

To create training and test sets with similar sequence variability, we analyzed the original SILVA alignments regarding the variability of the alignment columns and assigned a score to each sequence that reflects its distance from the consensus sequence/profile. Given an alphabet $\mathcal{A} = \{\text{A,C,G,T,N,-}\}$ and a sequence $\mathcal{S}$, the score $\mathcal{T}_{k,j}$ for a symbol $k \in \mathcal{A}$ and an alignment position j is calculated by

$$\mathcal{T}_{k,j} = \begin{cases} 0, & \text{if } \frac{\sum_{i=1}^{m} \mathcal{S}_{i,j='-'}}{m} > 0.5 \\ log_{10}(\frac{\sum_{i=1}^{m} \mathcal{S}_{i,j=k}}{m}), & \text{else} \end{cases}$$

whereby m represents the number of sequences. The sequence score is then calculated as $score(\mathcal{S}) = \sum_{j=1}^{L} \mathcal{T}_{\mathcal{S}_{k,j}}$. As a result, a lower sequence score indicates a higher deviation from the consensus sequence. Sequences for the final training and test datasets were then sampled from the reference datasets subject to similarly distributed sequence scores.

References

1 S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *J Mol Biol*, 215(3):403–410, Oct 1990.

2 L. C. Carvalhais, P. G. Dennis, G. W. Tyson, and P. M. Schenk. Application of metatranscriptomics to soil environments. *J. Microbiol. Methods*, 91(2):246–251, Nov 2012.

3 S. R. Eddy. Accelerated Profile HMM Searches. *PLoS Comput. Biol.*, 7(10):e1002195, Oct 2011.

4 R. C. Edgar. Search and clustering orders of magnitude faster than BLAST. *Bioinformatics*, 26(19):2460–2461, Oct 2010.

5 R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A Library for Large Linear Classification. *J. Mach. Learn. Res.*, 9:1871–1874, June 2008.

6 J. A. Gilbert, D. Field, Y. Huang, R. Edwards, W. Li, P. Gilna, and I. Joint. Detection of large numbers of novel sequences in the metatranscriptomes of complex marine microbial communities. *PLoS ONE*, 3(8):e3042, 2008.

7 J. A. Gilbert and M. Hughes. Gene expression profiling: metatranscriptomics. *Methods Mol. Biol.*, 733:195–205, 2011.

8 K. J. Hoff, M. Tech, T. Lingner, R. Daniel, B. Morgenstern, and P. Meinicke. Gene prediction in metagenomic fragments: a large scale machine learning approach. *BMC Bioinformatics*, 9:217, 2008.

9 Y. Huang, P. Gilna, and W. Li. Identification of ribosomal RNA genes in metagenomic fragments. *Bioinformatics*, 25(10):1338–1340, May 2009.

10 M. Kanehisa, S. Goto, S. Kawashima, and A. Nakaya. The KEGG databases at GenomeNet. *Nucleic Acids Res.*, 30(1):42–46, Jan 2002.

11 E. Kopylova, L. Noe, and H. Touzet. SortMeRNA: fast and accurate filtering of ribosomal RNAs in metatranscriptomic data. *Bioinformatics*, 28(24):3211–3217, Dec 2012.

12 J. H. Lee, H. Yi, and J. Chun. rRNASelector: a computer program for selecting ribosomal RNA encoding sequences from metagenomic and metatranscriptomic shotgun libraries. *J. Microbiol.*, 49(4):689–691, Aug 2011.

13 B. Leis, A. Angelov, and W. Liebl. Screening and expression of genes from metagenomes. *Adv. Appl. Microbiol.*, 83:1–68, 2013.

14 T. Lingner and P. Meinicke. Remote homology detection based on oligomer distances. *Bioinformatics*, 22(18):2224–2231, Sep 2006.

15 W. Ludwig, O. Strunk, R. Westram, L. Richter, H. Meier, Yadhukumar, A. Buchner, T. Lai, S. Steppi, G. Jobb, W. Förster, I. Brettske, S. Gerber, A. W. Ginhart, O. Gross, S. Grumann, S. Hermann, R. Jost, A. König, T. Liss, R. Lüssmann, M. May, B. Nonhoff, B. Reichel, R. Strehlow, A. Stamatakis, N. Stuckmann, A. Vilbig, M. Lenke, T. Ludwig, A. Bode, and K. H. Schleifer. ARB: a software environment for sequence data. *Nucleic Acids Res. 25;32(4):1363-71*, 32(4):1363–1371, Feb 2004.

16 R. S. Poretsky, S. Gifford, J. Rinta-Kanto, M. Vila-Costa, and M. A. Moran. Analyzing gene expression from marine microbial communities using environmental transcriptomics. *J Vis Exp*, (24), 2009.

17 M. Punta, P. C. Coggill, R. Y. Eberhardt, J. Mistry, J. Tate, C. Boursnell, N. Pang, K. Forslund, G. Ceric, J. Clements, A. Heger, L. Holm, E. L. Sonnhammer, S. R. Eddy, A. Bateman, and R. D. Finn. The Pfam protein families database. *Nucleic Acids Res*, 40:D290–D301, Jan 2012.

18 C. Quast, E. Pruesse, P. Yilmaz, J. Gerken, T. Schweer, P. Yarza, J. Peplies, and F. O. Glöckner. The SILVA ribosomal RNA gene database project: improved data processing and web-based tools. *Nucleic Acids Res.*, 41(Database issue):D590–596, Jan 2013.

19 D. C. Richter, F. Ott, A. F. Auch, R. Schmid, and D. H. Huson. MetaSim: a sequencing simulator for genomics and metagenomics. *PLoS ONE*, 3(10):e3373, 2008.

20 R. Schmieder and R. Edwards. Quality control and preprocessing of metagenomic datasets. *Bioinformatics*, 27(6):863–864, Mar 2011.

21 R. Schmieder, Y. W. Lim, and R. Edwards. Identification and removal of ribosomal RNA sequences from metatranscriptomes. *Bioinformatics*, 28(3):433–435, Feb 2012.

22 H. J. Tripp, I. Hewson, S. Boyarsky, J. M. Stuart, and J. P. Zehr. Misannotations of rRNA can now generate 90% false positive protein matches in metatranscriptomic studies. *Nucleic Acids Res.*, 39(20):8792–8802, Nov 2011.

Utilization of ordinal response structures in classification with high-dimensional expression data

Andreas Leha*, **Klaus Jung**, and **Tim Beißbarth**

Department of Medical Statistics
University Medical Center Göttingen
Humboldtallee 32, D-37073 Göttingen
andreas.leha@med.uni-goettingen.de

Abstract

Molecular diagnosis or prediction of clinical treatment outcome based on high-throughput genomics data is a modern application of machine learning techniques for clinical problems. In practice, clinical parameters, such as patient health status or toxic reaction to therapy, are often measured on an ordinal scale (e.g. *good*, *fair*, *poor*).
Commonly, the prediction of ordinal end-points is treated as a multi-class classification problem, disregarding the ordering information contained in the response. This may result in a loss of prediction accuracy. Classical approaches to model ordinal response directly, including for instance the cumulative logit model, are typically not applicable to high-dimensional data.
We present *hierarchical twoing (hi2)*, a novel algorithm for classification of high-dimensional data into ordered categories. *hi2* combines the power of well-understood binary classification with ordinal response prediction.
A comparison of several approaches for ordinal classification on real world data as well as simulated data shows that classification algorithms especially designed to handle ordered categories fail to improve upon state-of-the-art non-ordinal classification algorithms. In general, the classification performance of an algorithm is dominated by its ability to deal with the high-dimensionality of the data. Only *hi2* outperforms its competitors in the case of moderate effects.

1998 ACM Subject Classification I.5.2 Design Methodology

Keywords and phrases Classification, High-Dimensional Data, Ordinal Response, Expression Data

Digital Object Identifier 10.4230/OASIcs.GCB.2013.90

1 Introduction

In the pursuit of personalized medicine there is increasing demand to classify patients individually based on molecular features. Therefore, classification methods which are capable to handle high-dimensional data from high-throughput omics data are needed. In clinical problems it is oftentimes desired to classify patients into ordered categories, because many clinically relevant parameters, such as patient health status or toxic reaction, are measured on ordinal scales. Examples include the *TNM-status* [30] or the *Acute Toxicity Grades* [11, 32].

Standard classification methods for high-throughput data can only handle categorical responses [13]. In practice these methods are typically applied after dichotomization of an ordinal therapy response parameter [18, 20]. Not to use the ordered structure, however,

* corresponding author

© Andreas Leha, Klaus Jung, and Tim Beißbarth;
licensed under Creative Commons License CC-BY
German Conference on Bioinformatics 2013 (GCB'13).
Editors: T. Beißbarth, M. Kollmar, A. Leha, B. Morgenstern, A.-K. Schultz, S. Waack, E. Wingender; pp. 90–100

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

can lead to sub-optimal classification results, as exploiting the information contained in the ordering can improve the classification performance [2, 16].

Another approach to address the classification into ordered categories is regression. In this approach, the levels of the response (e.g. *good*, *fair*, *poor*) are mapped to numerical numbers (e.g. 1, 0, -1) and a regression model is fitted. While mapping the levels to numbers preserves the order, this approach imposes additional structure, that might not be actually present, as it restricts the distances between the levels. However, if the number of levels gets large (e.g. the CMTNS with 37 levels [28]), this bias is comparably small.

Several ordinal classification algorithms have been proposed, e.g. the cumulative logit model or the continuation ratio model [1], which are typically not suited for high-dimensional problems.

The set of ordinal classification methods that are suitable for high-dimensional data is small. It includes rpartOrdinal [2] and its variant rpartScore [17], which both extend classification trees by several methods to split the nodes to favour classification preserving the ordinality. Archer and Williams [3] propose a second method based on continuation ratio models and L1 penalization. Ordinal extensions exist for the k-nearest neighbours classification [19] as well as for support vector machines [10].

In this regard we propose *hierarchical twoing (hi2)*, a classification scheme for ordinal classification, that takes the idea of twoing from Breiman et al. [5] which is also used by Frank and Hall [16]. Twoing is the idea to take all possible ways of splitting the data into two sets and of constructing the overall classifier out of binary classifiers based on these splits. Hereby, hi2 extends the method of Frank and Hall to a forest of hierarchical configurations. hi2 is a classification scheme, s.t. the choice of the dichotomous classification algorithm used internally is free and can be adapted to best suit the data at hand.

This paper is organized as follows: Section 2 details the proposed hi2 method and describes the alternatives. Different methods to evaluate the performance of classifiers with ordinal response are discussed as well. After that, in section 3 we present a comparative evaluation of different ordinal classification methods, in a simulated setting as well as applied to real data. Following a discussion on the results in section 4 the paper concludes with section 5.

2 Methods

In the following the response variable is denoted as $\mathbf{C}$ and can take one of p ordered values $C_1 < C_2 < \cdots < C_p$. If the response variable of a sample takes the value C_j we also speak of the sample being *in class* j or *in class* C_j.

2.1 Related Work

Our comparison contains a *nullmodel* for comparison. This nullmodel does not use any information from the features for classification, but only relies on the relative group frequencies in the training set. If, for example, 50% of all samples in the training set are in class 1, the nullmodel will classify an unseen sample into class 1 with a probability of 50%. The nullmodel is called *relfreq* in the remainder of this paper.

As a second 'benchmark' we trained standard support vector machines, which are known to handle high-dimensional data well [4]. SVMs fit a hyperplane in the feature space which best separates the samples from two groups. By means of a kernel function (the radial basis function kernel was used in this paper) the data are mapped into a higher-dimensional space to enhance the linear separability. In order to accommodate more than two classes a

all binary subclass comparisons are performed and a voting mechanism decides the overall classification result.

A third method, which will be called *limma+lda* throughout this paper, first does a feature selection to reduce the dimensionality of the data. To that end it performs an ANOVA per feature where internally the residual mean squares are moderated between the features [29]. The resulting p-values reflect an overall relation between a feature's expression profile and the response and are used to filter the features. The number of features to use is a crucial parameter in this algorithm and an inner cross validation tunes that number. The retained features are then used as predictors in a linear discriminant analysis (lda) [31]. The lda projects the data onto linear subspaces in a way that maximises the separation of the projected means of the classes while normalizing for the inner-class variance, such that the ratio of inter- and intra-variance is maximized

Also shown are results from ordinal classification trees, implemented in the R-package rpartOrdinal [2]. Classification trees select the splitting features during the tree construction, thus, a explicit feature selection is not necessary. Several splitting functions are proposed in that package. We present results based on the ordered twoing approach, which – although being computationally the most demanding one – performed best in our experiments. To accommodate for overfitting rpartOrdinal proposes a bagging approach where several classification trees are grown on bootstrapped samples and the majority vote of these trees is used as the overall classification result.

Frank and Hall [16] present a classification framework, that extends binary classification to the ordinal case. In the remainder of this paper we will refer to their method as Frank&Hall. In short, given a p-class problem, Frank&Hall trains $p-1$ binary classifiers and uses them to assign a class probability to each of the p ordinal classes $C_1, \ldots, C_p$ when an unseen sample is classified. The class with the highest probability is used as classification result. The probability of the first class (C_1) is simply $1 - Pr(\text{sample} > C_1)$ and depends only on a single binary classifier that distinguishes C_1 from the other values. Analogously, the probability of C_p is also computed using a single binary classifier as $Pr(\text{sample} > C_{p-1})$. The probability of the remaining classes C_j is $Pr(\text{sample} > C_{j-1}) - Pr(\text{sample} > C_j), j = 2, \ldots, p-1$ and therefore dependent on two binary classifiers. Thus, Frank&Hall present a classification framework and the user can plug in any binary classifier suitable for the data.

2.2 Hierarchical Twoing (hi2)

We propose hi2 as an extension of Frank&Hall. The classification result in Frank&Hall is dependent on maximally 2 classifiers and information from more distant classes is not considered directly. We propose to apply a hierarchical tree-like classification scheme, that recursively partitions the data into two-class problems, so that in hi2 the information from distant classes has a more direct impact on the local binary classification.

hi2 has two main modes: the *all data* mode and the *split data* mode.

2.2.1 All Data Mode

In the all data mode, hi2 trains the same classifiers as Frank&Hall but does the prediction in a hierarchical way: hi2 chooses one of the $p-1$ trained classifiers as the root classifier, e.g. it might choose the first classifier that separates class 1 from the rest. In the ordinal setting each binary classifier can only separate lower classes from higher classes, which we call the *left side* and *right side* of the classifier, respectively. The hierarchical scheme of hi2 now works recursively on both sides of the chosen classifier. In our example the left side

Table 1 The number of classification trees in hi2 grows with the number of ordinal classes in the classification problem following the *Catalan Numbers*. The table shows the number of trees that hi2 has to construct for classification problems up to size 10. While it is still feasible to apply hi2 for 10 class problems, computing time constrains the applicability to problems with more classes. That does not represent a real constraint, though, as with a growing number of classes, classification into ordinal classes approximates a regression problem.

	number of classes							
	3	4	5	6	7	8	9	10
number of trees	2	5	14	42	132	429	1430	4862

consists only of one class – class 1 – so the classifier is done here. The right side consists of the $p-2$ remaining classes, and hi2 again chooses one classifier. This is recursively repeated until all classifiers have been chosen. That way a classification tree is built. Dependent on which classifier are chosen after each other, that classification tree will have a different topology. hi2 builds all possible classification trees and takes a weighted majority vote as final classification result.

The weight of each tree is the classification performance of that tree on the training data measured by Kendall's τ (see 2.3 below).

As hi2 generates all possible classification trees and the number of classification trees is dependent on the number of classes, hi2 is not suited for problems with many (> 10) classes (Table 1). We consider that to be not a strong limitation, as when the number of classes gets large, regression methods usually yield good results. The number of classes is given by the *Catalan Numbers*:

$$C_q = \frac{\binom{2q}{q}}{q+1} = \frac{(2q)!}{q!(q+1)!}, \tag{1}$$

where $q = p - 1$: number of binary classifiers

2.2.2 Split Mode

In split mode, the training phase of hi2 also follows the hierarchical scheme. That means that also the training set is split into samples that are classified into the left side of a classifier and samples that are classified into the right side. The recursive training is then carried out on the reduced training set in both sides. This approach poses an additional computational burden, as many more binary classifiers have to be trained compared to the all data mode. We found that computational burden to be acceptable. But furthermore, the reduction of the training set in each recursion might lead to an increase situations, where training of a classifier is not possible any more. The minimum number of necessary samples is dependent on the chosen binary classification method. The more unbalanced the group sizes are, the more frequent is that situation. So, we regard this mode suitable for classification problems with many samples and close-to balanced group sizes only. Therefore, the results presented below are from the all data mode.

2.3 Methods of Evaluation

In order to compare different classification methods we need a measure to compare their performance. The most common measure to evaluate a classifier is the accuracy, i.e. the

fraction of the number of correctly classified samples by the number of available samples. An equivalent measure is the *misclassification error rate (MER)* where MER = 1 − accuracy. These measures are not suitable for the ordinal case, as they do not have a notion of different levels of mis-classification, but treat a classification result as either correct or wrong. In the ordinal setting there are different levels of mis-classification, as classifying a sample into a neighbouring class should be considered a better result than classifying it into a distant class.

Therefore, different measures to evaluate a classifier have been proposed. One proposal is to look at all pairwise comparisons and to refrain from an overall measure [22]. It is, however, inconvenient to not have one overall measure. Most alternative performance measures that do an overall evaluation are based on the non-parametric notion of concordant and discordant pairs [12]. Here, the classification result of a pair of samples is called *concordant* if the relative order of their class values is the same in the classification compared to the true values. If the relative order is reverse to the true values, the pair is called *discordant*. Kendall's correlation coefficient τ_b [21] is the most common evaluation method of these rank based methods. τ_b is defined as

$$\tau_b := \frac{n_c - n_d}{\sqrt{n_c + n_d - n_t^{(r)}}\sqrt{n_c + n_d - n_t^{(t)}}}, \tag{2}$$

where n_c : number of concordant pairs,
n_d : number of discordant pairs,
$n_t^{(r)}$: number of ties in the classification result only,
$n_t^{(t)}$: number of ties in the true values only

Pairs which have ties in both the classification result as well as the true value are ignored by Kendall's τ_b. It is possible to calculate confidence intervals for Kendall's τ_b [23, p. 78] which is the main reason why we used τ_b as performance measure in the remainder of this paper.

As τ_b works on pairs of samples, it can not be used to compare different classifiers using one sample only. Alternative measures, include the *minimum/maximum mean average error* [12] which works on the absolute distance of classes, where all classes are mapped to integers or the *ordinal classification index* [8] which has both, a rank-based component and a distance-based component.

3 Results

In all settings the (not ordinal) limma+lda showed a very strong performance. Thus, we used limma+lda as the binary plug-in classifier in Frank&Hall as well as in hi2.

All analysis were implemented and performed in the statistical programming framework R [25]. Org mode [27] was used as environment for reproducible research.

3.1 Simulation

A simulation study was conducted to assess the influence of different data characteristics on the classification performance.

We simulated gene expression data for 1000 genes from 90 patients – 60 patients in a training set and 30 patients in a test set. The expression data was simulated to follow a multivariate normal distribution with expectation vector $\boldsymbol{\mu} = \mathbf{0}$ and a block-structured covariance matrix Σ: 20 blocks containing 50 genes each are placed along the diagonal. Each

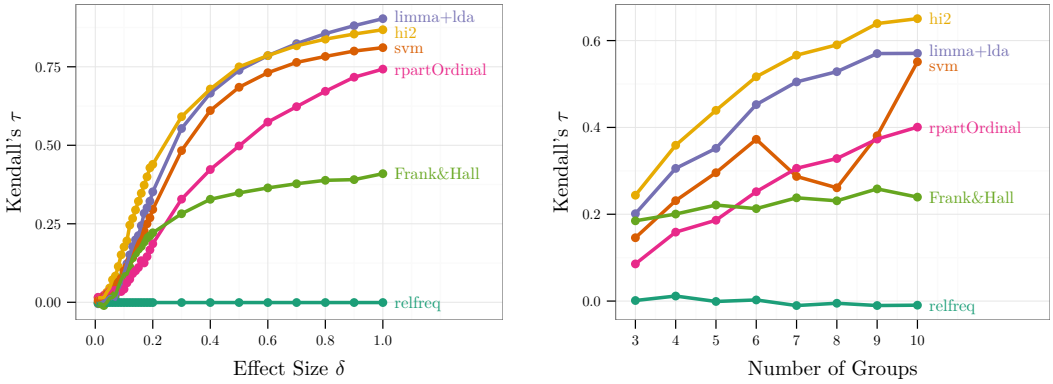


Figure 1 This Figure presents the results from a simulation study. Gene expression of 1000 genes was simulated for 90 patients belonging to 5 ordinal classes. 60 patients were used to train the classifiers and the remaining 30 patients formed the test set. Shown is Kendall's τ comparing the classification result (the predicted class) with the true value. Kendall's τ takes values between -1 (perfect negative correlation) and 1 (perfect correlation). The left panel (a) shows the results for different effect sizes δ. 50 genes cave been simulated to be differentially expressed across the 5 ordinal classes and δ is the level of differential expression. The right panel (b) shows results for a fixed effect $\delta = 0.2$ but different number of groups.

block has an autoregressive structure with parameter $\rho = 0.9$, i.e. the value is ρ^d where d is the unit distance to the diagonal. See the 'Autoblocks' panel in Figure 2a for a visualization.

In both groups, the training set and the test set, we simulated five ordinal groups of equal size. 50 randomly chosen genes were set to be differentially expressed following a linear trend pattern across the ordered groups with an effect size $\delta = 0.2$, i.e. the expectation of the expression for these 50 genes is $(j-1) * \delta$ for group $j, j = 1, \ldots, p$. All simulations were repeated 100 times.

In a first experiment the effect size δ was varied from 0 (no differentially expressed genes) to 1 (highly differentially expressed genes). Results are presented in Figure 1a. As expected all classification methods improve their performance with increasing effect size except the nullmodel which does not take the gene expression data into account. In the case $\delta = 0$ the performance of all classifiers is not better than guessing. Interestingly, the classification framework by Frank and Hall shows a much lower performance compared to the other methods and levels off at a moderately high effect size so that higher effect sizes do not lead to further improvements. The performance of rpartOrdinal, the second ordinal method under consideration, also does not match the other classifiers. Even the purely nominal methods svm and limma+lda perform better, where limma+lda again is the better choice. For small and moderate effect sizes, hi2 is the best classifier. Only for very large effect sizes, limma+lda again outperform hi2.

In a second experiment the effect size was fixed at $\delta = 0.2$ and the group size was varied between 3 (the smallest ordinal case) and 9. The aim of this experiment is to evaluate whether methods that exploit the ordinality of the response gain from more groups, as more groups can carry more ordinal information. We can observe, that the method Frank&Hall does not seem to gain from the presence of more groups. Also rpartOrdinal gains less from the increasing group number compared to hi2 or limma+lda. svm shows mixed results, as it takes advantage of more groups up until 6. For more groups, the performance shows a drop. But over all group numbers hi2 keeps the best performance.

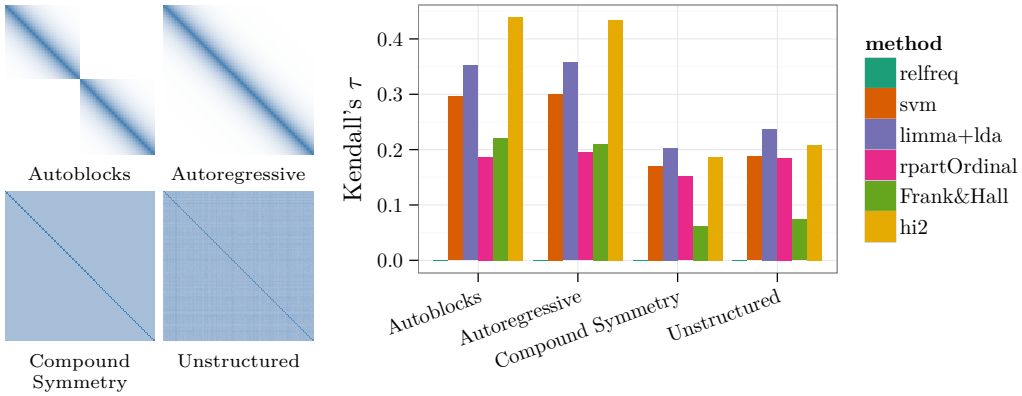


Figure 2 For the simulation study gene expression data of 1000 genes were simulated. We simulated the gene expression with different correlation structures. The left panel (a) shows the first 100 rows and columns of the used correlation matrices. The right panel (b) shows the behaviour of the classifiers under the different correlation structures. All classifiers have more difficulties in settings where all genes are correlated to each other. limma+lda and hi2 show the best performance across all settings.

The third experiment looks at different correlation structures. Besides the described block structure, a similar autoregressive structure without blocks, a compound symmetry with all values off the diagonal set to 0.5 and a random (unstructured) covariance have been simulated (see also Figure 2a). The settings with less correlation, namely autoblocks and autoregressive, are easier settings as all classifiers perform better in these two settings. Hi2 outperforms all other classifiers in these cases. Second best performs limma+lda followed by the svm. The ordinal method rpartScore and Frank&Hall perform similar, but are less potent compared to the others. In the settings with high correlation the difference between the methods is less pronounced. Limma+lda and hi2 change places and Frank&Hall has the most problem in these settings.

3.2 Analysis of a Data Examples

3.2.1 miRNA Expression in Breast Cancer

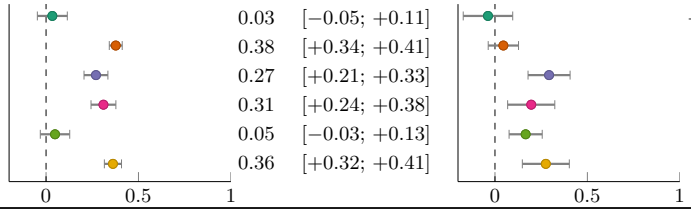
microRNAs have been shown to be important regulators of mRNA expression [9]. We analyzed a publicly available miRNA expression dataset [7] downloaded from the gene expression omnibus data base [14] (accession GSE22216). This data is part of a joint mRNA-miRNA analysis in 207 breast cancer patients. The annotation includes the tumor grade assigned following the modified method of Bloom and Richardson[15], which takes one of the values 1, 2, or 3. 42 patients have been assigned tumor grade 1, 87 and 65 have been assigned tumor grade 2 and 3, respectively. The annotation of the remaining patients was missing. The miRNA in that study had been measured using Illumina Human v1 MicroRNA expression beadchip which contains 735 miRNAs.

We present results from a 10-fold cross validation where the data was put into random order and split into 10 parts, each of which served in turn as test set, while the other 9 parts were used as training set.

On that setting (Table 2) the rpartOrdinal performs very well, but is still outperformed by hi2. The best classification, however, is delivered by the non-ordinal svm, but the confidence

Table 2 The classification algorithms under consideration have been applied to two publicly available datasets. This table shows their performance measured by Kendall's τ and includes the 95% confidence intervals. The dashed line in the visualizations marks the evaluation measure for a classification result that is completely uncorrelated to the truth. The left part of the table shows the results for miRNA expression data of 193 patients suffering from breast cancer split into 3 ordered groups. The right part shows results from mRNA data in 84 neuroblastoma patients of 5 ordered groups. The *nullmodel* relfreq performs consistently bad on both datasets. In contrast hi2 performs consistently strong on both datasets. The svm performs best on the miRNA data but surprisingly fails on the mRNA data.

	Breast Cancer (miRNA)		Neuroblastoma (mRNA)	
Method	τ	95% CI	τ	95% CI
relfreq	0.03	[−0.05; +0.11]	−0.04	[−0.17; +0.10]
svm	0.38	[+0.34; +0.41]	0.05	[−0.04; +0.13]
limma+lda	0.27	[+0.21; +0.33]	0.29	[+0.18; +0.41]
rpartOrdinal	0.31	[+0.24; +0.38]	0.20	[+0.07; +0.32]
Frank&Hall	0.05	[−0.03; +0.13]	0.17	[+0.08; +0.26]
hi2	0.36	[+0.32; +0.41]	0.28	[+0.15; +0.40]



intervals of the estimated performances overlap.

3.2.2 mRNA Expression in Neuroblastoma

As a second dataset an mRNA expression dataset [24] was downloaded from ArrayExpress [26] (identifier: E-TABM-38) and analyzed, again using a 10-fold cross validation scheme. This dataset includes mRNA expression levels for 10155 mRNAs from 251 patients suffering from neuroblastoma. 84 patients have been classified according to the International Neuroblastoma Staging System in its revised version [6] into one of the 5 classes $\{1, 2a, 2b, 3, 4\}$. The results from this dataset are presented in Table 2. While the svm performed best on the previous dataset, here it hardly outperforms the nullmodel. hi2 performs very strong again, and is only outperformed by limma+lda.

4 Discussion

Across all settings, the simulated ones and the real world data, hi2 shows a consistently strong performance: It performs best or second best result in all settings. We therefore consider hi2 a both good and safe choice for high-dimensional classification problems with ordered responses.

This stands in contrast to svm, for example, that is strong in some settings but fails completely on the mRNA data and also has problems in the simulation with 7-9 groups. We expect that better fine tuning of svm's parameters would help in these situations. But such fine tuning is not needed for hi2 with limma+lda as the used binary classifier. The main parameter of hi2 in this combination is the number of features to retain from limma and we propose to use an inner cross validation to determine that number.

limma+lda performs surprisingly strong even when it is applied on its own and not as part of the hi2 framework. When used on its own, the feature selection has information on all the groups and not only on a binarization of the grouping as in hi2. Thus, we take limma+lda's

strength as an indication that in these high-dimensional problems feature selection is a crucial step.

That point is supported by the observation that limma+lda even outperforms hi2 when the effect size δ between the groups is high. When the relevant features show a very strong effect, the task of selecting them gets easier and, thus, methods with a strong feature selection profit more than others. We also see a strong performance of limma+lda on the mRNA dataset which has many features. This again points to the importance of a good feature selection.

Another observation is that the performance of all classifiers drops considerably when we simulate features which are all strongly correlated. We take this as a hint that maybe other methods need to be used or developed that deal better with such highly correlated data. We target, however, data from mRNA or miRNA studies, where the correlation is expected to form blocks (e.g. higher correlation within pathways) where especially hi2 seems very well suited.

5 Conclusion

We presented a comparison of different classification methods applicable to high-dimensional data when the response lives on an ordinal scale. Both, simulated data and real data, have been used. The comparison includes the novel classification scheme hierarchical twoing (hi2), that performs consistently strong across all discussed settings, and seems especially strong in settings with small effects between the groups.

Acknowledgements This work was supported by the Deutsche Forschungsgemeinschaft (KFO 179) and by the German Federal Ministry of Education and Research (BMBF) within the NGFN network IG Prostate-Cancer(01GS0890) and within the Medical Systems Biology network BreastSys.

References

1 C V Ananth and D G Kleinbaum. Regression models for ordinal responses: a review of methods and applications. *International Journal of Epidemiology*, 26(6):1323–1333, 1997.

2 K J Archer. rpartordinal: An r package for deriving a classification tree for predicting an ordinal response. *J Stat Softw*, 34:7, Apr 2010.

3 K J Archer and A A A Williams. L 1 penalized continuation ratio models for ordinal response prediction using high-dimensional datasets. *Statistics in Medicine*, 2012.

4 Kristin P. Bennett and Colin Campbell. Support vector machines: hype or hallelujah? *SIGKDD Explor. Newsl.*, 2(2):1–13, December 2000.

5 L Breiman, J H Friedman, R A Olshen, and C J Stone. *Classification and Regression Trees*. Statistics/Probability Series. Wadsworth Publishing Company, Belmont, California, U.S.A., 1984.

6 GM Brodeur, J Pritchard, F Berthold, NL Carlsen, V Castel, RP Castelberry, B De Bernardi, AE Evans, M Favrot, and F Hedborg. Revisions of the international criteria for neuroblastoma diagnosis, staging, and response to treatment. *Journal of Clinical Oncology*, 11(8):1466–1477, 1993.

7 F M Buffa, C Camps, L Winchester, C E Snell, H E Gee, H Sheldon, M Taylor, A L Harris, and J Ragoussis. microrna-associated progression pathways and potential therapeutic targets identified by integrated mrna and microrna expression profiling in breast cancer. *Cancer Research*, 71(17):5635–5645, 2011.

8 J S Cardoso and R Sousa. Measuring the performance of ordinal classification. *International Journal of Pattern Recognition and Artificial Intelligence*, 25(08):1173–1195, December 2011.

9 T-C Chang and J T Mendell. microRNAs in vertebrate physiology and human disease. *Annual Review of Genomics and Human Genetics*, 8(1):215–239, 2007.

10 W Chu and S S Keerthi. Support vector ordinal regression. *Neural Computation*, 19(3): 792–815, Feb 2007.

11 J D Cox, J Stetz, and T F Pajak. Toxicity criteria of the radiation therapy oncology group (rtog) and the european organization for research and treatment of cancer (eortc). *International Journal of Radiation Oncology*Biology*Physics*, 31(5):1341 – 1346, 1995. Late Effects of Normal Tissues (LENT) Consensus Conference.

12 M Cruz-Ramirez, C Hervas-Martinez, J Sanchez-Monedero, and P A Gutierrez. A preliminary study of ordinal metrics to guide a multi-objective evolutionary algorithm. In *2011 11th International Conference on Intelligent Systems Design and Applications (ISDA)*, pages 1176–1181, 2011.

13 Sandrine Dudoit and Jane Fridlyand. Classification in microarray experiments. In Terry P. Speed, editor, *Statistical analysis of gene expression microarray data*, volume 1, pages 93–158. Chapman & Hall / CRC, New York, 2003.

14 R Edgar, M Domrachev, and A E Lash. Gene expression omnibus: NCBI gene expression and hybridization array data repository. *Nucleic Acids Research*, 30(1):207–210, January 2002.

15 C W Elston and I O Ellis. Pathological prognostic factors in breast cancer. i. the value of histological grade in breast cancer: experience from a large study with long-term follow-up. *Histopathology*, 19(5):403–410, November 1991.

16 E Frank and M Hall. A simple approach to ordinal classification. In *In: Proc 12th Europ Conf on Machine Learning*, pages 145–156. Springer, 2001.

17 G Galimberti, G Soffritti, and M Di Maso. Classification trees for ordinal responses in r: The rpartscore package. *Journal of Statistical Software*, 47(10), 5 2012.

18 B Michael Ghadimi, Marian Grade, Michael J Difilippantonio, Sudhir Varma, Richard Simon, Cristina Montagna, Laszlo Füzesi, Claus Langer, Heinz Becker, Torsten Liersch, and Thomas Ried. Effectiveness of gene expression profiling for response prediction of rectal adenocarcinomas to preoperative chemoradiotherapy. *Journal of clinical oncology: official journal of the American Society of Clinical Oncology*, 23(9):1826–1838, 2005.

19 K Hechenbichler and K Schliep. Weighted k-nearest-neighbor techniques and ordinal classification. In *Discussion Paper 399, SFB 386*, 2006.

20 Akihiro Ishizu, Utano Tomaru, Taichi Murai, Tomohiro Yamamoto, Tatsuya Atsumi, Takashi Yoshiki, Wako Yumura, Kunihiro Yamagata, Hidehiro Yamada, Shunichi Kumagai, Manae S. Kurokawa, Machi Suka, Hirofumi Makino, Shoichi Ozaki, and for JMAAV. Prediction of response to treatment by gene expression profiling of peripheral blood in patients with microscopic polyangiitis. *PLoS ONE*, 8(5), 2013.

21 M G Kendall. *Rank correlation methods*. Charles Griffin, London, London and High Wycombe, 1975. ISBN 0852641990.

22 S Natarajan, M McHenry, S Lipsitz, N Klar, and S Lipshultz. Agreement between two ratings with different ordinal scales. In N. Balakrishnan, Jean-Louis Auget, N. Balakrishnan, Mounir Mesbah, and Geert Molenberghs, editors, *Advances in Statistical Methods for the Health Sciences*, Statistics for Industry and Technology, pages 139–148. Birkhäuser Boston, 2007. ISBN 978-0-8176-4542-7.

23 G E Noether. *Elements of nonparametric statistics*. SIAM series in applied mathematics. Wiley, 1967.

24 A Oberthuer, F Berthold, P Warnat, B Hero, Y Kahlert, R Spitz, K Ernestus, R König, S Haas, R Eils, M Schwab, B Brors, F Westermann, and M Fischer. Customized oligonucleotide microarray gene expression-based classification of neuroblastoma patients outperforms current clinical risk stratification. *Journal of Clinical Oncology*, 24(31):5070–5078, 2006.

25 R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012. URL `http://www.R-project.org`.

26 G Rustici, N Kolesnikov, M Brandizi, T Burdett, M Dylag, I Emam, A Farne, E Hastings, J Ison, M Keays, N Kurbatova, J Malone, R Mani, A Mupo, R Pedro Pereira, E Pilicheva, J Rung, A Sharma, Y A Tang, T Ternent, A Tikhonov, D Welter, E Williams, A Brazma, H Parkinson, and U Sarkans. ArrayExpress update–trends in database growth and links to data analysis tools. *Nucleic Acids Research*, 41(D1):D987–D990, November 2012.

27 Eric Schulte, Dan Davison, Thomas Dye, and Carsten Dominik. A multi-language computing environment for literate programming and reproducible research. *Journal of Statistical Software*, 46(3):1–24, 1 2012.

28 M E Shy, J Blake, K Krajewski, D R Fuerst, M Laura, A F Hahn, J Li, R A Lewis, and M Reilly. Reliability and validity of the cmt neuropathy score as a measure of disability. *Neurology*, 64(7):1209–14, Apr 2005.

29 G K Smyth. Linear Models and Empirical Bayes Methods for Assessing Differential Expression in Microarray Experiments. *Statistical Applications in Genetics and Molecular Biology*, 3 : Iss. 1(3), 2004.

30 L H Sobin, M K Gospodarowicz, Ch Wittekind, and International Union against Cancer. *TNM classification of malignant tumours*. Wiley-Blackwell, Chichester, West Sussex, UK; Hoboken, NJ, 2010.

31 W N Venables and B D Ripley. *Modern Applied Statistics with S*. Statistics and Computing. Springer, 2002. ISBN 9780387954578.

32 H A Wolff, J Bosch, K Jung, T Overbeck, S Hennies, C Matthias, C F Hess, R M Roedel, and H Christiansen. High-grade acute organ toxicity as positive prognostic factor in primary radio(chemo)therapy for locally advanced, inoperable head and neck cancer. *Strahlentherapie und Onkologie*, 186(5):262–268, April 2010.

Extended Sunflower Hidden Markov Models for the recognition of homotypic *cis*-regulatory modules

Ioana M. Lemnian[1], Ralf Eggeling[1], and Ivo Grosse[1,2]

1 **Martin Luther University Halle-Wittenberg**
Institute of Computer Science
Von-Seckendorff-Platz 1, 06120 Halle, Germany
{lemnian, eggeling, grosse}@informatik.uni-halle.de

2 **German Centre for Integrative Biodiversity Research (iDiv) Halle-Jena-Leipzig**
Deutscher Platz 5e, 04103 Leipzig, Germany

Abstract

The transcription of genes is often regulated not only by transcription factors binding at single sites per promoter, but by the interplay of multiple copies of one or more transcription factors binding at multiple sites forming a *cis*-regulatory module. The computational recognition of *cis*-regulatory modules from ChIP-seq or other high-throughput data is crucial in modern life and medical sciences. A common type of *cis*-regulatory modules are homotypic clusters of binding sites, i.e., clusters of binding sites of one transcription factor. For their recognition the homotypic Sunflower Hidden Markov Model is a promising statistical model. However, this model neglects statistical dependences among nucleotides within binding sites and flanking regions, which makes it not well suited for *de-novo* motif discovery. Here, we propose an extension of this model that allows statistical dependences within binding sites, their reverse complements, and flanking regions. We study the efficacy of this extended homotypic Sunflower Hidden Markov Model based on ChIP-seq data from the Human ENCODE Project and find that it often outperforms the traditional homotypic Sunflower Hidden Markov Model.

1998 ACM Subject Classification J.3 Life and medical sciences

Keywords and phrases Hidden Markov Models, *cis*-regulatory modules, *de-novo* motif discovery

Digital Object Identifier 10.4230/OASIcs.GCB.2013.101

1 Introduction

The computational recognition of *cis*-regulatory modules (CRMs) is an important task in DNA sequence analysis. If the sequence motifs of the transcription factor binding sites (TFBSs) involved in putative CRMs are known, CRM recognition reduces to finding the composition of TFBS occurrences in a set of promoters or other unaligned sequences, and many methods exist for this task [13]. However, if the sequence motifs are unknown, CRM recognition becomes challenging, and reliable methods are still missing.

A promising model for CRMs is the Sunflower Hidden Markov Model (Sunflower HMM) proposed by Hoffmann and Birney [6], which allows multiple occurrences of TFBSs per sequence. However, the Sunflower HMM assumes statistical independence of the nucleotides within TFBSs and flanking regions, which limits its applicability to *de-novo* motif discovery. There is evidence about the presence of statistical dependences among adjacent nucleotides within TFBSs [9, 2, 1], and neglecting the dependences within flanking regions often leads to


© Ioana M. Lemnian, Ralf Eggeling, and Ivo Grosse;
licensed under Creative Commons License CC-BY
German Conference on Bioinformatics 2013 (GCB'13).
Editors: T. Beißbarth, M. Kollmar, A. Leha, B. Morgenstern, A.-K. Schultz, S. Waack, E. Wingender; pp. 101–109

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

the erroneous identification of repeats instead of putative TFBSs or to poor performance in the recognition of TFBSs [12].

Multiple TFBSs of the same transcription factor often build homotypic clusters, which we call homotypic CRMs. Such homotypic CRMs are frequent not only in invertebrates [8], but also in humans [4]. In this paper we focus on their recognition by a homotypic version of the Sunflower HMM, and by its extension that allows statistical dependences among adjacent nucleotides both within TFBSs and flanking regions.

The rest of the paper is structured as follows: in Section 2 we present the extended homotypic Sunflower HMM and corrresponding learning algorithms, and in Section 3 we study the efficacy of the extended homotypic Sunflower HMM in comparison to the traditional homotypic Sunflower HMM based on ChIP-seq data from the ENCODE project [11].

2 Extended homotypic Sunflower Hidden Markov Models

In the following two subsections, we introduce the extended homotypic Sunflower HMM and the Baum-Welch algorithm for estimating its model parameters. For the sake of convenience, we call the (traditional or extended) homotypic Sunflower HMM simply (traditional or extended) Sunflower HMM from now on.

2.1 Model

Consider a data set of N sequences $\underline{x}_1, \ldots, \underline{x}_N$, and denote the i-th sequence of length L_i by $\underline{x}_i = (x_{i,1}, \ldots, x_{i,L_i})$, where $i \in \{1, \ldots, N\}$. In analogy to the traditional Sunflower HMM, we define the probability of sequence $\underline{x}_i$ given model parameters π and $\underline{\phi}$ by

$$P(\underline{x}_i|\pi, \underline{\phi}) = \sum_{\underline{u}_i} P(\underline{u}_i|\pi) P(\underline{x}_i|\underline{u}_i, \underline{\phi}), \tag{1}$$

where $\underline{u}_i$ denotes a hidden path consisting of states $u_{i,j} \in \{m_1, \ldots, m_M, m_{\overline{1}}, \ldots, m_{\overline{M}}, f_1, f_2\}$, with M denoting the width of a putative TFBS. Here, π denotes the probability of a transition from a flanking region to a TFBS or its reverse complement, and $\underline{\phi}$ denotes all emission parameters of the model.

The states $u_{i,j}$ are indicator variables for TFBS occurrences in the following manner: $u_{i,j} = m_k$ indicates that $x_{i,j}$ is the k-th nucleotide of a TFBS on the forward strand, $u_{i,j} = m_{\overline{k}}$ indicates that $x_{i,j}$ is the k-th nucleotide (read in $3' \to 5'$ direction) of a TFBS on the reverse complementary strand, and $u_{i,j} = f_1$ and $u_{i,j} = f_2$ indicate that $x_{i,j}$ is part of the flanking region. State f_1 indicates the start position of a flanking region, while state f_2 indicates subsequent positions of a flanking region.

In analogy to the traditional Sunflower HMM, we define the probability of path $\underline{u}_i$ given model parameter π by

$$P(\underline{u}_i|\pi) = P(u_{i,1}|\pi) \cdot \prod_{j=2}^{L_i} P(u_{i,j}|u_{i,j-1}, \pi), \tag{2}$$

which states that the hidden path $\underline{u}_i$ is a realization of a homogeneous first-order Markov model.

The transition of one state to another is parameterized by a sparse transition matrix. There are three possible transitions from states f_1 and f_2: to m_1 with probability $\pi/2$, to $m_{\overline{M}}$ with probability $\pi/2$, and to f_2 with probability $1-\pi$. Here, we assume that the probabilities for the occurrence of a TFBS on the forward strand and on the reverse complementary strand

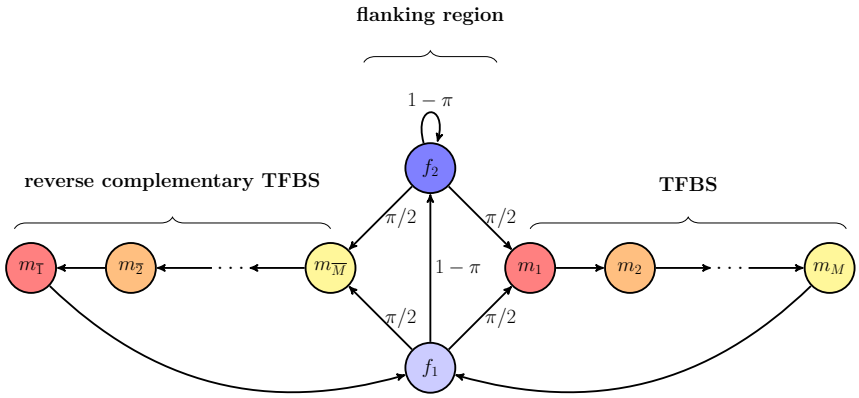


Figure 1 Graphical representation of the transition matrix of the extended Sunflower HMM. Circles denote states of the HMM. States f_1 and f_2 emit the flanking region, $m_1, m_2, \ldots, m_M$ emit the TFBS, and $m_{\overline{M}}, m_{\overline{M-1}}, \ldots, m_{\overline{1}}$ emit the reverse complementary TFBS. States in the same color correspond to the same position in the motif. Arrows represent transition probabilities between states with a probability greater than zero. The transition probability is either marked as a label of the corresponding arrow or is 1 in case of unlabeled arrows.

are equal. There are deterministic transitions from m_k to m_{k+1}, from $m_{\overline{k}}$ to $m_{\overline{k-1}}$, from m_M to f_1, and from $m_{\overline{M}}$ to f_1. All other transitions are forbidden, i.e., their propabilities are zero. The graphical representation of this transition matrix is shown in Figure 1.

In contrast to the traditional Sunflower HMM, we define the likelihood of sequence $\underline{x}_i$ given path $\underline{u}_i$ and model parameter $\underline{\phi}$ by

$$P(\underline{x}_i|\underline{u}_i, \underline{\phi}) = P(x_{i,1}|u_{i,1}, \underline{\phi}) \cdot \prod_{j=2}^{L_i} P(x_{i,j}|u_{i,j}, x_{i,j-1}, \underline{\phi}). \tag{3}$$

In the traditional Sunflower HMM, the conditional probabilities $P(x_{i,j}|u_{i,j}, x_{i,j-1}, \underline{\phi})$ from equation 3 are replaced by $P(x_{i,j}|u_{i,j}, \underline{\phi})$, which states that $x_{i,j}$ and $x_{i,j-1}$ are conditionally independent given $u_{i,j}$ and model parameter $\underline{\phi}$. This conditional independence of the traditional Sunflower HMM is responsible for the occasionally erroneous identification of repeats instead of putative TFBSs, and it is this conditional independence assumption that we drop in the extended Sunflower HMM.

Figure 2 shows the additional conditional dependences among adjacent nucleotides by red and blue arrows. Red arrows represent conditional dependences within TFBSs and within reverse complementary TFBSs, and blue arrows represent conditional dependences within flanking regions. We assume that nucleotides in TFBSs or reverse complementary TFBSs are independent of nucleotides in flanking regions and vice versa, so there are no arrows between TFBSs and flanking regions or between reverse complementary TFBSs and flanking regions.

We denote the probability of emitting symbol a in state f_1 by $\lambda_{1,a}$, the conditional probability of emitting nucleotide b in state f_2 given that the previous nucleotide is a by $\lambda_{2,a,b}$, and all of these model parameters by $\underline{\lambda}$. In analogy to $\underline{\lambda}$, we denote the probability of emitting nucleotide a in state m_1 by $\theta_{1,a}$, the conditional probability of emitting nucleotide b in state m_k given nucleotide a emitted by state m_{k-1} by $\theta_{k,a,b}$, for $k \in \{2, \ldots, M\}$, and all of these model parameters by $\underline{\theta}$. These parameters are equivalent to the parameters of the weight array model of [14], i.e., to those of an inhomogeneous first-order Markov model.

The emission probabilities of states $m_{\overline{1}}, \ldots, m_{\overline{M}}$ corresponding to the reverse complemen-

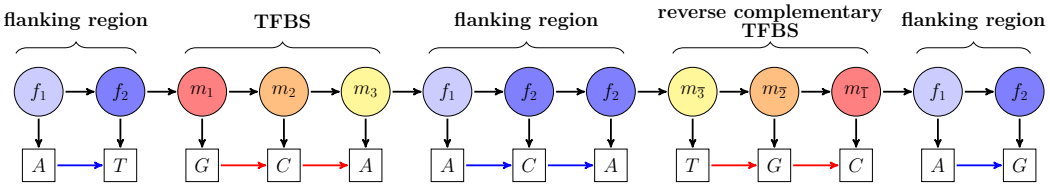


Figure 2 Extended Sunflower HMM for an example sequence of length 13 bp. The sequence contains a 3 bp long TFBS at positions 3-5 and its reverse complement at positions 9-11. The circles denote the values of the hidden states and the boxes the emitted nucleotides. Black arrows encode the dependences present in traditional Sunflower HMMs. Colored arrows encode the additional dependences modeled by the extended Sunflower HMMs: red arrows represent dependences within TFBSs, and blue arrows represent conditional dependences within flanking regions.

tary TFBS can be computed as a function of $\underline{\theta}$ by

$$\begin{aligned} P(x_{i,j} = a | u_{i,j} = m_{\overline{M}}, \underline{\theta}) &= \psi_{M,\overline{a}} \\ P(x_{i,j} = b | u_{i,j} = m_{\overline{k}}, x_{i,j-1} = a, \underline{\theta}) &= \frac{\theta_{k+1,\overline{b},\overline{a}} \cdot \psi_{k,\overline{b}}}{\psi_{k+1,\overline{a}}}, \end{aligned} \tag{4}$$

where $\overline{a}$ denotes the complementary nucleotide to a, and the auxiliary variables $\psi_{k,a}$ are given by the recursion

$$\begin{aligned} \psi_{1,a} &= \theta_{1,a} \\ \psi_{k,a} &= \sum_{b \in \{A,C,G,T\}} \theta_{k,b,a} \cdot \psi_{k-1,b}, \end{aligned} \tag{5}$$

where $k \in \{2, \ldots, M\}$.

We denote $\underline{\phi} = (\underline{\theta}, \underline{\lambda})$, and by plugging equations 2 and 3 into equation 1, we obtain the definition of the extended Sunflower HMM with model parameters π and $\underline{\phi}$.

2.2 Learning

In this section we describe how the model parameters π, $\underline{\theta}$, and $\underline{\lambda}$ of the extended Sunflower HMM can be learned and derive the corresponding Baum-Welch algorithm.

Model parameter π encodes the expected frequency with which TFBSs occur in a data set, and we allow the user to externally set this intuitive model parameter. Likewise, we treat the model parameter of the flanking regions $\underline{\lambda}$ as fixed, and we set it as maximum-likelihood estimator of a homogeneous first-order Markov model estimated from the entire data set. The reason for not learning $\underline{\lambda}$ via the Baum-Welch algorithm is that dynamically learning $\underline{\lambda}$ requires computing the sufficient statistics over almost the entire data set, which is unnecessarily time consuming given that the difference to the sufficient statistics of the full data set is only small, since the number of nucleotides in flanking regions and the number of nucleotides in the entire data set differ only slightly.

In analogy to [7] and many other *de-novo* motif discovery algorithms, we estimate the model parameter $\underline{\theta}$ by a maximum-likelihood approach. To this end, we derive a Baum-Welch algorithm [10] for the extended Sunflower HMM, which is a special case of the EM algorithm [3]. Formally, the Baum-Welch algorithm consists of two steps, which we will call E step and M step in analogy to the EM algorithm. The algorithm iterates between computing the expected sufficient statistics from the current values of the model parameters in the E step and computing the model parameters that maximize the log-likelihood of these expected values in the M step.

Here, we denote the conditional probability that nucleotide a is emitted in state m_1 or the complementary nucleotide $\overline{a}$ is emitted in state $m_{\overline{M}}$ by $\gamma_{1,a}$, and we denote the conditional probability that nucleotide b is emitted in state m_k given the previous nucleotide a or the reverse complementary nucleotide $\overline{b}$ is emitted in state $m_{\overline{k}}$ given the following nucleotide $\overline{a}$ by $\gamma_{k,a,b}$ for $k \in \{2, \ldots, M\}$. In the E step we compute $\gamma_{1,a}$ and $\gamma_{k,a,b}$ by using the current estimate of model parameter $\underline{\theta}^{(t)}$ by

$$\begin{aligned}
\gamma_{1,a}^{(t)} &= \sum_{i=1}^{N}\sum_{j=2}^{L_i} P(u_{i,j} = m_1 | \underline{x}_i, \underline{\theta}^{(t)}, \pi, \underline{\lambda}) \delta_{x_{i,j},a} \\
&\quad + \sum_{i=1}^{N}\sum_{j=2}^{L_i} P(u_{i,j} = m_{\overline{M}} | \underline{x}_i, \underline{\theta}^{(t)}, \pi, \underline{\lambda}) \delta_{x_{i,j},\overline{a}} \\
\gamma_{k,a,b}^{(t)} &= \sum_{i=1}^{N}\sum_{j=2}^{L_i} P(u_{i,j} = m_k | \underline{x}_i, \underline{\theta}^{(t)}, \pi, \underline{\lambda}) \delta_{x_{i,j-1},a} \delta_{x_{i,j},b} \\
&\quad + \sum_{i=1}^{N}\sum_{j=2}^{L_i} P(u_{i,j-1} = m_{\overline{k}} | \underline{x}_i, \underline{\theta}^{(t)}, \pi, \underline{\lambda}) \delta_{x_{i,j-1},\overline{b}} \delta_{x_{i,j},\overline{a}}.
\end{aligned} \tag{6}$$

In the M step we use the conditional probabilities from the E step to compute the next estimate of model parameter $\underline{\theta}^{(t+1)}$ by

$$\begin{aligned}
\theta_{1,a}^{(t+1)} &= \frac{\gamma_{1,a}^{(t)}}{\sum\limits_{a \in \{A,C,G,T\}} \gamma_{1,a}^{(t)}} \\
\theta_{k,a,b}^{(t+1)} &= \frac{\gamma_{k,a,b}^{(t)}}{\sum\limits_{b \in \{A,C,G,T\}} \gamma_{k,a,b}^{(t)}}.
\end{aligned} \tag{7}$$

For the computation of $P(u_{i,j} = k | \underline{x}_i, \underline{\theta}, \underline{\lambda})$ needed by each E step, we derive a Forward-Backward algorithm for the extended Sunflower HMM. First, we compute the forward variables $\alpha_{j,k}^i = P(x_{i,1}, \ldots, x_{i,j}, u_{i,j} = k | \underline{\theta}, \underline{\lambda})$ for $k \in \{m_1, \ldots, m_M, m_{\overline{1}}, \ldots, m_{\overline{M}}, f_1, f_2\}$ by the recursion

$$\begin{aligned}
\alpha_{1,k}^i &= P(x_{i,1} | u_{i,1} = k, \underline{\theta}, \underline{\lambda}) \delta_{k,f_1} \\
\alpha_{j,k}^i &= \sum_l \alpha_{j-1,l}^i P(u_{i,j} = k | u_{i,j-1} = l, \pi) P(x_{i,j} | u_{i,j} = k, x_{i,j-1}, \underline{\theta}, \underline{\lambda}).
\end{aligned} \tag{8}$$

Second, we compute the backward variables $\beta_{j,k}^i = P(x_{i,j+1}, \ldots, x_{i,L_i} | u_{i,j} = k, \underline{\theta}, \underline{\lambda})$ for $k \in \{m_1, \ldots, m_M, m_{\overline{1}}, \ldots, m_{\overline{M}}, f_1, f_2\}$ by the recursion

$$\begin{aligned}
\beta_{L_i,k}^i &= 1 \\
\beta_{j,k}^i &= \sum_l \beta_{j+1,l}^i P(u_{i,j+1} = l | u_{i,j} = k, \pi) P(x_{i,j+1} | u_{i,j+1} = l, x_{i,j}, \underline{\theta}, \underline{\lambda}).
\end{aligned} \tag{9}$$

Finally, we combine the forward and backward variables and obtain

$$P(u_{i,j} = k | \underline{x}_i, \underline{\theta}, \underline{\lambda}) = \frac{\alpha_{j,k}^i \beta_{j,k}^i}{\sum_k \alpha_{j,k}^i \beta_{j,k}^i}. \tag{10}$$

The Baum-Welch algorithm iterates the E step and the M step, yielding monotonically increasing log-likelihoods, and we terminate the algorithm when the difference of two subsequent log-likelihoods falls below $\varepsilon = 10^{-6}$. Typically, the algorithm reaches different local

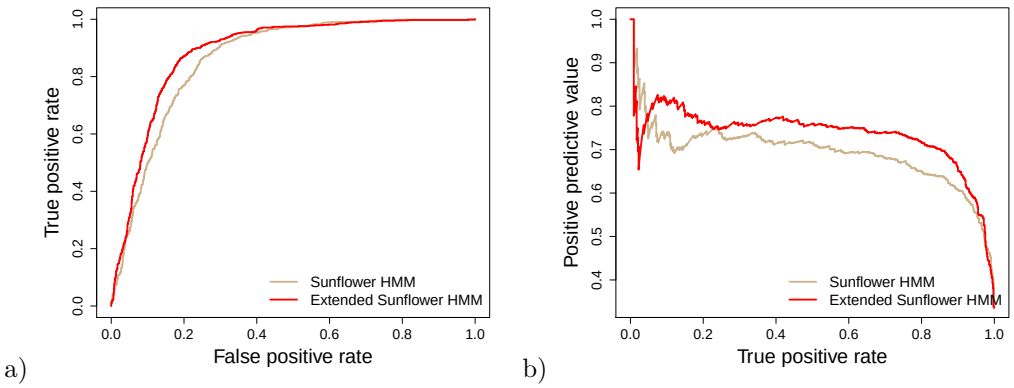


Figure 3 ROC curves (a) and PR curves (b) for the classification on the CREB1 data set. We find that in both cases the curves of classifier B, which uses the extended Sunflower HMM, lie above the curves of classifier A, which uses the traditional Sunflower HMM, except for recalls near zero, where the precisions of the traditional Sunflower HMM are greater than the precisions of the extended Sunflower HMM.

maxima or saddle points for different initializations, so we run it multiple times with different initializations and finally select the model parameter $\underline{\theta}$ with the highest log-likelihood.

3 Results

We have implemented the traditional and extended Sunflower HMMs including all algorithms in Java using Jstacs [5]. To assess the efficacy of the traditional and extended Sunflower HMM for the recognition of homotypic CRMs, we perform a classification of ChIP-seq positive versus negative regions based on data of human embryonic cells from the ENCODE project. We use the data for the six transcription factors CREB1, SP1, GABP, TEAD4, USF1, and YY1 from the HAIB TFBS track of the UCSC Genome Browser [1]. We select genomic regions covered by peaks with a score above 200 as positive sequences and the adjacent genomic regions of the same length as negative sequences. We split the positive and negative data sets for each transcription factors in two subsets, one for training and one for testing, at a ratio of 2:1.

We build two classifiers as follows: classifier A combines a traditional Sunflower HMM as foreground model for the positive sequences with a homogeneous Markov model of order 0 as background model for the negative sequences, while classifier B combines an extended Sunflower HMM for the positive sequences with a homogeneous Markov model of order 1 for the negative sequences.

For each transcription factor, we estimate the parameters of the homogeneous Markov models of order 0 and 1 by maximum likelihood from the union of positive and negative training data sets. We use these values as parameters of the background models of the classifiers and also as parameter $\underline{\lambda}$ of the Sunflower HMMs.

We learn the model parameters $\underline{\theta}$ on the positive data set by applying the traditional and extended Baum-Welch algorithm of Section 2.2. As a consequence, the only features discriminating between positive and negative sequences in each classifier are TFBSs in the

[1] `http://genome.ucsc.edu/cgi-bin/hgTables`

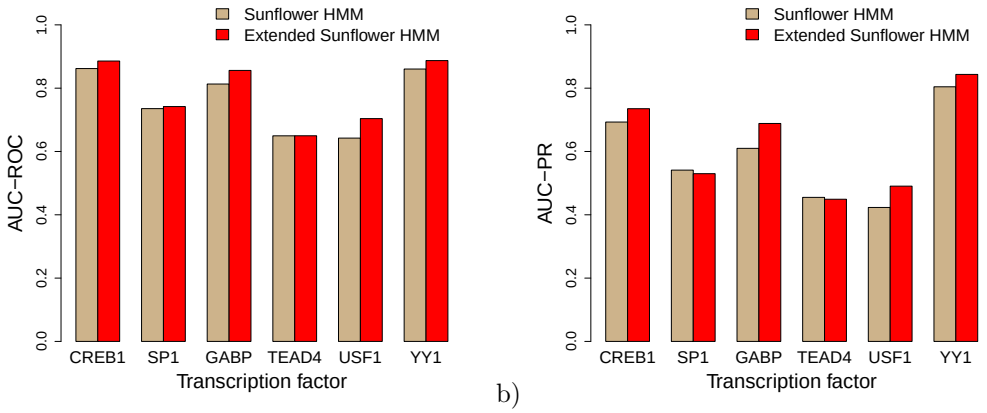


Figure 4 Classification results for six transcription factors. We classify the test data sets for CREB1, SP1, GABP, TEAD4, and USF1 using the two classifiers A and B trained on the training data sets. We show results of classifier A based on the traditional Sunflower HMM in light brown and those of classifier B based on the extended Sunflower HMM in red. Figure (a) shows the results in the area under the receiver operating characteristic curve (AUC-ROC), and Figure (b) shows the area under the precision-recall curve (AUC-PR).

foreground model. We may thus reason that a foreground model with a better classification performance has recognized TFBSs more accurately, so the classification performance may be regarded a measure of accuracy of recognition of homotypic CRMs from ChIP-seq data.

We compute the receiver operating characteristic (ROC) curves and the precision-recall (PR) curves for all six pairs of data sets. The ROC curve shows the true positive rate, also known as recall, as a function of false negative rate. The true positive rate is defined as the ratio of true positives and all positives. Analogously, the false positive rate is the ratio of false positives and all negatives. The PR curve shows the positive predictive value, also known as precision, as a function of the true positive rate. The positive predictive value is the ratio of true positives and the sum of true positives and false positives. Figure 3 shows the ROC and PR curves for the CREB1 data set. Both the ROC curves and the PR curves indicate that taking into account dependences among adjacent nucleotides leads to an improved recognition of CREB1 binding sites.

For calculating the overall classification performance we use the area under the ROC curve (AUC-ROC) and the area under the PR curve (AUC-PR). Figure 4 shows the AUC-ROC and AUC-PR values for both classifiers and each of the six transcription factors. The first pair of columns in each figure corresponds to the area under the curves shown in Figure 3. For CREB1, we observe that classifier B increases the AUC-ROC by 0.02 and the AUC-PR by 0.04 over classifier A. For the remaining transcription factors, we observe that the extended Sunflower HMM achieves higher AUC-ROC values and higher AUC-PR values than the traditional Sunflower HMM also for GABP, USF1, and YY1, whereas we do not observe an improved recognition of homotypic CRMs by taking into account dependences for SP1 and TEAD4.

4 Conclusions

In this work, we have extended the Sunflower HMM for homotypic CRMs by allowing statistical dependences among adjacent nucleotides within TFBSs and flanking regions. We

have derived a modified Baum-Welch algorithm including modified forward and backward algorithms, and we have found by case studies on ChIP-seq data that this extension improves the recognition of TFBSs for four out of six studied transcription factors.

However, this work is limited in several aspects. First, we have considered only one motif type and only first-order dependences. Second, the learning algorithm is based on the maximum likelihood principle, which neglects prior knowledge. Despite these limitations, the extended homotypic Sunflower HMM presented here might possibly be a useful starting point for the reliable recognition of CRMs. Further promising extensions could involve Bayesian or discriminative learning approaches or a generalization of the model to heterotypic CRMs, to higher-order nucleotide dependences.

Acknowledgements We thank Jesús Cerquides Bueno, Andreas Gogol-Döring, and Jan Grau for valuable discussions and DFG (grant no. GR 3526/2-1) and *Reisestipendium des allg. Stiftungsfonds der MLU Halle-Wittenberg* for financial support.

References

1 Yoseph Barash, Gal Elidan, Nir Friedman, and Tommy Kaplan. Modeling dependencies in protein-DNA binding sites. pages 28–37. ACM Press, 2003.

2 Martha L. Bulyk, Philip L. F. Johnson, and George M. Church. Nucleotides of transcription factor binding sites exert interdependent effects on the binding affinities of transcription factors. *Nucleic Acids Research*, 30(5):1255–1261, March 2002.

3 Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

4 Valer Gotea, Axel Visel, John M. Westlund, Marcelo A. Nobrega, Len A. Pennacchio, and Ivan Ovcharenko. Homotypic clusters of transcription factor binding sites are a key component of human promoters and enhancers. *Genome Research*, 20(5):565–577, May 2010.

5 Jan Grau, Jens Keilwagen, André Gohr, Berit Haldemann, Stefan Posch, and Ivo Grosse. Jstacs: A java framework for statistical analysis and classification of biological sequences. *Journal of Machine Learning Research*, (13):1967–1971, 2012.

6 Michael M. Hoffman and Ewan Birney. An effective model for natural selection in promoters. *Genome research*, 20(5):685–692, May 2010.

7 Charles E. Lawrence and Andrew A. Reilly. An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. *Proteins*, 7(1):41–51, 1990.

8 Alexander P. Lifanov, Vsevolod J. Makeev, Anna G. Nazina, and Dmitri A. Papatsenko. Homotypic regulatory clusters in Drosophila. *Genome Res*, 13(4):579–588, 2003.

9 Tsz-Kwong Man and Gary D. Stormo. Non-independence of Mnt repressor–operator interaction determined by a new quantitative multiple fluorescence relative affinity (QuMFRA) assay. *Nucleic Acids Research*, 29(12):2471–2478, June 2001.

10 Lawrence R. Rabiner and B. H. Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, 3(1):4–16, Jan 1986.

11 The ENCODE Project Consortium. Identification and analysis of functional elements in 1% of the human genome by the ENCODE pilot project. *Nature*, 447(7146):799–816, June 2007.

12 Gert Thijs, Magali Lescot, Kathleen Marchal, Stephane Rombauts, Bart De Moor, Pierre Rouzé, and Yves Moreau. A higher-order background model improves the detection of

promoter regulatory elements by Gibbs sampling. *Bioinformatics*, 17(12):1113–1122, December 2001.

13 Peter Van Loo and Peter Marynen. Computational methods for the detection of *cis*-regulatory modules. *Briefings in Bioinformatics*, 10(5):509–524, 2009.

14 M. Q. Zhang and T. G. Marr. A weight array method for splicing signal analysis. *Comput Appl Biosci*, 9(5):499–509, October 1993.

Avoiding Ambiguity and Assessing Uniqueness in Minisatellite Alignment

Benedikt Löwes and Robert Giegerich

Bielefeld University
Faculty of Technology and Center for Biotechnology
33501 Bielefeld, Germany
{bloewes,robert}@techfak.uni-bielefeld.de

Abstract

Several algorithms have been suggested for minisatellite alignment. Their time complexity is high—close to $O(n^3)$—due to the necessary reconstruction of duplication histories. We investigate the uniqueness of optimal alignments computed under the common single-copy duplication model. To this extent, it is necessary to avoid ambiguity in the algorithm employed. We re-code the ARLEM algorithm in the form of a grammar, and apply a disambiguation technique which uses a mapping to a canonical representation of minisatellite alignments. Having arrived at a non-ambiguous algorithm this way, we demonstrate that the underlying model—independent of the algorithm—gives rise to an exorbitant number of different, co-optimal alignments when applied to real-world data. We conclude that alignment-free methods should be considered for minisatellite comparison.

1998 ACM Subject Classification J.3.a [Life and Medical Sciences]: biology and genetics

Keywords and phrases minisatellite alignment, dynamic programming, ambiguity

Digital Object Identifier 10.4230/OASIcs.GCB.2013.110

1 Introduction

1.1 Background

The minisatellite comparison problem

Minisatellites are repetitive DNA sequences that have been used in population genetics and forensic studies [14,17,23]. They consist of short sequence motifs (6 – 100 bases), called units, which can spread over several kilobases as tandem repeats. The main mechanism behind their generation is unequal chromosomal crossover, which creates an extra copy of a unit in one chromosome, and a loss of such a unit in the other. In the course of evolution, units can pick up point mutations, which are then inherited by further copies. Thus, minisatellite alleles in a population differ not only in length, but also show microheterogeneities that make them useful—or even dangerous—genetic markers. For example, Jobling *et al.* have shown significant correlations between minisatellites on the human Y chromosome and the most common family names in England [16]. Recently, it has become of concern that these markers allow to re-personalize patient genome data [15].

Algorithms for minisatellite alignment

The minisatellite alignment problem has attracted substantial interest in bioinformatics. Analysis of highly repetitive sequences is difficult in general, as there is little information hidden in large data sets. Minisatellite sequences are compacted into "maps", i.e. sequences


© Benedikt Löwes and Robert Giegerich;
licensed under Creative Commons License CC-BY
German Conference on Bioinformatics 2013 (GCB'13).
Editors: T. Beißbarth, M. Kollmar, A. Leha, B. Morgenstern, A.-K. Schultz, S. Waack, E. Wingender; pp. 110–124

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

of characters, where each character denotes a unit that differs from others by point mutations. Compacting microsatellites, characterized by a shorter unit length, leads to the same representation. These maps are then aligned as one does with protein or DNA sequences, but placement and scoring of gaps receives special attention. A gap in a minisatellite alignment has the interpretation that at this point, the two alleles have undergone a divergent duplication history, and these duplication histories are reconstructed and scored in order to make optimal gap placements.

We only recall previous work that leads directly to our present approach. Bérard and Rivals proposed an $O(n^4)$ time and $O(n^3)$ space algorithm [6]. This was improved to $O(|\Sigma|n^3)$ and $O(|\Sigma|n^2)$ by Behzadi and Steyaert [5]. Here, Σ is the alphabet size, i.e. the number of different unit types. They also introduced run-length encoding, where a minisatellite map such as `aaabbbbbccacccc` is encoded as `a3b5c2a1c4`. Abouelhoda *et al.* gave an algorithm with a slightly more general model that runs in $O(n^3)$ time and $O(n^2)$ space, and also has a variant for run-length encoded maps [2]. This algorithm is the basis of the ARLEM web server[1] [1], and the starting point for the present work. A further improvement has been made by Pinhas *et al.* by using fast matrix multiplication techniques, achieving a running time of $O(\frac{|\Sigma|n^3 \log^3 \log n}{\log^2 n})$ [18]. While the underlying model of minisatellite evolution has become slightly more general along the way, the main concern of this line of work (including our own contributions) has been the improvement of efficiency. We do not pursue any further this intention here, but instead we raise the question: Should we align minisatellites at all?

1.2 Motivation of the present study

Significance of alignments

Alignment is a powerful method in biosequence analysis for two reasons: First of all, it gives us a quantitative measure of sequence similarity, which is taken as an indicator of evolutionary relatedness. But second, aside from such a distance or similarity score, the alignment from which this score is derived also gives us some qualitative information. It denotes homologous residues, elucidates sequence motifs more conserved than their context, and points out compensating base changes in RNA, which preserve secondary structure in the presence of sequence variation. There are many scenarios, however, where detailed information of the second type is not needed or considered too expensive to compute. In this case, alignment-free methods of sequence comparison are employed, such as k-mer profiles or word metrics [20, 24]. Hybrid methods are also common, the most prominent example being BLAST, where high-scoring segment pairs are found from k-mer matches, but in the output, sequences are aligned [3].

Statistical significance of alignments is commonly rated by P-values, but there is also the independent concern of uniqueness. Detailed information conveyed by an alignment is only dependable when the alignment is unique in the sense that there are no other alignments of similar score, or if so, they differ from the optimal one only in minor aspects. While it is common practice to speak of "the" optimal alignment or "the" minimum free energy structure found by dynamic programming, bioinformaticians are generally aware that the optimal answer to a combinatorial problem does not need to be unique, and co-optimal answers need not be similar. Although few programs do, it appears fair to ask a dynamic programming algorithm to report, together with a solution, the total number of co-optimal solutions (or even report all co-optimal answers on demand). In our case, we ask: How

[1] `http://www.nubios.nileu.edu.eg/tools/`, but currently unavailable due to political circumstances.

many co-optimal alignments are there for a given pair of minisatellites? How many different, but co-optimal duplication histories can be reconstructed? Only when the answer to either question is a very small number (such as 1), it makes sense to interpret the alignments in detail. However, there lies an intrinsic difficulty in this request, which is less widely known.

Ambiguity in combinatorial optimization

Dynamic programming algorithms used in biosequence analysis often explore a search space of exponential size in polynomial time. The candidates in the search space of the algorithm represent the features of our interest – homology assignments to residues, RNA structures, duplication histories. But they need not do so in a unique fashion. An algorithm may construct several candidates which *mean* the same feature of interest – this phenomenon is called *semantic* ambiguity. For example, with context-free grammars modeling RNA secondary structure, two different parses of an RNA sequence may indicate the same structure. This becomes apparent by mapping parse trees to dot-bracket strings, which constitute a canonical representation of structures. In a reasonable model, all candidates with the same meaning achieve the same score. If one of them is optimal, they all are. This is the crux of the question about co-optimal answers: The algorithm has no way to decide whether a large number of co-optimal candidates designates many different features of interest, or is merely a technical artifact of the algorithm. The meaning we associate with the candidates is not represented within the algorithm itself.

This problem has been defined formally and evaluated empirically, its undecidability was shown, and methods for testing or avoiding semantic ambiguity were suggested in [8, 9, 11, 12, 19]. We build on this work below when we eliminate semantic ambiguity from minisatellite alignment algorithms in order to correctly assess the uniqueness of their results.

1.3 Goals and preview of results

These are the goals and results of the present study:

- We raise the question: "How unique are minisatellite alignments constructed by the algorithms mentioned above?" and take the ARLEM algorithm as their representative. This question is relevant to decide whether the effort of constructing such alignments is well spent.
- We observe that the answer to this question is obscured by a high degree of semantic ambiguity in the ARLEM algorithm. We modify the algorithm such that it evaluates the same search space in a unambiguous fashion. The technique we use here is interesting in its own right.
- With the unambiguous algorithm we observe that the number of co-optimal alignments and duplication histories is extraordinarily large, leading to the conclusion that alignment-free methods should be considered as an alternative mode of minisatellite comparison.

Note that focusing on one particular algorithm does not limit the generality of our conclusion: While semantic ambiguity is a property of a particular algorithm, the question of result uniqueness is not: After weeding out semantic ambiguity, all algorithms implementing the same model must agree not only on the optimal score, but also on the number of co-optimal solutions. When ARLEM is plagued with co-optimals, so are all other algorithms implementing the same model. This stringency of the method employed here may render it useful also with other combinatorial optimization problems in biosequence analysis.

2 Abstracting the ARLEM algorithm

The ARLEM algorithm is given in [2] in the traditional form of dynamic programming recurrences, which makes it rather difficult to reason about ambiguity. We develop here a more abstract presentation of this algorithm in the form of a tree grammar. This representation is helpful to explain the underlying alignment and duplication history model, and serves as the starting point of disambiguation in the next section. Better than the recurrences, the grammar reflects the algorithm's division in two logically independent parts: (i) the alignment itself, with the well-known edit operations match, insertion and deletion, and (ii) the computation of duplication histories, which refer to units where the alleles have undergone a divergent development. Incorporated into the alignment, the duplication histories describe a more sophisticated gap model, which can be computed for either sequence individually.

The alignment model

Grammar ARLEM

Alignment

$$A^* \rightarrow \texttt{match}(a, A, b) \mid \texttt{lDupS}(\overleftarrow{L_o}, A) \mid \texttt{lDupT}(A, \overleftarrow{L_o}) \mid \texttt{rDup}(R_o, A, R_o) \mid \texttt{empty}$$

Duplication history

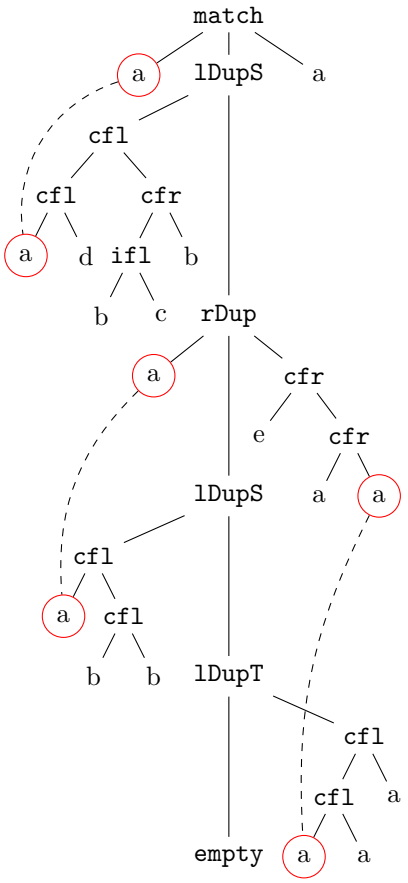
$$\begin{aligned} L_o &\rightarrow \texttt{cfl}(L_o, U) \mid \texttt{ifl}(L_o, U) \mid a \\ R_o &\rightarrow \texttt{cfr}(U, R_o) \mid \texttt{ifr}(U, R_o) \mid a \\ U &\rightarrow L \mid R \mid a \\ L &\rightarrow \texttt{cfl}(U, U) \mid \texttt{ifl}(U, U) \\ R &\rightarrow \texttt{cfr}(U, U) \mid \texttt{ifr}(U, U) \end{aligned}$$

The first rule of grammar ARLEM shows the classical edit distance model, adapted to the alignment of two minisatellite maps `S` and `T`. An asterisk marks the axiom of the grammar. It provides five edit operations: `match` (of two units in `S` and `T`), `lDupS` (left duplication in `S`), `lDupT` (left duplication in `T`), and `rDup` (right duplications in `S` and `T`).[2] Finally, `empty` represents the alignment of two empty minisatellites. The characters a and b are terminal symbols and denote arbitrary units from the unit alphabet. L_o and R_o denote duplications histories with their origin in the left/rightmost unit. They are further explained in the subgrammar for duplication histories. The left arrow indicates that the origin of duplication is one unit to the left and therefore also takes part in a previous `match` or `rDup` operation.

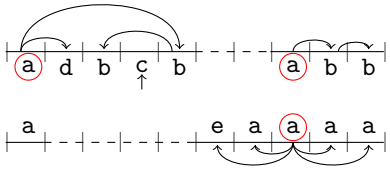
Duplication history model

A left (right) duplication is a sequence of units which originate from their leftmost (rightmost) unit. These sequences can be considered as gaps in the alignment. There is no need for duplication history rules with the origin in the middle of the sequence, since these cases can be constructed by simply using a right duplication followed by a left one. The `rDup` operation internally accounts for a match between the origin units of the two right duplications in `S` and `T`, where either right duplication may actually be empty. This asymmetric treatment of left and right duplications is a technical property of the ARLEM algorithm.

[2] Duplications in `S` could be called deletions, duplications in `T` insertions, but we avoid the directional bias of this terminology here.



(a) Tree structure showing sequence of used grammar rules



(b) Arched-arrow representation of the alignment

S	operations	T
a	(a) match a	**a**
a	lDupS	a
a**b**	(a) cfl b	a
a**d**b	(a) cfl d	a
ad**b**b	b cfr b	a
adb**c**b	b ifl c	a
adbcb**a**	(a) rDup (a)	a**a**
adbcba	e cfr (a)	a**e**a
adbcba	a cfr (a)	ae**a**a
adbcba	lDupS	aeaa
adbcba**b**	(a) cfl b	aeaa
adbcbab**b**	b cfl b	aeaa
adbcbabb	lDupT	aeaa
adbcbabb	(a) cfl a	aeaa**a**
adbcbabb	(a) cfl a	aeaa**a**a
adbcbabb	empty	aeaaaa

(c) Table showing the sequence of events together with the growing sequences of `S` and `T`.

Figure 1 Example of a minisatellite alignment of two sequences `S="adbcbabb"` and `T="aeaaaa"`. The red-circled units and dashed lines connect all units which occur twice, but in our implementation are only "read" once.

Duplication histories follow the single-copy model of ARLEM. They can be constructed for (sub)-sequences of `S` and `T` individually and use the following operations: `cfl` (copy from left) and `cfr` (copy from right) describe duplications from left to right (or vice versa) followed by an optional mutation of the new unit. The operations `ifl` (insertion from left) and `ifr` (insertion from right) seem syntactically similar, but describe an insertion of a single unit which is a foreign DNA fragment, most likely due to a transposition event. Since the unit is inserted, it has no unit of origin in the duplication history. But for the technical purpose of integrating this event in the duplication history, it takes a neighboring unit as its origin, but this unit does not contribute to the insertion score. In the grammar, nonterminals L_o and R_o mark histories that entail the left/right origin, whereas arbitrary histories are derived from nonterminal symbols L and R. Again, a is a terminal symbol and denotes an arbitrary unit.

Representation of alignments and histories

An example of an alignment with the incorporation of duplication histories is given in Figure 1. Figure 1(a) shows a tree-like representation of the alignment. Along the middle axis, we see the edit operations, noting that only two pairs of units are matched. The first is the

explicit `match` operator near the root of the tree, the second is implicit in the `rDup` operator. All others units arise from duplications, which branch from the stem in the tree towards the left for `S`, and towards the right for `T`, and are built from left and right duplication operators. Providing concrete scoring functions for the edit operations, such a tree can be directly evaluated to its alignment score. Figure 1(b) shows a graphical representation of an alignment and embedded duplication histories. Figure 1(c) shows a possible history of events that produces the two minisatellite sequences from a joint ancestor `aa`. (Note that one cannot reconstruct the exact order in time.)

Implementation in Bellman's GAP

A grammar like the above can be written in GAP-L, the language of the Bellman's GAP system [22], which generates recurrences automatically and functions as a programming system for Algebraic Dynamic Programming (ADP) [10, 13]. This was used to create a faithful emulation of the original ARLEM algorithm, and afterwards its dis-ambiguated version. Important for our intentions, Bellman's GAP allows to augment scoring schemes with "counting algebras", enabling us to evaluate the number of co-optimal solutions. This feature will be used in Section 4 to compare ARLEM with our dis-ambiguated version.

Simplifications

The implementation handles a number of subtleties that have been abstracted away in our presentation. Certain units take part in several edit operations. For instance, take the first left duplication in `S` from position 1 to 5 in Figure 1. The first unit takes part in the `match` operation as well as in a left duplication, as it is the leftmost unit and origin of a duplication history in S. So, the tree appears to hold this element twice. The dashed line connecting both instances indicates that in our implementation, the grammar "reads" this element just once. In the grammar, this is indicated by a red arrow above the nonterminal symbol.

The second problem arises while using the `cfl` and `cfr` operations of the duplication history part of the grammar. Consider the uppermost `cfl` operation in the tree of the first left duplication in `S` and the scenario that the score for the `cfl` operation is the sum of the scores of its two children plus the score of duplicating the unit `a` and mutating it to `b`. Here, it may be possible that there are two (sub-)trees with better scores than the ones of the uppermost `cfl` in Figure 1(a), but since this alignment is optimal, the mutation from the duplicated `a` to `c` has to score worse than the chosen one from `a` to `b`. In general, the optimization chooses the two best scoring (sub-)trees of `cfl` and `cfr` from the underlying sequence interval without considering the cost for the eventual duplication and mutation of the origin, which could lead to suboptimal results at this stage of the alignment and would therefore violate Bellman's Principle of Optimality. This problem is solved by a slightly different decomposition of duplication history intervals.

A third simplification concerns efficiency. The treatment of `rDup` in the grammar would lead to an $O(n^4)$ time algorithm. Using two separated operations like `rDupS` and `rDupT`[3], one can reduce the runtime to $O(n^3)$. This is a common speed-up technique in dynamic programming and is applied in ARLEM as well as in our Bellman's GAP implementations.

Finally, ARLEM allows inserted units to duplicate. We consider this an oversight in ARLEM, as foreign DNA inserts do not support unequal crossover. As this artificially blows up the search space, it will be disallowed in our next version of the algorithm.

[3] Both `rDupT` and `rDupS` include their origin (at the rightmost position), in contrast to `lDupS` and `lDupT`.

Table 1 Examples for the ambiguity of the ARLEM algorithm shown as tree structures as well as canonical representations of the alignments and the corresponding ambiguity type.

Type	Alignment tree structure	Canonical representation
1	$\texttt{match}(a, \texttt{lDupS}(\texttt{cfl}(a, b), \texttt{lDupT}(\texttt{empty}, \texttt{cfl}(a, c))), a)$	$a > (b) >$ $a > (c) >$
	$\texttt{match}(a, \texttt{lDupT}(\texttt{lDupS}(\texttt{cfl}(a, b), \texttt{empty}), \texttt{cfl}(a, c)), a)$	
	$\texttt{rDup}(a, \texttt{lDupS}(\texttt{cfl}(a, b), \texttt{lDupT}(\texttt{empty}, \texttt{cfl}(a, c))), a)$	
2	$\texttt{rDup}(e, \texttt{empty}, \texttt{cfr}(\texttt{cfl}(\texttt{cfr}(a, b), c), d))$	e $< ((a)b(c)) <$ d
	$\texttt{rDup}(e, \texttt{empty}, \texttt{cfr}(\texttt{cfr}(a, \texttt{cfl}(b, c)), d))$	
3	$\texttt{match}(a, \texttt{lDupS}(\texttt{cfl}(\texttt{ifl}(a, b), c), \texttt{empty}), a)$	$a > [b](c) >$ a
	$\texttt{match}(a, \texttt{lDupS}(\texttt{cfl}(a, \texttt{ifr}(b, c)), \texttt{empty}), a)$	$a > ([b]c) >$ a

3 Dis-ambiguation of the ARLEM algorithm

Semantic ambiguity in ARLEM

Running our emulation of the ARLEM algorithm, in addition to the original scoring scheme we employ a counting algebra to report the number of co-optimal alignments for each pair of minisatellites. Even for very short sequences, we obtain a large number of co-optimals. For example, aligning `adbcbabb` and `aeaaaa` from Figure 1 gives 1052 co-optimal answers.[4] Closer inspection shows that the same optimal alignment is returned several times. The algorithm is semantically ambiguous. For sources of ambiguity, consider Table 1.

1. Ambiguity type 1 shows three candidates that refer to the same "real" situation. Between the first two, the order of `lDupS` and `lDupT` is reversed, while in reality, these are individual duplication histories in `S` and `T` and do not happen in any order. The third entry uses the `rDup` operation with two empty histories, which is equivalent to a plain `match`.
2. Ambiguity type 2 shows a match between two `d` units, which is preceded by a right duplication in `T` showing a `b` unit originating from `d` and creating an `a` to the left and a `c` to the right. In the first example, the `a` is copied from `b` after the copy of `c`, while the second example reverses this order. This is within the duplication history of the same minisatellite. There is no evidence of which event happened first. Hence, we want to avoid reporting this situation twice.
3. Ambiguity type 3 shows an insertion of a unit `b`, which is modeled both as insertion from the left (unit `a`) and from the right (unit `c`). Remember that the insertion score depends neither on `a` nor on `c`. It is merely a technical requirement of the algorithm that the inserted unit has an origin, and hence one of the two cases should be ruled out.

These are just examples – note that there may be further sources of ambiguity!

Canonical alignment representation

Controlling ambiguity requires formalizing the "real world meaning" of the candidate alignments produced by the algorithm by means of a canonical representation [11], together with a semantic mapping μ that maps candidates to their meanings. For this mapping, we must show that $c \neq c'$ implies $\mu(c) \neq \mu(c')$ – then the algorithm is semantically unambiguous.

[4] For larger examples, numbers get so large that the counting algebra provided by Bellman's GAP overflows, and we had to use unlimited integers.

This is our canonical representation: Minisatellite alignments are represented as strings on two lines. Left duplications are enclosed in two "<"s, right duplications are enclosed in two "<"s. For empty right duplications, the "<"s may be omitted. Matched units of `S` and `T` are written below each other on the two lines; spaces have no meaning. For the duplication histories, we have to distinguish between the duplication (`cfl` and `cfr`) and insertion (`ifl` and `ifr`) operations. In case of duplications, the newly copied units are enclosed in parentheses ("()") and written to the right (left) of their origin. The same holds for insertions except that square brackets ("[]") are used. Examples of the canonical representation are given in the rightmost column of Table 1. Note that all the ambiguous cases there map to the same canonical representation, as intended. Formally, the canonical mapping μ is defined as follows:

Alignment

$$\mu(\texttt{lDupS}(S,A)) = \left\{\begin{matrix}\texttt{" > " ~+ } \sigma(S) \texttt{ ++ " > "}\end{matrix}\right\} \ddagger \mu(A)$$

$$\mu(\texttt{lDupT}(T,A)) = \left\{\begin{matrix}\texttt{" > " ~+ } \sigma(T) \texttt{ ++ " > "}\end{matrix}\right\} \ddagger \mu(A)$$

$$\mu(\texttt{rDup}(S,T,A)) = \left\{\begin{matrix}\texttt{" < " ++ } \sigma(S) \texttt{ +~ " < "} \\ \texttt{" < " ++ } \sigma(T) \texttt{ +~ " < "}\end{matrix}\right\} \ddagger \mu(A)$$

Duplication history

$$\sigma(\texttt{cfl}(A,B)) = \sigma(A) \texttt{ ++ "(" ++ } \sigma(B) \texttt{ ++ ")"}$$
$$\sigma(\texttt{ifl}(A,B)) = \sigma(A) \texttt{ ++ "[" ++ } \sigma(B) \texttt{ ++ "]"}$$
$$\sigma(\texttt{cfr}(A,B)) = \texttt{"(" ++ } \sigma(A) \texttt{ ++ ")" ++ } \sigma(B)$$
$$\sigma(\texttt{ifr}(A,B)) = \texttt{"[" ++ } \sigma(A) \texttt{ ++ "]" ++ } \sigma(B)$$
$$\sigma(a) = \texttt{"}a\texttt{"}$$

String concatenation operators

$\left\{\begin{matrix}A\\C\end{matrix}\right\}$	‡	$\left\{\begin{matrix}B\\D\end{matrix}\right\}$	=	$\left\{\begin{matrix}AB\\CD\end{matrix}\right\}$
A	++	B	=	AB
A	~+	xB	=	AB
A " < " x	+~	" < " B	=	AxB
A " < " $\tilde{C}x$	+~	" < " B	=	A " < " $\tilde{C}$ " < " xB

where capital letters A, B, C and D denote strings of length ≥ 0, x denotes a single unit and $\tilde{C}$ designates a string with length > 0 in which no "<" character is allowed.

Dis-ambiguating ARLEM

Grammar ARLEM-NONAMB

Alignment

$$\begin{aligned} A^* &\rightarrow \texttt{rDup}(R_o, B, R_o) \mid \texttt{empty} \\ B &\rightarrow \texttt{lDupS}(L_o, C) \mid C \\ C &\rightarrow \texttt{lDupT}(A, L_o) \mid A \end{aligned}$$

Duplication history

$$\begin{aligned} L_o &\rightarrow \texttt{cfl}(L,R) \mid L_i \\ L_i &\rightarrow \texttt{ifl}(L_i, a) \mid \texttt{ifl}(a,a) \\ L &\rightarrow L_o \mid a \\ R_o &\rightarrow \texttt{ifr}(a, R_o) \mid R_h \\ R_h &\rightarrow \texttt{cfr}(R, R_h) \mid a \\ R &\rightarrow \texttt{cfr}(R,R) \mid L \end{aligned}$$

The grammar ARLEM-NONAMB avoids the aforementioned types of ambiguity by the following means:

1. Type 1 ambiguity is avoided because the new grammar forces `lDupS` from nonterminal B before `lDupT` from nonterminal C, and the reverse order is not allowed. Also, we simply abandon the `match` operation, since it is covered by the more general `rDup`.
2. To avoid type 2, it is sufficient to restrict the left side of the `cfl` operation to the nonterminal L so that all `cfr` operations of one unit occur "before" the `cfl` operations.
3. We insist that the (merely technical) origin of an insertion is always on the left (unless it happens at the beginning of the minisatellite). If the direct left neighbor itself is inserted, the nearest left neighbor that is not inserted becomes origin of the `ifl` operation. This policy is reflected by the nonterminal L_i, whereas R_o holds the `ifr` operation in case the leftmost units of a right duplication have to be inserted. Finally, the nonterminal R_h is necessary to ensure that all right duplications starting with R_o have a right origin and L_o guarantees that a left duplication is at least of length two.

A proof is required to show that the grammar ARLEM-NONAMB is semantically unambiguous for the chosen canonical representation. We construct a context-free grammar that generates canonical representations in 1:1 correspondence with candidates constructed by ARLEM-NONAMB. Then the unambiguity of this grammar was proven with an automated ambiguity checker [8]. The full construction is included in Appendix B.

4 Assessing the co-optimal alignment search space

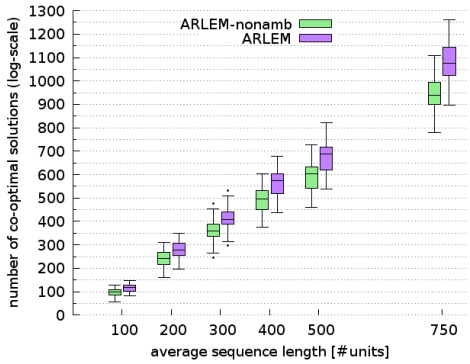
Using ARLEM-NONAMB, we investigate the number of co-optimal alignments, using three data sets: A set of random data (270 alignments), and the two "real-world" sets MSY1 and MS205 (59340 resp. 91806 alignments). See Appendix A for details. We use the same scoring system as in [2]. For comparison, we also describe the higher numbers reported by ARLEM due to its ambiguity. We find that while disambiguation has reduced the numbers reported, the co-optimal alignment space is still extremely large. See Figure 2. For sequences of average length 100, we find about 10^{100} co-optimal alignments, and 10^{500} for sequence length 400. The maps in the "real-world" data set are rather short, but still, 10^{20} co-optimal alignments discourage the idea that choosing a particular, optimal alignment for biological interpretation would be a meaningful effort.

We further investigated if the above situation might be alleviated by a version of the algorithm making use of run-length encoding. This has the effect that all duplication histories without modifications (a run) are considered equivalent and reported as one. While the counts are reduced further, an exponential pattern still prevails. See Appendix C for details.

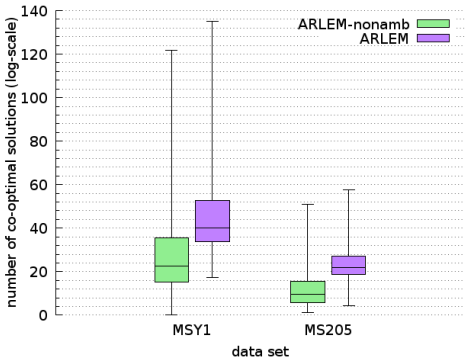
5 Conclusion

Dynamic programming algorithms as used in minisatellite alignment perform exact (rather than heuristic) combinatorial optimization. When such an algorithm is unambiguous and still reports a large number of co-optimal answers, the underlying problem is ill-posed. From our experiments with ARLEM-NONAMB, we conclude that the commonly used model of duplication histories lacks the distinctive power to designate a most plausible individual duplication history and pairwise minisatellite alignment.

A more fine-grained scoring scheme might produce sharper peaks for the same model. However, our own experiments with training `match`(a, A, b) from data sets has not led to an improvement. From this point, one could also move forward to advocate more sophisticated models of repeat evolution, such as the VNTR model in [21], where multiple copies are allowed and copies need not match unit boundaries. Ambiguity in such models has not yet



(a) Number of co-optimal alignments using random sequences



(b) Number of co-optimal alignments in MSY1 (length 48 – 114) and MS205 (length 23 – 75)

Figure 2 Numbers of co-optimal alignments reported by ARLEM-NONAMB and the emulated ARLEM algorithm. The y-axes show the logarithms ($\log_{10}$) of the actual values.

been addressed; a more elaborate model *may* lead to a smaller number of co-optimal results. Computational effort, however, will rise above $O(n^3)$.

We are not saying that minisatellite alignment methods have lost their merits. There may be data sets where alignments are unique. Observations like the directional bias of minisatellite growth, seen in [2], can only be made with an alignment method. Still, with large scale data and when only a similarity measure is required, we propose that minisatellite alignment should be abandoned in favor of alignment-free methods [20, 24]. These methods are based on k-mer composition and are often used for read clustering in meta-genomics. However, there is no off-the-shelf solution, as minisatellites are repetitive sequences and their k-mer profiles are dominated by a small number of k-mers that carry little information.

References

1 M. I. Abouelhoda, M. El-Kalioby, and R. Giegerich. WAMI: a web server for the analysis of minisatellite maps. *BMC Evolutionary Biology*, 10:167, 2010.

2 M. I. Abouelhoda, R. Giegerich, B. Behzadi, and J. M. Steyaert. Alignment of minisatellite maps based on run-length encoding scheme. *Journal of Bioinformatics and Computational Biology*, 7(2):287–308, April 2009.

3 S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, October 1990.

4 J. A. Armour, P. C. Harris, and A. J. Jeffreys. Allelic diversity at minisatellite MS205 (D16S309): evidence for polarized variability. *Human Molecular Genetics*, 2(8):1137–1145, August 1993.

5 B. Behzadi and J.-M. Steyaert. The Minisatellite Transformation Problem Revisited: A Run Length Encoded Approach. In Inge Jonassen and Junhyong Kim, editors, *WABI*, volume 3240 of *Lecture Notes in Computer Science*, pages 290–301. Springer, 2004.

6 S. Bérard and E. Rivals. Comparison of minisatellites. *Journal of Computational Biology*, 10(3-4):357–372, 2003.

7 N. Bouzekri, P. G. Taylor, M. F. Hammer, and M. A. Jobling. Novel mutation processes in the evolution of a haploid minisatellite, MSY1: array homogenization without homogenization. *Human Molecular Genetics*, 7(4):655–659, April 1998.

8 C. Brabrand, R. Giegerich, and A. Møller. Analyzing Ambiguity of Context-Free Grammars. *Science of Computer Programming*, 75(3):176–191, March 2010.

9 R. D. Dowell and S. R. Eddy. Evaluation of several lightweight stochastic context-free grammars for RNA secondary structure prediction. *BMC Bioinformatics*, 5:71, June 2004.

10 R. Giegerich. A systematic approach to dynamic programming in bioinformatics. *Bioinformatics*, 16(8):665–677, 2000.

11 R. Giegerich. Explaining and Controlling Ambiguity in Dynamic Programming. In Raffaele Giancarlo and David Sankoff, editors, *CPM*, volume 1848 of *Lecture Notes in Computer Science*, pages 46–59. Springer, 2000.

12 R. Giegerich and C. Höner zu Siederdissen. Semantics and ambiguity of stochastic RNA family models. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(2):499–516, 2011.

13 R. Giegerich, C. Meyer, and P. Steffen. A discipline of dynamic programming over sequence data. *Science of Computer Programming*, 51(3):215–263, June 2004.

14 P. Gill, A. J. Jeffreys, and D. J. Werrett. Forensic application of DNA 'fingerprints'. *Nature*, 318(6046):577–579, 1985.

15 M. Gymrek, A. L. McGuire, D. Golan, E. Halperin, and Y. Erlich. Identifying personal genomes by surname inference. *Science*, 339(6117):321–324, January 2013.

16 M. A. Jobling, N. Bouzekri, and P. G. Taylor. Hypervariable digital DNA codes for human paternal lineages: MVR-PCR at the Y-specific minisatellite, MSY1 (DYF155S1). *Human Molecular Genetics*, 7(4):643–653, April 1998.

17 M. A. Jobling and C. Tyler-Smith. The human Y chromosome: an evolutionary marker comes of age. *Nature Reviews Genetics*, 4(8):598–612, August 2003.

18 T. Pinhas, D. Tsur, S. Zakov, and M. Ziv-Ukelson. Edit Distance with Duplications and Contractions Revisited. In Raffaele Giancarlo and Giovanni Manzini, editors, *CPM*, volume 6661 of *Lecture Notes in Computer Science*, pages 441–454. Springer, 2011.

19 J. Reeder, P. Steffen, and R. Giegerich. Effective ambiguity checking in biosequence analysis. *BMC Bioinformatics*, 6:153, 2005.

20 G. Reinert, D. Chew, F. Sun, and M. S. Waterman. Alignment-free sequence comparison (I): statistics and power. *Journal of Computational Biology*, 16(12):1615–1634, December 2009.

21 M. Sammeth and J. Stoye. Comparing Tandem Repeats with Duplications and Excisions of Variable Degree. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(4):395–407, 2006.

22 G. Sauthoff, M. Möhl, S. Janssen, and R. Giegerich. Bellman's GAP–a language and compiler for dynamic programming in sequence analysis. *Bioinformatics*, 29(5):551–560, March 2013.

23 B. Sykes and C. Irven. Surnames and the Y chromosome. *American Journal of Human Genetics*, 66(4):1417–1419, April 2000.

24 S. Vinga and J. Almeida. Alignment-free sequence comparison-a review. *Bioinformatics*, 19(4):513–523, March 2003.

A Data sets used in evaluation

In the evaluation of the different algorithms, we used three data sets, a random one and two from human. We used random sequences of lengths 100, 200, 300, 400, 500 and 750 for the pairwise comparison of 10 sequences per length, resulting in 270 different alignments. In order to create some variance, the sequences were allowed to be 20% longer or shorter than the default length and the maximal repeat length of a unit was set to 25% of the complete length. The alphabet size of the 10 sequences per length was allowed to be in the range of 5% to 10% of the sequence length.

The MSY1 data set [7, 16] is based on a locus which lies on the human Y chromosome and shows sequence lengths varying between 48 and 114 units with a single unit having a length of 25 nt. Initially, five different unit types were identified, which differ only by few base substitutions at fixed sites. By using an extended approach, three additional unit types, which are mutated versions of the previous ones, could be identified, resulting in an alphabet size of 8. Overall, the data set consists of 345 distinct sequences which results in 59340 different pairwise alignments.

The locus of the MS205 data set [4] maps to the subtelomeric part of the short arm of the 16^{th} human chromosome. The complete length of the tandem repeat region is less than 5 kb and consists of 23 to 75 repeats with a unit length ranging from 45 to 54 bp. Overall, the data set consists of 429 different sequences with an alphabet size of only two, but with several sites of variation between the two types. This results in 91806 different pairwise alignments.

B Proof of unambiguity

Grammar ARLEM-NONAMB-STRING

Alignment

$$
\begin{array}{rcll}
A^* & \to & \text{" < "}\ R_o\ \text{" < "}\ B\ \text{" < "}\ R_o\ \text{" < "} & \mid \text{"\$"} \\
B & \to & \text{" > "}\ L_o\ \text{" > "}\ C & \mid C \\
C & \to & A\ \text{" > "}\ L_o\ \text{" > "} & \mid A
\end{array}
$$

Duplication history

$$
\begin{array}{rcll}
L_o & \to & L\ \text{"("}\ R\ \text{")"} & \mid L_i \\
L_i & \to & L_i\ \text{"["}\ U\ \text{"]"} & \mid U\ \text{"["}\ U\ \text{"]"} \\
L & \to & L_o & \mid U \\
R_o & \to & \text{"["}\ U\ \text{"]"}\ R_o & \mid R_h \\
R_h & \to & \text{"("}\ R\ \text{")"}\ R_h & \mid U \\
R & \to & \text{"("}\ R\ \text{")"}\ R & \mid L \\
U & \to & \text{"}a\text{"} \mid \ldots \mid \text{"}z\text{"} &
\end{array}
$$

In order to show that the (tree) grammar ARLEM-NONAMB is semantically unambiguous for the canonical representation μ, we transform the given grammar into a context-free (string) grammar ARLEM-NONAMB-STRING. Productions of both grammars are isomorphic, such that for a minisatellite alignment $t \in L(\text{ARLEM-NONAMB})$, $\mu(t)$ is generated by ARLEM-NONAMB-STRING by the same derivation. If ARLEM-NONAMB was semantically ambiguous, then ARLEM-NONAMB-STRING would be syntactically ambiguous. We use an ambiguity analyzer for context-free grammars [8] to prove that ARLEM-NONAMB-STRING is unambiguous, i.e. there is only one derivation for each valid canonical alignment string. This establishes the desired result.

A few technical problems must be solved before we can encode $\mu(L(\text{ARLEM-NONAMB}))$ as a context-free grammar. First, the two alignment strings for `S` and `T` are now written on a single line, separated by a "$" character. The string for `T` shows the alignment operations in reverse order, while between them, duplication histories are in their original order. For instance,

$$a > (b) > d$$
$$a > (a) > c$$

is now produced as

$$< a <> a(b) >< d < \$ < c <> a(a) >< a <,$$

and describes the alignment starting with a match of two `a` units, followed by a left duplication in `S` and `T`. It ends with a match between units `c` and `d`.

The example also shows two further deviations from the original canonical representation: right duplications include their origin within the <-brackets, rather than showing it outside to their right. Since for the sake of unambiguity, matches are treated as right duplications of length zero in ARLEM-NONAMB, the matched units are now shown as singletons between < and <.

With respect to left duplications, the string grammar is allowed to produce the same unit twice, first as a match between S and T and then again as the origin inside the duplication history. A match of unit `a` followed by a left duplication in S now reads

$$< a <> a(b) > \$ < a <$$

where the a in S actually exists only once. Still, this notation is in 1:1 correspondence with that generated by μ, although it is less readable for the human eye.

However, being context-free, the grammar also generates additional strings such as

$$< a <> c(b) > \$ < a <$$

with $a \neq c$. This does not correspond to a canonical representation generated by μ, so we have a language inclusion: $L(\text{ARLEM-NONAMB-STRING}) \supset \mu(L(\text{ARLEM-NONAMB}))$ modulo the aforementioned notation changes.

Syntactic ambiguity of context-free grammars, in general, is an undecidable problem. However, there are powerful semi-decision procedures. Finally, grammar ARLEM-NONAMB-STRING is submitted to the ACLA ambiguity checker at `http://www.brics.dk/grammar`, which confirms that it is unambiguous.

C Co-optimal alignment search space using run-length encoding

Presentation of new algorithms using run-length encoding

In addition to the two algorithms ARLEM and ARLEM-NONAMB, we used the idea introduced by Behzadi *et al.* [5] to incorporate run-length encoding (RLE) into the design of the algorithm. In Abouelhoda *et al.* [2], the initial goal was to use RLE to speed up the computation while ensuring that the optimal overall score is found. Speed plays only a minor role in our study; we want to use RLE to reduce the number of co-optimal alignments. We have realized this idea in two different ways. First, the algorithm RLE V1 uses the same approach as ARLEM-NONAMB, but with RLE sequences as input, such as `aaabbcddd` now represented as `a3b2c1d3`. Obviously, the basic algorithm ARLEM-NONAMB can be easily

Table 2 Scoring statistics of the algorithms RLE v1 and RLE v2 using the random, MSY1 and MS205 data set. Here, v1 and v2 are used synonymously for the scores of the algorithms RLE v1 and RLE v2.

	Random	**MSY1**	**MS205**
# alignments	270	59340	91806
v1 = v2	4%	34%	25%
mean $\frac{\mathbf{v1}}{\mathbf{v2}}$	1.04	1.20	1.64
max. $\frac{\mathbf{v1}}{\mathbf{v2}}$	1.29	7.66	7.66

modified to parse a unit as a character and an integer instead of just a character. Additionally, the scoring functions have to take the changes into account. For example, a run of length m scores as $m-1$ duplications, with no mutations involved. Overall, the basic approach of the algorithm remains unchanged. Taking advantage of the RLE input significantly speeds up the algorithm, but introduces the problem that we can no longer guarantee that the optimal alignment score will be found. This occurs when optimality requires to split runs, e.g. when aligning `a41` to `a20b1a20`.

Therefore, we tried a second approach, resulting in the algorithm RLE v2, to use the RLE idea and to guarantee that the optimal score can be found. Such an algorithm was already given by Abouelhoda *et al.*, but with the problem that it introduces yet another source of semantic ambiguity. A further issue is that this algorithm is not exclusively based on dynamic programming, but also makes us of combinatorics to compute the scores for the left/right duplications of the whole sequences from the scores of left/right duplications of the RLE sequences. In general, this approach uses the full-length sequences as input, but exploits the advantages of the RLE approach in the matrix recurrences. Despite these problems, we have also implemented RLE v1 and the slower but correct RLE v2 using the Bellman's GAP system and have eliminated the semantic ambiguity in similar fashion as before. In fact, our RLE v2 algorithm considers all duplication histories without modification equivalent and reports them as one.

Scoring statistics of RLE v1 versus RLE v2

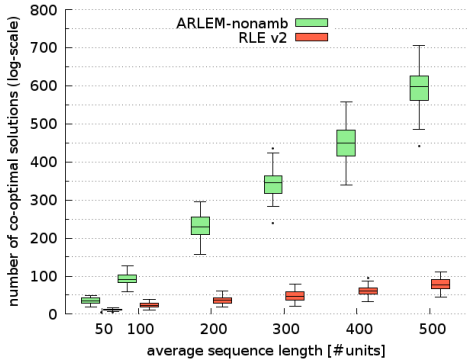
Using these two new grammars, we have investigated in what percentage the RLE v1 algorithm was able to compute the optimal alignment score. The result can be seen in Table 2. While the scores for the random sequences show no major differences between the two algorithms, the "real-word" data sets show large differences in cases of their maximum difference. In some cases the calculated score is not even near the optimum, which discourages the use of the RLE v1 algorithm.

Co-optimal alignments under run-length encoding

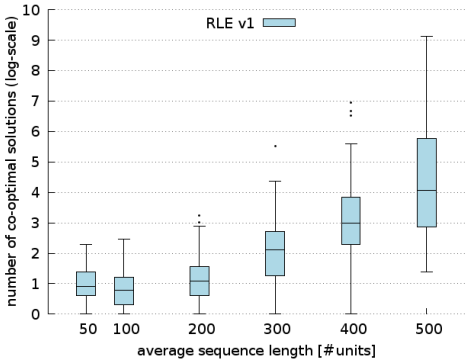
To evaluate if RLE reduces the number of co-optimal alignments significantly, see Figure 3.

For the RLE v2 algorithm, we observe that for random sequences of 300 units still 10^{50} different alignments exist. The same holds for the MSY1 and MS205 data sets where the median of RLE v2 is only a fraction smaller than the one of ARLEM-NONAMB. While the measured maximum value has decreased drastically, the median value is still high and discourages the use of this algorithm.

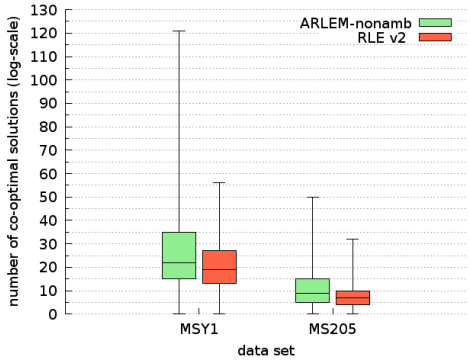
In contrast, the RLE v1 algorithm is characterized by a relatively small number of co-optimal alignments since the medians for the random sequences of lengths 200 and 300 are



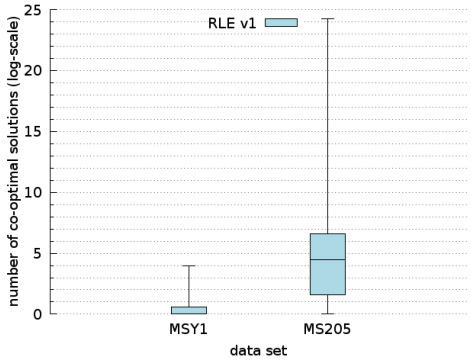
(a) Number of co-optimal alignments using random sequences and the ARLEM-NONAMB and RLE V2 algorithms



(b) Number of co-optimal alignments using random sequences and the RLE V1 algorithm



(c) Number of co-optimal alignments using the MSY1 and MS205 data sets and the ARLEM-NONAMB and RLE V2 algorithms



(d) Number of co-optimal alignments using the MSY1 and MS205 data sets and the RLE V1 algorithm

Figure 3 Numbers of co-optimal alignments reported by the ARLEM-NONAMB, RLE V1 and RLE V2 algorithm. The y-axes show the logarithms ($\log_{10}$) of the actual values.

about 10 and 200. This impression is confirmed when looking at the MSY1 data set, whereas the MS205 record shows a higher number of co-optimal alignments. This may be due to the small alphabet size of MS205, which inevitably leads to fewer different alignment scores.

In the end, the RLE V2 algorithm is not able to eliminate the exponential pattern of co-optimal alignments, while RLE V1 performs well in this regard, but strongly diverges from the optimal alignment score.

Aligning Flowgrams to DNA Sequences

Marcel Martin and Sven Rahmann

Genome Informatics, Institute of Human Genetics, Faculty of Medicine
University of Duisburg-Essen, Essen, Germany
marcel.martin@tu-dortmund.de, sven.rahmann@uni-due.de

Abstract

A read from 454 or Ion Torrent sequencers is natively represented as a *flowgram*, which is a sequence of pairs of a nucleotide and its (fractional) intensity. Recent work has focused on improving the accuracy of base calling (conversion of flowgrams to DNA sequences) in order to facilitate read mapping and downstream analysis of sequence variants. However, base calling always incurs a loss of information by discarding fractional intensity information. We argue that base calling can be avoided entirely by directly aligning the flowgrams to DNA sequences. We introduce an algorithm for *flowgram-string alignment* based on dynamic programming, but covering more cases than standard local or global sequence alignment. We also propose a scoring scheme that takes into account sequence variations (from substitutions, insertions, deletions) and sequencing errors (flow intensities contradicting the homopolymer length) separately. This allows to resolve fractional intensities, ambiguous homopolymer lengths and editing events at alignment time by choosing the most likely read sequence given both the nucleotide intensities and the reference sequence. We provide a proof-of-concept implementation and demonstrate the advantages of flowgram-string alignment compared to base-called alignments.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, J.3 Life and Medical Sciences

Keywords and phrases flowgram, sequencing, alignment algorithm, scoring scheme

Digital Object Identifier 10.4230/OASIcs.GCB.2013.125

1 Introduction

Pyrosequencing or Sequencing by Synthesis Pyrosequencing, also sequencing by synthesis, is a technology for DNA sequencing that does not sequence single nucleotides, but one run of nucleotides (*homopolymer*), at a time. There are two commercial sequencing technologies that use this approach: 454 (now owned by Roche) [10, 5] and Ion Torrent's "Ion semiconductor sequencing" (now owned by Life Technologies) [6].

Sequencing by synthesis starts with a single-stranded DNA template with an initial sequencing primer. Nucleotides of a single type are added and extend the primer if the next free bases on the template strand are complementary. The activity of the enzyme that catalyzes this reaction can be measured optically through its intermediate release of pyrophosphate (454). Alternatively, a change in pH value caused by the incorporation of nucleotides into the double strand can be measured directly with a semiconductor chip (Ion Torrent). The intensity of the measured signal is, in principle, proportional to the number of bases incorporated. All remaining free nucleotides are then removed and a different type of nucleotide is added. By cyclically flowing all four nucleotides and measuring the signal intensity, the sequence of the template DNA fragment can be reconstructed.

Using initial key sequences for each read, the signal intensities are normalized such that an intensity of 1.0 represents the incorporation of a single base. Due to the linearity of the


© Marcel Martin and Sven Rahmann;
licensed under Creative Commons License CC-BY

signal, 2.0 represents two bases and so on. This linearity of signal intensity can be maintained up to eight bases on a 454 system [5], but errors can occur at lower intensities.

The sequencing results from both mentioned technologies are natively output not as regular DNA sequences, but as so-called *flowgrams*, which associate each nucleotide homopolymer with its measured fractional intensity. It is therefore possible, for example, that a nucleotide homopolymer was measured at an "intensity of 2.4" (see Section 2).

Information Loss from Base Calling By rounding intensities to the nearest integer, a regular DNA sequence can be inferred (for a thymine at 2.4, this could be `TT` or `TTT`), a step known as *base calling*. Subsequently, standard read-mapping and sequence alignment algorithms can be used to compare the obtained sequence reads with reference sequences.

Recent work has focused on improving base calling from straightforward rounding to the nearest integer towards more elaborate statistical methods based on HMMs [3]. Nevertheless, base calling always incurs a loss of information by replacing the fractional intensity with a sequence of integer length. For example, the distinction between a `C` observed at an intensity of 5.4 vs. an intensity of 4.6 is lost. Both are called as `CCCCC`, but in the first case, alignment to six `C`s is much more plausible than in the latter case.

Previous Work Avoiding Base Calling We put forward the hypothesis that it makes more sense to invent alignment algorithms that directly work on flowgrams, instead of on a base-called sequence. A few publications on flowgram-based alignment already exist, but none clearly separates the two processes of sequence editing and flowgram under- and overcalling.

- Vacic et al. [12] model the distribution of flowgram intensities and derive a probabilistic model to compute the log-odds score that a given flowgram originates from a given genomic sequence. Their software FLAT is intended for mapping sequenced small RNA molecules to a reference and not for aligning diverged DNA sequences, so they do not take into account editing events. We use a similar way of deriving log-odds scores for differences between aligned reference and flow intensity.
- Quince et al. [9] use an algorithm adapted from global alignment [8] to align two flowgrams, first converting the reference sequence into flowspace. The authors' idea is to introduce gaps only in steps of four in order to take into account the cyclic nature of the flow order. The remaining description in the paper is brief, but one can deduce that a single flow is aligned to a homopolymer. It is unclear how editing is handled. The cost function used is $-\log P(f \mid \ell)$, where $P(f \mid \ell)$ is the probability of observing flow intensity f given a homopolymer of length ℓ.
- Lysholm et al. [4] propose a different method of aligning flowgrams, which is an extension of the Smith-Waterman local alignment algorithm [11] and can handle substitutions and indels with affine gap costs. FAAST's alignment is computed between the reference string and the base-called flowgram. Its modified scoring system reduces gap costs at points of uncertain homopolymer lengths.

Our Contributions In contrast to previous work, we do neither convert the reference into flowspace nor the flowgram to a string. Instead, we present the first algorithm that directly aligns a flowgram to a reference sequence, being aware of two processes in between: sequence editing between the reference and the (unknown) sequenced sample, and sequencing errors resulting in imprecise flow intensities.

After stating basic definitions (Section 2), we introduce a dynamic programming algorithm for optimal flowgram-string alignment (Section 3). The key component is a detailed scoring

scheme that models both sequence editing events and flow intensity measurement errors; it is described in detail in Section 4, where we also explain how the scoring parameters can be set to reasonable values. In Section 5, we demonstrate how flowgram-string alignment improves upon aligning a base-called sequence. A discussion and outlook on future work concludes the paper. A proof-of-concept implementation is available as a Python module from `http://www.rahmannlab.de/software`.

2 Basic Definitions and Ideas

Let Σ be the DNA alphabet. Let b^ℓ be the character $b \in \Sigma$ repeated ℓ times. Such a string is called a *homopolymer* of length ℓ.

The output of a 454 or Ion Torrent sequencer for a single read is a sequence of pairs of a nucleotide character and an intensity, called a flowgram [5], which we now define formally.

▶ **Definition 1** (Flow). A *flow* is a pair (b, f), where $b \in \Sigma$ is the *flow character* (or *flow nucleotide*) and $f \in \mathbb{R}_0^+$ is the *flow intensity*.

In analogy to exponentiation, we also write a single flow as $b_i^{f_i}$, that is, as the flow character followed by the flow intensity as a superscript. For example, instead of $(\mathtt{A}, 3.4)$, we write $\mathtt{A}^{3.4}$.

▶ **Definition 2** (Flowgram). A *flowgram* is a finite sequence $F = (F_1, F_2, \ldots, F_m)$ of flows $F_i = (b_i, f_i)$. The *flowgram length* is m. For $k = 1, \ldots, m-1$, we require that $b_i \neq b_{i+1}$.

The four nucleotides are typically added in repeating cycles. While our algorithm does not depend on it, we assume in the following that the order is $(\mathtt{T}, \mathtt{A}, \mathtt{C}, \mathtt{G}, \ldots)$, the typical order used in 454 instruments (that for Ion Torrent is different), and that the first flow character is always `T`. We may also say that a read is in *flowspace* to indicate that it is a flowgram.

Given flowgram $F = (F_1, \ldots, F_m)$, the sequence $F_{j\ldots k} := (F_j, \ldots, F_k)$ is a *subflowgram*. For $j = 1$, it is a *flowgram prefix*, and for $k = m$, it is a *flowgram suffix*.

▶ **Example 3.** A possible measured flowgram for the sequence `TTCGG` is $\mathtt{T}^{2.3}\mathtt{A}^{0.1}\mathtt{C}^{0.9}\mathtt{G}^{1.9}$.

The data output by both 454 and Ion Torrent sequencers contains flowgrams and uses the Standard Flowgram Format (`.sff`). The cycle order is stored once globally, and each flow intensity is stored as a 16-bit unsigned integer value and scaled such that a value of 100 represents an intensity of 1, thus allowing values from 0.00 up to $(2^{16} - 1)/100 = 655.35$.

▶ **Definition 4** (Canonical flowgram). Given a string s, the *canonical flowgram* for s is the flowgram that arises when we substitute all runs of character b of length n with the flow b^n and insert appropriate flows of intensity zero in between or in the beginning in order to get the correct order of nucleotides according to cycle order.

▶ **Example 5.** The canonical flowgram for `ACTT` (using cycle order `TACG`) is $\mathtt{T}^0\mathtt{A}^1\mathtt{C}^1\mathtt{G}^0\mathtt{T}^2$.

▶ **Definition 6** (Canonical DNA sequence). Given a flowgram F, let $\bar{F}$ be the flowgram with each intensity rounded to the nearest integer. The *canonical DNA sequence* for F is the DNA sequence which has the canonical flowgram $\bar{F}$, if it exists, and is undefined otherwise.

Note that a canonical DNA sequence for $F = \mathtt{T}^{1.1}\mathtt{A}^{0.1}\mathtt{C}^{0.4}\mathtt{G}^{0.2}\mathtt{T}^{2.3}$ does not exist, as rounding leads to $\bar{F} = \mathtt{T}^1\mathtt{A}^0\mathtt{C}^0\mathtt{G}^0\mathtt{T}^2$, but `TTT` has a canonical flowgram starting with $\mathtt{T}^3$. (A base caller that is cleverer than rounding [3] calls `TCTT` in this example instead of a non-existing canonical DNA sequence.)

reference r substring t	editing $\longrightarrow$	read/sample s homopolymer b^ℓ	sequencing $\longrightarrow$	flowgram F flow b^f
(known)		(unknown)		(observed)

Figure 1 Differences between an observed flowgram F and a reference sequence r arise from two different processes that cannot be distinguished by the observer: Sequence editing, responsible for differences between the sequenced sample and the reference in databases, and sequencing errors (intensity overcalls and undercalls) incurred during the sequencing process.

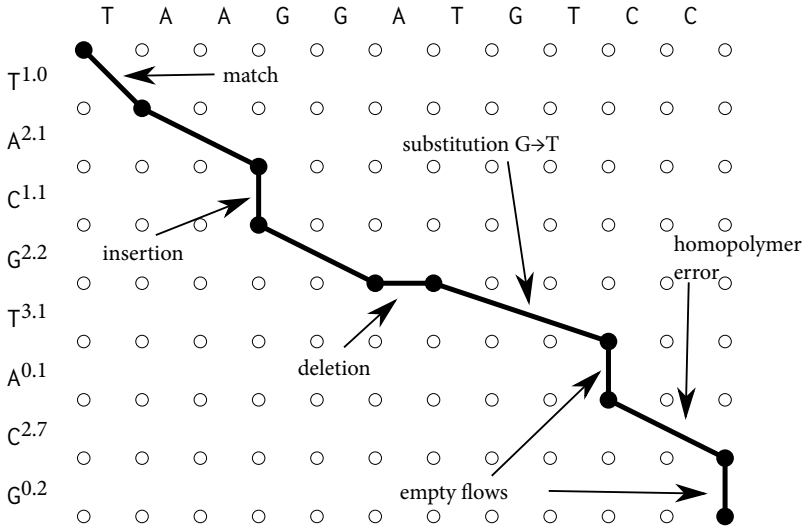


Figure 2 A visualization of the alignment of Example 8. The black path represents the alignment. The path may skip an arbitrary number of columns, but it cannot skip rows.

The problem just illustrated is sufficient motivation to find methods that skip base calling and directly align a flowgram to a reference sequence. Our main idea is to conceptually model a two-stage process (sequence editing, errors during sequencing) within one model and scoring function. It is best visualized with Figure 1.

We align a flowgram directly to a reference, without converting it to a string. Fractional intensities and ambiguous run lengths are resolved at alignment time by choosing the most likely read sequence given flowgram and reference sequence. In contrast to Vacic et al.'s work [12], we also model differences due to editing events. Every observed flow b^f must be explained by a substring t of the reference. The substring and the flow need not necessarily agree: If there is a non-b character in t, then there is a substitution or an insertion, and if the absolute difference between f and $|t|$ is sufficiently large, then there is an insertion or deletion event or a homopolymer error. Thus, a flow b^f can be explained as a sequenced homopolymer b^ℓ (where ℓ is integer), which in turn is an edited version of t (cf. Figure 1).

3 A Flowgram-String Alignment Algorithm

3.1 Alignments

▶ **Definition 7** (Flowgram-string alignment)**.** A *flowgram-string alignment* $\mathcal{F}$ between a flowgram F of length m and a string s of length n is a finite sequence of pairs $\mathcal{F} = (F'_i, t_i)$, where each F'_i is a flow or the space character $(-)$ such that the concatenation of all non-space

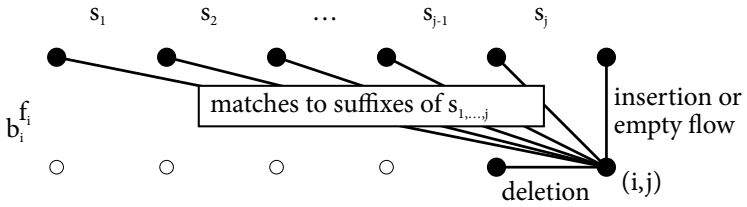


Figure 3 Visualization of different types of edges of the alignment graph and of the recurrence for cell (i, j) in the dynamic programming matrix.

F'_i is the flowgram F, and where the t_i are (possible empty) substrings of s such that their concatenation is equal to s.

▶ **Example 8.** Given are flowgram $F = \mathtt{T}^{1.0}\mathtt{A}^{2.1}\mathtt{C}^{1.1}\mathtt{G}^{2.2}\mathtt{T}^{3.1}\mathtt{A}^{0.1}\mathtt{C}^{2.7}\mathtt{G}^{0.2}$ and the string $s =$ TAAGGATGTCC. Using a notation in which F'_i is written above t_i, separating elements (F'_i, t_i) with a vertical line, a possible alignment is the following (see also Figure 2):

$\mathtt{T}^{1.0}$	$\mathtt{A}^{2.1}$	$\mathtt{C}^{1.1}$	$\mathtt{G}^{2.2}$	–	$\mathtt{T}^{3.1}$	$\mathtt{A}^{0.1}$	$\mathtt{C}^{2.7}$	$\mathtt{G}^{0.2}$
T	AA	ε	GG	A	TGT	ε	CC	ε

We see that flowgram-string alignment can describe all editing events: $\mathtt{T}^{3.1}$ aligned to TGT involves a mismatch (G instead of T); $\mathtt{C}^{1.1}$ aligned to ε means that there is an insertion; and the space aligned to an A is a deletion. We will also see below that, with the proper scoring function, flowgram-string alignment can distinguish between homopolymer errors and insertions. The scoring function will inform us whether the rightmost flow $\mathtt{G}^{0.2}$ aligned to an empty string ε of the reference needs to be interpreted as a homopolymer error of 0.2 or as an insertion. Our alignment algorithm picks the option with the better score.

A flowgram-string alignment describes how (1) editing events and (2) sequencing errors due to over- or undercalling add up to result in an observed flowgram. In contrast to previous flowgram alignment ideas, there is no need to convert the reference to a flowgram or to convert the flowgram into a string. Instead, a flowgram-string alignment describes a direct relationship between flowgram and reference.

3.2 The Flowgram-String Alignment Graph

A flowgram-string alignment can be interpreted as a path through a graph of $(m+1) \times (n+1)$ vertices $(i, j) \in \{0, \ldots, m\} \times \{0, \ldots, n\}$ that has different types of edges with different scores (see Figures 2 and 3). The score of an alignment $\mathcal{F}$ is the sum of the scores of the individual edges used by the alignment, and so finding the optimal alignment is equivalent to finding a highest-scoring path. The following two edge type exist:

- horizontal edges that connect (i, j) to $(i, j-1)$. These represent deletions, i.e., the flowgram indicates that a nucleotide from the reference is missing in the sample. The score $del < 0$ is assigned to these edges.
- vertical and diagonal edges that connect (i, j) to $(i-1, j-k)$ for all $k \in \{0, \ldots, j\}$. For $k = 0$, the edge is vertical and interpreted as either an empty flow aligned to an empty substring, or as an insertion, where the flowgram indicates a homopolymer not present in the reference. These edges use a complex scoring function $v(b, f, t)$ for aligning flow b^f to substring $t = s_{k+1\ldots j}$. This scoring function is central to our method and discussed in detail in Section 4.

3.3 Recurrence

Let (b, f) be a flow and $t \in \Sigma^*$ a string. We assume that scoring parameter del and scoring function $v(b, f, t)$ are available (see Section 4).

Let $S(i, j)$ be the optimal score between the length-i prefix of flowgram F of total length m and the length-j prefix of string s of total length $n = |s|$. The recurrence for $S(i, j)$ follows from the structure of the alignment graph, in which the optimal flowgram-string alignment is a highest-scoring path, analogously to standard global alignment. Other variants (local, free end gaps, etc.) are possible; for ease of exposition, we focus on the global case. We have

$$\begin{aligned} S(0, j) &= j \cdot del, \\ S(i, 0) &= \sum_{k=1}^{i} v(b_k, f_k, \varepsilon), \\ S(i, j) &= \max \left\{ \begin{array}{l} S(i, j-1) + del, \\ \max\limits_{k=0,\ldots,j} \left(S(i-1, k) + v(b_i, f_i, s_{k+1\ldots j}) \right) \end{array} \right\}. \end{aligned} \tag{1}$$

The two cases for $S(i, j)$ correspond to the two types of edges. The inner maximization corresponds to the vertical and diagonal edges, in which the score of aligning the current flowgram to all suffixes of $s_{1\ldots j}$ (including the empty suffix for case $k = j$) is found. It is the main difference to regular global alignment. With dynamic programming, $S(m, n)$ can be computed in time $\mathcal{O}(mn^2)$, assuming v can be evaluated in constant time.

4 Scoring

The score $v(b, f, t)$ for pairing flow (b, f) with string t must take into account two different processes that cannot be distinguished by an observer (Figure 1). First, editing events occur that change a substring t of the reference into b^ℓ, but ℓ is unknown. Second, an intensity f is measured for b^ℓ. We score the first process by $s_{\text{edit}}(b, \ell, t)$, which is the score of an optimal alignment between t and b^ℓ. The score $\sigma(f, \ell)$ is assigned to measuring intensity f for a homopolymer run of length ℓ; we assume that it does not depend on the nucleotide b.

Since ℓ is unknown, to obtain $v(b, f, t)$ we maximize over all possible lengths in order to pick the most plausible explanation:

$$v(b, f, t) := \max_{\ell=0,1,2,\ldots} \left(s_{\text{edit}}(b, \ell, t) + \sigma(f, \ell) \right) \tag{2}$$

As we will see in the two following subsections, this potentially infinite maximization is in fact finite, since a value of $\ell \gg \max\{|t|, f\}$ will yield a strongly negative score in both terms and cannot achieve the maximum. In practice, positive scores are only obtained if $f \approx |t|$ for a choice of ℓ close to both f and $|t|$.

To reconstruct the most plausible process to flow b^f via homopolymer b^ℓ from sequence t, we also store the value of ℓ maximizing $v(b, f, t)$ in (2),

$$L(b, f, t) := \operatorname{argmax}_{\ell=0,1,2,\ldots} \left(s_{\text{edit}}(b, \ell, t) + \sigma(f, \ell) \right). \tag{3}$$

It is this (unknown but inferred) value of $\ell = L(b, f, t)$ that links the two processes of sequence editing and sequencing shown in Figure 1.

As we will show, the score $v(b, f, t)$, and hence $L(b, f, t)$, depends on b and t only through the number $e = e(t, b)$ of characters in t that are equal to b and the number $\bar{e} = \bar{e}(t, b)$ of characters different from b (see Section 4.1). Therefore we can write $v(b, f, t) = v'(f, e, \bar{e})$ and

$L(b, f, t) = L'(f, e, \bar{e})$. A table of L' (or tables of both L' and v' for realistic flow intensities $f \in \{0.00, 0.01, \ldots, 9.99\}$ and values of e and $\bar{e}$, both in $\{0, \ldots, 9\}$, i.e., 100 000 values overall, is pre-computed. As the recurrence (1) considers different substrings t that differ in length by 1, the bs in t can be counted in amortized constant time for each t, so each value of $v(b, f, t)$ is available in constant time. The non-tabulated rare cases can be computed on demand without measurably affecting the running time.

4.1 Scoring of Editing Events

In this section, we derive the edit score $s_{\text{edit}}(b, \ell, t)$ to align two sequences: b^ℓ and t. This is, in fact, a classical sequence alignment problem, with the special property that one sequence b^ℓ is a homopolymer. Instead of using a standard global alignment algorithm every time when s_{edit} is called, we can give a closed formula because of the special structure.

We assume that scores for insertion (*ins*), deletion (*del*), mismatch (*mis*) and match (*mat*) are available and fulfill $ins, del < mis < 0 < mat$.

Let e and $\bar{e} = |t| - e$ be the number of characters in t that are equal to b and not equal to b, respectively. If $|t| = \ell$, the score is composed of only match (e times) and mismatch ($\bar{e}$ times) scores. If t is longer than ℓ, then $|t| - \ell$ characters must be deleted from t to obtain length ℓ, and it is advantageous to delete only non-b characters, as long as there are any. If t is shorter than ℓ, we have e matches and $\bar{e}$ mismatches, and $\ell - |t|$ characters must be inserted into t. Thus the score for aligning t to b^ℓ can be expressed as

$$s_{\text{edit}}(b, \ell, t) = \begin{cases} e \cdot mat + \bar{e} \cdot mis & \text{if } \ell = |t|, \\ e \cdot mat + \bar{e} \cdot mis + (\ell - |t|) \cdot ins & \text{if } \ell > |t|, \\ \min\{e, \ell\} \cdot mat + \max\{\ell - e, 0\} \cdot mis + (|t| - \ell) \cdot del & \text{if } \ell < |t|. \end{cases}$$

The parameter values for *mat*, *mis*, *ins*, *del* must be compatible with the scores for scoring flow intensities f against substring lengths ℓ, which we discuss next. We come back to choosing appropriate values in Section 4.3.

4.2 Scoring of Flow Intensities Against Substring Lengths

Here we describe how to set the scores $\sigma(f, \ell)$ for scoring the event that a flow of intensity f is aligned to a DNA sequence of length ℓ. Our approach is similar to that of Vacic et al. [12], but we go further by analyzing the resulting empirical scores parametrically.

Intuitively, the score should be positive if $f \approx \ell$ and drop into the negative range when $|f - \ell|$ gets large. A consistent set of score values is obtained by using log-odds scores [2, 7], having their roots in the theory of score matrices for amino acids, such as the famous PAM matrices [2]. There the score Σ_{ij} between amino acids i and j is computed as the log-odds $\Sigma_{ij} = \log(P_{ij}/(\pi_i \cdot \pi_j))$, where P_{ij} is the probability of observing i and j paired in an alignment and π_i, π_j are the background frequencies of amino acids i, j, respectively. Moreover, the joint probabilities P_{ij} depend on the divergence time t of the aligned sequences, and so different score matrices $\Sigma_{ij}^{(t)}$ are used for differently diverged sequences.

Here we follow a similar idea for deriving scores for evaluating differences between f and ℓ. We estimate frequencies from (assumedly correctly) aligned flowgrams to DNA sequences. For ease of exposition, we do not discuss different divergence times, and we assume that the flowgrams have been obtained from the DNA reference by sequencing, or at least from a very closely related reference sequence.

Given a large number of such aligned flowgram-DNA alignments, we construct a count matrix $C = (C_{f,\ell})$ for all reasonable genomic lengths $\ell \in \{0, 1, 2, \ldots\}$ and flow intensities

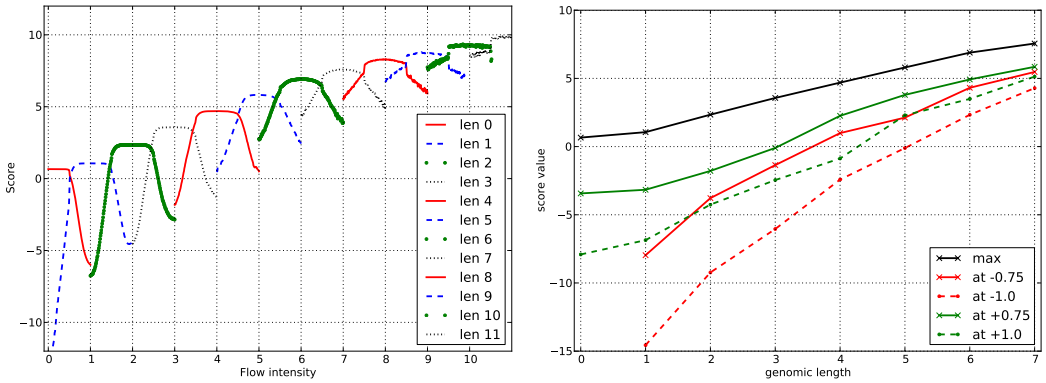


Figure 4 Left: Empirically determined score functions $\sigma(f, l)$ for each genomic sequence length ℓ (see legend) from carefully crafted alignments of flowgrams against an *Arabidopsis* reference. Right: As each function on the left panel can be described by a piecewise affine function with three components, one of them constant, we estimated five parameters from five characteristic score values: for each length ℓ, the score values at $f \in \{\ell - 1.0, \ell - 0.75, \ell, \ell + 0.75, \ell + 1.0\}$. The plot shows these five score values as a function of ℓ.

$f \in \{0.00, 0.01, 0.02, \ldots, 1.00, \ldots\}$, such that $C_{f,\ell}$ counts the number of times we observe a flow of intensity f aligned to a genomic sequence of length ℓ. We obtain a joint probability matrix $P = (P_{f,\ell})$ by dividing C through the sum of its entries. Background frequencies $\pi = (\pi_\ell)$ for genomic lengths are obtained as marginal probabilities $\pi_\ell = \sum_f P_{f,\ell}$, and similarly background frequencies $\tau = (\tau_f)$ for flow intensities. The score component for aligning a flow of intensity f to homopolymer of length ℓ is defined in units of nats as

$$\sigma(f, \ell) := \log \frac{P_{f,\ell}}{\tau_f \cdot \pi_\ell}.$$

To obtain such scores, we used three `.sff` files containing *Arabiopsis* reads (from an unspecified strain), provided by the chair of Genome Research at Bielefeld University. To measure only the effects of homopolymer errors, only reads aligning close-to-perfectly to the *A. thaliana* reference sequence were considered further, and the empirical joint distribution of flow intensities f and homopolymer lengths ℓ was tabulated where $f \in [\ell - 1, \ell + 1]$. The resulting scores are shown in Figure 4, one curve for each ℓ with sufficient data. Unsurprisingly, the maximum score occurs at flow $f = \ell$ for each ℓ. More remarkably, the score stays almost constant at the same level in the interval $f \in [\ell - 0.5, \ell + 0.5]$. At $\ell \pm 0.5$, there appears to be a sudden drop in the scoring function, beyond which we can observe an affine-linear course in the intervals $[\ell - 1.0, \ell - 0.5]$ and $[\ell + 0.5, \ell + 1.0]$. Therefore, for each ℓ, the score function can be described by five parameters, namely the values of $S_{\ell,f}$ for $f \in \{\ell - 1.0, \ell - 0.75, \ell, \ell + 0.75, \ell + 1.0\}$. Scores at other values of f are obtained by linear resp. constant interpolation (or extrapolation outside the 1.0-neighborhood). The parameters for lengths $\ell \leq 7$ are shown in Figure 4 (right). As empirical data becomes sparser for larger ℓ, it is advisable to extrapolate the parameters instead of relying on data.

In summary, we implement $\sigma(f, \ell)$ for each ℓ as a piecewise affine function consisting of three components, given by the empirically determined parameters shown in Figure 4 (right).

	(A)	(B)
original reference r	TAAGAAC	TAAACACCCGG
mutated sample s	TAAAAAC	TAAAACCCCGG
flowgram F	$T^{0.9}A^{4.4}C^{1.1}$	$T^{0.9}A^{3.4}C^{3.4}G^{2.0}$
base-called	TAA-AAC	TAAA--CCCGG
reconstructed	TAAAAAC	TAAAACCCCGG

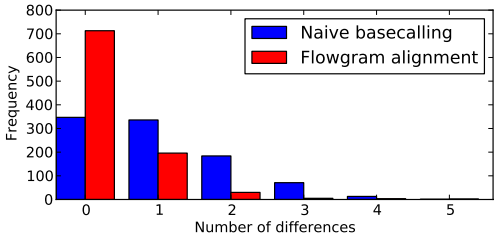


Figure 5 Left: Example errors made by alignment after base calling in contrast to flowgram-string alignment: (A) The G → A substitution is mistakenly reported as a 1 bp deletion because of the low flow value. For flowgram-string alignment with the surrounding context, the case that AAGAA generated $A^{4.4}$ is plausible. (B) Similarly, but slightly more complex, the CA → AC flip is mistakenly reported as a 2 bp deletion after base calling. For our method, however, two mismatches plus small intensity errors are more plausible than two deletions. Right: Histograms of the number of differences due to sequencing and base-calling errors for naive base calling (blue) and our method (red). Note how the red distribution is shifted towards zero.

4.3 Parameters for Editing Events

It remains to appropriately set the match, mismatch, insertion and deletion score parameters *mat*, *mis*, *ins* and *del*, respectively. These depend on the assumed degree of divergence of the sequenced sample and the reference and can be obtained by (approximate) log-odds.

Assuming 3% divergence (i.e., 97% matches, as opposed to about 30% in alignments of random sequences), and rare insertions/deletion with a rate of 1/3000, it is reasonable to use

- $mat \approx \log(0.97/0.3) = 1.173 \approx 1.2$,
- $mis \approx \log(0.03/0.7) = -3.1498 \approx -3.1$,
- $ins = del \approx \log\big((1/3000)/C\big) \approx -8.0$ with some $C \approx 1$.

These are the scores that we use for evaluation; other assumptions will result in different scores. It is important to use the same logarithm (and scaling, if any) as for $\sigma(f, \ell)$ in order to keep both score components compatible.

5 Evaluation

Before we evaluate flowgram-string alignment against base-called alignment, let us illustrate typical miscalls made by base calling. Obviously, the most common case is that a homopolymer length is simply off by 1 because of rounding in the wrong direction. This can always be corrected by post-processing the alignments. However, there are more complex errors, such as spurious indels in the middle or between two homopolymers, as illustrated in Figure 5 (left).

We now demonstrate that flowgram-string alignment reduces the number of differences between observed sequence and reference that are due to sequencing errors, but leaves actual mutation events untouched (see Figure 1). We simulate DNA fragments of *E. coli K12* (NC_000913); call this the *original* data. We introduce mutations by adding 3% substitutions and 0.05% indels (*mutated* data). Then the 454 sequencing process is simulated with *flowsim* [1]. Reads in the resulting `.sff` file are base-called by rounding flow intensities (*basecalled*) and aligned to the original sequence. Alternatively, we use our flowgram alignment algorithm to align each flowgram to the *original* sequence. During the process, the most likely base-space *mutated* sequence is reconstructed using function $L(b, f, t)$ from (3) (*reconstructed*).

All differences between *mutated* and *basecalled* are necessarily due to sequencing errors and wrong base calls. Similarly, all differences between *mutated* and *reconstructed* are due to

sequencing errors and errors by our alignment method. Figure 5 compares histograms of the number of differences, measured by unit-cost edit distance. The differences are considerably reduced for flowgram-string alignment in comparison to base-calling: The distribution is shifted towards the left side. Thus, flowgram alignment is able to distinguish editing events and true mutations.

6 Discussion and Conclusion

We presented a dynamic programming alignment algorithm that optimally aligns flowgrams output by Roche/454 or Ion Torrent sequencers to DNA reference sequences directly, without explicit base calling. Our approach can also be interpreted as calling bases *conditional* on the reference we align to, i.e., doing both steps at the same time instead of sequentially. Our algorithm is based on a two-stage process model (Figure 1) that explains both sequence editing and homopolymer sequencing errors. In particular, in the process, we can reconstruct the most plausible homopolymer length ℓ for each flow b^f and thus separate flow intensity over- and under-calling from sequence editing. Our method is the first one that cleanly separates the two processes.

A major challenge is to design a scoring scheme for flowgram-DNA alignment that is of low complexity (i.e., has few parameters) and statistically well-founded. We here started from a classical log-odds framework [2] that was also used by Vacic et al. [12]. Going a step further, we noted that for each length ℓ, the score function has a simple three-component piecewise affine form that can be described by only five parameters. This yields the first low-complexity scoring scheme for directly aligning 454 flowgrams to DNA sequences.

There are several ways to extend this work in the future. For example, finding a more robust way to estimate the divergence rate between reference and sample than guessing it before computing alignments would be of interest. On the practical side, several optimizations of the basic alignment algorithm are possible, improving the running time from $O(mn^2)$ to $O(mn)$ by restricting the considered predecessors in each node (i, j) of the alignment graph (cf. Figure 3). It is clear that for a flow b^f, the best choices for t and ℓ have $|t| \approx \ell \approx f$. Extending the algorithm to be able to use affine gap costs would be of high practical relevance. This is not entirely trivial, as gaps could extend over several flows, which in the current model can only be considered separately.

Our approach should of course work on Ion Torrent datasets as well, using a different scoring function. The 454 technology can be used for bisulfite amplicon sequencing to determine CpG methylation. The resulting datasets contain long `T`- or `A`-homopolymers after bisulfite conversion and have different characteristics than standard 454 datasets. Thus it is of interest to estimate scoring parameters for this application.

Acknowledgements We thank Bernd Weisshaar (Genome Research, Bielefeld University) for providing `.sff` files from which we estimated the scoring function parameters.

References

1 Susanne Balzer, Ketil Malde, Anders Lanzén, Animesh Sharma, and Inge Jonassen. Characteristics of 454 pyrosequencing data–enabling realistic simulation with flowsim. *Bioinformatics*, 26(18):i420–i425, September 2010.

2 M. Dayhoff, R. Schwartz, and B. Orcutt. A model of evolutionary change in proteins. *Atlas of Protein Sequences and Structure*, 5:3345–352, 1978.

3 David Golan and Paul Medvedev. Using state machines to model the Ion Torrent sequencing process and to improve read error rates. *Bioinformatics*, 29(13):i344–i351, July 2013.

4 Fredrik Lysholm, Björn Andersson, and Bengt Persson. FAAST: Flow-space assisted alignment search tool. *BMC Bioinformatics*, 12:293, 2011.

5 M. Margulies *et al.* Genome sequencing in microfabricated high-density picolitre reactors. *Nature*, 437(7057):376–380, September 2005.

6 Barry Merriman, Ion Torrent R&D Team, and Jonathan M. Rothberg. Progress in Ion Torrent semiconductor chip based sequencing. *Electrophoresis*, 33(23):3397–3417, December 2012.

7 Tobias Müller, Sven Rahmann, and Marc Rehmsmeier. Non-symmetric score matrices and the detection of homologous transmembrane proteins. *Bioinformatics*, 17(Suppl.1):S182–S189, 2001.

8 Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, March 1970.

9 Christopher Quince, Anders Lanzén, Thomas P. Curtis, Russell J. Davenport, Neil Hall, Ian M. Head, L Fiona Read, and William T. Sloan. Accurate determination of microbial diversity from 454 pyrosequencing data. *Nature Methods*, 6(9):639–641, September 2009.

10 M. Ronaghi, S. Karamohamed, B. Pettersson, M. Uhlén, and P. Nyrén. Real-time DNA sequencing using detection of pyrophosphate release. *Analytical Biochemistry*, 242(1):84–89, November 1996.

11 T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, March 1981.

12 Vladimir Vacic, Hailing Jin, Jian-Kang Zhu, and Stefano Lonardi. A probabilistic method for small RNA flowgram matching. *Pacific Symposium on Biocomputing*, pages 75–86, 2008.