German Conference on Bioinformatics 2012

GCB'12, September 19–22, 2012, Jena, Germany

Edited by

Sebastian Böcker
Franziska Hufsky
Kerstin Scheubert
Jana Schleicher
Stefan Schuster

OASICS

Editors

Sebastian Böcker
sebastian.boecker@uni-jena.de
Franziska Hufsky
franziska.hufsky@uni-jena.de
Kerstin Scheubert
kerstin.scheubert@uni-jena.de

Chair of Bioinformatics
Faculty of Mathematics and Computer Science
Friedrich-Schiller-University Jena

Jana Schleicher
jana.foerster@uni-jena.de
Stefan Schuster
stefan.schuster@uni-jena.de

Department of Bioinformatics
Faculty of Biology and Pharmacy
Friedrich-Schiller-University Jena

ACM Classification 1998
J.3 Life and Medical Sciences

ISBN 978-3-939897-44-6

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/978-3-939897-44-6.

Publication date
September, 2012

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.

Digital Object Identifier: 10.4230/OASIcs.GCB.2012.i

ISSN 2190-6807

www.dagstuhl.de/oasics

Contents

Preface

This volume contains papers presented at the German Conference on Bioinformatics (GCB 2012) held in Jena, Germany, September 19 – 22, 2012. The German Conference on Bioinformatics is an annual, international conference, which provides a forum for the presentation of current research in bioinformatics and computational biology. The GCB 2012 was organized by the Jena Center of Bioinformatics (JCB) in cooperation with the German Society for Chemical Engineering and Biotechnology (DECHEMA) and the Society for Biochemistry and Molecular Biology (GBM). The conference was open to all fields of bioinformatics and theoretical systems biology.

Five satellite workshops that took place on 19 September 2012 placed thematic emphasis on diverse aspects of systems biology: "Systems Biology of Aging" organized by J. Sühnel, "Organ-oriented Systems Biology" by D. Driesch and R. Mrowka, "Network Reconstruction and Analysis in Systems Biology" by W. Wiechert and T. Lengauer, "Computational Proteomics and Metabolomics" by S. Böcker, and "Image-based Systems Biology" by M.T. Figge.

Six leading scientists accepted our invitation to give keynote lectures at the conference:

- Claude dePamphilis (Pennsylvania State University, University Park, USA)
  The draft genome sequence of *Amborella trichopoda* sheds light on the ancestral angiosperm genome
- Oliver Fiehn (University of California, Davis, USA)
  Comprehensive metabolomic databases and annotation workflows: The U.S. West Coast Metabolomics Center
- Arndt von Haeseler (Max F. Perutz Laboratories, Vienna, Austria)
  Exploring the sampling universe of RNA-seq
- Tom Kirkwood (Newcastle University, Newcastle, GB)
  Probing the deep complexity of ageing
- Erik van Nimwegen (University of Basel, Basel, Switzerland)
  A democracy of transcription factors: Inferring transcription regulatory interactions from high-throughput data
- Ruth Nussinov (National Cancer Institute, Frederick, USA)
  Structural proteome scale prediction of protein-protein interactions using interfaces

With the topics of these talks the meeting indeed succeeded in '***J***oining ***E***volution, ***N***etworks, and ***A***lgorithms', according to this year's conference motto.

From 39 submissions, the program committee selected 10 highlight papers and 11 regular papers as contributed talks for the conference. Additionally, about 95 poster abstracts were accepted for presentation. All regular papers are collected in this volume. The highlight papers, the abstracts from the invited speakers, and the poster abstracts are collected in a separate volume available online (www.gcb2012-jena.de).

We thank all members of the program committee as well as all local organizers and helpers for their efforts. We are also very grateful to the participants who presented their work at the lively panel sessions and poster party. Our special thanks go to the sponsors who supported the conference financially.

August 2012,
Sebastian Böcker, Franziska Hufsky, Kerstin Scheubert, Jana Schleicher, and Stefan Schuster

Program Committee

Program Chairs
Sebastian Böcker
Stefan Schuster

Program Committee

Mario Albrecht
Rolf Backofen
Jan Baumbach
Michael Beckstette
Niko Beerenwinkel
Tim Beissbarth
Sebastian Böcker
Thomas Dandekar
Dmitrij Frishman
Georg Fuellen
Robert Giegerich
Ivo Große
Reinhard Guthke
Andreas Hildebrandt
Ivo Hofacker
Daniel Huson
Christoph Kaleta
Gunnar W. Klau
Ina Koch
Oliver Kohlbacher
Stefan Kurtz
Hans-Peter Lenhof
Manja Marz
Burkhard Morgenstern
Steffen Neumann
Kay Nieselt
Stefan Posch
Sven Rahmann
Matthias Rarey
Knut Reinert
Uwe Scholz
Dietmar Schomburg
Falk Schreiber
Michael Schroeder
Stefan Schuster
Joachim Selbig
Jens Stoye
Korbinian Strimmer
Andrew Torda
Martin Vingron
Arndt Von Haeseler
Edgar Wingender
Ralf Zimmer

Additional Referees

Nicola Bonzanni
Stefan Canzar
Christian Colmsee
Simona Constantinescu
Fabrizio Costa
Simone Daminelli
Mohammed El-Kebir
Andre Gohr
Giorgio Gonnella
Niels Grabe
Udo Hahn
Volker Heun
Christian Höner Zu Siederdissen
Benny Kneissl
Tina Koestler
Frank Kramer
Andreas Leha
Ioana Lemnian
Manuel Landesfeind
Fernando Meyer
Bui Quang Minh
Konstantin Riege
Fabian Schmich
Heiko Schmidt
Juliane Siebourg
Peter Stadler
Sascha Steinbiss
Patrick Trampert
George Tsatsaronis
Claus Weinholdt
Stephan Weise
Dirk Willrodt

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/`

Jena Centre for Bioinformatics
`http://www.jcb-jena.de/`

Hans-Knöll-Institute Jena
`http://www.hki-jena.de/`

Leibniz Institute for Age Research – Fritz Lipmann Institute
`http://www.fli-leibniz.de/`

Friedrich-Schiller-Universiät Jena
`http://www.uni-jena.de/`

Jena Centre for Systems Biology of Ageing
`http://www.jenage.de/`

Jena School for Microbial Communication
`http://www.jsmc.uni-jena.de/`

International Leibniz Research School for Microbial and Biomolecular Interactions
`http://www.ilrs.hki-jena.de/`

Sponsors and Donors

BioControl Jena GmbH
`http://www.biocontrol-jena.com/`

Jenaer Universitäts-Buchhandlung Thalia
`http://www.thalia.de/`

TimeLogic biocomputing solutions
`http://www.timelogic.com/`

HMK Supercomputing GmbH
`http://conveycomputer.com/lifesciences/`

antibodies-online GmbH
`http://www.antikoerper-online.de/`

Index of Authors

ModeScore: A Method to Infer Changed Activity of Metabolic Function from Transcript Profiles

Andreas Hoppe and Hermann-Georg Holzhütter

Charité University Medicine Berlin, Institute for Biochemistry, Computational Systems Biochemistry Group
hoppe@bioinformatics.org

Abstract

Genome-wide transcript profiles are often the only available quantitative data for a particular perturbation of a cellular system and their interpretation with respect to the metabolism is a major challenge in systems biology, especially beyond on/off distinction of genes.

We present a method that predicts activity changes of metabolic functions by scoring reference flux distributions based on relative transcript profiles, providing a ranked list of most regulated functions. Then, for each metabolic function, the involved genes are ranked upon how much they represent a specific regulation pattern. Compared with the naïve pathway-based approach, the reference modes can be chosen freely, and they represent full metabolic functions, thus, directly provide testable hypotheses for the metabolic study.

In conclusion, the novel method provides promising functions for subsequent experimental elucidation together with outstanding associated genes, solely based on transcript profiles.

1998 ACM Subject Classification J.3 Life and Medical Sciences

Keywords and phrases Metabolic network, expression profile, metabolic function

Digital Object Identifier 10.4230/OASIcs.GCB.2012.1

1 Background

The comprehensive study of the cell's metabolism would include measuring metabolite concentrations, reaction fluxes, and enzyme activities on a large scale. Measuring fluxes is the most difficult part in this, for a recent assessment of techniques, see [31]. Although mass spectrometry allows to assess metabolite concentrations in a more comprehensive way, the larger the set of potential metabolites, the more difficult [8]. Enzyme activities are currently measured only for selected enzymes [15, 33]. In contrast, the measurement of transcript RNA [35, 41] and protein levels [4] on a large scale is an established technique. Chip assays allow the high-throughput estimation for different conditions in a single experiment at manageable cost. Thus, we are confronted with the situation that for many cellular systems there are plenty of measured RNA profiles and a lack of accurate data for metabolite concentrations and enzyme activities. Even though the RNA will not allow accurate quantitative predictions of fluxes it is desirable to draw as many conclusions from that data as possible.

In the context of metabolic networks, transcript values are often used to select a subset of *active* reactions using a threshold [3, 38, 20, 23, 11, 22]. This has two immediate problems: (i) the selection of a proper threshold value and (ii) the subnetwork selected in that way may not be a functional network. The first problem is tackled in the GIMME algorithm [3] by gradually penalizing a reaction below a certain threshold, or in the approach by Shlomi et al. [38] by using a three-valued system — *off*, *on* and *intermediate* — with two thresholds. The second problem is solved either by the successive addition of necessary reactions considered *off* [3, 38] or by maximizing the number of concordant genes in an optimization framework

[20]. More elegantly, both problems are combined in a method which sets the threshold according to network function [22].

However, the main problem of reducing the transcript information to *on* and *off* genes is that it discards any information on gradual metabolic changes. But a complete switch-off of a metabolic enzyme is a rare event, most regulation is just the reduction or amplification of a particular function. Lee et al. predicted the metabolic state from absolute expression profiles without thresholding [28] but for expression *changes* it has not yet been done. This prompted us to develop a method focusing on quantitative changes of metabolic functionality.

To tackle this ambitious task the problem is *restated* first. While in transcript-based flux-balance analysis [3, 38, 22] the flux distribution resp. activity network is the result of the computation, here we assume that a plethora of functional flux distributions is already known. The transcript profile are used to decide which of them change their activity, in which direction, and how much.

In fact, functional reaction paths are studied throughout the last decades and are well-known for a large number of functions. On the basis of well-curated metabolic networks [10, 40] these functional flux modes can even be computed automatically [16].

In the context of the metabolism, transcript profiles are often used to find co-expressed genes, e.g. to understand the pattern of regulation [37, 21] and, thus, to predict targets of transcription factors. The Mamitsuka applied the approach to predefined metabolic reaction paths [30, 39] based on the KEGG database [25]. They used the results to rank the reaction paths [14], an idea which is also used here. However, the question of co-expression is not the main aspect here, instead it is the relative change of activity. Thus, we combine the idea of using expression profiles to score metabolic paths [14] with the idea of functional flux distributions [16]. The approach is supported by a finding of Notebaart et al. that the strongest gene co-expression occurs for coupled fluxes [32].

2 Results

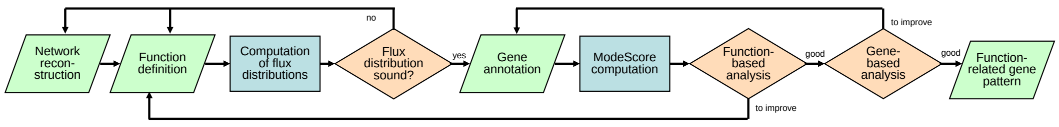


Figure 1 Flow chart of the ModeScore analysis.

The final goal of a ModeScore analysis is a set of remarkable function-related gene patterns from a series of full-genome transcript (or protein) profiles. It is a multi-step method with backwards jumps depicted in Figure 1, where the novel scoring algorithm is one of the steps. While some of the steps are computed (blue boxes in Figure 1) others involve manual curation (green rhomboids) and selective evaluation (khaki rhombi).

Preparation of flux distributions and gene annotations

For the predictivity of the ModeScore analysis sound representative flux distributions are critical. Sometimes, reference modes are already the result of the network testing process [12] but it still it may be worthwhile to tailor the flux distributions to a particular application. In a broad screening of metabolic functions, 1000s of different functions can be used with several alternate solutions computed automatically [18]. But the focus can be shifted, and also a confined analysis with just a few functions is sensible.

The gene–reaction association may be given in the reconstruction process [7] but even then it is advisable to manually refine this with the help of UniProt [1] and BRENDA [36] and the literature referenced with, e.g. if a respective isoform is expressed but at such a low level that another isoform is dominant it should be discarded, or if the enzyme is present but has a low affinity to the substrate. If several genes remain for one reaction, and if several repeat experiments have been performed their average expression is used. In the comparison between two cellular states, their expression values (as $\log_2$) are subtracted.

Scoring function

Let the k-th reference mode be denoted with $M_k = (m_i)_i$, where m_i is the flux rate of the i-th reaction. The relative expression profile is denoted by $V = (v_i)_i$, where v_i is the difference of the $\log_2$ values of the transcript abundances of one state and a reference state. Then the score of the mode is

$$\mathsf{Score}(M_k, V) = \frac{\sum_{i \in I_k} w_i \mathsf{score}_i(m_i, v_i)}{\sum_{i \in I_k} w_i}$$

where

$$I_k = \{i |\, m_i \neq 0\} \quad \text{and} \quad w_i = \sqrt{|m_i| \omega_i}$$

and

$$\mathsf{score}_i(m_i, v_i) = e^{-\frac{1}{2}\left(2\frac{\lambda v_i - m_i}{|m_i|}\right)^2}.$$

λ is chosen such that $\mathsf{Score}(M_k, V)$ is maximal. The non-negative numbers ω_i are fixed adjustments to modify the impact of a reaction on the score. If the gene expression changes are proportional to flux rates of a reference mode, the score would be equal to 1, and λ is the scaling factor between fluxes and expression changes. The rationale of this approach is that the larger the required flux increase, the larger the necessary enzyme amount and, thus, the increase in RNA transcription.

In the application below, reactions with stoichiometric factors larger than one (except protons) receive a weight ω_i which is the number of individual conversions for lumped reactions, and zero for non-enzymatic reactions, see Supplementary file 6.

The number $1/\lambda$, called amplitude in the sequel, is a measure for the function's expression difference. In fact, if all genes related to a specific function are regulated by the same amount α, then the score would be 1 and the amplitude equal to α. In this sense, the amplitudes are compatible to the expression changes.

Optimization procedure

The scoring function has been observed to have many local maxima for larger reference flux distributions, especially near zero. Therefore a specifically tuned calculation method has been recruited.

To find the global optimum of the scaling factor λ the derivation of the scoring function has been calculated algebraically. A set of probe points is calculated with the following procedure: (i) for every m_i/v_i a first level probe point is defined (ii) for every pair of first level probe points the arithmetic mean is added as another probe point (iii) for every consecutive pair (x, y) of second level probe points $n = 10 + 10(\frac{1}{1+x})$ intermediate points with equal distance are added as another probe point. This procedure is tailored to search for maxima in superposed GAUSSian bell curves with centers at m_i/v_i. It uses the fact that a higher density of local maxima can be expected where the density of m_i/v_i is higher (→ii) and the steepness of the bell curves for m_i near zero is higher, thus more probe points are generated. The huge abundance of probe points (compared to just a few maxima) to ensure a very low probability to miss solutions. The zeros of the derivation function are computed with the function `fzero` in octave [9] by supplying the probe points with different signs of the derivation function value. The zeros of the derivation function are the candidates for the global maximum which is subsequently selected.

ModeScore analysis

There are two different layers on which the ModeScore results can be analyzed, on the layer of functions and on the layer of individual genes (for one selected function).

The ranking of the amplitudes of the different functions reveals which are the most up and down-regulated functions in the comparison between two expression profiles. Additionally, three or four expression profiles can be compared in a way that the difference of amplitudes is ranked. This way, the functions with the most remarkable pattern are selected.

The next step is to check the layer of individual reactions and genes. For each function and pair of expression profiles the individual score components $\mathsf{score}_i(m_i, v_i)$ are analyzed. A high $\mathsf{score}_i(m_i, v_i)$ means that the reaction is considered belonging to the regulation pattern that led to the evaluation of the function with the amplitude $1/\lambda$. If the score is small or zero this reaction has been overruled by other reactions.

Possibly, a reaction's score $\mathsf{score}_i(m_i, v_i)$ has been the result of different genes, e.g. subunits of a protein complex. If it becomes for instance apparent that only one subunit is responsible for the regulation, other subunits can be deleted from the gene assignment to the reaction. The same should be done if several isozymes are recorded, but some of them are apparently not active in this particular experiment. It should be clear that a general network reconstruction includes all the possible reaction assignments, and only at this point of the analysis it is possible to select from the genes. After this modification, the ModeScore calculation must be repeated.

After this process has been iteratively repeated, the final reaction scores will reflect the function's regulation as shown by the expression profiles. For visualization it is now advisable to discard the genes which are not part of the regulation pattern.

In a nutshell, the following criteria are used to select the functions:

- functions with high absolute amplitudes are favored,
- central, well-known functions are favored to more peripheral functions,
- if a function can be seen as a sub-function of another function, check if the super-function is also a good candidate,
- prefer functions which are top/bottom scorer in more than one of the comparisons,
- prefer functions which show a plausible pattern of genes regulated in a consistent manner,
- prefer functions which have already been noted as specifically sensitive in the experiment,
- prefer functions which contain genes with highly remarkable regulation amplitude or pattern, and
- prefer functions whose pattern is not based on a single gene regulation.

Note that this selection process takes into account available domain knowledge.

The following criteria are used to select for each selected function the relevant genes which led to the amplitude estimation of the whole function are selected. This is also a manual process which takes into account several information sources and is guided by the following criteria:

- prefer genes with large transcript value changes,
- prefer genes with large absolute values,
- prefer genes associated to reactions which are adjacent to reactions whose genes have already been selected,
- prefer genes of highly specific enzymes or transporters,
- prefer genes of reactions which are most important for the function under regard and not for any more important function, and
- close gaps in an otherwise preferable reaction path.

Mostly, a connected reaction path (as a subset for the whole flux distribution) appears to be responsible for the regulation. Transporters are also integrated if consistent with the regulation of the enzymes. This aspect is noteworthy in comparison to the PathRanker [14] concept which ignores transporters altogether. Another important aspect in comparison to PathRanker (based on the KEGG MODULE reaction paths [26]) is that the definition of the relevant paths is not predefined (and fixed) but flexible and dependent on the actual expression values of the experiment. Often, they are in accordance with well-known reaction paths but in some cases there is a unique and noteworthy different selection of genes forming a functional unit.

Application of the method to TGFβ treatment of hepatocyte cultures

As an application of the proposed method, transcript profiles of primary mouse hepatocytes in monolayer culture (3 time points, 1h, 6h, and 24h, control versus TGFβ stimulation, 3 repeats) [6, 13, 5] are screened for remarkable functional alterations.

The functions with very high and low amplitudes of the profile comparisons 1h vs. 24h in the untreated sample and the control vs. TGFβ comparison at 24h have been analyzed for their functional relevance, see Supplementary File 4. Functions related to intermediates are replaced by their superordinate function, similar functions have been collected. For each of these functions the scores for individual reactions, see Supplementary File 5, have been analyzed for the set of genes which are responsible that the function in question has appeared with a particular regulation pattern.

The most prominent functions are the down-regulation of tyrosine degradation and ethanol degradation. The ethanol degradation pathway has already been verified to be sensitive to TGFβ [5]

However, the strong down-regulation of tyrosine degradation is a novel finding yet to be confirmed on the metabolic level, see Figure 2. The amplitudes $\frac{1}{\lambda}$ of the C1h/24h and the C/T24h comparisons are -3.5 and -0.87, respectively, belonging to the largest negative amplitudes among all functions. The scores $\mathsf{Score}(M_k, V)$ are 0.41 and 0.44 indicating a relatively strong coherence of the expression changes. The scores $\mathsf{score}_i(m_i, v_i)$ for the individual genes are especially high ($\geq$0.5) for the genes coding the reaction cascade from Phenylalanine to Acetoacetate+Fumarate, see Supplementary File 5 for details. Thus, the complete degradation pathway is down-regulated in time, amplified by TGFβ treatment, in a very consistent pattern, even suggesting a common regulation factor.

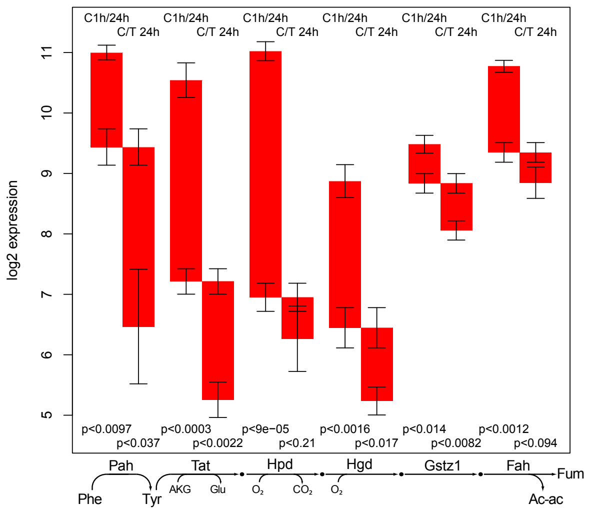


Figure 2 Regulation of the degradation cascade of phenylalanine and tyrosine. Red bars indicate down-regulation. For each gene, the respective first bar (C1h/24h) shows the comparison of the beginning of the experiment (1h) with the 24h time point of the control experiment, the second bar (C/T 24h) shows the comparison of control vs. treated sample at 24h. Error bars indicate average standard deviation of the 3 repeat experiments. P-values refer to the probability that there is an equal expression of two respective probe values, determined from the 3 repeats using Welch's t test [42], implemented in R [29]. The chart was prepared with R.

3 Discussion

Scoring algorithm

The scoring function is based on the superposition of GAUSSian bell curves. The GAUSSian bell curve is the typical approximation in situation where an expected behavior (the expression pattern matches the flux pattern) is blurred by a number of not quantitatively described factors which is the case for the relation of the RNA expression to the expressed enzyme and further to the reaction flux: RNA can be selectively degraded before translation, the rate of misfolded protein differs for different proteins, the life span of the enzymes differs, the reaction flux may deviate from the maximal catalytic rate, the enzyme life span is different and so is the catalytic efficiency.

The influence of the scores of individual spots to the total score of the metabolic functions is not equal but controlled by a weighting scheme. The weight mainly depends on the flux in the reference flux distribution. This is based on the assumption that the importance of the individual reactions to indicate the metabolic function as a whole can be approximated by the flux. However, that one or very few reactions dominate the scoring of the whole scoring function must be avoided. A good compromise has been found in a number of different trials as the square root of the absolute flux rate.

The luminescence signal on a gene chip is not actually measuring the RNA concentration in the sample as the affinity of the RNA to the probes is not equal and unknown for most genes. However, with a high signal the expectation value of the RNA concentration is high as well and the same is true for the resulting protein concentration, the enzyme activity, and finally the reaction flux. As the relation can be assumed to be normally distributed, the use of GAUSSian bell curves is the most robust way to account for the uncertainty if the signal-flux relation.

The ModeScore concept would also be applicable if not just gene array luminescence data but quantitatively more reliable qPCR [41], protein data, or even enzyme activity data were available. The salient point is that array data suffices to draw first conclusions before the laborious and costly other techniques are applied. On the other side, if data more closely representing metabolic function (such as enzyme activity data) was available then methods based on actual enzyme mechanisms could and should be applied such as kinetic modeling with time-resolved differential equations.

Maxima in the scoring function

In the scoring formula the global maximum of the scoring function has been used to obtain the scaling factor from the relative expression profile to the reference flux distribution. For some examples alternative local maxima have been observed with only slightly smaller scoring value. These alternative solutions can be seen as alternative ways to match the expression pattern to the flux distribution. The alternative solutions can be analyzed in the same way but generally their existence can be interpreted as an ambiguity of the expression profiles.

Reference flux distributions

The method critically depends on the quality of the predicted reference flux distributions. For this work they have been obtained in a similar way as in [12]. Different computation series have been preformed combining thermodynamic realizability [19] and flux minimization [17] in different proportions. The obtained flux distributions have individually been checked on biochemical plausibility while the one with the fewest reactions used is preferred, see Supplementary File 3.

An advantage of the ranking concept is that the flux distributions need not be representative (in the sense that overlaps are critical). Only a sufficient coverage of the whole metabolic network should be ensured (not to miss remarkable transcript changes). Not to miss a remarkable regulation abundance rather plentiful flux distributions should be used.

Comparison with other approaches

One may want to compare the ModeScore approach with a simple concept using the same reference modes: average the expression changes of all genes associated to the function. This has been tested and it showed almost no discriminative power. The problem is that even if a function contains a recognizable pattern of regulation, other genes are constant or regulated in the opposite direction. Thus the average expression changes are always close to zero. The ModeScore function however finds an amplitude that is consistent with the changes of several genes even if it is far from zero.

One may want to compare ModeScore with just filtering the largest changes of genes associated to the metabolic network. The functional context of these genes then had to be added manually. But many genes play a role in different functions, so they should be discriminated again manually. Even if this has been done, the result would not be able to detect moderately sized but very consistent regulation patterns (such as the Tyrosine degradation function above).

Often, lists of transcript changes are filtered by a certain threshold and the regulated genes (in both directions) are counted for each metabolic subsystem in either the gene ontology [2] or in the KEGG maps [24]. The main problem is that in such a collection of reactions/genes, the genes may have a completely different functional context: for instance in the amino acid metabolism maps there are reactions specific to synthesis, degradation, and trans-amination, some of the reactions are needed to synthesize further cellular components from amino acids, for gluconeogenesis or urea synthesis. A pathway-oriented approach can not distinguish between these functions. Occasionally, even the same reaction located in different compartments participates in different functions: HMG-CoA synthase in the cytosol is involved in cholesterol synthesis while HMG-CoA synthase in the mitochondrium is only recruited for the generation of ketone bodies (in mammalian hepatocytes).

4 Methods and Materials

The stoichiometric network of the human hepatocyte's metabolism [12] has been adapted to the mouse and a plethora of about 1000 metabolic functions defined, see Supplementary File 2. The gene assignments have been obtained from the following resources: (i) Metabolic reactions with a KEGG reaction annotation in HepatoNet can be mapped to a mouse gene using KEGG [25]. (ii) Other metabolic functions with a EC number can also be mapped to a mouse gene using KEGG. (iii) Transporters with an annotation in TCDB can be mapped to a mammalian enzyme (UniProt nomenclature) using the TCDB [34]. (iv) Protein synthesis reactions have been mapped to the protein directly which makes sense for this work as the RNA is the most specific requisite for the synthesis reaction using Uniprot [1]. (v) Proteins have been mapped to their coding gene using Ensembl/BioMart [27]. (vi) Genes of the different species have been mapped to mouse genes (Ensembl nomenclature) using the computed homologies contained in BioMart database. (vii) BioMart database has also been used to assign the Affymetrix probeset identifiers to Ensembl annotations. HepatoNet was enlarged by the synthesis reactions of collagens. See Supplementary file 1 for the final network with gene annotations.

Availability

Calculation of the ModeScore scores is implemented in the freely available software FASIMU [18] in the function `modeset-score`. Supporting scripts including the score optimization process is available from the authors upon request.

Acknowledgements

The author thanks Iryna Ilkavets, Carsten Sticht, and Steven Dooley for discussions on the application example. The works was funded by the BMBF Germany as part of the VirtualLiver Network.

Supplementary files

1. Network definition in SBML (level 2 version 4) format including Ensembl annotation (xml).
2. Definition of metabolic functions (pdf).
3. Flux distributions satisfying the functional definitions (pdf).
4. Sorted tables of scores and amplitudes for functions (pdf).
5. Table of detailed reaction scores for selected functions (pdf).
6. List of weights by reaction (pdf).

The files can be downloaded from:
`http://www.bioinformatics.org/fasimu/_GCB_Modescore_Supplementary_files`

References

1 Rolf Apweiler, Amos Bairoch, Cathy H Wu, Winona C Barker, Brigitte Boeckmann, Serenella Ferro, Elisabeth Gasteiger, Hongzhan Huang, Rodrigo Lopez, Michele Magrane, Maria J Martin, Darren A Natale, Claire O'Donovan, Nicole Redaschi, and Lai-Su L Yeh. Uniprot: the universal protein knowledgebase. *Nucleic Acids Res*, 32(Database issue):D115–D119, Jan 2004.

2 M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nat Genet*, 25(1):25–29, May 2000.

3 Scott A Becker and Bernhard Ø Palsson. Context-specific metabolic networks are consistent with experiments. *PLoS Comput Biol*, 4(5):e1000082, May 2008.

4 Grigoriy S Chaga. Antibody arrays for determination of relative protein abundances. *Methods Mol Biol*, 441:129–151, 2008.

5 Loredana Ciuclan, Sabrina Ehnert, Iryna Ilkavets, Hong-Lei Weng, Haristi Gaitantzi, Hidekazu Tsukamoto, Elke Ueberham, Nadja M Meindl-Beinker, Manfred V Singer, Katja Breitkopf, and Steven Dooley. Tgf-beta enhances alcohol dependent hepatocyte damage via down-regulation of alcohol dehydrogenase i. *J Hepatol*, 52(3):407–416, Mar 2010.

6 Steven Dooley, Jafar Hamzavi, Loredana Ciuclan, Patricio Godoy, Iryna Ilkavets, Sabrina Ehnert, Elke Ueberham, Rolf Gebhardt, Stephan Kanzler, Andreas Geier, Katja Breitkopf, Honglei Weng, and Peter R Mertens. Hepatocyte-specific smad7 expression attenuates tgf-beta-mediated fibrogenesis and protects against liver damage. *Gastroenterology*, 135(2):642–659, Aug 2008.

7 Natalie C Duarte, Scott A Becker, Neema Jamshidi, Ines Thiele, Monica L Mo, Thuy D Vo, Rohith Srivas, and Bernhard Ø Palsson. Global reconstruction of the human metabolic network based on genomic and bibliomic data. *Proc Natl Acad Sci U S A*, 104(6):1777–1782, Feb 2007.

8 Warwick B Dunn. Mass spectrometry in systems biology an introduction. *Methods Enzymol*, 500:15–35, 2011.

9 John W. Eaton. *GNU Octave Manual*. Network Theory Limited, 2002.

10 Adam M Feist, Markus J Herrgård, Ines Thiele, Jennie L Reed, and Bernhard Ø Palsson. Reconstruction of biochemical networks in microorganisms. *Nat Rev Microbiol*, 7(2):129–143, Feb 2009.

11 Ori Folger, Livnat Jerby, Christian Frezza, Eyal Gottlieb, Eytan Ruppin, and Tomer Shlomi. Predicting selective drug targets in cancer through metabolic networks. *Mol Syst Biol*, 7:501, 2011.

12 Christoph Gille, Christian Bölling, Andreas Hoppe, Sascha Bulik, Sabrina Hoffmann, Katrin Hübner, Anja Karlstädt, Ramanan Ganeshan, Matthias König, Kristian Rother, Michael Weidlich, Jörn Behre, and Herrmann-Georg Holzhütter. HepatoNet1: a comprehensive metabolic reconstruction of the human hepatocyte for the analysis of liver physiology. *Mol Syst Biol*, 6:411, Sep 2010.

13 Patricio Godoy, Jan G Hengstler, Iryna Ilkavets, Christoph Meyer, Anastasia Bachmann, Alexandra Müller, Gregor Tuschl, Stefan O Mueller, and Steven Dooley. Extracellular matrix modulates sensitivity of hepatocytes to fibroblastoid dedifferentiation and transforming growth factor beta-induced apoptosis. *Hepatology*, 49(6):2031–2043, Jun 2009.

14 Timothy Hancock, Ichigaku Takigawa, and Hiroshi Mamitsuka. Mining metabolic pathways through gene expression. *Bioinformatics*, 26(17):2128–2135, Sep 2010.

15 T. K. Harris and M. M. Keshwani. Measurement of enzyme activity. *Methods Enzymol*, 463:57–71, 2009.

16 Sabrina Hoffmann, Andreas Hoppe, and Hermann-Georg Holzhütter. Composition of metabolic flux distributions by functionally interpretable minimal flux modes (minmodes). *Genome Inform*, 17(1):195–207, 2006.

17 Hermann-Georg Holzhütter. The principle of flux minimization and its application to estimate stationary fluxes in metabolic networks. *Eur J Biochem*, 271(14):2905–2922, Jul 2004.

18 Andreas Hoppe, Sabrina Hoffmann, Andreas Gerasch, Christoph Gille, and Hermann-Georg Holzhütter. FASIMU: flexible software for flux-balance computation series in large metabolic networks. *BMC Bioinformatics*, 12:28, 2011.

19 Andreas Hoppe, Sabrina Hoffmann, and Hermann-Georg Holzhütter. Including metabolite concentrations into flux balance analysis: thermodynamic realizability as a constraint on flux distributions in metabolic networks. *BMC Syst Biol*, 1:23, 2007.

20 Carola Huthmacher, Andreas Hoppe, Sascha Bulik, and Hermann-Georg Holzhütter. Antimalarial drug targets in plasmodium falciparum predicted by stage-specific metabolic network analysis. *BMC Syst Biol*, 4(120), August 2010.

21 Jan Ihmels, Ronen Levy, and Naama Barkai. Principles of transcriptional control in the metabolic network of saccharomyces cerevisiae. *Nat Biotechnol*, 22(1):86–92, Jan 2004.

22 Paul A Jensen and Jason A Papin. Functional integration of a metabolic network model and expression data without arbitrary thresholding. *Bioinformatics*, 27(4):541–547, Feb 2011.

23 Livnat Jerby, Tomer Shlomi, and Eytan Ruppin. Computational reconstruction of tissue-specific metabolic models: application to human liver metabolism. *Mol Syst Biol*, 6:401, Sep 2010.

24 Minoru Kanehisa. The KEGG database. *Novartis Found Symp*, 247:91–101; discussion 101–3, 119–28, 244–52, 2002.

25 Minoru Kanehisa, Michihiro Araki, Susumu Goto, Masahiro Hattori, Mika Hirakawa, Masumi Itoh, Toshiaki Katayama, Shuichi Kawashima, Shujiro Okuda, Toshiaki Tokimatsu, and Yoshihiro Yamanishi. KEGG for linking genomes to life and the environment. *Nucleic Acids Res*, 36(Database issue):D480–D484, Jan 2008.

26 Minoru Kanehisa and Susumu Goto. KEGG: kyoto encyclopedia of genes and genomes. *Nucleic Acids Res*, 28(1):27–30, Jan 2000.

27 Rhoda J Kinsella, Andreas Kähäri, Syed Haider, Jorge Zamora, Glenn Proctor, Giulietta Spudich, Jeff Almeida-King, Daniel Staines, Paul Derwent, Arnaud Kerhornou, Paul Kersey, and Paul Flicek. Ensembl biomarts: a hub for data retrieval across taxonomic space. *Database (Oxford)*, 2011:bar030, 2011.

28 Dave Lee, Kieran Smallbone, Warwick B Dunn, Ettore Murabito, Catherine L Winder, Douglas B Kell, Pedro Mendes, and Neil Swainston. Improving metabolic flux predictions using absolute gene expression data. *BMC Systems Biology*, 6:73, 2012.

29 Uwe Ligges. *Programmieren mit R*. Springer-Verlag, Heidelberg, 3 edition, 2009. In German.

30 Hiroshi Mamitsuka, Yasushi Okuno, and Atsuko Yamaguchi. Mining biologically active patterns in metabolic pathways using microarray expression profiles. *SIGKDD Explorations*, 5(2):113–121, 2003.

31 Totte Niittylae, Bhavna Chaudhuri, Uwe Sauer, and Wolf B Frommer. Comparison of quantitative metabolite imaging tools and carbon-13 techniques for fluxomics. *Methods Mol Biol*, 553:355–372, 2009.

32 Richard A Notebaart, Bas Teusink, Roland J Siezen, and Balázs Papp. Co-regulation of metabolic genes is better explained by flux coupling than by network distance. *PLoS Comput Biol*, 4(1):e26, Jan 2008.

33 Andrew Razgulin, Nan Ma, and Jianghong Rao. Strategies for in vivo imaging of enzyme activity: an overview and recent advances. *Chem Soc Rev*, 40(7):4186–4216, Jul 2011.
34 Milton H Saier, Can V Tran, and Ravi D Barabote. Tcdb: the transporter classification database for membrane transport protein analyses and information. *Nucleic Acids Res*, 34(Database issue):D181–D186, Jan 2006.
35 M. Schena, D. Shalon, R. W. Davis, and P. O. Brown. Quantitative monitoring of gene expression patterns with a complementary dna microarray. *Science*, 270(5235):467–470, Oct 1995.
36 Ida Schomburg, Antje Chang, Oliver Hofmann, Christian Ebeling, Frank Ehrentreich, and Dietmar Schomburg. BRENDA: a resource for enzyme data and metabolic information. *Trends Biochem Sci*, 27(1):54–56, Jan 2002.
37 Aswin S N Seshasayee, Gillian M Fraser, M. Madan Babu, and Nicholas M Luscombe. Principles of transcriptional regulation and evolution of the metabolic system in e. coli. *Genome Res*, 19(1):79–91, Jan 2009.
38 Tomer Shlomi, Moran N Cabili, Markus J Herrgård, Bernhard Ø Palsson, and Eytan Ruppin. Network-based prediction of human tissue-specific metabolism. *Nat Biotechnol*, 26(9):1003–1010, Sep 2008.
39 Ichigaku Takigawa and Hiroshi Mamitsuka. Probabilistic path ranking based on adjacent pairwise coexpression for metabolic transcripts analysis. *Bioinformatics*, 24(2):250–257, Jan 2008.
40 Ines Thiele and Bernhard Ø Palsson. A protocol for generating a high-quality genome-scale metabolic reconstruction. *Nat Protoc*, 5(1):93–121, 2010.
41 Zhong Wang, Mark Gerstein, and Michael Snyder. RNA-Seq: a revolutionary tool for transcriptomics. *Nat Rev Genet*, 10(1):57–63, Jan 2009.
42 B L Welch. The generalisation of student's problems when several different population variances are involved. *Biometrika*, 34(1-2):28–35, 1947.

Comparing Fragmentation Trees from Electron Impact Mass Spectra with Annotated Fragmentation Pathways

Franziska Hufsky[1,2] and Sebastian Böcker[1]

1 **Chair for Bioinformatics, Friedrich-Schiller-University, Jena, Germany,** {franziska.hufsky,sebastian.boecker}@uni-jena.de

2 **Max Planck Institute for Chemical Ecology, Beutenberg Campus, Jena, Germany**

Abstract

Electron impact ionization (EI) is the most common form of ionization for GC-MS analysis of small molecules. This ionization method results in a mass spectrum not necessarily containing the molecular ion peak. The fragmentation of small compounds during EI is well understood, but manual interpretation of mass spectra is tedious and time-consuming. Methods for automated analysis are highly sought, but currently limited to database searching and rule-based approaches. With the computation of hypothetical fragmentation trees from high mass GC-MS data the high-throughput interpretation of such spectra may become feasible. We compare these trees with annotated fragmentation pathways. We find that fragmentation trees explain the origin of the ions found in the mass spectra in accordance to the literature. No peak is annotated with an incorrect fragment formula and 78.7 % of the fragmentation processes are correctly reconstructed.

1998 ACM Subject Classification J.2 Physical Sciences and Engineering (Chemistry)

Keywords and phrases metabolomics, GC-MS, computational mass spectrometry, fragmentation trees

Digital Object Identifier 10.4230/OASIcs.GCB.2012.12

1 Introduction

Metabolomics, also called "metabonomics" or "metabolic profiling", is a rapidly developing field of 'omics' research, dealing with the detection, identification and quantification of low molecular-weight compounds (typically below 1000 Da) in cells, organs or organisms. The analysis and identification of small molecules is important in many areas of biology and medicine such as biomarker discovery, diagnostics, pharmaceutical chemistry and functional genomics [2, 9, 36]. The metabolome consists of various compounds that belong to a wide array of compound classes, including sugars, acids, bases, lipids, hormonal steroids, and many others [5, 18]. The structural diversity of metabolites is extraordinarily large despite of their small size [21]. Unlike biopolymers such as proteins and glycans, metabolites are not made up of repeated building blocks. The genome sequence does not reveal information about metabolite structure, as it does for protein structure. The number of metabolites in any higher eukaryote is estimated between 4000 and 20 000 [7]. Unfortunately, an astounding number of these metabolites remain uncharacterized with respect to their structure and function [26].

At the moment there is no single instrumental platform that can analyze all metabolites [5, 21]. Mass spectrometry (MS), typically coupled with a chromatographic separation technology, is one of the key technologies for the identification of small molecules. It has excellent compound specificity and high sensitivity. In particular, MS sensitivity is orders of magnitude higher than that of nuclear magnetic resonance (NMR) [21, 29]. Several kinds of analytical apparatus have been developed and most of these combine chromatography with a fragmentation technique to increase compound specificity. Gas chromatography coupled to mass spectrometry (GC-MS) is one of the most frequent tools for profiling metabolites and it was in existence decades before liquid chromatography MS (LC-MS) [8, 14]. The amount of data produced during metabolomic analysis is hard to process and analyze manually [18].

The most common ionization technique in GC-MS is electron impact ionization (EI). The resulting fragment-rich mass spectra are in general consistent and specific for each molecule [20, 21] and fragmentation mechanisms are already well described [23]. Reference spectra were collected over many years, allowing for automated interpretation via database search [24]. Where the compound is unknown, comparing the spectrum obtained to a spectral library will result in imprecise or incorrect hits, or no hits at all [8, 18, 20]. To cover a wider range of compounds in silico fragmentation is used to predict spectra of compounds with known structure [11, 12, 19, 37]. A first step towards the structural elucidation of fully unknown compounds is feature-based identification of the compound class [17, 19, 34, 35]. See Kind and Fiehn [20] for a comprehensive review of computational techniques for small molecule mass spectrometry.

Böcker and Rasche [3] introduced fragmentation trees for the *de novo* interpretation of metabolite fragmentation data. The fragmentation tree concept helps to identify the molecular formulas of the compound and to interpret the fragmentation process. Nodes are annotated with the molecular formulas of the fragments, and edges represent fragmentation events, that is, neutral or radical losses. Computing fragmentation trees does not require databases of compound structures or mass spectra or expert knowledge of fragmentation. The trees can be compared to each other to identify compound classes of unknowns [28]. Expert evaluation suggests that the fragmentation trees from LC-MS^2 data are of very good quality [29]. Fragmentation trees can also be computed from LC-MS^n data [31]. Recently, Hufsky *et al.* [16] presented a computational method for the *de novo* interpretation of EI fragmentation data, based on fragmentation tree construction, and applied it to real world data. Besides a list of common losses, this method does not use any chemical expert knowledge, but does require high mass accuracy of the measurements [16].

In this study, we evaluate the quality of fragmentation trees computed from EI fragmentation data [16]. To evaluate the potential of fragmentation trees to reconstruct fragmentation processes we compare them to annotated fragmentation pathways of 22 compounds from the literature. The constructed fragmentation trees were not supposed to depict the actual fragmentation reactions. They however agree well with the annotated pathways explaining the origin of the respective ions found in the mass spectra. No peak was annotated with an incorrect fragment formula and 78.7 % of the fragmentation processes were correctly reconstructed. For the annotation of the fragmentation processes in the literature the molecular structures of the compounds were used. In contrast, the computation of fragmentation trees works without this knowledge. The assignment of molecular formulas to all fragments and explanation of relevant fragmentation reactions independent of existing library knowledge, supports the structural elucidation of unknown compounds. Combined with a method for the automated comparison of fragmentation trees [15, 28] it will enable the automated analysis of metabolites that are not included in common libraries.

2 Methods

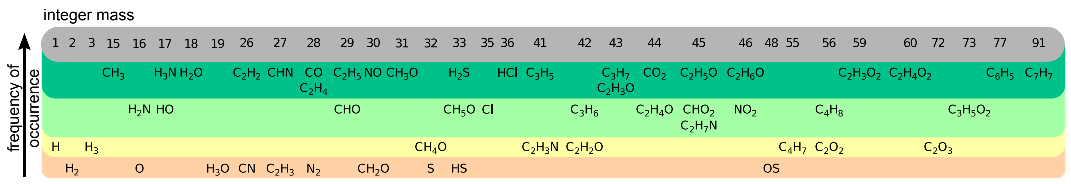
For the interpretation of the GC-MS spectra we use fragmentation trees as introduced by Böcker and Rasche [3] for LC-MS^2 spectra. A hypothetical fragmentation tree models fragmentation cascades by annotating nodes with the molecular formulas of fragments, and edges with fragmentation events, that is, neutral or radical losses. The root of the fragmentation tree is labeled with the molecular formula of the unfragmented ion. For LC-MS^2 data the molecular ion mass is known. EI results in a mass spectrum not necessarily containing the molecular ion peak. Hufsky *et al.* [16] proposed a method for computing fragmentation trees from such data.

To compute a fragmentation tree from an EI fragmentation spectrum a fragmentation graph is constructed. All candidate molecular formulas within the mass accuracy of the instrument are computed for each peak. The fragmentation graph contains a node for each decomposition. The nodes are colored, such that all explanations of the same peak receive the same color. Nodes are weighted using mass deviation and peak intensity [3]. Two nodes are connected by an edge (corresponding to a loss) if the second molecular formula is a subformula of the first. Edges are weighted according to their plausibility as real fragmentation steps considering the mass of the loss, the ratio between carbon and hetero atoms, and common losses for EI fragmentation (see Table 1). See [16] for a detailed description of the scoring.

The colorful subtree with maximum sum of edge weights is the explanation of the observed fragments, that fits best with the given conditions. Considering trees every fragment is explained by a unique fragmentation pathway, see [29]. Considering only colorful trees every peak is explained by a single fragment. Several fragments resulting in a single peak is an extremely rare event in practice.

By demanding that each fragment in the fragmentation spectrum is generated by a single fragmentation pathway we slightly oversimplify the problem. Our optimization algorithm will choose the mainly occurring pathway to compute a fragmentation tree. There are two exceptions to this reasoning: (1) In the resulting fragmentation tree, assume that some fragment f_3 is cleaved from f_2, which is in turn cleaved from f_1. Solely from the EI fragmentation pattern and without additional structural information, it cannot be ruled out that fragment f_3 is in truth cleaved directly from f_1. Both interpretations are implicitly encoded in the fragmentation tree: the fragmentation may occur from the fragment's direct parent in the tree or from any of its parents (see Figure 1(a)). (2) In the resulting fragmentation tree, assume that some fragment f_2 is cleaved from a fragment f_1 by loosing l_1 and another fragment f_3 is cleaved from f_1 by loosing l_2. Further, another fragment f_4 is cleaved from f_2 by loosing l_2. Solely from the data, it cannot be ruled out that fragment f_4 is in truth cleaved from f_3 by loosing l_1. Again, both interpretations are implicitly encoded

Table 1 List of common losses over the alphabet used throughout this study (CHNOPSCl) for scoring EI fragmentation reactions [13]. The losses are sorted by integer mass and their probability of occurrence in a GC-MS spectrum [16]. Losses in the first row (dark green) are very common and thus score high, while losses in the last row (orange) are not-that-common and thus score comparatively low.



integer mass	1	2	3	15	16	17	18	19	26	27	28	29	30	31	32	33
frequency of occurrence ↑				CH_3		H_3N	H_2O		C_2H_2	CHN	CO C_2H_4	C_2H_5	NO	CH_3O		H_2S
					H_2N	HO						CHO				CH_5O
	H		H_3												CH_4O	
		H_2			O			H_3O	CN	C_2H_3	N_2		CH_2O		S	HS

integer mass	35	36	41	42	43	44	45	46	48	55	56	59	60	72	73	77	91
frequency of occurrence ↑		HCl	C_3H_5		C_3H_7 C_2H_3O	CO_2	C_2H_5O	C_2H_6O				$C_2H_3O_2$	$C_2H_4O_2$			C_6H_5	C_7H_7
	Cl			C_3H_6		C_2H_4O	CHO_2 C_2H_7N	NO_2			C_4H_8				$C_3H_5O_2$		
			C_2H_3N	C_2H_2O						C_4H_7	C_2O_2			C_2O_3			
									OS								

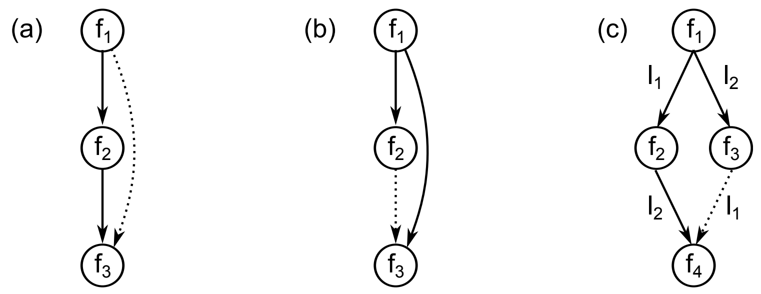


Figure 1 Comparing fragmentation trees (solid edges) with annotated pathways from the literature (dashed edges). (a) In the fragmentation tree, fragment f_3 is cleaved from f_2, which is in turn cleaved from f_1. It cannot be ruled out that fragment f_3 is in truth cleaved directly from f_1 (dotted edge). Both interpretations are implicitly encoded in the fragmentation tree. We evaluate these edges as *correct*. (b) In the fragmentation tree, fragment f_3 is cleaved directly from f_1, while in truth it is cleaved from f_2 (dotted edge), which is in turn cleaved from f_1. We evaluate these edges as inserted *to high*. (c) *Parallelogram:* Fragment f_4 is cleaved from f_2 by loosing l_2, which is in turn cleaved from f_1 by loosing l_1. In truth fragment f_4 is cleaved by loosing l_2 first and l_1 afterwards (dotted edge). Both interpretations are implicitly encoded in the fragmentation tree. We evaluate these edges as *correct*.

in the fragmentation tree: the fragmentation may occur by loosing l_1 first and l_2 afterwards, or the other way (see Figure 1(c)). In the following, we call this constellation *parallelogram*.

EI is a hard ionization technique often resulting in missing or low intensity molecular ion peaks [20, 21]. Different from LC-MS^2 analysis the mass of the molecular ion is not known. All nodes in the graph are possible roots of the fragmentation tree. Molecular formulas explaining each peak cannot be restricted to sub-molecular formulas as proposed in [29]. Therefore the identification of the molecular ion and formula and the computation of the complete fragmentation tree is done in two separate steps.

We first identify the molecular ion and molecular formula using only a set of peaks that appear to be most relevant for the compound. These peaks are selected using three different criteria. We choose the k_1 most intense peaks, the k_2 peaks with highest *score*, and the k_3 peaks with highest *score* in the *upper m/z range*. The *score* is a combination of m/z value and relative intensity $m/z \cdot \ln(100 \cdot int_{rel})$ and the *upper m/z range* is the m/z region from $0.9\tilde{M}$ to $\tilde{M}$ where $\tilde{M}$ is the highest m/z of a peak detected in the spectrum. In this step fragmentation trees are computed using Dynamic Programming [3]. Afterwards we compute a fragmentation tree for the complete spectrum assuming that we know the correct molecular ion and molecular formula of the compound. The resulting fragmentation tree is rooted in this molecular formula. In this step, fragmentation trees are computed using Integer Linear Programming [30].

3 Data

The EI induced fragmentation of small molecules is well described in the literature. To evaluate the potential of fragmentation trees to reconstruct fragmentation processes we extract annotated fragmentation pathways for 22 compounds from different compound classes (see Table 2). In [1] Acheson *et al.* describe the fragmentation of alkyl acridines. We choose two simple alkylacridines and two reduced acridines containing chlorine. Further, we choose seven compounds from a study on alkyl isocyanides and methyl branched alkyl cyanides [10]. From [25] we select four dihydro-1,4-oxathiines with fragmentation paths additionally invest-

Table 2 Overview of the 22 reference compounds with fragmentation pathways annotated in the literature. If more than one compound of the same class is used, we denote the class name and give the mass range and average mass.

		mass	
compound (class)	**#**	**range**	**average**
alkyl acridines [1]	4	207.1 - 399.2 Da	276.1 Da
alkyl isocyanides & α-branched alkyl cyanides [10]	7	41.0 - 83.1 Da	69.1 Da
dihydro-1,4-oxathiines [25]	4	146.0 - 235.1 Da	178.8 Da
gossypol [27]	1	518.2 Da	
ephedrine [33]	1	165.1 Da	
2,1-benzisothiazoline 2,2-dioxide nitro derivatives [4]	5	214.0 - 242.0 Da	228.0 Da
all	22	41.0 - 518.2 Da	187.6 Da

igated with qualitative collisionally induced dissociation (CID) measurements. From [27] we extract the fragmentation pathway of gossypol and from [33] the one from ephedrine. Further, we choose five 2,1-benzisothiazoline 2,2-dioxide nitro derivatives from [4].

As the measured spectra are not available to us, we simulate spectra from the pathways. From the molecular formulas in the fragmentation pathway, we compute exact peak masses, and simulate “measured” spectra by adding a normal distributed error of 10 ppm on the mass of the fragment formula (without considering ionization). Peak intensities of the fragment peaks are taken from the literature. They are either given as actual number or estimated from the plotted spectrum. In addition, we add 70 % noise peaks with uniformly distributed masses smaller than the parent mass, and pareto distributed intensities.

4 Results

4.1 Molecular Ion Peak and Formula Identification

Fragmentation trees enable the identification of the molecular ion and the molecular formula of a metabolite if the molecular ion is present in the spectrum. EI is a hard ionization technique resulting in missing molecular ion peaks in about 30 % of the spectra [22]. For two compounds in our dataset, namely gossypol and ephedrine, the relative intensity of the molecular ion peak given in the literature was 0 %. We test the identification of the molecular ion and the molecular formula on the remaining 20 spectra containing a molecular ion peak.

To identify the molecular ion peak and its formula an alphabet of potential elements must be provided to the method. For all compounds, we use the six elements most abundant in metabolites, namely carbon (C), hydrogen (H), nitrogen (N), oxygen (O), phosphorus (P), and sulfur (S) [18]. When analyzing the two compounds in our dataset containing chlorine (Cl), we also add this element to the alphabet. Information on whether a compound contains chlorine can be usually obtained from isotope pattern analysis.

Computing the molecular ion peak and molecular formula requires an average of 4.6 s for each compound. This time includes peak decomposition and graph construction. We discard peaks with no decomposition. We then choose the subset of peaks that appear to be most relevant for the compound as described above. We choose $k_1 = 10$ and $k_2 = k_3 = 5$, resulting in at most 20 peaks if the sets are not overlapping.

For all compounds, the method correctly detect the molecular ion peak. This is not surprising, since the molecular ion peaks have the highest m/z values in all spectra, based on the generation of noise peaks as described above. For 17 of the 20 compounds (85 %), the highest scoring suggestion for both the molecular ion peak and its molecular formula is

Table 3 Results of the tree evaluation. (a) Peak explanations in the annotated pathways compared to the computed fragmentation trees. [1]Percent of the explanations in the annotated pathways. [2]Percent of the explanations in the computed fragmentation trees. (b) Evaluation of the fragmentation events annotated in the fragmentation trees. For 5 of the 277 correct peak explanations, the fragmentation process leading to this fragment is not given in the literature. (c) Evaluation of the frequency of parallelograms in the annotated pathways. A parallelogram is *closed* if both fragmentation ways are annotated, and *"open"* otherwise. [3]Percent of the "open" parallelograms.

(a) fragments	pathway total	tree total	correct	missing	additional
peak explanations	296	284	277	19	7
precentage			93.6 %[1]	6.4 %[1]	2.5 %[2]

(b) losses	total	*correct*	correct, but *to deep*	correct, but *reverse order*	*to high*	*wrong*
losses	272	214	31	8	19	39
percentage		78.7 %	11.4 %	2.9 %	7.0 %	14.3 %

(c) parallelograms	total	*closed*	*"open"*	different in tree
parallelograms	99	29	70	8
percentage		29.3 %	70.7 %	11.4 %[3]

correct. For the remaining three compounds, the correct molecular formula is the second suggestion.

4.2 Fragmentation Tree Quality

We compute a hypothetical fragmentation for every compound, assuming that we know the correct molecular ion and molecular formula of the compound. In this step, all peaks of the spectrum are used for computation. Computation, including decomposition and graph construction, requires 1.5 s on average and a maximum of 18.5 s for the largest compound, namely gossypol. For this compound with mass of 518.2 Da, decomposition of all peaks requires 17.9 s (97 % of the total running time).

We compare the computed fragmentation trees with annotated fragmentation patterns from the literature. The fragmentation trees annotate 284 peaks in total (see Table 3(a)). Only seven of this explanations (2.5 %) are false positives, that is explanations of noise peaks as fragments. The remaining 277 peaks are annotated with the correct fragment formula. From all 296 fragments described in the pathways from literature 19 (6.4 %) could not be explained. There are different reasonings for a peak not being explained in the tree. For some peaks, the mass deviation of the measured peak mass to the exact mass is to high. This effect is getting stronger for smaller peaks, since mass deviation penalty is dependent of the peak intensity [16]. For other peaks, the fragmentation step resulting in this fragment gets a bad score. For example, the loss C_2H_2N that was annotated in the literature as a first fragmentation step for three of the alkyl isocyanides is not included in the list of common losses for EI fragmentation and is not even a combination of these (see Table 1 and [16]). Therefore the fragments resulting from this step could not be identified. Nevertheless, the method is capable of identifying losses that are very specific for a single compound or compound class and therefore not listed as a common loss (see [16]).

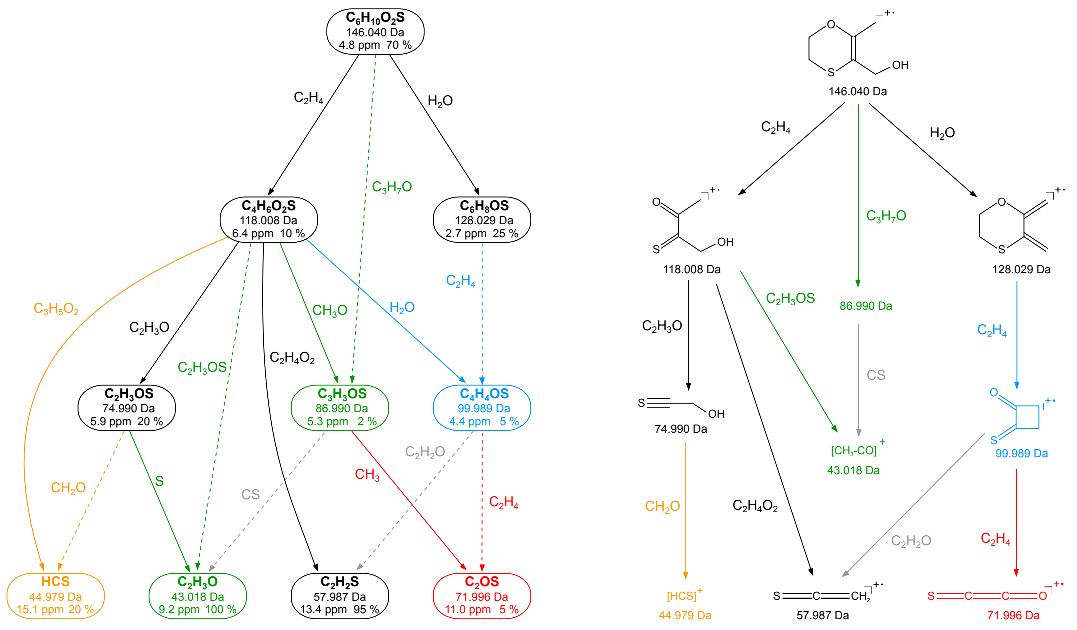

$C_6H_{10}O_2S$ 146.040 Da 4.8 ppm 70 %
C_2H_4
H_2O
C_3H_7O
$C_4H_6O_2S$ 118.008 Da 6.4 ppm 10 %
C_6H_8OS 128.029 Da 2.7 ppm 25 %
$C_3H_5O_2$
C_2H_3O
C_2H_3OS
$C_2H_4O_2$
CH_3O
H_2O
C_2H_4
C_2H_3OS 74.990 Da 5.9 ppm 20 %
C_3H_3OS 86.990 Da 5.3 ppm 2 %
C_4H_4OS 99.989 Da 4.4 ppm 5 %
CH_2O
S
CS
C_2H_2O
CH_3
C_2H_4
HCS 44.979 Da 15.1 ppm 20 %
C_2H_3O 43.018 Da 9.2 ppm 100 %
C_2H_2S 57.987 Da 13.4 ppm 95 %
C_2OS 71.996 Da 11.0 ppm 5 %
146.040 Da
118.008 Da
128.029 Da
86.990 Da
74.990 Da
$[CH_3\text{-}CO]^+$ 43.018 Da
99.989 Da
$[HCS]^+$ 44.979 Da
57.987 Da
71.996 Da

Figure 2 Computed fragmentation tree (solid edges) of 5,6-hydro-3-hyroxymethyl-2-methyl-1,4-oxathiine (left) compared to the annotated pathways [25] from the literature (right). This compound is a worst-case example to visualize all the things that can go wrong. All fragments are annotated with the correct molecular formula. Dashed edges in the tree are losses from the annotated pathways. Black edges in the fragmentation tree agree with the annotated pathways. Grey dashed edges are additional pathways that could not be computed since the tree property would have been violated. The blue fragment was actually cleaved in *reverse order* from the molecular ion. The green fragments were inserted *to deep*, and the orange fragment was inserted *to high* in the fragmentation tree. The red fragment was inserted into a completely different pathway. Note that mass errors of more than 10 ppm occur as we added the simulated mass error on the mass of the fragment formula (without considering ionization).

Individual edges from the fragmentation tree were compared to those in the annotated pathways, and matching losses were assigned as *correct*. In some cases, consecutive edges of the fragmentation tree can be combined to give the molecular formula of a single fragmentation step in the annotated fragmentation pathways (see Figure 1(a)). In some other cases two consecutive losses in the fragmentation tree are described in reverse order in the annotated fragmentation pathways (see Figure 1(c)). We evaluate those fragments that were inserted *to deep* or in *reverse order* in the fragmentation trees as *correct*, since without a given structural formula and solely from the EI fragmentation data, the correct case cannot be distinguished from our method's suggestion. If the fragmentation step in the resulting fragmentation tree is explained by several consecutive steps in the annotated pathway, the fragment was inserted *to high* (see Figure 1(b)). If the fragment was inserted into a completely different pathway the edge is assigned as *wrong*.

For 5,6-hydro-3-hyroxymethyl-2-methyl-1,4-oxathiine [25], we now describe in more detail how we evaluate the edges of the fragmentation tree (see Figure 2). We choose this compound as worst-case example to visualize all the things that can go wrong. The loss of ethene from the molecular ion (146-118) followed by a loss of C_2H_3O (118-75) as well as a loss of $C_2H_4O_2$ (118-58) are annotated as *correct*, as they can be found in the annotated pathways. The water loss from the molecular ion (146-128) is also annotated in the literature. In the fragmentation

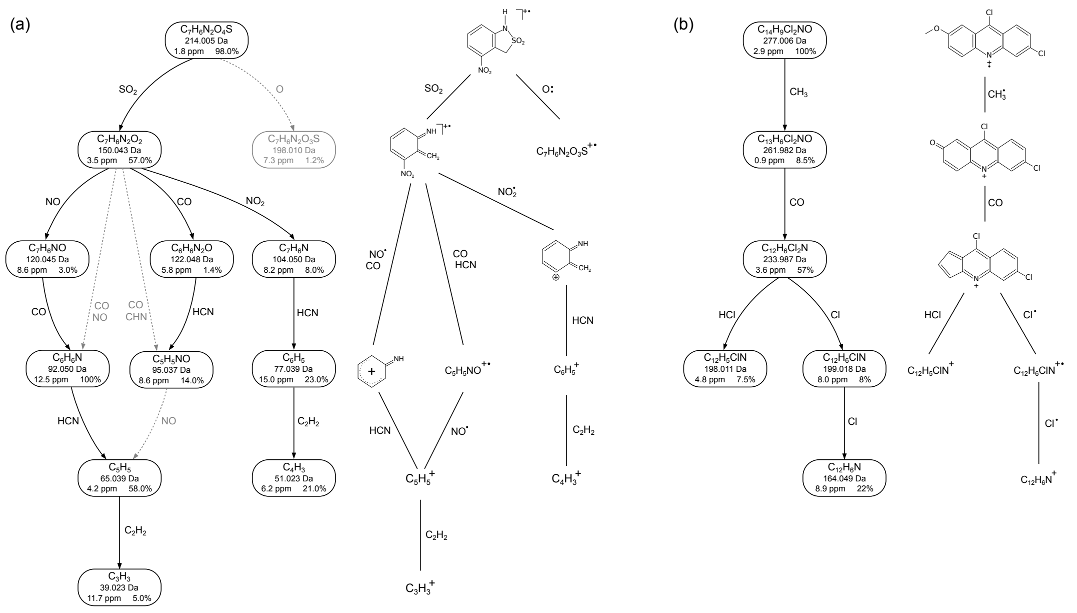

(a)
C7H6N2O4S 214.005 Da 1.8 ppm 98.0%
SO2
O
C7H6N2O2 150.043 Da 3.5 ppm 57.0%
C7H6N2O3S 198.010 Da 7.3 ppm 1.2%
NO
CO
NO2
C7H6NO 120.045 Da 8.6 ppm 3.0%
C6H6N2O 122.048 Da 5.8 ppm 1.4%
C7H6N 104.050 Da 8.2 ppm 8.0%
CO
CO NO
CO CHN
HCN
HCN
C6H6N 92.050 Da 12.5 ppm 100%
C5H5NO 95.037 Da 8.6 ppm 14.0%
C6H5 77.039 Da 15.0 ppm 23.0%
HCN
NO
C2H2
C5H5 65.039 Da 4.2 ppm 58.0%
C4H3 51.023 Da 6.2 ppm 21.0%
C2H2
C3H3 39.023 Da 11.7 ppm 5.0%
SO2
O:
C7H6N2O3S+•
NO2•
NO• CO
CO HCN
HCN
C5H5NO+•
C6H5+
HCN
NO•
C2H2
C5H5+
C4H3+
C2H2
C3H3+
(b)
C14H9Cl2NO 277.006 Da 2.9 ppm 100%
CH3
C13H6Cl2NO 261.982 Da 0.9 ppm 8.5%
CO
C12H6Cl2N 233.987 Da 3.6 ppm 57%
HCl
Cl
C12H5ClN 198.011 Da 4.8 ppm 7.5%
C12H6ClN 199.018 Da 8.0 ppm 8%
Cl
C12H6N 164.049 Da 8.9 ppm 22%
CH3•
CO
HCl
Cl•
C12H5ClN+
C12H6ClN+•
Cl•
C12H6N+

Figure 3 Fragmentation trees compared to annotated pathways from the literature. (a) Fragmentation tree (left) and annotated pathway (right) of a 2,1-benzisothiazoline 2,2-dioxide nitro derivative (compound 6 from [4]). The grey fragment is not explained in the fragmentation tree as it has very low intensity and results from a rather uncommon loss (see Table 1). Dashed edges in the tree are additional losses from the annotated pathways that cannot be explained by our method since the tree property would be violated. In the literature the edge (150-92) combines two fragmentation steps (150-120-92), since the 120 Da peak is very small. In truth, it is very likely, that this fragmentation always proceeds in two steps, but that the lifetime of the intermediate ions is too short [4]. The same applies to edge (150-95) combining the two fragmentation steps (150-122-95). (b) The fragmentation tree (left) and the annotated pathway (right) of 6,9-dichloro-2-methoxyacridine [1] match completely. Note that mass errors of more than 10 ppm occur as we added the simulated mass error on the mass of the fragment formula (without considering ionization).

tree ethene gets lost first and water afterwards (146-118-100), while in the annotated pathway these losses are cleaved in *reverse order*. Edges between nodes 118-75-43 can be combined to the expected loss of C_2H_3OS so the loss of sulfur is considered as *correct*. Pulling up the edges between nodes 146-118-87 results in a total loss of C_3H_7O, so the CH_3O loss was inserted *to deep* and is considered as *correct* by pull-up. Cleaving fragment 45 directly from 118 is considered as *to high*. Fragment 72 was cleaved by loosing ethene from fragment 100 in the annotated pathway. Therefore the methyl loss (87-72) in the fragmentation tree is annotated as *wrong*.

We use similar reasoning processes to evaluate all hypothetical fragmentation trees (see Figure 3 for two examples and Table 3(b) for an overview). For 5 of the 277 correct peak explanations, the fragmentation process leading to this fragment is not given in the literature. From the remaining 272 losses in our data set, 214 losses (78.7 %) are assigned as *correct*. From these, 31 fragments (11.4 %) are inserted *to deep* and 8 fragments (2.9 %) are actually cleaved in *reverse order*. Further, we find that 19 fragments (7.0 %) are inserted *to high* and 39 edges (14.3 %) are annotated as *wrong*. We stress that, unlike for the annotation of the fragmentation processes in the literature, our method has no information about the molecular structure of the compounds.

4.3 Parallelograms

We evaluate the frequency of *parallelograms* in the annotated pathways from the literature (see Table 3(c)). As mentioned above, these are constellations where it cannot be decided solely from the data, whether a fragment results from cleaving loss l_1 first and l_2 afterwards or the other way round (see Figure 1), since both intermediate fragment ions are present in the spectrum. In total, we find 99 parallelograms in all but three compounds. 29 of these are *closed*, that is both fragmentation ways are annotated. This is possible since pathways from the literature not necessarily have to be trees. In contrast, our method has to choose one of these fragmentation pathways. For the remaining 70 parallelograms, either the one or the other way is annotated. From these 70 parallelograms, our method selects the other (possibly wrong) pathway in only 8 (11.4 %) cases.

5 Conclusion

We show that hypothetical fragmentation trees agree in their general information very well with annotated EI fragmentation patterns. We stress that for the computation of the trees no information about the molecular structure of the compounds is used. It is important to note that fragmentation trees are not a tool to reflect the specific mechanisms of EI fragmentation. We find that often the combination or inversion of edges results in pathways that correspond to the true fragmentation. This is not a major set-back since the relevant fragmentation can be constructed based on the trees. Fragmentation trees are a basis for the further interpretation of EI mass spectra. The information obtained, such as fragment formulas, can be used within other methods, for example to simplify in silico fragmentation and presumably improve its results.

Many available methods for analyzing fragmentation spectra of metabolites are rule-based. Mass spectral features are used for classifying compounds [35], Scott [32] uses rules for different classes to estimate the molecular mass of the compound, and rules are used to predict the fragmentation pattern of compounds not included in spectral libraries [19]. Completely unknown compounds may not necessarily follow these known rules for classification or fragmentation. In contrast, the computation and alignment of fragmentation trees is a fully automated and “rule-free” approach that is not limited to known compound classes [28]. It allows to find similar, not necessarily identical, compounds in a library search and unlike other methods it can report the significance of these hits using a decoy database. Consensus substructures of these hits may be key structural elements of the unknown compound and can be used within molecular isomer generators to enumerate all structural isomers containing these substructures [6]. This pipeline will suggest only a few molecular structures and thus can greatly reduce manual analysis time.

Fragmentation tree alignment already accounts for the combination of two consecutive edges [15]. In addition, we find that for some consecutive fragmentation steps the respective ions do not allow to determine the correct fragmentation order solely from the EI data. These constellations occur in 86 % of the compounds. Our method cannot discern the correct fragmentation order solely from the data and will select, based on the scoring properties, the more common and smaller loss twice. In the future, both fragmentation ways should be considered in the fragmentation tree alignment.

Acknowledgments F. Hufsky was supported by the International Max Planck Research School Jena. We thank Martin Rempt for pointing out EI fragmentation pathways annotated in the literature.

References

1 R. M. Acheson, R. T. Aplin, and R. G. Bolton. Electron impact induced alkyl-group fragmentation on the acridine nucleus. *Organic Mass Spectrometry*, 12:518–530, 1977.

2 R. J. Bino, R. D. Hall, O. Fiehn, J. Kopka, K. Saito, J. Draper, B. J. Nikolau, P. Mendes, U. Roessner-Tunali, M. H. Beale, R. N. Trethewey, B. M. Lange, E. S. Wurtele, and L. W. Sumner. Potential of metabolomics as a functional genomics tool. *Trends Plant Sci*, 9(9):418–425, 2004.

3 S. Böcker and F. Rasche. Towards de novo identification of metabolites by analyzing tandem mass spectra. *Bioinformatics*, 24:I49–I55, 2008. Proc. of *European Conference on Computational Biology* (ECCB 2008).

4 W. Danikiewicz, K. Wojciechowski, R. H. Fokkens, and N. M. M. Nibbering. Electron impact-induced fragmentation of 2,1-benzisothiazoline 2,2-dioxide. *Org Mass Spectrom*, 28:853–859, 1993.

5 K. Dettmer, P. A. Aronov, and B. D. Hammock. Mass spectrometry-based metabolomics. *Mass Spectrom Rev*, 26(1):51–78, 2007.

6 J.-L. Faulon, D. P. Visco, and D. Roe. Enumerating molecules. In K. B. Lipkowitz, R. Larter, and T. R. Cundari, editors, *Reviews in Computational Chemistry*, volume 21, chapter 3, pages 209–286. John Wiley & Sons, Inc., 2005.

7 A. R. Fernie, R. N. Trethewey, A. J. Krotzky, and L. Willmitzer. Metabolite profiling: from diagnostics to systems biology. *Nat Rev Mol Cell Biol*, 5(9):763–769, 2004.

8 O. Fiehn. Extending the breadth of metabolite profiling by gas chromatography coupled to mass spectrometry. *Trends Analyt Chem*, 27(3):261–269, 2008.

9 O. Fiehn, J. Kopka, P. Dörmann, T. Altmann, R. N. Trethewey, and L. Willmitzer. Metabolite profiling for plant functional genomics. *Nat Biotechnol*, 18(11):1157–1161, 2000.

10 W. Heerma and J. J. D. Ridder. The electron-impact-induced fragmentation of some alkyl isocyanides and α-branched alkyl cyanides. *Org Mass Spectrom*, 3:1439–1456, 1970.

11 M. Heinonen, A. Rantanen, T. Mielikäinen, J. Kokkonen, J. Kiuru, R. A. Ketola, and J. Rousu. FiD: a software for ab initio structural identification of product ions from tandem mass spectrometric data. *Rapid Commun Mass Spectrom*, 22(19):3043–3052, 2008.

12 M. Heinonen, A. Rantanen, T. Mielikäinen, E. Pitkänen, J. Kokkonen, and J. Rousu. Ab initio prediction of molecular fragments from tandem mass spectrometry data. In *Proc. of German Conference on Bioinformatics (GCB 2006)*, volume P-83 of *Lecture Notes in Informatics*, pages 40–53, 2006.

13 M. Hesse, B. Zeeh, and H. Meier. *Spectroscopic Methods in Organic Chemistry*. Thieme Medical Pub, 1997.

14 E. C. Horning and M. G. Horning. Metabolic profiles: gas-phase methods for analysis of metabolites. *Clin Chem*, 17(8):802–809, 1971.

15 F. Hufsky, K. Dührkop, F. Rasche, M. Chimani, and S. Böcker. Fast alignment of fragmentation trees. *Bioinformatics*, 28:i265–i273, 2012. Proc. of *Intelligent Systems for Molecular Biology* (ISMB 2012).

16 F. Hufsky, M. Rempt, F. Rasche, G. Pohnert, and S. Böcker. De novo analysis of electron impact mass spectra using fragmentation trees. *Anal Chim Acta*, 739:67–76, 2012.

17 J. Hummel, N. Strehmel, J. Selbig, D. Walther, and J. Kopka. Decision tree supported substructure prediction of metabolites from GC-MS profiles. *Metabolomics*, 6(2):322–333, 2010.

18 H. J. Issaq, Q. N. Van, T. J. Waybright, G. M. Muschik, and T. D. Veenstra. Analytical and statistical approaches to metabolomics research. *J Sep Sci*, 32(13):2183–2199, 2009.

19 A. Kerber, R. Laue, M. Meringer, and K. Varmuza. MOLGEN-MS: Evaluation of low resolution electron impact mass spectra with MS classification and exhaustive structure generation. *Adv Mass Spectrom*, 15:939–940, 2001.

20 T. Kind and O. Fiehn. Advances in structure elucidation of small molecules using mass spectrometry. *Bioanal Rev*, 2(1-4):23–60, 2010.

21 T. Kind, G. Wohlgemuth, D. Y. Lee, Y. Lu, M. Palazoglu, S. Shahbaz, and O. Fiehn. FiehnLib: mass spectral and retention index libraries for metabolomics based on quadrupole and time-of-flight gas chromatography/mass spectrometry. *Anal Chem*, 81(24):10038–10048, 2009.

22 S. J. Lehotay, K. Mastovska, A. Amirav, A. B. Fialkov, T. Alon, P. A. Martos, A. de Kok, and A. R. Fernández-Alba. Identification and confirmation of chemical residues in food by chromatography-mass spectrometry and other techniques. *Trends Analyt Chem*, 27(11):10170–1090, 2008.

23 F. W. McLafferty and F. Tureček. *Interpretation of Mass Spectra*. University Science Books, Mill valley, California, fourth edition, 1993.

24 S. Neumann and S. Böcker. Computational mass spectrometry for metabolomics – a review. *Anal Bioanal Chem*, 398(7):2779–2788, 2010.

25 V. Nevalainen and P. Vainiotalo. Electron impact induced fragmentation of dihydro-1,4-oxathiines. 1. 2,3-substituted 5,6-dihydro-1,4-oxathiines. *Org Mass Spectrom*, 21(9):543–548, 1986.

26 G. J. Patti, O. Yanes, and G. Siuzdak. Innovation: Metabolomics: the apogee of the omics trilogy. *Nat Rev Mol Cell Biol*, 13(4):263–269, 2012.

27 P. Przybylski, T. Pospieszny, A. Huczyński, and B. Brzezinski. EI MS and ESI MS studies of the bisesquiterpene from cotton seeds: Gossypol and its Aza-derivatives. *J Mass Spectrom*, 43(5):680–686, 2008.

28 F. Rasche, K. Scheubert, F. Hufsky, T. Zichner, M. Kai, A. Svatoš, and S. Böcker. Identifying the unknowns by aligning fragmentation trees. *Anal Chem*, 84(7):3417–3426, 2012.

29 F. Rasche, A. Svatoš, R. K. Maddula, C. Böttcher, and S. Böcker. Computing fragmentation trees from tandem mass spectrometry data. *Anal Chem*, 83:1243–1251, 2011.

30 I. Rauf, F. Rasche, F. Nicolas, and S. Böcker. Finding maximum colorful subtrees in practice. In *Proc. of Research in Computational Molecular Biology (RECOMB 2012)*, volume 7262 of *Lect Notes Comput Sci*, pages 213–223. Springer, Berlin, 2012.

31 K. Scheubert, F. Hufsky, F. Rasche, and S. Böcker. Computing fragmentation trees from metabolite multiple mass spectrometry data. *J Comput Biol*, 18(11):1383–1397, 2011.

32 D. R. Scott. Rapid and accurate method for estimating molecular weights of organic compounds from low resolution mass spectra. *Chemometr Intell Lab*, 16(3):193–202, 1992.

33 M. Thevis and W. Schänzer. Mass spectrometry in sports drug testing: Structure characterization and analytical assays. *Mass Spectrom Rev*, 26(1):79–107, 2007.

34 H. Tsugawa, Y. Tsujimoto, M. Arita, T. Bamba, and E. Fukusaki. GC/MS based metabolomics: development of a data mining system for metabolite identification by using soft independent modeling of class analogy (SIMCA). *BMC Bioinformatics*, 12:131, 2011.

35 K. Varmuza and W. Werther. Mass spectral classifiers for supporting systematic structure elucidation. *J Chem Inf Comp Sci*, 36(2):323–333, 1996.

36 D. S. Wishart. Current progress in computational metabolomics. *Brief Bioinform*, 8(5):279–293, 2007.

37 S. Wolf, S. Schmidt, M. Müller-Hannemann, and S. Neumann. In silico fragmentation for computer assisted identification of metabolite mass spectra. *BMC Bioinformatics*, 11:148, 2010.

Finding Characteristic Substructures for Metabolite Classes

Marcus Ludwig[1], Franziska Hufsky[1,2], Samy Elshamy[1], and Sebastian Böcker[1]

1 Chair for Bioinformatics, Friedrich-Schiller-University, Jena, Germany, {m.ludwig,franziska.hufsky,sebastian.boecker}@uni-jena.de

2 Max Planck Institute for Chemical Ecology, Beutenberg Campus, Jena, Germany

Abstract

We introduce a method for finding a characteristic substructure for a set of molecular structures. Different from common approaches, such as computing the maximum common subgraph, the resulting substructure does not have to be contained in its exact form in all input molecules. Our approach is part of the identification pipeline for unknown metabolites using fragmentation trees. Searching databases using fragmentation tree alignment results in hit lists containing compounds with large structural similarity to the unknown metabolite. The characteristic substructure of the molecules in the hit list may be a key structural element of the unknown compound and might be used as starting point for structure elucidation. We evaluate our method on different data sets and find that it retrieves essential substructures if the input lists are not too heterogeneous. We apply our method to predict structural elements for five unknown samples from Icelandic poppy.

1998 ACM Subject Classification J.2 Physical Sciences and Engineering (Chemistry)

Keywords and phrases metabolites, substructure prediction, mass spectrometry, FT-BLAST

Digital Object Identifier 10.4230/OASIcs.GCB.2012.23

1 Introduction

The rapidly developing field of metabolomics deals with the detection, identification and quantification of low molecular-weight compounds (typically below 1000 Da). All organisms synthesize many different metabolites and a large portion of them remain uncharacterized regarding their structure and function [25]. Metabolites cover a wide array of compound classes and, due to their physical and chemical properties, show large structural diversity. The analysis and identification of small molecules plays an important role in biomarker discovery, diagnostics, pharmaceutical chemistry, and functional genomics [9, 21, 41].

Currently, no single instrumental platform can analyze all metabolites. Mass spectrometry (MS) is a key technique to analyze small molecules. It is orders of magnitude more sensitive than nuclear magnetic resonance (NMR). MS enables high throughput experiments and the amount of data produced is hard to process and analyze manually [19]. Usually, a combination of a chromatography with a fragmentation technique is used to obtain information beyond the compound mass. Most common combinations are gas chromatography MS (GC-MS) using electron impact (EI) fragmentation and liquid chromatography MS (LC-MS) using collision-induced dissociation (CID) for fragmentation.

The identification of unknown compounds is a major bottleneck in the interpretation of metabolomics MS data. When the compound is unknown, comparison with library entries or spectra of known standards will result in imprecise or incorrect hits, or no hits at all [10, 20].

For GC-MS spectra Demuth *et al.* [7] propose a method for finding similar compounds in a spectral library if the sample molecule is not contained. Further, methods for the identification of chemical substructures of the unknown sample molecule based on the direct comparison of fragmentation spectra have been proposed [17, 34, 37, 39]. For the automated analysis of LC-MS there are few computational pioneering studies [13, 14, 26, 42].

Besides classification of unknown compounds, not much progress has been made towards the *de novo* interpretation of mass spectra of small molecules, that cannot be found in any, not even a structural, database. When manually analyzing fragmentation mass spectra experts annotate fragmentation peaks and pathways to explain the data and determine the molecular structure. Böcker and Rasche [3] developed a method for the *de novo* interpretation of such spectra, resulting in hypothetical fragmentation trees. Fragmentation tree nodes are annotated with the molecular formula of the fragments, whereas edges represent fragmentation events, that is, neutral or radical losses. Here the computational analysis does not require any knowledge about compound structures or databases of compound structures or mass spectra. Only lists of common and implausible losses are required as expert knowledge about fragmentation mechanisms. Recently, methods to calculate fragmentation trees from multiple MS and GC-MS data have been developed [16, 32].

Rasche *et al.* [28] propose local tree alignments for the automated comparison of fragmentation trees. A local tree alignment contains those parts of the two trees where similar fragmentation cascades occurred. The authors could show, that this method is superior to spectral comparison in applications such as database searching. Fragmentation tree alignments even allow for inter-dataset comparisons for data sets measured on different instruments [28]. The authors present FT-BLAST, a database search tool for the identification of unknown metabolites based on fragmentation tree alignment. The received hit lists contain compounds with large structural similarity to the unknown metabolite. The common substructure of these compounds can be a starting point for its structural elucidation. A similar approach has been proposed by [31].

Finding the largest substructure common to a collection of graphs is denoted as maximum common subgraph (MCS) problem [29]. This problem is NP-hard even for two graphs [11]. There exists a rich literature in chemoinformatics and molecular modelling on this topic, often targeted at searching in molecular databases [27]. See Brown [5] for an introduction to chemoinformatics, and Raymond and Willet [29] for a review of exact and approximation algorithms for the maximum common subgraph problem. When computing the MCS for a set of molecules no deviation to the subgraphs in the molecules is allowed. This rather strict definition may not reflect the chemical similarity between compounds [36]. Due to their different physical and chemical properties, the structure of metabolites is heterogeneous even within the same compound classes. Finding the maximum common substructure to the hit lists received from FT-BLAST will often result in a very small structure that is not meaningful, or even in a single (carbon-) atom or an empty graph. An alternative approach is searching for frequent substructures in the molecule set. Frequent subgraph mining has been studied extensively in the last decade [4, 12, 15, 18, 23, 24, 43].

In this work, we loosen the strict definition of a common substructure that has to be contained in its exact form in either some or all of the input molecules. We rather try to find a *characteristic substructure* that is build of frequent (representative) substructures and reflects the specific features of the molecule set. We present a method to compute this structure and evaluate it on different compound classes and hit lists received from FT-BLAST. We find that our method is suitable to deal with structural outliers and retrieves characteristic substructures if the input lists are not too heterogeneous. We use this method to predict substructures for unknown samples from Icelandic poppy.

In addition we developed *MoleculePuzzle* to combine this characteristic substructure with information from fragmentation trees, in silico fragmentation prediction [13, 26, 33, 42], rule based substructure prediction [17], or structural isomer generators [8]. This tool can help with the assembly of structural pieces to elucidate the structure of an unknown compound.

2 Molecule graphs

We model the chemical structures of molecules using an undirected, labeled graph $M = (V, E)$ with vertices V representing the atoms of the molecule, and edges E representing the covalent bonds. Vertices are labeled with the corresponding element. Edge labels consider single or multiple bonds. In the following, we call these graphs *molecule graphs.* Molecule graphs only reflect the topology of the molecule and do not indicate the geometric distance between a pair of atoms (vertices). As hydrogen atoms are of limited importance for elucidating the core structure of a molecule, they are ignored in the following.

Two graphs are *isomorphic* if there is a one-to-one correspondence between their vertices such that an edge exists between two vertices in one graph if and only if an edge exists between the two corresponding vertices in the other graph. A (vertex-) *induced subgraph* of M is a set of vertices $S \subseteq V$ and those edges from M that connect two vertices from S. A graph $M_{1,2}$ is a *common induced subgraph* of graphs M_1 and M_2 if $M_{1,2}$ is isomorphic to induced subgraphs of M_1 and M_2. A *maximum common induced subgraph* (MCIS) is the common induced subgraph with maximum number of vertices. The related *maximum common edge subgraph* (MCES) is a subgraph consisting of the largest number of edges common to both M_1 and M_2.

Since the structures of metabolites are heterogeneous, finding the MCS (either MCIS or MCES) will often result only in small subgraphs or even a single vertex. In this work we loosen the strict definition of an MCS and try to find a *characteristic substructure* CS, that is a graph reflecting the characteristics of a set of molecule graphs best.

3 Methods

To compute a *characteristic substructure* CS for a set $\mathcal{M}$ of m molecule graphs $M_1, \ldots, M_m$ we start with the computation of representative paths, since paths can be found fast in (molecule) graphs. The relative frequency $h_{\mathcal{M}}(p)$ of a path p is the number of molecule graphs $M_i \in \mathcal{M}$ that contain a path q that is isomorphic to p divided by the total number of molecule graphs. We call a path *representative* if it has a relative frequency above a certain threshold $h_t \in [0, 1]$. For the computation of representative paths edge labels are ignored.

A path p induces a subgraph $M(p)$ containing all edges from M that connect two vertices v_i, v_j from p. We call this subgraph *path structure* P. Note that two isomorphic paths do not necessarily have to induce two isomorphic subgraphs (see Figure 1). Path structures contain the information that is necessary to construct the characteristic substructure. The relative

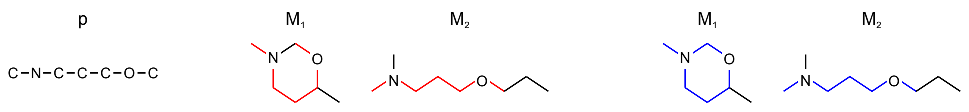


Figure 1 Molecules M_1 and M_2 both contain a path isomorphic to path p (red). The two isomorphic paths induce two subgraphs in M_1 and M_2 that are not isomorphic (blue).

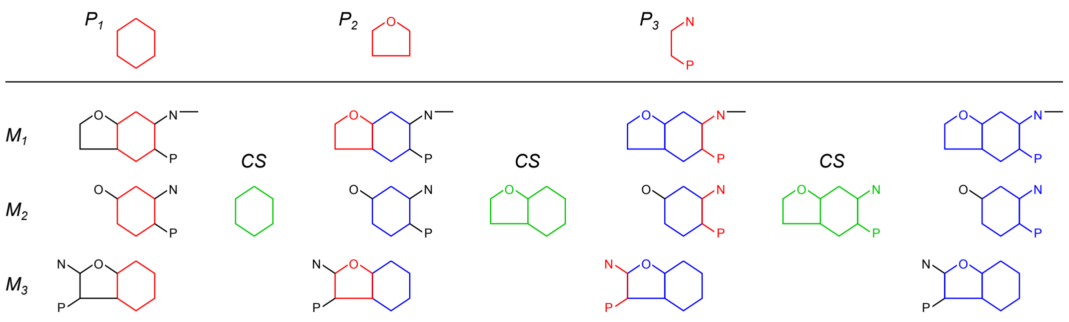


Figure 2 Adding path structures to the *characteristic substructure* CS. The figure is not a snapshot of a specific stage of the algorithm but rather an illustration of how to build up the CS from several path structures. C atoms are implicit. The drawn structures are not meant to depict real molecules. First, all subgraphs isomorphic to the first path structure P_1 are located in the molecule graphs (red) and P_1 is used as skeletal structure to build CS (green). We store the locations of the subgraphs (blue) to locate the subgraphs isomorphic to the next path structures. The second path structure P_2 is only isomorphic to subgraphs in M_1 and M_2 (red). Location of the subgraphs is the same in both molecules. Path structure P_2 is added to CS at this location. Path structure P_3 is isomorphic to subgraphs in all molecule graphs (red). The subgraphs in M_1 and M_2 are located at the same position while the subgraph in M_3 is located somewhere else. Path structure P_3 is added to CS at the most frequently occurring location and the subgraphs in M_1 and M_2 are marked (blue).

frequency $h_{\mathcal{M}}(P_i)$ of a path structure P_i is the number of molecule graphs $M_j \in \mathcal{M}$ that contain a subgraph that is isomorphic to P_i divided by the total number of molecule graphs. We call a path structure *representative* if it has a relative frequency above a certain threshold h_t. For the computation of representative path structures, edge labels are reconsidered.

For a certain length l we compute all representative paths of length l, find all representative path structures induced by these paths, and add these path structures to the characteristic substructure. We start with a certain length l_{start} decreasing it stepwise until we find a path structure that can be added to the characteristic substructure. Afterwards we increase the step size to s to skip path structures that add only few information to the characteristic substructure. We stop if $l < l_{\text{end}}$ since adding shorter paths seems to worsen the results.

Assume we have computed all representative path structures for a certain length l. We sort these path structures by their relative frequency in descending order $P_1, \ldots, P_n$. Lets assume that each path structure is isomorphic to at most one induced subgraph in each molecule graph. We start building the *characteristic substructure* CS with the most frequent path structure P_1 using it as skeletal structure. After P_1 is added to CS we remember the locations of all subgraphs within the molecule graphs that are isomorphic to P_1 (see Figure 2).

By definition path structure P_i is a common induced subgraph of at least h_t molecule graphs $M_j \in \mathcal{M}$. To add P_i to CS we have to compare the locations of these subgraphs with the location of the subgraphs of recently added $P_1, \ldots, P_{i-1}$. Comparison can be done easy, since we have stored the locations of all recently added path structures in the molecule graphs. Path structure P_i gets the location in CS that most of the subgraphs have within the molecules. Again, we memorize this subgraph location for the molecules in which it occurs (see Figure 2).

If a path structure P_i is isomorphic to more than one subgraph in a molecule graph we want to add all subgraphs instead of choosing one. If at least $k \geq 1$ isomorphic subgraphs within the same molecule graph in at least $|\mathcal{M}| \cdot h_{\text{iso}}$ molecule graphs occur, we proceed as

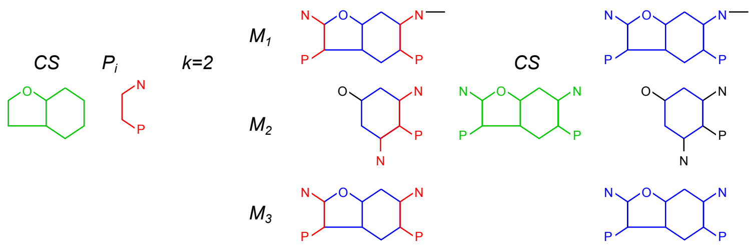


Figure 3 Adding a path structures that is isomorphic to $k = 2$ subgraphs in all molecule graphs. C atoms are implicit. The drawn structures are not meant to depict real molecules. In all molecules we consider the two subgraphs isomorphic to P_i as a single subgraph (red). This subgraph is disconnected in M_1 and M_3, and connected in M_2. We add the most occurring subgraph to CS (green) and mark its location in M_1 and M_3 (blue).

follows: In all molecules that contain exactly k subgraphs isomorphic to P_i we consider these k subgraphs as a single subgraph. This subgraph is either connected or disconnected. We compare the location of these subgraphs and add the most occurring to CS as described above (see Figure 3).

4 MoleculePuzzle

Computing a *characteristic substructure* for a set of molecules that have presumably large structural similarity to an unknown compound will greatly support its structure elucidation. In combination with the information from the fragmentation tree this substructure can be used to "puzzle" the molecular structure of the unknown compound. If need be, molecular isomer generators [8] can be used to enumerate all structural isomers of the annotated fragment formulas.

We present a puzzle tool to assemble molecular structures (see Figure 7 in the Appendix). *MoleculePuzzle* is a JAVA based plugin for the SIRIUS^2 framework[1] [2]. It is based on JChemPaint [22] which provides an interface that allows to draw chemical compounds. JChemPaint already contains templates for ring structures and different bond types and allows atom coloring, different render settings, as well as loading and storing structures from and into various file formats. In addition, it offers automated highlighting of chemically incorrect structures, which simplifies structure assembly.

We extend JChemPaint to work with a list of molecular structures which are transformed into puzzle pieces. These pieces can be added to the painting panel to modify the structures and fit them together. JChemPaint's drag and drop feature indicates when it is possible to connect one structure to another. *MoleculePuzzle* helps to assemble several structural pieces to a full structure.

5 Experimental results and discussion

We evaluate our method on three different data sets: First, we compute characteristic substructures for molecules from the same compound class and compare these structures to all molecules within this class. Second, we compute the characteristic substructures for molecules in the hit lists received from the FT-BLAST search tool for a reference data set

[1] `http://bio.informatik.uni-jena.de/sirius2/`

and compare them to the structure of the query compound. Third, we use our method on FT-BLAST hit lists for structure prediction of unknowns from Icelandic poppy.

To evaluate the quality of the computed substructures we use binary fingerprint representations to compute Tanimoto similarity scores (Jaccard indices) [30] and Tversky similarity scores [38]. We use fingerprints of the PubChem database [40] in the Chemistry Development Toolkit version 1.3.7 [35] for our computations. To compute the asymmetric Tversky similarity we weight the features only occurring in the CS with 1 and these only occurring in input molecules with 0. Tversky similarity gives an indication whether the CS is a substructure of the input molecules. Tanimoto similarity indicates whether the CS reconstructs important features of the input list.

In the following, we choose $l_{\text{start}} = 20$, $l_{\text{end}} = 5$, $s = 4$, $h_{\text{iso}} = 0.8$, $h_t = 0.8$. We implemented the algorithm in Java and carried out all computations on a quad-core Intel® Core™ i7-820QM with 8 GB RAM under Windows 7 operating system.

5.1 Characteristic substructures for compound classes

This data set consists of 395 molecules from 15 different compound classes (based on MeSH[2] categories) downloaded from PubChem Compound[3] (see Table 3 in the Appendix for CIDs). For each compound class, we compute the characteristic substructure (see Figure 8 in the Appendix). Computation required 6.4 s on average.

We calculate a score Tanimoto$_{CS}$ that is the average of Tanimoto scores of all input molecules to the CS (see Table 1). To report the average Tanimoto structural similarity score of a class, we calculate pairwise Tanimoto scores for all molecules in the class. Tanimoto$_{\text{class}}$ is the average of all these pairwise Tanimoto scores. This score may be seen as an upper bound for Tanimoto$_{CS}$. Tversky$_{CS}$ is the average of Tversky scores of all input molecules to the CS.

Table 1 Quality of the characteristic substructures for 15 metabolite classes. Tanimoto$_{\text{class}}$ is the average of all pairwise Tanimoto scores within the compound class. Tanimoto$_{CS}$ (Tversky$_{CS}$) is the average Tanimoto (Tversky) score of all input molecules to the CS.

compound class	# of molecules	Tanimoto$_{\text{class}}$	Tanimoto$_{CS}$	Tversky$_{CS}$	running time (s)
2-acetylaminofluorenes	12	0.88	0.82	0.99	1.6
adenines	44	0.81	0.71	0.97	8.1
benzothiadiazines	25	0.64	0.54	0.99	4.3
chlorothiazides	24	0.58	0.51	0.98	7.1
cytosines	22	0.62	0.45	0.98	1.5
erythromycins	24	0.77	0.53	0.99	8.0
glucosinolates	66	0.67	0.52	1	4.9
guanines	18	0.62	0.26	0.97	2.7
lipids	22	0.92	0.37	1	28.0
neuraminic acids	14	0.77	0.57	0.97	1.8
peroxides	24	0.59	0.18	0.98	4.7
pregnadienes	26	0.79	0.68	0.99	12.5
thymines	24	0.66	0.49	1	0.9
trichothecenes	26	0.86	0.74	1	8.9
uraciles	24	0.56	0.46	1	0.7

[2] http://www.nlm.nih.gov/mesh/
[3] http://www.ncbi.nlm.nih.gov/pccompound

For all compound classes, Tversky_{CS} is at least 0.97, indicating that the CS is contained in all molecules to a large extend. For 9 compound classes, Tanimoto_{CS} is above 0.5. We argue that $\text{Tanimoto}_{CS} > 0.5$ indicates good quality of the result, as large parts of the input structures have been reconstructed. From the remaining, we found that for three classes the difference between Tanimoto_{CS} to $\text{Tanimoto}_{\text{class}}$ is less than 0.2, so the classes already seem to be very heterogeneous and therefore the resulting substructures are only small. For the other three compound classes the difference is larger than 0.2: For guanines our method only found the 6-membered-ring but unfortunately the 5-membered-ring is missing. Thus Tanimoto_{CS} is low (0.26). Lipids are typically build of several 6-membered-rings and several carbon sidechains of different lengths. Due to the different number of rings and sidechains, our method only reconstructs a single 6-membered-ring with the typical oxygen and phosphor structure and a carbon sidechain of eleven atoms connected via a nitrogen. We argue, that this is the main substructure which can be duplicated and expanded (for example using the *MoleculePuzzle* tool). Our method works worst for the class of peroxides, which we found to be structurally more heterogeneous (at least for our method) than the Tanimoto score suggests.

For erythromycins the difference between Tanimoto_{CS} and $\text{Tanimoto}_{\text{class}}$ is also above 0.2. Erythromycins are huge compounds (56 heavy atoms on average) all containing a macrocycle (14 atoms). Our method reconstructs this ring correctly but has problems with the different sidechains, resulting in a somewhat cluttered side structure. Nevertheless, Tanimoto_{CS} is still above 0.5 and Tversky_{CS} is 0.99.

5.2 Characteristic substructures for FT-BLAST hit lists

We compute the characteristic substructures for hit lists reported by FT-BLAST [28]. Rasche *et al.* [28] evaluated their method on 97 compounds measured on an Orbitrap XL instrument using Collision Induced Dissociation (CID) or High-energy Collision Dissociation (HCD) for fragmentation. They computed fragmentation trees for all compounds and carried out a *leave-one-out* FT-BLAST search. For each compound they removed the correct answer from the database and searched for the compound in the remainder. They reported hits up to a False Discovery Rate (FDR) of 30%. For the 60 reported lists with at least two hits we compute characteristic substructures and compare them to the structure of the query compound. We also use the hit lists with FDR 20 % (57 lists with at least two hits) and 10 % (56 lists with at least two hits).

To report the average structural similarity of the hits returned by FT-BLAST, we calculate the Tanimoto score of the query compound to each hit list entry. $\text{Tanimoto}_{\text{hitlist}}$ is the

Table 2 Quality of the characteristic substructures computed for FT-BLAST hit lists. $\text{Tanimoto}_{\text{hitlist}}$ is the average of all Tanimoto scores of the query compound to the hit list entries. Tanimoto_{CS} (Tversky_{CS}) is the Tanimoto (Tversky) score of the query compound to the CS. We average $\text{Tanimoto}_{\text{hitlist}}$ over all hit lists and Tanimoto_{CS} (Tversky_{CS}) over all non empty substructures.

FDR	nr of hit lists	average hit list length	average $\text{Tanimoto}_{\text{hitlist}}$	empty CS	average Tanimoto_{CS}	average Tversky_{CS}	running time (s)
30 %	60	11	0.76	9	0.49	1	1.4
20 %	57	9	0.77	5	0.50	0.99	1.1
10 %	56	8	0.79	4	0.50	0.99	0.9

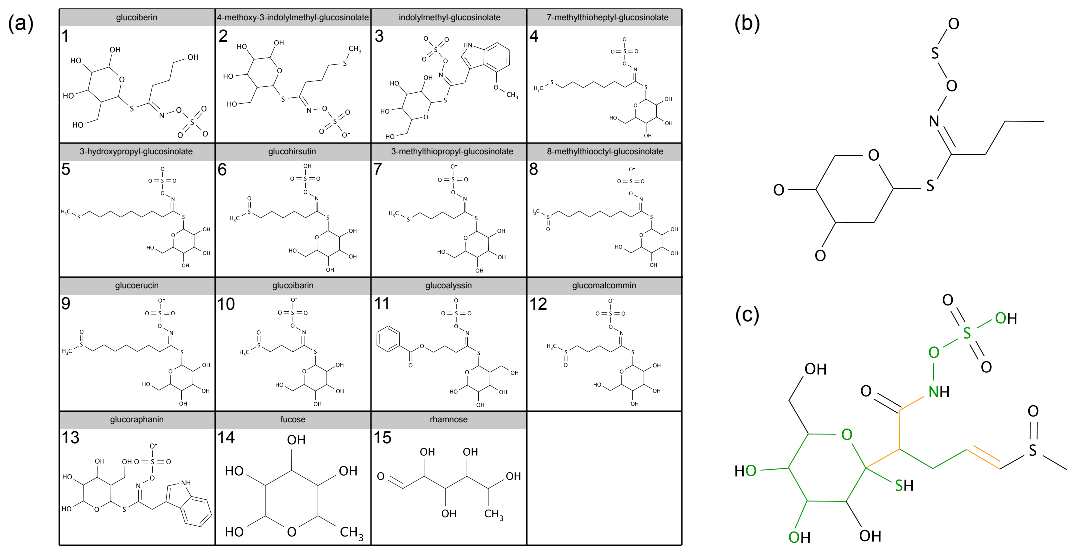

(a)
glucoiberin
4-methoxy-3-indolylmethyl-glucosinolate
indolylmethyl-glucosinolate
7-methylthioheptyl-glucosinolate
3-hydroxypropyl-glucosinolate
glucohirsutin
3-methylthiopropyl-glucosinolate
8-methylthiooctyl-glucosinolate
glucoerucin
glucoibarin
glucoalyssin
glucomalcommin
glucoraphanin
fucose
rhamnose
(b)
(c)

Figure 4 Characteristic substructure for the FT-BLAST hit list of glucoraphenin (CID 6443008). (a) The FT-BLAST hit list with FDR 30 % contains several glucosinolates (1-13) and two sugars (14,15). Computing the maximum common substructure would result in a very small structure. The characteristic substructure computed by our method (b) is contained in the query compound (c) (green) with slight variation (orange). Note that H atoms are ignored by our method.

average over all these scores. Again, this is somewhat an upper bound for the Tanimoto score between the query compound and the characteristic substructure (Tanimoto$_{CS}$). The average Tanimoto$_{\text{hitlist}}$ similarity for the complete dataset, using the leave-one-out strategy described above, is 0.76 for an FDR of 30 % and increases slightly with decreasing FDR to 0.79 (see Table 2).

For each hit list, we compute the characteristic substructure (see Table 2 for an overview of the results). Computation required on average 1.4 s for the hit lists with FDR 30 % (average hit list length 11), 1.1 s for hit lists with FDR 20 % (average hit list length 9), and 0.9 s for hit lists with FDR 10 % (average hit list length 8). We calculate Tanimoto$_{CS}$ (Tversky$_{CS}$), that is the Tanimoto (Tversky) similarity between the query compound and the characteristic substructure. For the hit lists with FDR 10 % and 20 % Tanimoto$_{CS}$ was 0.5 on average and for FDR 30 % it was about the same, namely 0.49. Tversky$_{CS}$ was 0.99 for 10 % and 20 % FDR and even increased to 1.0 for FDR 30 %. Since using the hit lists with smaller FDR seems not to improve the results significantly, we use the FDR 30 % hit lists reported by Rasche *et al.* [28] for further evaluation (see Table 2 in [28]).

The method works best if many compounds of the same class are included in the hit lists (as for sugars, zeatins and glucosinolates). Nevertheless, the method also computes good results even if some outliers are contained in the list. For example the hit list for an anthocyanin (CID 44256805) contains benzopyrans, amino acids, anthocyanins and one sugar. The method finds the main component, a 6-membered-ring with an oxygen and a carbon side atom. This ring is contained several times in the structure of the query compound. Another example is glucoraphenin (see Figure 4). Computing the maximum common substructure would have result in a very small, as fucose and rhamnose do not share the large structure with the remaining hits. Our method computes a characteristic substructure which is close to the query compound.

Very heterogeneous hit lists are hard to process. For 9 query compounds the characteristic

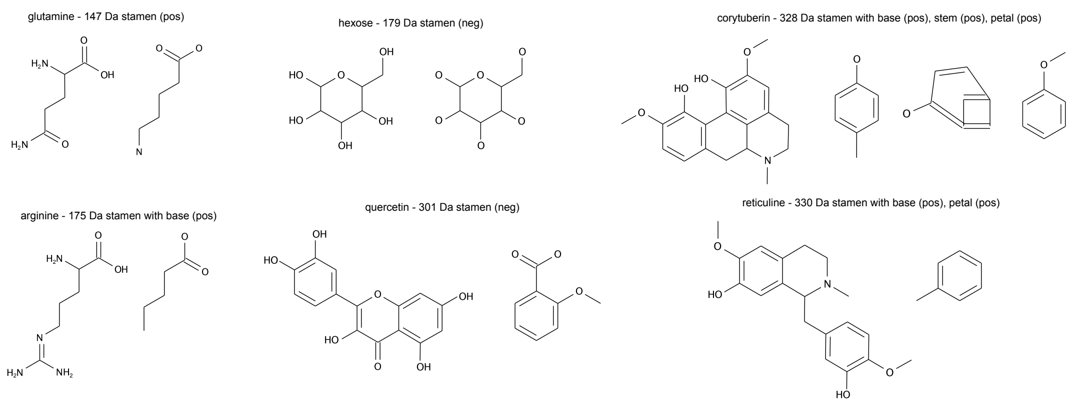

glutamine - 147 Da stamen (pos)
hexose - 179 Da stamen (neg)
corytuberin - 328 Da stamen with base (pos), stem (pos), petal (pos)
arginine - 175 Da stamen with base (pos)
quercetin - 301 Da stamen (neg)
reticuline - 330 Da stamen with base (pos), petal (pos)

Figure 5 Resulting characteristic substructures for the FT-BLAST hit lists for the six compounds from Icelandic poppy that were manually identified. For each sample, the query compound (left) is compared to the resulting substructure(s) (right). For corytuberin the hit lists for the different samples from stamen, stem and petal supply different characteristic substructures that could be further combined with the *MoleculePuzzle* tool. For reticuline the samples from stamen and petal result in the same substructure, while no substructure was found for the stem sample. Processing the hit lists of the samples for the unspecified palmatine derivates (370 Da and 386 Da) results in the same substructure as for reticuline. Note that H atoms are ignored by our method.

substructure was empty. For other compounds (for example bergapten), the characteristic substructure looks somewhat cluttered.

5.3 Characteristic substructures for unknowns from Icelandic poppy

As a real-world example Rasche *et al.* [28] tried to identify unknown metabolites from Icelandic poppy (*P. nudicaule*). They computed fragmentation trees for 32 compounds and compared them with the Orbitrap data set as reference using FT-BLAST. Again, they reported hits up to an FDR of 30 %. We use this lists, since smaller FDRs seem not to improve the results for the previous data set.

Eight compounds from the dataset were manually identified by Rasche *et al.* [28]. For arginine, glutamine, quercetin and a hexose the correct compound was in the hit list. FT-BLAST results for reticuline (330.17 Da) and corytuberine (328.15 Da) included chemical precursors of these alkaloids. Two other unknowns (370 and 386 Da) were manually classified as palmatine-derivatives. For all these compounds our method predicts correct substructures (see Figure 5).

We use our method to predict substructures for the remaining unknown compounds in this data set. For one compound only one hit is received and substructure prediction is not necessary. For seven compounds the hit lists are very heterogeneous ($\text{Tanimoto}_{\text{hitlist}}=0$) and our method finds no characteristic substructure. This can be attributed to the small database (97 compounds) we are searching in. For the remaining five samples we compute characteristic substructures (see Figure 6) that can be used for further structure elucidation of the compounds.

6 Conclusion

We have developed a method for computing a characteristic substructure for a set of molecules. Different from the maximum common substructure this substructure does not have to be

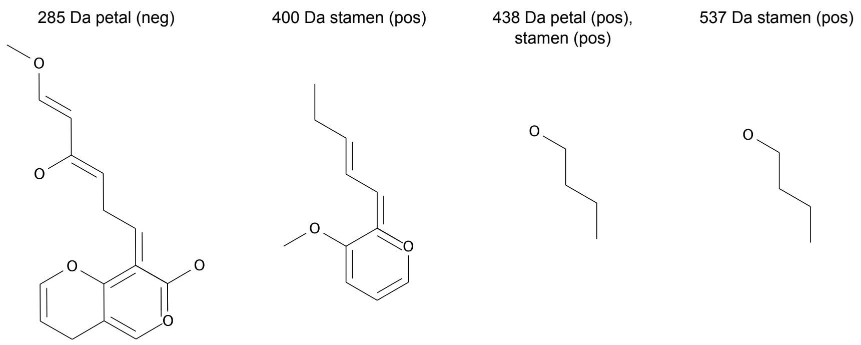


Figure 6 Substructure prediction for five samples from Icelandic poppy that are unknown. FT-BLAST hit list for 285 Da petal (neg) contains six compounds with $\text{Tanimoto}_{\text{hitlist}} = 0.6$. The hit list for 400 Da stamen (pos) contains six compound with $\text{Tanimoto}_{\text{hitlist}} = 0.4$. For both compounds we compute large characteristic substructures. The hit lists for 438 Da petal (pos) and 438 Da stamen (pos) both contain ten compounds with $\text{Tanimoto}_{\text{hitlist}} = 0.1$. FT-BLAST hit list for 537 Da stamen (pos) contains twelve compounds with $\text{Tanimoto}_{\text{hitlist}} = 0.1$. The resulting characteristic substructures for these three samples are rather small.

contained in its exact form in all input molecules. Finding characteristic substructures is an important step in the identification of unknown metabolites. It is part of a pipeline which is based on the computation of fragmentation trees from mass spectral data. Fragmentation tree alignment allows to find similar, not necessarily identical, compounds in a library search. Characteristic substructures of these hits may be key structural elements of the unknown metabolite.

We have evaluated our method on classes of molecules and FT-BLAST hit lists. We found that our method reconstructs many structural features contained in the input lists. Different from finding the maximum common substructure our method can deal with structural irregularities. Nevertheless, if the input lists are to heterogeneous no characteristic substructure can be predicted. We used the Tanimoto score for estimating structural homogeneity, but by visual inspection we found that some input lists are structurally more heterogeneous (at least for our method) than the Tanimoto score suggests.

We have predicted substructures for five unknown samples from Icelandic poppy. Substructure prediction is strongly dependent on the FT-BLAST hit lists. The reference data set for FT-BLAST used in the study of Rasche *et al.* [28] was pretty small. The resulting hit lists will become more homogeneous as more reference compounds become available, and we expect that the predicted substructures will be better and probably larger.

The presented method is not taking atom valences into account, but rather adopts the bond order from the input molecules. Bond orders can be corrected using automated bond order assignment [1,6]. The characteristic substructure and information from fragment formulas can be assembled in our *MoleculePuzzle* tool with results from in silico fragmentation prediction [13, 26,33,42], rule based substructure prediction [17], or structural isomer generators [8]. Together, the two presented tools form an important step towards the structure elucidation of unknown compounds.

Acknowledgments F. Hufsky was supported by the International Max Planck Research School Jena.

References

1 Sebastian Böcker, Quang Bao Anh Bui, and Anke Truss. Computing bond orders in molecule graphs. *Theoretical Computer Science*, 412(12–14):1184–1195, 2011.

2 Sebastian Böcker, Matthias Letzel, Zsuzsanna Lipták, and Anton Pervukhin. SIRIUS: Decomposing isotope patterns for metabolite identification. *Bioinformatics*, 25(2):218–224, 2009.

3 Sebastian Böcker and Florian Rasche. Towards de novo identification of metabolites by analyzing tandem mass spectra. *Bioinformatics*, 24:I49–I55, 2008. Proc. of *European Conference on Computational Biology* (ECCB 2008).

4 Christian Borgelt and Michael R. Berthold. Mining molecular fragments: Finding relevant substructures of molecules. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002)*, page 51. IEEE Computer Society, 2002.

5 Nathan Brown. Chemoinformatics — an introduction for computer scientists. *ACM Comput Surv*, 41(2):8, 2009.

6 Anna Katharina Dehof, Alexander Rurainski, Quang Bao Anh Bui, Sebastian Böcker, Hans-Peter Lenhof, and Andreas Hildebrandt. Automated bond order assignment as an optimization problem. *Bioinformatics*, 27(5):619–625, 2011.

7 W. Demuth, M. Karlovits, and K. Varmuza. Spectral similarity versus structural similarity: mass spectrometry. *Anal Chim Acta*, 516(1-2):75–85, 2004.

8 Jean-Loup Faulon, Donald P. Visco, and Diana Roe. Enumerating molecules. In Kenny B. Lipkowitz, Raima Larter, and Thomas R. Cundari, editors, *Reviews in Computational Chemistry*, volume 21, chapter 3, pages 209–286. John Wiley & Sons, Inc., 2005.

9 O. Fiehn, J. Kopka, P. Dörmann, T. Altmann, R. N. Trethewey, and L. Willmitzer. Metabolite profiling for plant functional genomics. *Nat Biotechnol*, 18(11):1157–1161, 2000.

10 Oliver Fiehn. Extending the breadth of metabolite profiling by gas chromatography coupled to mass spectrometry. *Trends Analyt Chem*, 27(3):261–269, 2008.

11 Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* 1979.

12 Ehud Gudes, Solomon Eyal Shimony, and Natalia Vanetik. Discovering frequent graph patterns using disjoint paths. *IEEE Trans Knowl Data En*, 18(11):1441–1456, 2006.

13 Markus Heinonen, Ari Rantanen, Taneli Mielikäinen, Juha Kokkonen, Jari Kiuru, Raimo A Ketola, and Juho Rousu. FiD: a software for ab initio structural identification of product ions from tandem mass spectrometric data. *Rapid Commun Mass Spectrom*, 22(19):3043–3052, 2008.

14 Dennis W. Hill, Tzipporah M. Kertesz, Dan Fontaine, Robert Friedman, and David F. Grant. Mass spectral metabonomics beyond elemental formula: Chemical database querying by matching experimental with computational fragmentation spectra. *Anal Chem*, 80(14):5574–5582, 2008.

15 Jun Huan, Wei Wang, and Jan Prins. Efficient mining of frequent subgraphs in the presence of isomorphism. In *Proceedings of the Third IEEE International Conference on Data Mining (ICDM 2003)*, page 549, 2003.

16 Franziska Hufsky, Martin Rempt, Florian Rasche, Georg Pohnert, and Sebastian Böcker. De novo analysis of electron impact mass spectra using fragmentation trees. *Anal Chim Acta*, 739:67–76, 2012.

17 Jan Hummel, Nadine Strehmel, Joachim Selbig, Dirk Walther, and Joachim Kopka. Decision tree supported substructure prediction of metabolites from GC-MS profiles. *Metabolomics*, 6(2):322–333, 2010.

18 Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In Jan Komorowski Djamel A. Zighed and Jan Zytkow, editors, *Proceedings of the 4th European Conference on Principles of Data*

Mining and Knowledge Discovery (PKDD 2000), volume 1910 of *Lect Notes Comput Sci*, pages 13–23. Springer, Berlin, 2000.

19 Haleem J Issaq, Que N Van, Timothy J Waybright, Gary M Muschik, and Timothy D Veenstra. Analytical and statistical approaches to metabolomics research. *J Sep Sci*, 32(13):2183–2199, 2009.

20 Tobias Kind and Oliver Fiehn. Advances in structure elucidation of small molecules using mass spectrometry. *Bioanal Rev*, 2(1-4):23–60, 2010.

21 Sing Teang Kong, Hai-Shu Lin, Jianhong Ching, and Paul C Ho. Evaluation of dried blood spots as sample matrix for gas chromatography/mass spectrometry based metabolomic profiling. *Anal Chem*, 83(11):4314–4318, 2011.

22 Stefan Krause, Egon Willighagen, and Christoph Steinbeck. JChemPaint - using the collaborative forces of the internet to develop a free editor for 2D chemical structures. *Molecules*, 5(1):93–98, 2000.

23 Michihiro Kuramochi and George Karypis. An efficient algorithm for discovering frequent subgraphs. *IEEE Trans Knowl Data En*, 16(9):1038–1051, 2004.

24 Siegfried Nijssen and Joost N. Kok. The Gaston tool for frequent subgraph mining. *Electron Notes Theor Comput Sci*, 127(1):77–87, 2005.

25 Gary J. Patti, Oscar Yanes, and Gary Siuzdak. Innovation: Metabolomics: the apogee of the omics trilogy. *Nat Rev Mol Cell Biol*, 13(4):263–269, 2012.

26 Anna Pelander, Elli Tyrkkö, and Ilkka Ojanperä. In silico methods for predicting metabolism and mass fragmentation applied to quetiapine in liquid chromatography/time-of-flight mass spectrometry urine drug screening. *Rapid Commun Mass Spectrom*, 23(4):506–514, 2009.

27 Syed Asad Rahman, Matthew Bashton, Gemma L Holliday, Rainer Schrader, and Janet M Thornton. Small molecule subgraph detector (SMSD) toolkit. *J Cheminform*, 1(1):12, 2009.

28 Florian Rasche, Kerstin Scheubert, Franziska Hufsky, Thomas Zichner, Marco Kai, Aleš Svatoš, and Sebastian Böcker. Identifying the unknowns by aligning fragmentation trees. *Anal Chem*, 84(7):3417–3426, 2012.

29 John W Raymond and Peter Willett. Maximum common subgraph isomorphism algorithms for the matching of chemical structures. *J Comput Aided Mol Des*, 16(7):521–533, 2002.

30 D. J. Rogers and T. T. Tanimoto. A computer program for classifying plants. *Science*, 132(3434):1115–1118, 1960.

31 Miguel Rojas-Cherto, Julio E. Peironcely, Piotr T. Kasper, Justin Johan Jozias van der Hooft, Ric C. H. De Vos, Rob J. Vreeken, Thomas Hankemeier, and Theo Reijmers. Metabolite identification using automated comparison of high resolution ms(n) spectral trees. *Anal Chem*, 84(13):5524–5534, 2012.

32 Kerstin Scheubert, Franziska Hufsky, Florian Rasche, and Sebastian Böcker. Computing fragmentation trees from metabolite multiple mass spectrometry data. In *Proc. of Research in Computational Molecular Biology (RECOMB 2011)*, volume 6577 of *Lect Notes Comput Sci*, pages 377–391. Springer, Berlin, 2011.

33 Emma Louise Schymanski, Christine M J Gallampois, Martin Krauss, Markus Meringer, Steffen Neumann, Tobias Schulze, Sebastian Wolf, and Werner Brack. Consensus structure elucidation combining GC/EI-MS, structure generation and calculated properties. *Anal Chem*, 84(7):3287–3295, 2012.

34 Stephen Stein. Chemical substructure identification by mass spectral library searching. *J Am Soc Mass Spectrom*, 6:644–655, 1995.

35 Christoph Steinbeck, Christian Hoppe, Stefan Kuhn, Matteo Floris, Rajarshi Guha, and Egon L Willighagen. Recent developments of the Chemistry Development Kit (CDK) - an open-source java library for chemo- and bioinformatics. *Curr Pharm Des*, 12(17):2111–2120, 2006.

36 Yoshimasa Takahashi, Yuzuru Satoh, Hajime Suzuki, and Shin-ichi Sasaki. Recognition of largest common structural fragment among a variety of chemical structures. *Anal Sci*, 3:23–28, 1987.
37 Hiroshi Tsugawa, Yuki Tsujimoto, Masanori Arita, Takeshi Bamba, and Eiichiro Fukusaki. GC/MS based metabolomics: development of a data mining system for metabolite identification by using soft independent modeling of class analogy (SIMCA). *BMC Bioinformatics*, 12:131, 2011.
38 Amos Tversky. Features of similarity. *Psychol Rev*, 84(4):327–352, 1977.
39 K. Varmuza and W. Werther. Mass spectral classifiers for supporting systematic structure elucidation. *J Chem Inf Comp Sci*, 36(2):323–333, 1996.
40 Yanli Wang, Jewen Xiao, Tugba O Suzek, Jian Zhang, Jiyao Wang, and Stephen H Bryant. PubChem: a public information system for analyzing bioactivities of small molecules. *Nucleic Acids Res*, 37(Web Server issue):W623–W633, 2009.
41 David S Wishart. Current progress in computational metabolomics. *Brief Bioinform*, 8(5):279–293, 2007.
42 Sebastian Wolf, Stephan Schmidt, Matthias Müller-Hannemann, and Steffen Neumann. In silico fragmentation for computer assisted identification of metabolite mass spectra. *BMC Bioinformatics*, 11:148, 2010.
43 Xifeng Yan and Jiawei Han. gSpan: Graph-based substructure pattern mining. In *Proceedings of the Second IEEE International Conference on Data Mining (ICDM 2002)*, pages 721–724, 2002.

A Appendix

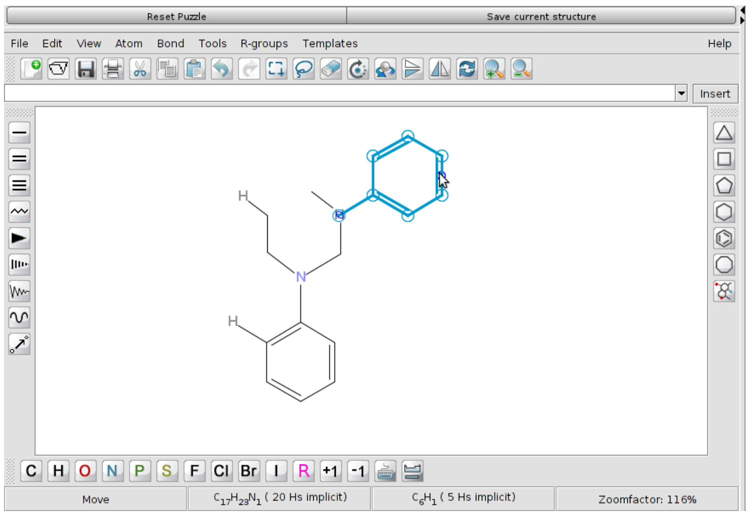

Reset Puzzle
Save current structure
File Edit View Atom Bond Tools R-groups Templates Help
Insert
H
N
H
C H O N P S F Cl Br I R +1 -1
Move
$C_{17}H_{23}N_1$ (20 Hs implicit)
C_6H_1 (5 Hs implicit)
Zoomfactor: 116%

Figure 7 Screenshot of the *MoleculePuzzle* plugin for the SIRIUS² framework. Several structural pieces can be puzzled together to receive the full structure of a molecule.

Table 3 PubChem CIDs for the 395 molecules from 15 compound classes.

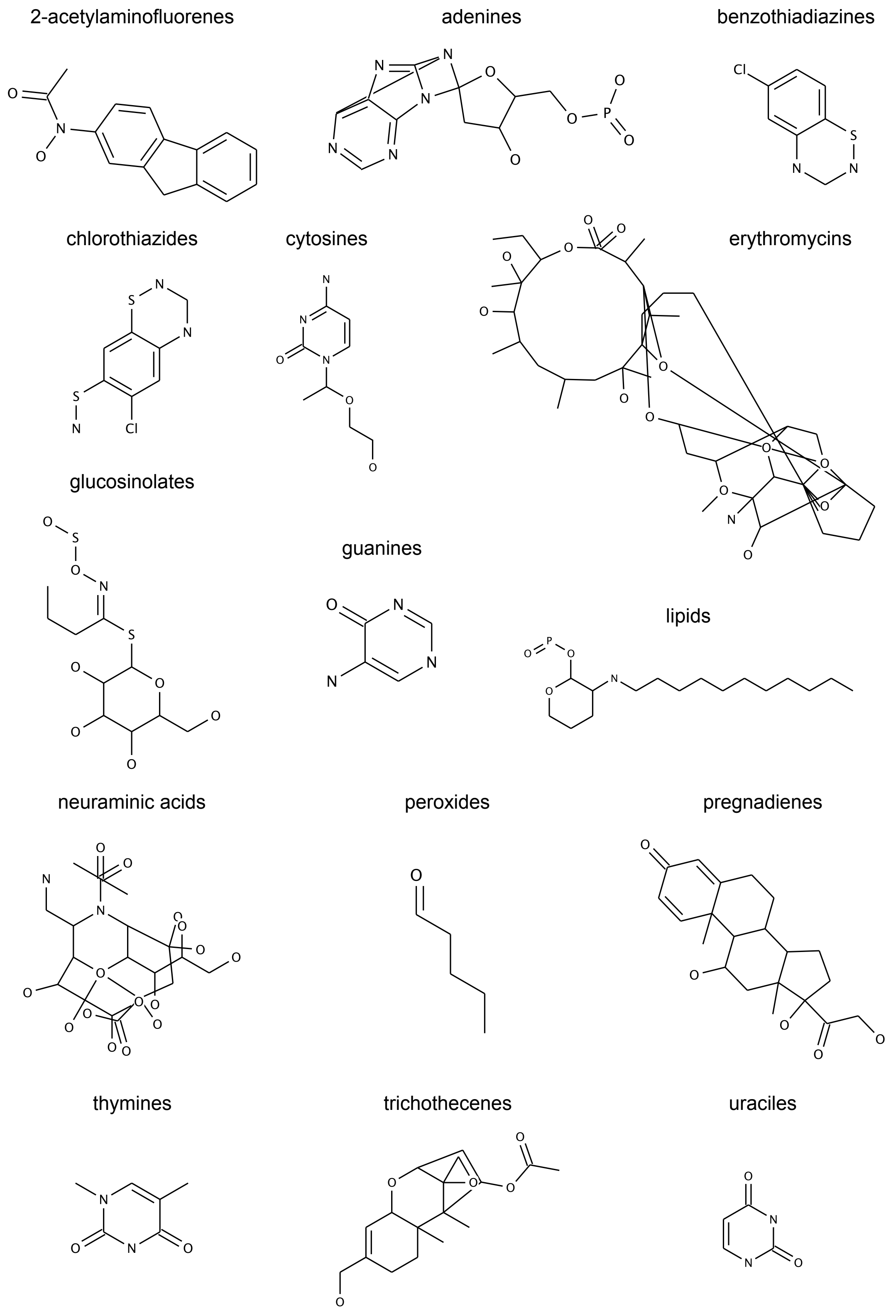
compound class	# molecules	PubChem CIDs
glucosinolates	66	23682211, 25244590, 46173878, 9548633, 9548621, 9576240, 9576416, 656498, 9548605, 656539, 656538, 9576241, 46173877, 44237373, 656547, 46173876, 44237368, 5281133, 656555, 656557, 46173880, 44237257, 656543, 656523, 44237260, 656527, 6325242, 9548892, 25244892, 441524, 46173879, 6442557, 5281135, 656568, 46173882, 9576738, 46173875, 656545, 656525, 7098673, 17756749, 656541, 656531, 46173881, 44237206, 5281136, 44237203, 9548619, 5281138, 46173884, 25245521, 6443008, 5281134, 656537, 25244538, 25244201, 17756744, 5281139, 46173883, 25246161, 25245774, 25244220, 25243874, 9548618, 656562, 656548
adenines	44	3083432, 3083316, 3082029, 3081390, 3080770, 3080762, 3036950, 703739, 466837, 465383, 440867, 25246029, 15938965, 12358355, 7059571, 7058055, 41211, 34768, 32014, 10238, 9687, 6083, 6076, 1913, 224, 50909893, 44134557, 25244014, 25201135, 23615303, 23615194, 23421209, 22848660, 16078938, 9578273, 9589376, 6992262, 6452236, 6449870, 6426627, 5748329, 5491933, 5399013, 4617095
cytosines	22	455597, 5276954, 597, 452713, 441224, 374908, 492030, 492031, 500131, 6473860, 471292, 477169, 477168, 467421, 455604, 455603, 455602, 455598, 16727509, 477170, 455605, 455601
guanines	18	764, 374910, 160219, 129161, 133387, 161069, 145817, 129136, 406591, 130450, 478537, 25082899, 471293, 485625, 485629, 485626, 485624, 195385

Continued on next page

Table 3 – continued from previous page

compound class	# molecules	PubChem CIDs
thymines	24	1135, 452067, 451954, 6439704, 135557, 6450954, 452068, 452063, 452715, 607039, 196362, 6477693, 370631, 406592, 6477695, 6477692, 492029, 465896, 445213, 495405, 374907, 457298, 485384, 452946
uraciles	24	1174, 6194, 18323, 9412, 5386, 68216, 13712, 5899, 5360852, 5282192, 688297, 6971263, 6029, 69672, 13268, 208432, 453162, 1177, 452948, 125110, 3067786, 55281, 456513, 54929
lipids	22	440885, 46173186, 46173313, 50909837, 46173444, 25246183, 25246208, 25200834, 46173394, 25246224, 3083382, 25202130, 25244473, 51351771, 51351791, 46173153, 50909852, 46173409, 127960, 13831140, 440886, 160295
benzothiadiazines	25	62940, 43148, 22425, 2910, 2348, 2343, 2122, 12933, 5560, 4870, 4121, 44154271, 44152755, 44151672, 6441852, 871720, 216293, 198367, 194167, 188359, 174783, 173791, 107748, 72070, 71652
trichothecenes	26	11968047, 11968045, 6540635, 6450461, 6444304, 6438947, 6438478, 6437354, 6437353, 45266518, 11969549, 6431315, 6321400, 5459303, 5284461, 3000635, 529495, 442403, 442400, 30552, 50987470, 50986319, 44144558, 44144549, 44144548, 11969469
chlorothiazides	24	127085, 116034, 107748, 71656, 62940, 2348, 2720, 5560, 3639, 50987261, 44151672, 44147212, 24847808, 23717274, 11354874, 3083286, 3083063, 242921, 216293, 193444, 188359, 174783, 172393, 159328
erythromycins	24	5284534, 3033819, 447043, 429694, 84029, 55185, 24847865, 17753754, 5748242, 5282045, 3002190, 44629874, 16212992, 6713919, 6426643, 83935, 83933, 12560, 44629879, 25102720, 17753750, 11969952, 9604450, 6915744
peroxides	24	5497123, 5464098, 641668, 637882, 45266618, 16760624, 16219283, 5311493, 445049, 1035, 45027791, 45027789, 44202131, 44145773, 25245484, 25244877, 25244708, 23690934, 22169438, 16394563, 6476300, 6543478, 6450800, 6454765
pregnadienes	26	63043, 63042, 63041, 62961, 45006164, 44266812, 24867475, 20054915, 23671691, 11957468, 11954369, 16490, 9793, 9782, 9642, 5876, 9571040, 6714002, 6713977, 6452749, 5388957, 5388959, 656804, 633091, 443958, 443936
2-acetylaminofluorenes	12	5897, 22469, 5896, 168033, 135827, 130776, 130694, 119334, 108117, 22722, 19347, 17270
neuraminic acids	14	46878426, 23679065, 20112027, 16760374, 6857396, 448209, 445063, 444885, 439197, 349960, 60855, 18292, 8565, 906

Figure 8 Characteristic substructures for the 15 different compound classes.



A Two-Step Soft Segmentation Procedure for MALDI Imaging Mass Spectrometry Data

Ilya Chernyavsky[1], **Theodore Alexandrov**[2,3], **Peter Maass**[2], **and Sergey I. Nikolenko**[1,4]

1 **St. Petersburg Academic University, ul. Khlopina, d. 8, korp. 3, St. Petersburg, 194021, Russia,** `ilya.chernyavsky@gmail.com`
2 **Center for Industrial Mathematics, University of Bremen, Bibliothekstr. 1, 28359 Bremen, Germany,** `theodore@uni-bremen.de`
3 **Steinbeis Innovation Center SCiLS, Richard-Dehmel-Str. 69, 28211 Bremen, Germany**
4 **Steklov Mathematical Institute, nab. r. Fontanka, 27, St. Petersburg, 191023, Russia,** `sergey@logic.pdmi.ras.ru`

Abstract

We propose a new method for soft spatial segmentation of matrix assisted laser desorption/ionization imaging mass spectrometry (MALDI-IMS) data which is based on probabilistic clustering with subsequent smoothing. Clustering of spectra is done with the Latent Dirichlet Allocation (LDA) model. Then, clustering results are smoothed with a Markov random field (MRF) resulting in a soft probabilistic segmentation map. We show several extensions of the basic MRF model specifically tuned for MALDI-IMS data segmentation. We describe a highly parallel implementation of the smoothing algorithm based on GraphLab framework and show experimental results.

1998 ACM Subject Classification I.4.6 Segmentation, J.3 Life and Medical Sciences, J.2 Physical Sciences and Engineering, H.2.8 Database Applications

Keywords and phrases MALDI imaging mass spectrometry, hyperspectral image segmentation, probabilistic graphical models, latent Dirichlet allocation, Markov random field

Digital Object Identifier 10.4230/OASIcs.GCB.2012.39

1 Introduction

1.1 MALDI-imaging mass spectrometry and the segmentation problem

Untargeted spatially-resolved biochemical analysis of biological tissue sections using imaging mass spectrometry (IMS) is an emerging field of biochemistry which has received considerable attention over the last decade [11,16,35], with the most popular IMS technique called matrix-assisted laser desorption/ionization (MALDI-IMS) or MALDI-imaging [9,32]. IMS acquires a set of mass spectra in a rasterized way across the surface of a prepared tissue section; one mass spectrum corresponds to one pixel. A mass spectrum is a vector of intensities associated with mass-to-charge (m/z) values representing in a semi-quantitative way the abundances of ions separated by their m/z-values; a high intensity of a mass spectrum at an m/z-value represents a high abundance of an ion of this m/z. In MALDI mass spectrometry, an m/z-value is usually interpreted as the molecular mass, since ions with charge $+1$ prevail. An IMS dataset is large, normally with around 100×100 spectra acquired across the section with each spectrum having around 10000 intensity values. An IMS dataset can be considered as a hyperspectral image whose image channels correspond to m/z-values.

Automatic mining of such massive hyperspectral imaging data is essential for applications of the IMS technique to biomedical challenges. One of the established methods of MALDI-IMS data mining is spatial segmentation that allows one to represent a hyperspectral MALDI-IMS dataset with just one image, the so-called segmentation map. In this map, pixels of similar mass spectra are pseudo-colored with the same color. There are several methods proposed for spatial segmentation of MALDI-IMS data based on clustering of mass spectra. Spatial segmentation of MALDI-IMS data is challenging because of the size of the datasets and strong pixel-to-pixel variation inherent to the sample preparation and acquisition techniques used for MALDI-IMS. It has been shown that the best-performing segmentation methods reduce pixel-to-pixel variation, e.g. by means of incorporating spatial relations between pixels into the distance function [2].

1.2 Results and structure of the paper

We propose a two-step procedure for spatial segmentation of MALDI-IMS data. On the first step, we cluster MALDI-IMS spectra with the probabilistic graphical model called Latent Dirichlet Allocation (LDA) [8]. LDA has recently obtained recognition in text mining because it can remain scalable and efficient while extracting hidden features (“topics”) from a set of documents. In the context of MALDI-IMS data segmentation, LDA has the following advantages as compared to other published methods: (i) reduced memory requirements as compared to hierarchical clustering recommended by [12]; (ii) reduced complexity as compared to high-dimensional discriminant clustering recommended by [1]; (iii) most importantly, additional information provided by LDA as compared to other clustering methods. This information associates each data point (pixel) in the LDA model with the probabilities to belong to each cluster and, moreover, the LDA model learns a distribution of the “words” (m/z values) to be generated by different topics; this “soft” clustering is better suited for further analysis such as extracting characteristic masses for each cluster. We describe our clustering procedures in Section 2.

On the second step, we smooth soft clustering results provided by LDA with a probabilistic smoothing model which we base upon Markov random fields. LDA results are convenient to use in such a probabilistic setting. We introduce several extensions to the model that let us enhance MRF smoothing results specifically for MALDI-imaging data processing; in particular, we learn the weights of the MRF network with an EM-like procedure. Moreover, we have developed a fast highly parallel implementation of MRF smoothing for clustering results based on the GraphLab parallelization paradigm. These results are described in Section 3. In Section 4, we describe the dataset and show experimental results of our segmentation procedures. Section 5 concludes the paper.

2 Clustering of MALDI-IMS spectra

2.1 LDA

Latent Dirichlet allocation (LDA) is a clustering model that was originally intended for use with natural language text processing [8, 24]. LDA takes the naive Bayes model one step further: while naive Bayes has only one latent variable, namely the topic, and words are conditionally independent given a topic, LDA view documents as *mixtures* of several topics. In essence, LDA is a hierarchical Bayesian model: (i) on the first level, it has a mixture whose components correspond to the “topics”; (ii) on the second level, a multinomial variable with Dirichlet prior that corresponds to the “distribution of topics” in a document.

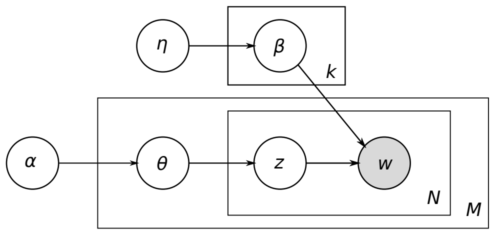


Figure 1 LDA graphical model.

This gives rise to the following generative model (see Fig. 1): (1) choose the number of words $N \sim p(N \mid \xi)$; (2) choose the topic distribution $\theta \sim \mathrm{Dir}(\alpha)$; (3) for each word w_n, $n = 1..N$: (i) choose a topic $z_n \sim \mathrm{Mult}(\theta)$; (ii) choose a word $w_n \sim p(w_n \mid z_n, \beta)$. The joint distribution in the LDA model (for a fixed number of topics k and parameters $\beta_{ij} = p(w^j = 1 \mid z^i = 1)$) is $p(\theta, \mathbf{z}, \mathbf{w}, N \mid \boldsymbol{\alpha}, \boldsymbol{\beta}) = p(N \mid \xi)p(\theta \mid \boldsymbol{\alpha}) \prod_{n=1}^{N} p(z_n \mid \theta)p(w_n \mid z_n, \boldsymbol{\beta})$. Inference in LDA can be done, for example, with variational methods or with Gibbs sampling.

The LDA model has had success in text processing, but its applications are not limited to information retrieval problems. In this work, we have applied LDA to the clustering of the spectra. Note that in MALDI-imaging, one spectrum is acquired at one pixel. We propose to use individual pixels of the image as "documents" and masses corresponding to peaks in the spectrum as "words". The results of applying LDA to such "documents" are topic distributions in each pixel of the image and a mass distribution for every topic. The number of topics is a parameter defined in advance. Fig. 3a shows the results of a direct application of LDA with 10 topics to the original dataset. Each color corresponds to a topic, and a pixel represents a topic with maximal posterior probability. Fig. 3b shows a soft segmentation map representing the second, fourth, and tenth clusters in the segmentation map from Fig 3a. A soft segmentation map is an RGB-image where three clusters are encoded with red, blue, and green channel respectively. Intensities of each channel represent probabilities of a pixel to belong to the corresponding cluster. For example, a highly red pixel has high probability to be in the second cluster; a blue pixel probably belongs to the fourth cluster, and a green pixel probably belongs to the tenth cluster. Compared to a standard segmentation map (Fig. 3a), a soft segmentation map, although restricted to visualize only three clusters, provides more detailed information on probability distribution of pixels among the clusters. In particular, it shows pixels with comparable probabilities of belonging to different clusters.

2.2 k-means

For comparison, we have implemented k-means as a simple yet effective clustering algorithm [21, 23]. In k-means, we fix the number of clusters and initialize their centers (usually at random dataset points that are chosen to be relatively far from each other). Then k-means proceeds in alternating steps: (1) assign points to clusters; each point is assigned to the nearest cluster center; (2) move cluster centers to the centers of mass for the points assigned to them.

3 Smoothing probabilistic clustering results

3.1 Markov random fields

An *undirected graphical model* (or *Markov random field*) [5, 22, 27, 29] is a representation of a complex probability distribution factorized into a product of *potentials*; the model consists of an undirected graph with vertices corresponding to random variables $X_1, \ldots, X_k$ and a potential for each maximal clique in this graph, i.e., a nonnegative function of the variables; the joint probability distribution corresponding to the model is the product of potentials: $p(X_1 = x_1, \ldots, X_k = x_k) = \frac{1}{Z} \prod_C \psi_C(x_C)$, where C are the maximal cliques, ψ_C

are nonnegative functions (potentials), and Z is the normalization constant often called the *partition function.* Since ψ_C are nonnegative, they are often represented in the exponential form: $\psi_C(x_C) = \exp(-E_C(x_C))$; E_C are often called *energy functions*, and the exponential representation of the total energy $p(X) = \exp(-\sum_C E_C(x_C))$ is called the *Boltzmann distribution* (this field borrows much of its terminology from statistical physics).

This definition implies a simple conditional independence statement: $p(x_i,\ x_j \mid x_{k\neq i,j}) = p(x_i \mid x_{k\neq i,j})p(x_j \mid x_{k\neq i,j})$ if x_i and x_j are not connected by an edge. Moreover, $p(\mathcal{X},\ \mathcal{Y} \mid \mathcal{Z}) = p(\mathcal{X} \mid \mathcal{Z})p(\mathcal{Y} \mid \mathcal{Z})$ if every path from $\mathcal{X}$ to $\mathcal{Y}$ goes through $\mathcal{Z}$, where $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$ are disjoint subsets of random variables.

Undirected graphical models have a rich history; in particular, undirected graphical models have been widely used for image denoising and similar problems, such as image segmentation, which is the primary motivation for our current study [13, 25, 28, 29]. Image denoising was one of the first applications for undirected graphical models [3, 4, 14].

In image processing problems, undirected models usually appear in the form of rectangular grids of pixels, with potentials on the edges intended to smooth the transitions; i.e., potentials usually serve to bring the neigboring nodes closer together. A simple yet useful model on a rectangular grid is the *Ising model*: energy functions on the edges of the grid are given by $\epsilon_{i,j}(x_i, x_j) = -w_{i,j}x_ix_j$, and the resulting joint distribution is $p(x_1, \ldots, x_n) = \frac{1}{Z}\exp\left(\sum_{(i,j)\in E} w_{i,j}x_ix_j + \sum_i u_ix_i\right)$. Inference in undirected graphical models can be done via *belief propagation.* In the basic message passing algorithm, a node collects messages from other nodes and sends out the marginalized results; for a detailed description of belief propagation, we refer to [5, 27]. When the underlying graph of the model contains cycles (and a grid model certainly does), belief propagation is not guaranteed to converge but when it converges, it does find a local minimum of the total energy function; this version of the algorithm is called *loopy belief propagation.*

These models also allow for a relatively simple approximate inference algorithm based on Gibbs sampling [5, 10, 14, 30]. For an Ising model with weights $w_{i,j}$ and u_i, the Gibbs sampler sequentially updates all variables by fixing all values of the variables except one, say x_i, and then sampling x_i from the conditional distribution: $x_i \sim p(x_i \mid \mathbf{x}_{-i})$. The conditional distribution is easy to compute in this case:

$$p(x_i = k \mid \mathbf{x}_{-i}) = \frac{p(x_i = k, \mathbf{x}_{-i})}{\sum_s p(x_i = s, \mathbf{x}_{-i})} = \frac{\prod_{j\in\mathrm{nei}(i)} \psi_{ij}(x_i = k, x_j)}{\sum_s \prod_{j\in\mathrm{nei}(i)} \psi_{ij}(x_i = s, x_j)},$$

where $\mathrm{nei}(i)$ is the neighbors set of the i^{th} vertex. Gibbs sampling is a special case of the Metropolis-Hastings algorithm for Monte-Carlo Markov chain sampling.

3.2 Our two-step approach

Markov random fields have already been successfully applied in the field of imaging mass spectrometry [15]. In this paper, we present a two-stage procedure for the segmentation of MALDI-imaging data. The first step is based on latent Dirichlet allocation, and the second step is based on Markov random fields used for denoising.

In the first step we learn the LDA model from the data. To do this, we need to define “documents” and their contents (“words”). The method we propose treats each pixel of a MALDI-imaging dataset as a document and considers the masses with high intensity values (peaks) in the associated spectrum as the words of the document. The result of this step consists of topic distributions θ_i for each pixel and word (peak) distributions for each topic.

In the second step, we use a Markov random field similar to the Ising model in order to denoise (smooth) the segmentation map. The model has two types of factors: binary factors

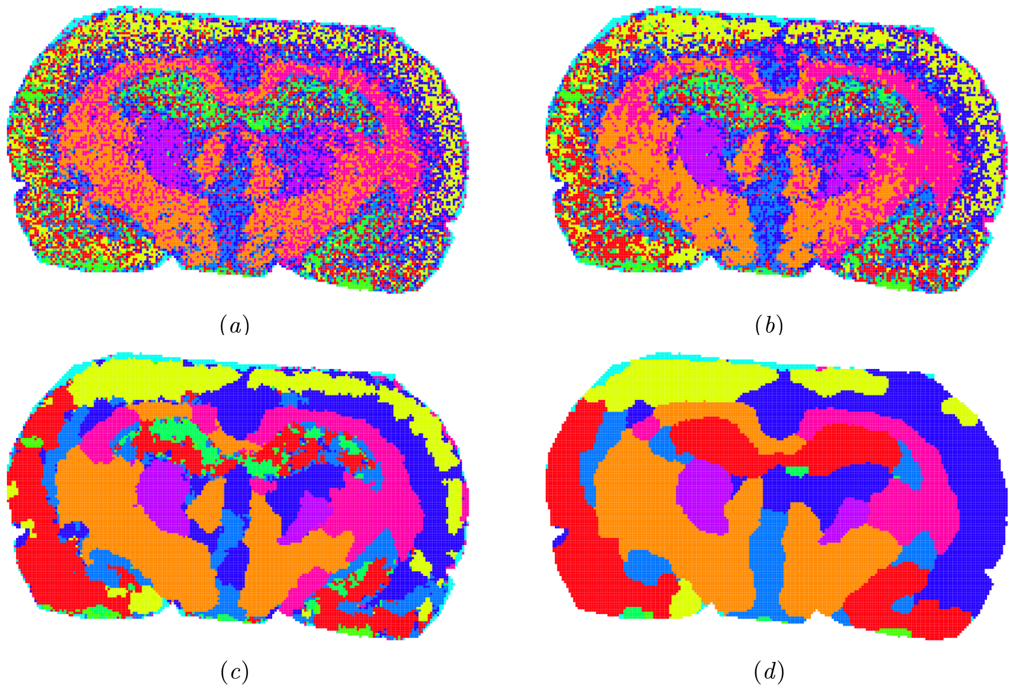


Figure 2 Segmentation maps obtained with k-means and MRF: (a) no denoising; (b) $w = 0.3$; (c) $w = 0.6$; (d) $w = 0.9$.

that correspond to the edges and unary factors that correspond to the vertices. A binary factor $\phi_{i,j}(t_1, t_2)$ takes the value $\exp(w)$ whenever $t_1 = t_2$ and $\exp(-w)$ otherwise. These factors are responsible for the smoothing part of the model. The unary factors are responsible for the incorporation of topic distributions found on the previous step; we initialize unary factors simply as $u_i(t) = \theta_i(t)$, where θ_i is the topic distribution for pixel i and $\theta_i(t)$ is the probability of topic t.

Previously, we developed other two-step approaches, where denoising was applied prior to clustering [1] and where denoising was embedded into clustering [2]. In this paper, we considered the potential of the two-step approach where the denoising is applied after probabilistic clustering. Note that our approach is different from calculating the segmentation map first and then denoising it for better visualization that might lead to losing details during denoising. In line with [15], we first reduce the dataset to k images, where each image represents the probabilities of pixels to belong to the corresponding topic. Then, the Markov random field exploits all k probabilities calculated for one pixel to decide on the assignment of this pixel to a specific cluster of the segmentation map.

For comparison, we have also implemented an approach in which the first step is based on the k-means algorithm. In this approach, we consider a MALDI-imaging dataset as a set of vectors and apply k-means clustering to it. The k-means algorithm outputs a set of cluster centers. Next we use a Markov random field for segmentation map smoothing. Binary factors are the same as above while unary factors look like $u_i(t) = 1/||s_i - c_t||$, where s_i is the spectrum associated with pixel i and c_t is the center of the t^{th} cluster.

Thus, in both cases the models try to smooth the image and at the same time preserve the segmentation map found on the first step. The use of LDA model on the first step not only increases the quality of the resulting segmentation but also provides additional information concerning the structure of segments through the distributions on topics for each pixel and distribution on words, i.e., masses, for each topic.

3.3 MRF extensions for MALDI-imaging

One problem with MRF smoothing for MALDI-IMS data is that the prior distribution induced by the smoothing factors with constant weights might not be flexible enough to represent the complex anatomical structures of the brain. In this section, we present two modifications of the MRF approach that significantly improve segmentation results for MALDI-IMS data.

The first modification is based on the use of Kullback-Leibler divergence. Instead of a constant weight w, we set the smoothing weight between neighboring pixels i and j depending on the Kullback-Leibler divergence between the topic distributions θ_i and θ_j that correspond to these pixels. In particular, we decrease the value of the smoothing weight when the Kullback-Leibler divergence exceeds the chosen threshold. This modification allows for different smoothing weights depending on the similarity of the pixels which results in reducing of the noise while preserving the details of the picture.

The second modification works through a simple version of the EM algorithm. Here, we use a more complicated form of the smoothing binary factor. We assume that the model has a set of parameters $\{\lambda_{t_1,t_2}\}$ and that the binary factor $\phi_{i,j}(t_1,t_2)$ has the following form: $\phi_{i,j}(t_1,t_2) = \exp(w\lambda_{t_1,t_2})$. Then we use one iteration of the EM algorithm to estimate the parameters. To do this, we first initialize binary factors with unit factors, $\lambda_{t_1,t_2} = 1$ when $t_1 = t_2$ and $\lambda_{t_1,t_2} = -1$ otherwise. After that, we use the Markov random field to get the denoised segmentation map. Then we reestimate λ_{t_1,t_2} as $\lambda_{t_1,t_2} = N_{t_1,t_2}/N$, where N_{t_1,t_2} is the number of neighboring pixel pairs with values t_1 and t_2 and N is the number of edges. After that, we normalize each row in the matrix of λ_{t_1,t_2} values and use the Markov random field with updated binary factors to get the final segmentation map.

3.4 Parallel implementation

MRF inference algorithms take up the greater portion of the running time in our two-step system. We have developed a highly parallel implementation of loopy belief propagation based on GraphLab, a recently developed framework for parallelizing large-scale machine learning tasks [26]. In the GraphLab paradigm, the data is represented as a graph, and the algorithm is represented as two basic routines: (i) Update runs in a single node and can use the data in this node, on its adjacent edges, and on its neighboring vertices; (ii) Sync collects the results from several subgraphs and propagates them further up. GraphLab can be viewed as a generalization of MapReduce. However, while MapReduce requires data subsets for the Map function to be completely independent, GraphLab allows them to form a complex highly connected graph, requiring only that the graph is relatively sparse (otherwise, there will be no speed improvement from parallelization).

GraphLab guarantees that during an Update procedure the data that it uses cannot be changed by any other Update, thus ensuring that there are no deadlocks and data races. There are several levels of data independence: an Update in a node may lock either its entire neighborhood (adjacent nodes and incident edges), incident edges only, or no other graph elements. In the case of loopy belief propagation in MRF, it suffices to lock incident edges since the message passing algorithm changes messages on the edges but does not change anything in other nodes. The resulting system shows promising speed and accuracy; it performs loopy belief propagation on an MRF until convergence in time which rapidly decreases with the number of cores; see Fig. 4 for a comparison.

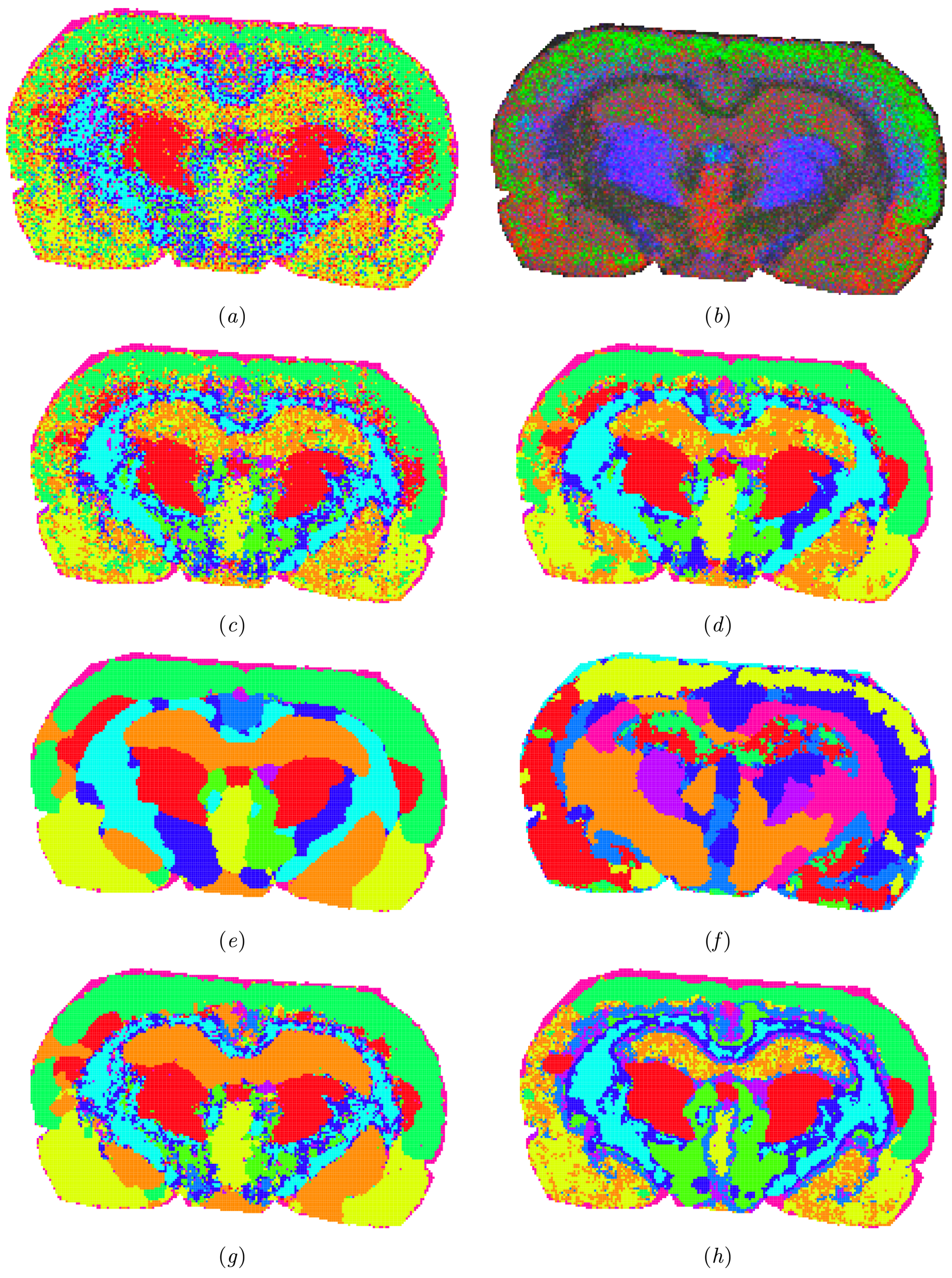


Figure 3 Segmentation maps obtained with LDA and MRF. (a) LDA clustering results; *(b)* a soft segmentation map for three clusters; (c) MRF smoothing with $w = 0.4$; (d) $w = 0.55$; (e) $w = 0.7$; (f) Segmentation map obtained with k-means ($w = 0.6$); (g) Kullback-Leibler modification; (h) EM algorithm modification.

4 Experimental results

Samples preparation and MALDI-IMS measurements are described in detail in [1]. Shortly, the cryosections of $10\mu m$ thickness were cut on a cryostat, transferred to a conductive indium-tin-oxide coated glass slide (Bruker Daltonik GmbH, Bremen, Germany) and measured using a MALDI-TOF instrument (Autoflex III; Bruker Daltonik GmbH) using flexControl 3.0 and flexImaging 2.1 software (Bruker Daltonik GmbH). The lateral resolution was set to $80\mu m$. Preprocessing was done in ClinProTools 2.2 software (Bruker Daltonik GmbH). The spectra were baseline-corrected with the TopHat algorithm (minimal baseline width set to 10 %, the default value in ClinProTools). No normalization or binning was done. The dataset comprises 20185 spectra acquired within the area of the slice (120×201 pixels), each of 3045 data points covering the mass range 2.5–10 kDa. For examples of intensity images for several m/z-values, see [1]. Afterwards, we applied a peak picking algorithm selecting 110 m/z-values; see [1] for the details and a visualization of the peak picking results.

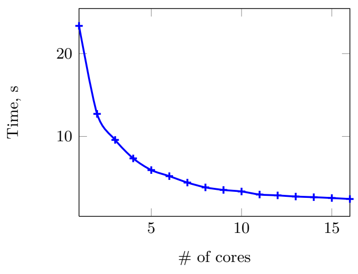


Figure 4 GraphLab implementation of loopy belief propagation on multiple cores.

Segmentation maps obtained with k-means and MRF are shown on Fig. 2; the ones obtained with LDA and MRF with varying values of w, on Fig. 3c–e. We consider the best-looking denoised picture to be the one with $w = 0.55$ (Fig. 3d). For comparison with LDA pictures, an example of segmentation map obtained with k-means is shown on Fig. 3f. We also experimented with two modifications described in Section 3.3. Results from the modification based on Kullback-Leibler divergence are shown on Fig. 3g; from the EM-based modification, on Fig. 3h. It is evident that our modification based on Kullback-Leibler divergence results in a cleaner image with less noise and at the same time better preserved details in the central part of the image. The EM-based modification produces an image that more closely resembles the anatomical structure of the brain.

5 Conclusion

In this work, we have presented a method for processing MALDI-imaging data; our method consists of two steps: probabilistic clustering with the LDA model and MRF smoothing. We have presented two novel modifications for the MRF smoothing method that significantly improve segmentation results for MALDI-imaging datasets. Further work in this direction is twofold. First, we plan to further improve clustering and denoising: improve the LDA model by learning hierarchical structures and/or correlated topics [6, 7], extend the undirected smoothing model by hidden factors, thus passing from an MRF to a restricted Boltzmann machine [17–20], incorporate ideas from segmentation algorithms based on hierarchical Dirichlet processes and hierarchical Pitman-Yor processes [31, 33, 34]. But the main direction of further work is to make the next step in analyzing the “spectrometry data cube” and apply the resulting models for a deeper analysis of mass-spectrometry data: find m/z values that are most characteristic for each segment, analyze spectra of the segments, construct most characteristic mass sections, find least characteristic sections (outliers) and so on. We hope that our approach will significantly assist us in this analysis.

Acknowledgements The authors thank Michael Becker (Bruker Daltonik GmbH, Bremen, Germany) for providing the rat brain MALDI-imaging dataset. This work was supported by the Russian Fund for Basic Research grant 11-01-00760-a, Russian Presidential Grant for Young Ph.D.s MK-6628.2012.1, for Leading Scientific Schools NSh-3229.2012.1, Russian Governmental grant 11.G34.31.0018, and the Federal Target Programme "Scientific and Scientific–Pedagogical Personnel of Innovative Russia", 2009–2013.

References

1 Theodore Alexandrov, Michael Becker, Soeren-Oliver Deininger, Guenther Ernst, Liane Wehder, Markus Grasmair, Ferdinand von Eggeling, Herbert Thiele, and Peter Maass. Spatial segmentation of imaging mass spectrometry data with edge-preserving image denoising and clustering. *Journal of Proteome Research*, 9(12):6535–6546, 2010.

2 Theodore Alexandrov and Jan H. Kobarg. Efficient spatial segmentation of large imaging mass spectrometry datasets with spatially aware clustering. *Bioinformatics*, 27:i230–i238, 2011.

3 J. Besag. On spatio-temporal models and Markov fields. In *Transactions of the 7th Prague Conference on Information Theory, Statistical Decision Functions and Random Processes*, pages 47–75. Academia, 1974.

4 J. Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society*, B-48:259–302, 1986.

5 Christopher M. Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006.

6 David M. Blei, Michael I. Jordan, Thomas L. Griffiths, and Joshua B. Tennenbaum. Hierarchical topic models and the nested chinese restaurant process. *Advances in Neural Information Processing Systems*, 13, 2004.

7 David M. Blei and John D. Lafferty. Correlated topic models. *Advances in Neural Information Processing Systems*, 18, 2006.

8 David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

9 R. M. Caprioli, T. B. Farmer, and J. Gile. Molecular imaging of biological samples: localization of peptides and proteins using maldi-tof ms. *Analytical Chemistry*, 69(23):4751–4760, 1997.

10 G. Casella and E. I. George. Explaining the gibbs sampler. *The American Statistician*, 46:167–174, 1992.

11 Kamila Chughtai and Ron M. A. Heeren. Mass spectrometric imaging for biomedical tissue analysis. *Chemical Reviews*, 110(5):3237–77, 2010.

12 Soeren-Oliver Deininger, Matthias P. Ebert, Arne Fuetterer, Marc Gerhard, and Christoph Roecken. MALDI imaging combined with hierarchical clustering as a new tool for the interpretation of complex human cancers. *Journal of Proteome Research*, 7(12):5230–5236, 2008.

13 Mario A. T. Figueiredo. Bayesian image segmentation using wavelet-based priors. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, pages 437–443, Washington, DC, USA, 2005. IEEE Computer Society.

14 S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

15 Michael Hanselmann, Ullrich Köthe, Marc Kirchner, Bernhard Y. Renard, Erika R. Amstalden, Kristine Glunde, Ron M. A. Heeren, and Fred A. Hamprecht. Toward digital staining using imaging mass spectrometry and random forests. *Journal of Proteome Research*, 8(7):3558–3567, 2009. PMID: 19469555.

16 Ron M. A. Heeren, Donald F. Smith, Jonathan Stauber, Basak Kuekrer-Kaletas, and Luke MacAleese. Imaging mass spectrometry: hype or hope? *Journal of the American Society for Mass Spectrometry*, 20(6):1006–1014, 2009.

17 Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14:1771–1800, 2002.

18 Geoffrey E. Hinton. What kind of a graphical model is the brain? In *Proceedings of the* 19th *International Joint Conference on Artificial Intelligence*, pages 1765–1775, 2005.

19 Geoffrey E. Hinton. A practical guide to training restricted boltzmann machines. `http://www.cs.toronto.edu/~hinton/absps/guideTR.pdf`, 2012.

20 Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

21 Paul Hudak, John Hughes, Simon Peyton Jones, and Philip Wadler. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297. Berkeley, University of California Press, 1967.

22 Michael I. Jordan and Y. Weiss. Graphical models: probabilistic inference. In M. Arbib, editor, *Handbook of Neural Networks and Brain Theory.* MIT Press, 2002.

23 T. Kanungo, D. M. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Y. Wu. An efficient k–means clustering algorithm: Analysis and implementation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24:881–892, 2002.

24 S. Lacoste-Julien, F. Sha, and Michael I. Jordan. Disclda: Discriminative learning for dimensionality reduction and classification. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, Cambridge, MA, 2008. MIT Press.

25 S. Z. Li. *Markov Random Field Modeling in Image Analysis.* Advances in Pattern Recognition. Springer, 2009.

26 Yucheng Low, Joseph Gonzalez, Aapo Kyrola, Danny Bickson, Carlos Guestrin, and Joseph M. Hellerstein. Graphlab: A new parallel framework for machine learning. In *Proceedings of the* 26th *Conference on Uncertainty in Artificial Intelligence*, 2010.

27 D. J. MacKay. *Information Theory, Inference and Learning Algorithms.* Cambridge University Press, 2003.

28 Patrick Perez. Markov random fields and images. *CWI Quarterly*, pages 413–437, 1998.

29 Simon Prince. *Computer Vision: Models, Learning, and Inference.* Cambridge University Press, 2012.

30 C. P. Robert and G. Casella. *Monte Carlo Statistical Methods.* Springer-Verlag, New York, 2004.

31 Alex Shyr, Trevor Darrell, Michael I. Jordan, and Raquel Urtasun. Supervised hierarchical pitman-yor process for natural scene segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2281–2288, 2011.

32 M. Stoeckli, P. Chaurand, D. E. Hallahan, and R. M. Caprioli. Imaging mass spectrometry: a new technology for the analysis of protein expression in mammalian tissues. *Nature Medicine*, 7(4):493–496, 2001.

33 E. Sudderth and Michael I. Jordan. Shared segmentation of natural scenes using dependent Pitman-Yor processes. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, Cambridge, MA, 2008. MIT Press.

34 Y. W. Teh, Michael I. Jordan, M. J. Beal, and David M. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2004.

35 Jeramie D. Watrous, Theodore Alexandrov, and Pieter C. Dorrestein. The evolving field of imaging mass spectrometry and its impact on future biological research. *Journal of Mass Spectrometry*, 46:209–222, 2011.

Building and Documenting Workflows with Python-Based Snakemake

Johannes Köster[1,2] and Sven Rahmann[1]

1 **Genome Informatics, Institute of Human Genetics, Faculty of Medicine, University of Duisburg-Essen, Hufelandstr. 55, 45122 Essen, Germany,** `{johannes.koester,sven.rahmann}@uni-due.de`

2 **Paediatric Oncology, University Childrens Hospital Essen, Hufelandstr. 55, 45122 Essen, Germany**

Abstract

Snakemake is a novel workflow engine with a simple Python-derived workflow definition language and an optimizing execution environment. It is the first system that supports multiple named wildcards (or variables) in input and output filenames of each rule definition. It also allows to write human-readable workflows that document themselves. We have found Snakemake especially useful for building high-throughput sequencing data analysis pipelines and present examples from this area. Snakemake exemplifies a generic way to implement a domain specific language in python, without writing a full parser or introducing syntactical overhead by overloading language features.

1998 ACM Subject Classification D.3.2 Language Classifications, D.3.4 Processors, J.3 Life and Medical Sciences

Keywords and phrases workflow engine, dependency graph, knapsack problem, Python, high-throughput sequencing, next-generation sequencing

Digital Object Identifier 10.4230/OASIcs.GCB.2012.49

1 Introduction

Workflow systems manage the ever-increasing complexity of computational pipelines in large-scale bioinformatics applications, e.g. for image processing or next-generation sequencing data analysis. Existing solutions range from graphical systems, e.g. Biopipe [5], Taverna [11], Galaxy [2] and GeneProf [4], to frameworks supporting only text based workflow definitions, e.g. Ruffus [3], Pwrake [14], GXP Make [15] and Bpipe [12]. While graphical solutions are easy to learn, text-based representations can be advantageous: Workflows can be edited faster and directly on servers without graphical environments and developers can collaborate on them via source code management tools.

A basic way to define a workflow is provided by the build system GNU Make [13] that was originally intended to compile source code but can be used to execute arbitrary pipelines of command line applications. In contrast to many of the above systems, here the actual workflow (dependencies, parallelization) is inferred from a given set of rules with input and output files, rather than relying on an explicit specification. This idea was adapted by Pwrake and GXP Make, and is also the foundation of *Snakemake*.

In the spirit of the Python language, Snakemake complements these prior works with a syntax close to pseudocode. The resulting workflows are human-readable and serve as both documentation and formal definition. Further, Snakemake provides scalability because it optimizes the number of parallel processes with respect to provided CPU cores and needed threads and can make use of single machines as well as cluster engines without the need to modify the workflow. In contrast to Pwrake and GXP Make, Snakemake does not rely on any password-less SSH setup or custom server processes runing on the cluster nodes. Finally, Snakemake is the first system to support multiple named wildcards and output files in rules, which we show to be useful in a common bioinformatics workflow.

This paper also presents a uniform way to implement a domain specific language (DSL) that avoids to write a full parser. The approach avoids unnecessary syntactic overhead to be exposed to the user (e.g., Python decorators or unintuitively overloaded operators).

The remainder of this paper is structured as follows: In Section 2.1 we describe the DSL for defining and documenting Snakemake workflows, which is illustrated by an exemplary pipeline for calling single nucleotide polymorphisms (SNPs) in Section 2.2. Section 2.3 focuses on the technical implementation of the Snakemake language as a Python-based DSL. The engine that executes the workflow is described in Section 3, and a final conclusion is drawn in Section 4.

2 The Snakemake Workflow Language

2.1 Language Definition

A workflow is defined in a *Snakefile* via a domain specific language close to Python syntax. It consists of rules that define how to create output files from input files. The workflow is implied by dependencies between the rules that arise from one rule needing an output file of another as an input file.

A rule definition in a Snakefile specifies (1) a name, (2) any number of input and output files and (3) either a shell command or Python code that creates the output from the input. Input and output files may contain multiple named wildcards and are specified as Python strings. In extended Backus-Naur Form (EBNF), the Snakemake syntax can be defined as follows.

$$\begin{aligned}
\text{snakemake} &= \textit{statement} \mid \text{rule} \\
\text{ni} &= \textit{NEWLINE INDENT} \\
\text{rule} &= \text{"rule" } (\textit{identifier} \mid \text{""}) \text{ ":" ruleparams} \\
\text{ruleparams} &= [\text{ni input}]\ [\text{ni output}]\ [\text{ni threads}]\ [\text{ni (run} \mid \text{shell)}]\ \textit{NEWLINE} \text{ snakemake} \\
\text{input} &= \text{"input" ":" } \textit{parameter\_list} \\
\text{output} &= \text{"output" ":" } \textit{parameter\_list} \\
\text{threads} &= \text{"threads" ":" } \textit{integer} \\
\text{run} &= \text{"run" ":" ni } \textit{statement} \\
\text{shell} &= \text{"shell" ":" } \textit{stringliteral}
\end{aligned}$$

The terminal symbols *NEWLINE* and *INDENT* and the italic non-terminals refer to plain Python syntax, e.g. *statement* allows any Python statement, *stringliteral* denotes a Python string (i.e. `"..."` or `"""..."""`), and *parameter_list* enables the same syntax as for the parameters of Python functions.

2.2 Example: A SNP-Calling Pipeline

Listing 1 shows a single nucleotide polymorphism (SNP) calling pipeline in the context of next generation sequencing, implemented as a Snakemake workflow. SNP calling is a typical

Listing 1 Example Snakefile for read mapping and SNP calling.

```
SAMPLES = "100 101 102 103".split()
REF = "hg19.fasta"
DBSNP = "dbsnp.vcf"

rule all:
    input:  "calls/{sample}.vcf".format(sample = sample)
            for sample in SAMPLES

rule map_reads:
    input:  ref = REF, reads = "{sample}.{group}.fastq"
    output: temp("{sample}.{group}.sai")
    threads: 8
    shell:  "bwa aln -t{threads} {input.ref} {input.reads} > {output}"

rule sai_to_bam:
    input:  REF, "{sample}.1.sai", "{sample}.2.sai",
                "{sample}.1.fastq", "{sample}.2.fastq"
    output: protected("{sample}.bam")
    shell:  "bwa sampe \
            -r'@RG\\tID:{wildcards.sample}\\tSM:{wildcards.sample}'\
            {input} | samtools view -Sbh - > {output}"

rule sort:
    input: "{name}.bam"
    output: "{name}.sorted.bam"
    shell: "samtools sort {input} {wildcards.name}.sorted"

rule index:
    input:  "{name}.sorted.bam"
    output: "{name}.sorted.bam.bai"
    shell:  "samtools index {input}"

rule realign_targets:
    input:  ref = REF, dbsnp = DBSNP
    output: "known.intervals"
    shell:  "gatk -T RealignerTargetCreator -R {input.ref}\
            -known {input.dbsnp} -o {output}"

rule realign:
    input:  bam = "{sample}.sorted.bam",
            bai = "{sample}.sorted.bam.bai",
            ref = REF, intervals = "known.intervals"
    output: "realigned/{sample}.sorted.bam"
    shell:  "gatk -T IndelRealigner -I {input.bam} -R {input.ref}\
            --targetIntervals {input.intervals} -o {output}"

rule call_snps:
    input:  bam = "realigned/{sample}.sorted.bam",
        bai = "realigned/{sample}.sorted.bam.bai",
        dbsnp = DBSNP, ref = REF
    output: "calls/{sample}.vcf"
    shell:  "gatk -T UnifiedGenotyper --dbsnp {input.dbsnp}\
            -I {input.bam} -R {input.ref} -o {output}"
```

task e.g. in cancer genomics [10]. In this example, paired-end sequence reads are given as `.fastq` files for four samples with IDs from 100 to 103, and shall be mapped to the human reference genome. Afterwards, reads are realigned to avoid artifacts due to the mapping heuristic, and finally SNPs are called. The Snakefile consists of seven rules that each start with the keyword **`rule`** followed by their name and the indented definitions of input and output files and executable shell commands.

In lines 1–3, a list of sample IDs is created, the desired genome reference and the database of known SNPs is specified in plain Python. As Snakemake rules can intermix with any Python statement, accessing configuration files or environment variables, even waiting for user input or opening a graphical user interface is possible. In line 9, the rule `map_reads` aligns the sequence reads to the reference genome with the BWA software [7], which is assumed to be installed on the system. Specified in braces inside the input and output files, the rule uses two *named wildcards*, since it shall be applied to the first and the second read of each read pair (wildcard `{group}`) from each sample (wildcard `{sample}`). BWA creates suffix array intervals (`.sai`-files) which are converted to the common format for aligned reads in the rule `sai_to_bam` (line 15). These two rules illustrate major patterns for accessing input and output files inside a rule. In the rule `map_reads`, input files are named and can be accessed as attributes of the **`input`**-object (e.g. `{input.ref}` in line 13). In rule `sai_to_bam`, all input files are accessed at once (separated by a space) via `{input}` (line 21).

BWA's internal `.sai`-files can be deleted once the `.bam`-files are created, which are in turn worth to be write-protected to avoid accidental deletion. Snakemake supports this by marking files as `temp` (line 11) or `protected` (line 18), respectively.

Next, the rule `realign` (line 39) uses the GATK [9] to ensure that the read alignments are free of artifacts from the mapping heuristic. It depends on the rules `sort` (line 23) and `index` (line 28) being applied to the alignments from BWA to create an index for the `.bam` file using samtools [8].

The actual SNP calling with GATK takes place in the rule `call_snps` (line 47); the result is saved to `.vcf`-files. Here, the alignments created by the rule `realign` are used. Further, another `.bam`-index has to be created, hence the rule `index` is used a second time.

When Snakemake is invoked without a specific target, the first rule (here called `all`) in line 5 is executed, which ensures that the `.vcf`-files and hence all needed intermediate files are created for each sample.

In place of shell commands, Snakemake allows to write pure Python code using a **`run`** block instead of a **`shell`** block to create the output files of a rule. For example, we may want to plot the coverage histogram as a quality control.

```
from matplotlib.pyplot import hist, savefig
rule plot_coverage_histogram:
    input: "{sample}.bam"
    output: hist = "{sample}.coverage.pdf"
    run:
        hist(list(map(int,
            shell("samtools mpileup {input} | cut -f4",
            iterable = True))))
        savefig(output.hist)
```

Here, we use the `hist` function of Python's matplotlib module [6] and parse the output of Snakemakes **`shell`** function, that invokes samtools to calculate a per-nucleotide coverage. This demonstrates how to apply Python code to the result of shell commands.

The example workflow illustrates the readability of the language and how the workflow rules document themselves. The advantage of conceptually separating a rule's name from the output file(s) it produces becomes evident, as does the benefit of mutiple named wildcards for input and output files.

2.3 Implementation

In general, two extreme approaches for implementing a domain specific language are thinkable: a full parser can be generated, or in-language facilities can be overloaded. The first allows a concise syntax, the second (that can be approached e.g. by Python decorators or operator overloading) can sometimes lead to syntactical overhead and unintuitive constructs. Here we present a third way, using a finite automaton that translates tokens from the Snakemake language to a sequence of Python tokens, that are interpreted by the ordinary Python interpreter. Thereby, only a small subset of tokens needs to be altered. Formally, the procedure can be specified by an automaton that recognizes the Snakemake grammar (Section 2.1) and an emission function that emits Python language terminal symbols based on the current state and token.

▶ **Definition 1** (Snakemake Language Automaton). Given the Snakemake grammar, let NT be the set of non-terminals and T be the set of terminal symbols. Over the alphabet of tokens $\Sigma = T$, the Snakemake language automaton is denoted as $(Q, q_0, F, \Sigma, \delta)$ with states $Q = NT$, start state $\mathit{snakemake} \in NT$, and a single final state $F = \{\mathit{statement}\}$. The transition function $\delta : \Sigma \times Q \to Q$ is given by the EBNF grammar of Section 2.1.

The above finite automaton definition is possible because in the Snakemake grammar each non-terminal is uniquely determined by a prefix of terminal symbols. We extend the automaton with an emission function $e : Q \times \Sigma \to \{(t_0, t_1, \dots) \mid t_0, t_1, \dots \in T'\}$, where T' is the set of terminal symbols in the Python language, such that rule definitions are converted to Python API calls. For example, the rule `map_reads` from Listing 1 is translated to the following Python code using the (less user-readable) function decorator syntax (function decorators are higher-order functions that operate on Python functions). This internal representation is evaluated by the Python interpreter.

```
@rule("map_reads")
@input(ref = REF, reads = "{sample}.{group}.fastq")
@output(temp("{sample}.{group}.sai"))
@threads(8)
@run
def __map_reads(input, output, wildcards, threads):
    shell("bwa aln -t {threads} {input.ref} {input.reads} > {output}")
```

In Snakemake, multiple named wildcards are allowed in input and output files. To determine if a rule can create a requested file, its defined output files are matched against the requested file. Thereby, wildcards are replaced by the Python regular expression `.+` per default (i.e. any non empty string is matched in a greedy fashion). The substring a wildcard matches to is then propagated to the input files. When the rule `map_reads` is requested to create a certain file, e.g. `100.1.sai`, the wildcard `{sample}` matches `100` and `{group}` matches `1`, so that the input file `100.1.fastq` is expected. Multiple wildcards may in some cases introduce the problem of ambigous matches. To avoid such problems, wildcards can be restricted to particular regular expressions. In this case, to force the greedy matching to the intended solution, `{group}` could be replaced by `{group,[12]}`, such that the regular expression `[12]` is used instead of `.+`. The regular expression syntax is that of Pythons `re`

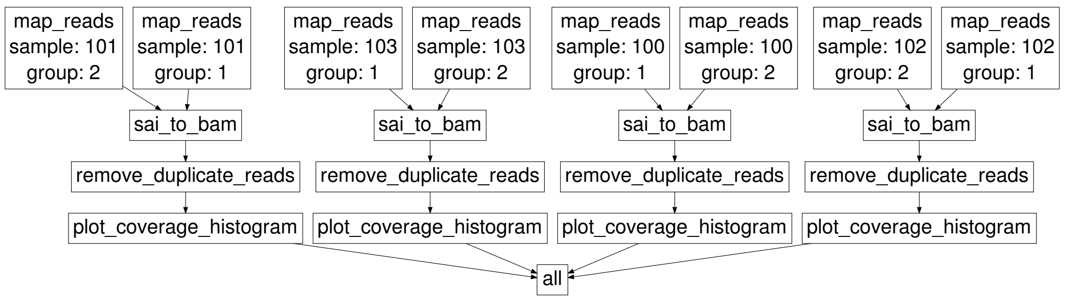

map_reads sample: 101 group: 2
map_reads sample: 101 group: 1
map_reads sample: 103 group: 1
map_reads sample: 103 group: 2
map_reads sample: 100 group: 1
map_reads sample: 100 group: 2
map_reads sample: 102 group: 2
map_reads sample: 102 group: 1
sai_to_bam
remove_duplicate_reads
plot_coverage_histogram
all

Figure 1 Directed acyclic graph of jobs for the example workflow from Listing 1.

module, which is almost identical to PERL 5's regular expressions, except for some corner cases.

3 The Snakemake Execution Engine

When Snakemake is invoked with a Snakefile, it determines a plan of rule executions that are needed to create the requested output. We call the planned execution of a rule a *job*. Since one rule can be executed for multiple wildcard values, more than one job for each rule is possible. A job may need certain input files that are created by other jobs, which gives rise to a directed acyclic graph (DAG) that represents the execution plan (see Figure 1). The nodes of the DAG are jobs, and a directed edge between job A and B means that the rule underlying job B needs the output of job A as an input file. A path in the DAG represents a sequence of jobs that have to be executed serially. Importantly, two disjoint paths in the DAG can be executed independently from each other, i.e., in parallel.

On top of per-job parallelism, the number of threads that a single job uses can be specified in a rule, via the parameter `threads` (line 12 of Listing 1), which defaults to 1 if it is not specified. The number of threads can be accessed in the shell command by the variable `threads` and thus passed to external tools that support multithreading. Snakemake tries to maximize parallelism (i.e. minimize the number of idle cores) up to a given threshold of available CPU cores. Considering that each job can use one or more threads as explained above, the parallel job scheduling problem can be cast as follows.

▶ **Definition 2** (Parallel Job Scheduling). Let J be the set of jobs that are ready to execute (i.e. do not depend on missing input files), and t_j be the number of threads for job $j \in J$. Let T be the given threshold of available cores; if a job requests more threads (i.e. $t_j > T$), the threads are reduced to T. Thus the problem is to find E^* among all $E \subseteq J$ such that $\sum_{j \in E} \min(t_j, T)$ is maximized while the sum remains bounded by the number of currently idle cores.

The parallel job scheduling problem is a classical binary knapsack problem and can be solved by dynamic programming in pseudo-polynomial time. Given current CPU core numbers, the solution is instantaneous. The problem is solved every time a new job becomes ready to execute and idle cores exist. Snakemake first executes the jobs in the solution E^*.

Solving above problem to schedule jobs allows Snakemake to scale to environments with a hard limit of used CPU cores, e.g., when multiple users share the same compute server, by choosing T appropriately. Further, using only as many threads as there are cores available can be beneficial for performance since it reduces the amount of context switching.

Apart from running on single machines, Snakemake contains a generic mechanism that allows the execution of jobs on a batch system or a compute cluster engine. For this, a submit command that handles shell scripts (e.g. `qsub`) and a shared file system accessible by all cluster nodes has to be available. Snakemake compiles every job into a shell script and submits it with the given command once it is ready to execute. It then queries about the existence of a certain guard file that indicates that the job is finished in regular intervals.

Per default, Snakemake only executes rules if the output files are not present or the modification time of the input files is newer. Together with the automatic deletion of output files from incomplete rule executions (e.g. due to a failing shell command), this enables Snakemake to avoid duplicate work when resuming workflows.

To analyse the workflow, Snakemake provides options to perform a dry-run without actual execution of jobs, give the reason for each executed job and print the DAG to the *graphviz* `.dot` format [1] for visualization (see Figure 1).

4 Conclusion

Snakemake is a pythonic workflow system inspired by GNU Make, that allows to define a workflow in terms of rules that create output files from input files and thereby automatically handles dependencies and parallelization. It is the first workflow system to support multiple named wildcards and output files in rules. We show that this is especially useful in bioinformatics workflows. Further, Snakemake offers a simple Python-like syntax with high readability, and allows to efficiently intermix Python and shell commands. This removes the need to write external shell or PERL scripts for simple format conversions, and additionally allows to run complex algorithms within the workflow. Further, this allows to make use of web services, e.g. using specialized libraries or the built-in Python http client and XML parsing facilities.

Our approach exemplifies an automaton-based implementation of a domain-specific language without the need to generate a full parser or creating syntactical overhead by overloading language features.

By optimizing the schedule of jobs against a given threshold of available cores while taking also intra-job parallelism into account, a Snakefile scales from single core workstations over multi-core servers to compute clusters of different architectures, without the need to modify the workflow.

Acknowledgements We thank Tobias Marschall (CWI Amsterdam) and Marcel Martin (TU Dortmund) for their tremendously helpful testing work, feature requests and comments.

References

1 Emden R. Gansner and Stephen C. North. An open graph visualization system and its applications to software engineering. *Software - Practice and Experience*, 30(11):1203–1233, 2000.

2 Jeremy Goecks, Anton Nekrutenko, James Taylor, and The Galaxy Team. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biology*, 11(8):R86, 2010.

3 Leo Goodstadt. Ruffus: A lightweight python library for computational pipelines. *Bioinformatics*, 26(21):2778–2779, November 2010.

4 Florian Halbritter, Harsh J. Vaidya, and Simon R. Tomlinson. GeneProf: analysis of high-throughput sequencing experiments. *Nature Methods*, 9(1):7–8, December 2011.

5 Shawn Hoon, Kiran Kumar Ratnapu, Jer-Ming Chia, Balamurugan Kumarasamy, Xiao Juguang, Michele Clamp, Arne Stabenau, Simon Potter, Laura Clarke, and Elia Stupka. Biopipe: A flexible framework for Protocol-Based bioinformatics analysis. *Genome Research*, 13(8):1904–1915, August 2003.

6 J.D. Hunter. Matplotlib: A 2D graphics environment. *Computing in Science Engineering*, 9(3):90 –95, June 2007.

7 Heng Li and Richard Durbin. Fast and accurate short read alignment with Burrows–Wheeler transform. *Bioinformatics*, 25(14):1754–1760, July 2009.

8 Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, and Richard Durbin. The sequence Alignment/Map format and SAMtools. *Bioinformatics*, 25(16):2078–2079, August 2009.

9 Aaron McKenna, Matthew Hanna, Eric Banks, Andrey Sivachenko, Kristian Cibulskis, Andrew Kernytsky, Kiran Garimella, David Altshuler, Stacey Gabriel, Mark Daly, and Mark A. DePristo. The genome analysis toolkit: A MapReduce framework for analyzing Next-Generation DNA sequencing data. *Genome Research*, 20(9):1297–1303, September 2010.

10 Matthew Meyerson, Stacey Gabriel, and Gad Getz. Advances in understanding cancer genomes through second-generation sequencing. *Nature Reviews Genetics*, 11(10):685–696, September 2010.

11 Tom Oinn, Matthew Addis, Justin Ferris, Darren Marvin, Martin Senger, Mark Greenwood, Tim Carver, Kevin Glover, Matthew R Pocock, Anil Wipat, and Peter Li. Taverna: A tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, November 2004.

12 Simon P Sadedin, Bernard Pope, and Alicia Oshlack. Bpipe: A tool for running and managing bioinformatics pipelines. *Bioinformatics*, April 2012.

13 Richard M. Stallman and Roland McGrath. *GNU Make - A Program for Directing Recompilation.* 1991.

14 Masahiro Tanaka and Osamu Tatebe. Pwrake: a parallel and distributed flexible workflow management tool for wide-area data intensive computing. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, HPDC '10, page 356–359, New York, NY, USA, 2010. ACM.

15 K. Taura, T. Matsuzaki, M. Miwa, Y. Kamoshida, D. Yokoyama, N. Dun, T. Shibata, C. S Jun, and J. Tsujii. Design and implementation of GXP make - a workflow system based on make. *Future Generation Computer Systems*, 2011.

Online Transitivity Clustering of Biological Data with Missing Values

Richard Röttger*[1,2], Christoph Kreutzer[1], Thuy Duong Vu[4], Tobias Wittkop[5], and Jan Baumbach[1,2,3]

1 **Max Planck Institute for Informatics, Saarbrücken, Germany,** `{roettger,ckreutzer,jbaumbac}@mpi-inf.mpg.de`
2 **Center for Bioinformatics, Saarbrücken, Germany**
3 **Cluster of Excellence on Multimodal Computing and Interaction, Center for Bioinformatics, Saarland University, Saarbrücken, Germany**
4 **Bioinformatics group, CBS-KNAW Fungal Biodiversity Centre, Utrecht, The Netherlands,** `d.vu@cbs.knaw.nl`
5 **Buck Institute for Age Research, Navato, NV, USA,** `twittkop@buckinstitute.org`

Abstract

Motivation: Equipped with sophisticated biochemical measurement techniques we generate a massive amount of biomedical data that needs to be analyzed computationally. One long-standing challenge in automatic knowledge extraction is clustering. We seek to partition a set of objects into groups such that the objects within the clusters share common traits. Usually, we have given a similarity matrix computed from a pairwise similarity function. While many approaches for biomedical data clustering exist, most methods neglect two important problems: (1) Computing the similarity matrix might not be trivial but resource-intense. (2) A clustering algorithm itself is not sufficient for the biologist, who needs an integrated online system capable of performing preparative and follow-up tasks as well.

Results: Here, we present a significantly extended version of Transitivity Clustering. Our first main contribution is its' capability of dealing with missing values in the similarity matrix such that we save time and memory. Hence, we reduce one main bottleneck of computing all pairwise similarity values. We integrated this functionality into the Weighted Graph Cluster Editing model underlying Transitivity Clustering. By means of identifying protein (super)families from incomplete all-vs-all BLAST results we demonstrate the robustness of our approach. While most tools concentrate on the partitioning process itself, we present a new, intuitive web interface that aids with all important steps of a cluster analysis: (1) computing and post-processing of a similarity matrix, (2) estimation of a meaningful density parameter, (3) clustering, (4) comparison with given gold standards, and (5) fine-tuning of the clustering by varying the parameters.

Availability: Transitivity Clustering, the new Cost Matrix Creator, all used data sets as well as an online documentation are online available at http://transclust.mmci.uni-saarland.de/.

Contact: roettger@mpi-inf.mpg.de

1998 ACM Subject Classification I.5.3 Clustering

Keywords and phrases Transitivity Clustering, Large Scale clustering, Missing Values, Web Interface

Digital Object Identifier 10.4230/OASIcs.GCB.2012.57

* to whom correspondence should be addressed

1 Introduction

Sophisticated biomedical measurement techniques generate a massive amount of data that needs to be analyzed efficiently. One example is the availability of next-generation sequencing (NGS) technology that provided us with almost two thousand whole-genome sequences [23]. This data is easily accessible for any researcher in the world-wide scientific community. However, the sequences are worthless without annotations of the biological meaningful elements, i.e. genes and non-coding functional elements, such as microRNAs. Analyzing millions of such DNA sequences individually to determine their function is an impossible task without the aid of specific bioinformatics tools [3, 25]. Just to give some numbers, the NCBI database hosts data about approximately 5.2 million bacterial and 3.2 million eukaryotic genes [22].

Although it appears that we have finally arrived in the post-genome era we still lack fundamental knowledge about crucial genetic programs, the interplay of genes and proteins as well as their biochemical regulation networks. We know very little about how cells, organs and tissues regulate survival, reproduction, movement, etc. in response to altering environmental conditions [20]. Therefore, we use so-called OMICS technology to generate even more data in order to unravel this crucial knowledge about interactions of all kinds of biological entities. The database content of the Gene Expression Omnibus (GEO) with data on more than 630,000 samples may exemplify this trend for the case of gene expression data [8].

Analyzing this impressive amount of data manually is infeasible. In order to draw conclusions from such huge data sets successfully, we often start with compressing the raw data such that the contained information becomes visible. One typical computational tool is clustering [2]. Clustering is a computer science method that partitions data objects into groups such that the objects share common traits, i.e. the elements within the groups are more similar to each other than to objects from other groups. In bioinformatics, for many tasks we have given a pair-wise similarity function $s(u, v)$ that assigns each pair of objects (u and v) a similarity value [12]. The corresponding similarity matrix serves as input for many computational biology clustering problems: protein complex detection from protein-protein interaction networks, cancer sub-typing from gene expression data and protein homology detection from all-vs.-all BLAST [1] results. In our paper we exemplarily focus on the later data type: protein sequence similarities utilized for finding clusters of potentially homologous proteins. Various tools have been developed for this purpose: k-means, Affinity Propagation, Markov Clustering, and FORCE as well as Transitivity Clustering, to name just a few [11, 10, 9, 16, 27].

Here, we concentrate on the problem of finding clusters of homologous proteins with Transitivity Clustering. While this task was extensively studied and published before (refer to [26, 28, 29]), we will focus on two sideline challenges arousing in this context: (1) running time aspects for computing the similarity matrix and (2) online access to a web interface that allows for performing all necessary steps of a typical cluster analysis efficiently without the need for downloading, installing and running the software tool on the local desktop PC. While there exist clustering tools, which allow for clustering with incomplete information, they mainly focus on repairing noise and incompleteness due to the underlying assays (e.g. microarray studies or patient data collections) [13, 7, 18, 21]. Here, we present a systematic approach of saving runtime and memory by omitting the calculation of a share of the similarity values for homology detection while still using the well accepted standard NCBI BLAST. This approach allows the user to save time and enables large-scale studies with almost no drawbacks in the quality of the clustering results. Furthermore, the web front-end offers easy-to-use interfaces to typical data pre-processing, clustering as well as post-processing tasks. In the following, we briefly describe our previous work on Transitivity Clustering followed by a summary of our new contributions. Afterwards, we describe our approach in detail. Finally, we will evaluate our improved method and discuss its' robustness for protein homology detection.

1.1 Our previous work

We recently developed Transitivity Clustering, an integrated software package dedicated to partitioning biomedical data sets. In previous publications, we studied and evaluated its' performance for remote homology detection and protein (super)family reconstruction [27, 28] as well as protein complex identification [29]. Based on gold standard data from Paccanaro *et al.* [16] and Brown *et al.* [5] we demonstrated that Transitivity Clustering performs equally well or outperforms other popular bioinformatics clustering tools for these tasks. It is based on exact and heuristic algorithms for solving the Weighted Transitive Graph Projection (WTGP) problem [19]. Given a similarity matrix and a user-given threshold (density parameter), we compute a cost graph by removing all edges below the threshold. Afterwards, we make the graph transitive by adding/deleting edges with respect to a cost function that we minimize. Finally, we report the emerging cliques as clusters (see *Definitions* in the Methods section). As demonstrated in [29], the average similarity within the clusters is above the threshold while the average similarity between elements from different clusters is below the density cutoff. So far, Transitivity Clustering comes as stand-alone tool, Java library and as Cytoscape [24] plugin.

1.2 Our new contributions

However, Transitivity Clustering as well as most other approaches neglect two important problems: First, computing the similarity matrix might not be trivial but resource-intense (main memory, running time, costs for allocating appropriate data sets, etc.). Second, a clustering algorithm itself is not sufficient for the biomedical researcher, who needs an integrated online system capable of performing preparative and follow-up tasks as well. The goal of Transitivity Clustering is to assist the user through the entire clustering pipeline. We present a significantly extended version of Transitivity Clustering:

1. Missing similarity values: We added the capability of dealing with missing values in the similarity matrix to the Transitivity Clustering framework such that we may save time and memory. Hence, we reduce one main bottleneck of computing all pairwise similarity values. We will describe how we integrated this functionality into the underlying Weighted Graph Cluster Editing model such that we still achieve accurate results.
2. Evaluation of the robustness: We use the gold standard data from Brown *et al.* [5], as well as a larger data set containing the protein sequences of 27 different *corynebacteria*, to study the robustness of our extended approach. We utilize all-vs.-all BLAST results and remove a varying amount of similarity values based on a block-wise scheme. We study how this affects the accuracy of the clustering performance as well as the running time improvement.
3. Web interface: We present a new, intuitive web interface that aids with all important steps of a typical cluster analysis. We give examples and describe a workflow through the interface.

2 Methods

2.1 Extension of the Weighted Transitive Graph Projection model

Given a set of objects V, a threshold $t \in \mathbb{R}$, and a pairwise similarity function $\text{sim}\colon \binom{V}{2} \to \mathbb{R}$, we define a graph G as

$$G = (V, E);\; E = \left\{ uv \in \binom{V}{2}; \text{sim}(uv) > t \right\}.$$

The WTGP problems is now defined as the determination of a transitive graph $G' = (V, E')$, such that there exists no other transitive graph $G'' = (V, E'')$, with $\text{cost}\,(G \to G'') < \text{cost}\,(G \to G')$. The costs for edge additions/deletions are defined as

$$\text{cost}\,(G \to G') := \underbrace{\sum_{uv \in E \setminus E'} |\text{sim}\,(uv) - t|}_{\text{deletion cost}} + \underbrace{\sum_{uv \in E' \setminus E} |\text{sim}\,(uv) - t|}_{\text{addition cost}}.$$

This problem is NP-hard [14] and APX-hard [6] and we tackle it with a combination of exact and heuristic algorithms (refer to [19, 27]). This problem is also known as "Cluster Editing" and several other exact approaches are mentioned in literature. For an overview of exact methods refer to [4].

Now we need to integrate a possibility to deal with missing edges in the graph G. Therefore, we slightly adjust the underlying similarity function.

$$\text{sim}\,(uv) := \begin{cases} \in \mathbb{R} & \text{if similarity available} \\ t & \text{if missing value} \end{cases}$$

The similarity of the missing values is set to the user-given threshold t. As a result, the costs for adding/remove an edge with a missing value is

$$\underbrace{|\text{sim}\,(uv) - t|}_{\text{deletion cost}} = \underbrace{|\text{sim}\,(uv) - t|}_{\text{addition cost}} = |t - t| = 0$$

In consequence, the overall costs for transforming a graph G into the transitive G' is not affected by the missing values:

$$\text{cost}\,(\text{missing values}) = \sum_{uv \in E \setminus E'} |\text{sim}\,(uv) - t| = \sum_{uv \in E \setminus E'} |t - t| = 0$$

In summary, since the missing values have no information content we adapted our approach such that they do not impact the clustering process. The remaining edges with existing similarity values have to account for the clustering result. We are confident that this does not affect the clustering quality much, as for most transitive most of the similarity information is redundant anyways. Thus, whenever we find a cluster with high intra-cluster similarity it is likely that missing similarities within this cluster are comparably high as well. Our transitivity model is perfectly suited for this key feature, as the missing values do not influence the clustering process itself. Thus, the remaining high similarities are capable of forming the final cluster.

2.2 Costmatrix creation with missing values

With the presented method, the user is able to include missing values in the similarity input files for Transitivity Clustering. In comparison to available methods [13, 7, 18, 21], we not only allow for missing values, which are repaired either *a priori* or during the clustering process (in our case by exploiting the transitivity properties of the clusters), but we also provide the user with the possibility to systematically benefit from missing values in order to save calculation time of the similarity file and resulting cost matrices.

In this paper we concentrate on protein homology detection based on a BLAST all-vs-all run. In our approach, we do not use missing values on randomly chosen positions but omit the calculation of entire blocks of the similarity file. Therefore, we have developed the new CostMatrixCreator (CMC) assisting the user in creating a similarity file with missing values. First, a BLAST database for all protein sequences is created. Afterwards, only a certain percentage of the proteins are BLASTed against the database. Figure 1 depicts this process. CMC can either choose these proteins randomly (as in our evaluation) or the user can provide a manually created list if the user wants to incorporate prior knowledge. Consequently, we compute similarity values for only those pairs of sequences where at least one of them is in the list of proteins compared against the database. The similarities for the other sequence pairs are missing. That keeps the similarity file small, as the missing values are not required to be explicitly marked as missing. The only additional information needed are the IDs of the proteins left out. Note that this procedure distinguishes between missing similarity values due the BLAST cut-off and missing values due to an omitted comparison. The missing values resulting from the BLAST cut-off indeed carry information: They are very dissimilar and are set to a minimum value. The missing values resulting from omitted comparisons do not carry any information and are set to the density threshold.

This approach has four major advantages: (1) The user is still able to utilize the well accepted standard BLAST. (2) The user saves the calculation time for the missing similarities. (3) As the missing values are not stored explicitly, which reduces the memory consumption of the result file drastically. (4) The created cost matrices can afterwards be analyzed with the standard Transitivity Clustering workflow as before. Also note that our approach would also work with any other similarity function than BLAST as well.

2.3 Data sets

We utilize the gold standard data set from Brown *et al.* for studying the effect of missing values to the clustering performance, i.e. accuracy and running time. The data set comprises of five enzyme superfamilies (amidohydrolase, crotonase, enolase, haloacid dehalogenase, and vicinal oxygen chelate) with different levels of sequence diversity. On the one hand, the enolase and crotonase superfamilies contain a very discrete set of sequences, i.e. high sequence similarities. The other extreme are the haloacid dehalogenase and parts of the amidohydrolase, which include a very divers set of sequences with a comparably high number of outliers. Therefore, in congruence with Brown *et al.*, this set of protein sequences serves as an evaluation data set for clustering tools.

The five superfamilies consist of 4,887 proteins that are further divided into 91 families. Each of the amino acid sequences is either annotated to a gold or a silver standard family. Gold standard families only contain sequences with experimentally determined functions, while silver standard families are less restrictive. As done in previous studies, when we compared Transitivity Clustering to other approaches [27], we only used the 866 sequences that are assigned to a gold standard family.

As the data set from Brown *et al.* resembles a rather small problem instance, we can not expect huge improvements in terms of reduced computational time. Thus, we also apply our methods to a larger remote homology detection data set consisting of 66,000 proteins of 27 different *corynebacteria*. In the remainder of this manuscript, we refer to this data set as the "Coryne-Data".

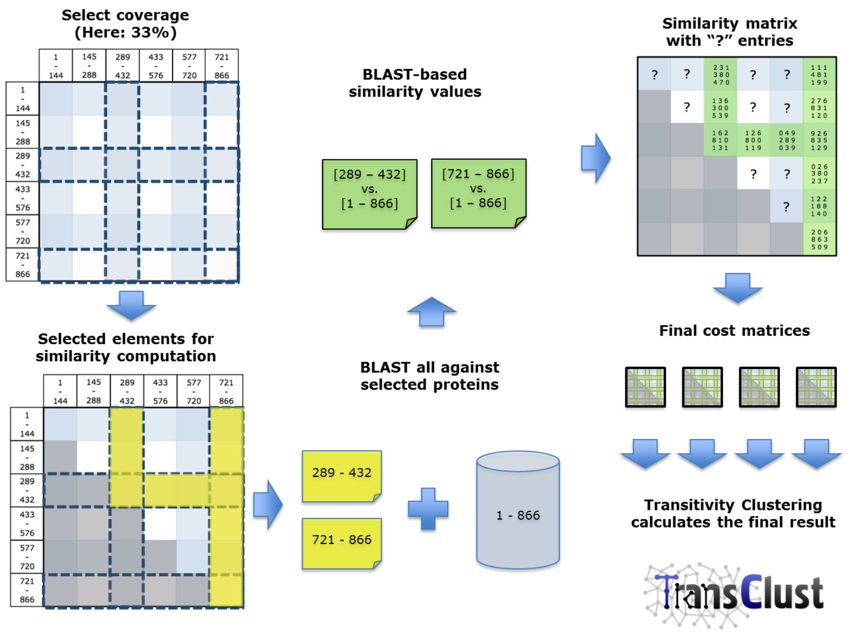

Select coverage
(Here: 33%)
Selected elements for similarity computation
BLAST-based similarity values
[289 – 432] vs. [1 – 866]
[721 – 866] vs. [1 – 866]
BLAST all against selected proteins
289 - 432
721 - 866
1 - 866
Similarity matrix with "?" entries
Final cost matrices
Transitivity Clustering calculates the final result
TransClust

Figure 1 Illustration of the block-based scheme for saving similarity value computation time. For illustration reasons, we arbitrarily ordered the data to form six blocks. Now, only the data from block three and six are selected to be BLASTed against the database. The remaining values in the similarity matrix are displayed as "?". These are the steps used by the CostMatrixCreator to create cost matrices for Transitivity Clustering.

2.4 Evaluation

We use the F-measure for comparing the results of Transitivity Clustering to the above described gold standard. Essentially, the F-measure is an equal combination of precision and recall. It gives a value between 0 and 1, where values near 0 mean "bad" results and a value of 1 means a perfect overlap between clustering result and gold standard, i.e. protein families in our case.

Let $C = \{C_1, \ldots, C_n\}$ be the clusters obtained by Transitivity Clustering and $K = \{K_1, \ldots, K_m\}$ be the gold standard. Furthermore let $T = (t_{i,j}) \in \mathbb{N}^{m \times n}$ denote the matrix where each entry is the number of common objects between K_i and C_j,

$$t_{i,j} := |\{K_i \cap C_j\}|,\ 1 \leq i \leq m,\ 1 \leq j \leq n.$$

We follow the scheme of Paccanaro *et al.* [16] for computing the F-measure for a clustering C against a reference clustering K in the following way:

$$\text{F-measure}(K_i) = \max_{1 \leq j \leq n} \frac{2 \cdot t_{i,j}}{|C_j| + |K_i|}.$$

The overall F-measure is then defined as

$$\begin{aligned}\text{F-measure}(C, K) &= \frac{1}{\sum_{i=1}^{m} |K_i|} \sum_{i=1}^{m} (|K_i| \cdot \text{F-measure}(K_i)) \\ &= \frac{1}{\sum_{i=1}^{m} |K_i|} \sum_{i=1}^{m} \left(|K_i| \cdot \max_{1 \leq j \leq n} \frac{2 \cdot t_{i,j}}{|C_j| + |K_i|} \right).\end{aligned}$$

Note that we refer to the F-measure when we use the term "accuray".

In order to use the above described Brown *et al.* gold standard data set we first downloaded all amino acid sequences for all proteins and BLASTed them all-vs.-all (E-value threshold 0.01). We furthermore downloaded the corresponding protein family annotations. For the Coryne-Data, we obtained all protein sequences of all 27 fully-sequenced *corynebacteria* from the NCBI website. As there is no gold standard for this data set, we performed an all-vs.-all BLAST and performed a TC clustering with a threshold of 20 in order to produce our own "gold standard". Anyway, we may use this data set for studying the effect of missing values compared a full-coverage similarity matrix and for evaluating run time savings.

For creating a similarity matrix with missing values we utilize the new CostMatrixCreator to construct a BLAST database containing all proteins of a dataset. Here, only a certain percentage of the available proteins are compared to the database (again with E-value threshold 0.01). Figure 1 depicts this process. For systematic evaluation, we vary the coverage from 1% to 90% and create the cost matrices accordingly. We call this method the "block-based scheme", as we exclude no single entries from the similarity matrix calculation but entire blocks or rather stripes (Figure 1 illustrates that as well).

The emerging cost matrices are then used to perform the clustering with TC. For each clustering we compute the accuracy (F-Measure) and measure the runtime. The running time for the entire clustering is the time for CMC plus the successive clustering process. In order to assess the variability (robustness) of the best threshold, we clustered both data sets with different thresholds and always picked (and reported) the threshold leading to the best F-measure.

2.5 Web interface

TransClust, our Transitivity Clustering implementation, and the new CostMatrixCreator can be downloaded as stand-alone Java programs. Nevertheless, in order to increase the accessibility of the presented tools, a web interface helps in attracting more users. Especially non computer experts benefit from the now obsolete step of an installation process. Furthermore, the results can be accessed from any computer with Internet connection, which allows for an optimized workflow and eases collaborations. For the web interface, we mainly used the following libraries and programming languages:

LAP: We use Linux, Apache, and PHP to generate the HTML code, for data handling and for executing the Transitivity Clustering software.

JavaScript and jQuery: jQuery is a JavaScript library under GPL or MIT license that can be used to automate common tasks in web design. We make use of this for the toggle switches that show/hide information such as advanced options for different steps in the clustering process. We also used JavaScript to check input forms for correctness and for changing contents of input forms, when using sliders, for instance.

jQueryUI: jQueryUI is an extension of jQuery specifically designed for providing extended functionality such as widgets and animations. We use jQueryUI for the tabs that can be used to toggle the different outputs and for the sliders that can be used to change different input parameters.

Highcharts: Highcharts is a charting library from Highsoft Solutions under the Creative Commons Attribution-NonCommercial 3.0 License (free for non-academics). We use to create the graphical representations of the results. We use jQuery/JavaScript to process the output files and feed them to Highcharts.

The new web interface is publicly available online at http://transclust.mmci.uni-saarland.de/.

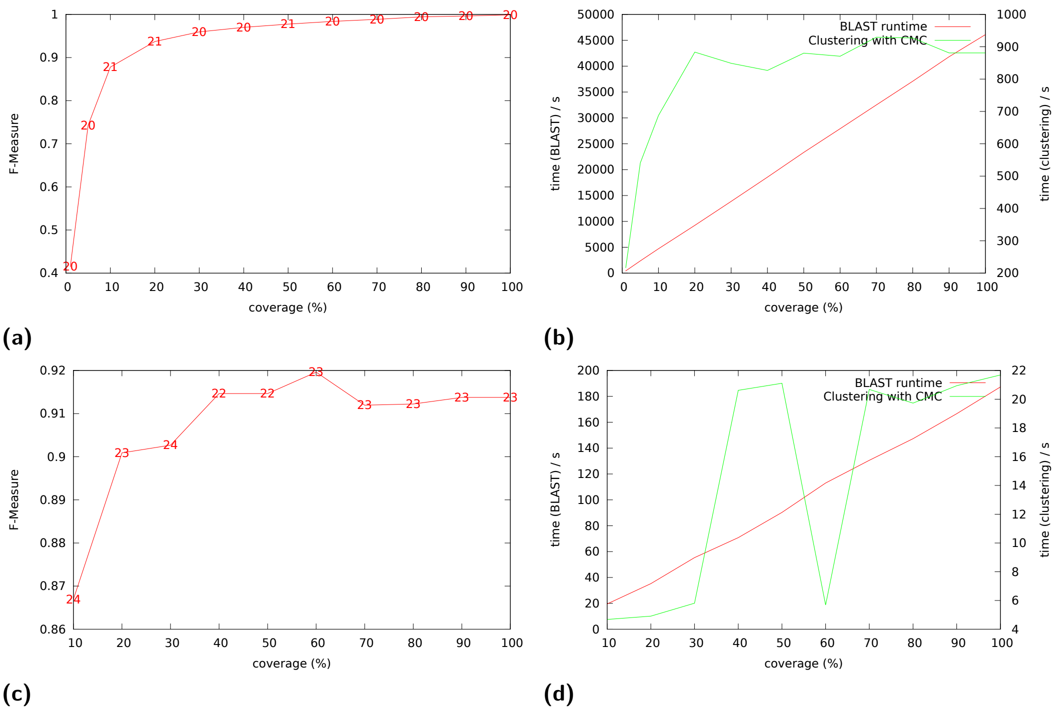


Figure 2 The plots depict the runtime and accuracy of both datasets, Coryne-Data is on the top (**(a)** and **(b)**), the data set of Brown *et al.* on the bottom (**(c)** and **(d)**). Figures **(a)** and **(c)** display the F-measure as a function of the coverage. The numbers on the line give the threshold, which yielded to the corresponding F-meassure. Figures **(b)** and **(d)** give two runtime measures, again as a function of the coverage: **red** displays the time consumed for calculating the BLAST results, **green** depicts the runtime for the clustering plus CMC. Note that both runtime plots have different timescales displayed on the left for BLAST and on the right for clustering respectively.

3 Results and discussion

3.1 Clustering with missing values

We first investigate the robustness of our approach by comparing the F-Measure for different percentages of missing values (inverse similarity coverage). The results are depicted in Figure 2. We can see that even for a low coverage (high amount of missing values) the F-Measure only drops by a few percent when comparing against the protein families from Brown *et al.*, as well as for the Coryne-Data. Furthermore, it is important to notice, that the threshold delivering the best F-measure is very stable for all coverages. That means in conclusion, that all methods for finding a good threshold (e.g. using a smaller gold standard dataset for parameter training or utilizing the same threshold from a comparable study) can be applied for clustering with missing values as well.

Running time scales as expected: BLAST computing times grow linearly with the number of sequence comparisons while the runtime for the clustering process is essentially constant with little variation.

Figure 3 plots the average runtime of the cost matrix creation CMC with missing values against the F-Measures. It shows that with our approach the runtime can be drastically reduced, while the drop of the F-Measure is comparably moderate, i.e. less than 10% F-Measure drop for about 70% runtime saving. All runtimes are based on a single thread

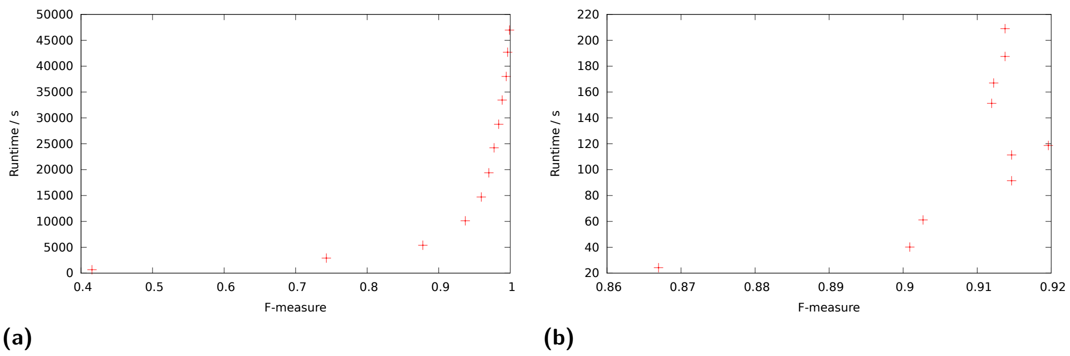


Figure 3 Runtime of CMC as a function of the F-Measure for **(a)** *Coryne-Data* and **(b)** Brown *et al.* dataset. Due to the small size of the Brown *et al.* dataset and the resulting short BLAST runs, we observe a certain variation of the clustering runtime. For the larger dataset, this is not observed anymore.

execution of BLAST and CMC. Note, that CMC also supports parallel execution, which may further reduce the runtime.

3.2 Web interface

The clustering process with the web interface is divided into three steps. In the first step, the user provides the input (cost-matrix file, similarity file, or BLAST and FASTA files). In the second step, a review of the given similarities is presented as a similarity distribution plot. The user may then specify further clustering options, such as the clustering threshold(s) or a gold standard file to compare against, if available/desired. In the third step the results are presented as intra/inter-cluster similarity distribution graphs. The best way, however, to evaluate our new web interface is navigating with your browser to the new Transitivity Clustering web site. It is easy-to-use and provides start-to-end solutions for each important step of a typical cluster analysis. Briefly, the new interface now assists with the following typical data processing tasks: (1) computing and post-processing of a similarity matrix, (2) estimation of a meaningful density parameter, (3) running the clustering process itself, (4) automatic comparison with given gold standards, and (5) fine-tuning of the clustering by varying the threshold and by visualizing inter-cluster vs. intra-cluster similarity distributions.

4 Conclusion and outlook

To sum up, we directly integrated the concept of missing similarity values with the weighted transitive graph projection model of Transitivity Clustering in a straight forward fashion. We demonstrated the power of the approach for protein homology detection based on all-vs.-all BLAST results. The accuracy only drops slightly while run time for computing the similarity matrix can be reduced linearly, with the presented tool. The new web interface saves time and effort with download, configuration and installation.

The new CostMatrixCreator is a JAVA implementation and supports the user with the creation of cost matrices with missing values step by step. Note that we do not use an own BLAST implementation but we execute the standard NCBI Blast binaries from within CMC. Thus our CMC implementation could easily be adapted for performing the necessary steps with any kind of similarity function. The resulting cost matrices can finally be included easily into the normal clustering workflow by using our new Transitivity Clustering web interface, for instance.

In the future, we aim to apply our method to more run time intense similarity computation problems (3D structures of proteins, for instance). We will also integrate the new version of Transitivity Clustering into clusterMaker [15]. The most important future work, however, is to evaluate our method on more data sets. We are working closely together with the SFLD (Structure-Function Linkage Database) team [17] to make this possible in the near future. Another example for future applications is huge gene expression data sets.

Acknowledgement

JB and RR are grateful for financial support from the Cluster of Excellence on Multimodal Computing and Interaction of the German Research Foundation (DFG). RR's work was also supported by the International Max Planck Research School in Computer Science. JB, RR and CK wish to thank the Center for Bioinformatics Saar (ZBI).

References

1 S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, 1997.

2 Bill Andreopoulos, Aijun An, Xiaogang Wang, and Michael Schroeder. A roadmap of clustering algorithms: finding a match for a biomedical application. *Briefing in Bioinformatics*, Feb 2009.

3 Enrique Blanco and Josep F Abril. Computational gene annotation in new genome assemblies using geneid. *Methods Mol Biol*, 537:243–61, 2009.

4 S. Böcker, S. Briesemeister, and G.W. Klau. Exact algorithms for cluster editing: Evaluation and experiments. *Algorithmica*, 60(2):316–334, 2011.

5 Shoshana D Brown, John A Gerlt, Jennifer L Seffernick, and Patricia C Babbitt. A gold standard set of mechanistically diverse enzyme superfamilies. *Genome Biology*, 7(1):R8, 2006.

6 Moses Charikar, Venkatesan Guruswami, and Anthony Wirth. Clustering with qualitative information. *Journal of Computer and System Sciences*, 71:360–383, 2003.

7 S. Dubnov, R. El-Yaniv, Y. Gdalyahu, E. Schneidman, N. Tishby, and G. Yona. A new nonparametric pairwise clustering algorithm based on iterative estimation of distance profiles. *Machine Learning*, 47(1):35–61, 2002.

8 Ron Edgar, Michael Domrachev, and Alex E Lash. Gene expression omnibus: NCBI gene expression and hybridization array data repository. *Nucleic Acids Res*, 30(1):207–10, Jan 2002.

9 A. J. Enright, S. Van Dongen, and C. A. Ouzounis. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Research*, 30(7):1575–1584, Apr 2002.

10 A. J. Enright and C. A. Ouzounis. GeneRAGE: a robust algorithm for sequence clustering and domain detection. *Bioinformatics*, 16(5):451–457, May 2000.

11 B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, 2007.

12 John A. Hartigan. *Clustering Algorithms.* Wiley, 1975.

13 Dae-Won Kim, Ki-Young Lee, Kwang H Lee, and Doheon Lee. Towards clustering of incomplete microarray data without the use of imputation. *Bioinformatics*, 23(1):107–113, Jan 2007.

14 Mirko Křivánek and Jaroslav Morávek. NP-hard problems in hierarchical-tree clustering. *Acta Informatica*, 23(3):311–323, 1986.

15 John H Morris, Leonard Apeltsin, Aaron M Newman, Jan Baumbach, Tobias Wittkop, Gang Su, Gary D Bader, and Thomas E Ferrin. clustermaker: a multi-algorithm clustering plugin for cytoscape. *BMC Bioinformatics*, 12(1):436, Nov 2011.

16 A. Paccanaro, J. A. Casbon, and M. A. Saqi. Spectral clustering of protein sequences. *Nucleic Acids Research*, 34(5):1571–1580, 2006.

17 Scott C-H Pegg, Shoshana D Brown, Sunil Ojha, Jennifer Seffernick, Elaine C Meng, John H Morris, Patricia J Chang, Conrad C Huang, Thomas E Ferrin, and Patricia C Babbitt. Leveraging enzyme structure-function relationships for functional inference and experimental design: the structure-function linkage database. *Biochemistry*, 45(8):2545–55, Feb 2006.

18 J. Poland and T. Zeugmann. Clustering pairwise distances with missing data: Maximum cuts versus normalized cuts. pages 197–208, 2006.

19 Sven Rahmann, Tobias Wittkop, Jan Baumbach, Marcel Martin, Anke Truss, and Sebastian Böcker. Exact and heuristic algorithms for weighted cluster editing. *Comput Syst Bioinformatics Conf*, 6:391–401, 2007.

20 Richard Röttger, Ulrich Rückert, Jan Taubert, and Jan Baumbach. Towards the size of gene regulatory networks – how little do we actually know? *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, in press, 2012.

21 M. Sarkar and T. Y. Leong. Fuzzy k-means clustering with missing values. *Proc AMIA Symp*, pages 588–592, 2001.

22 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.

23 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, 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, 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 Research*, Nov 2009.

24 Paul Shannon, Andrew Markiel, Owen Ozier, Nitin S Baliga, Jonathan T Wang, Daniel Ramage, Nada Amin, Benno Schwikowski, and Trey Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Research*, 13(11):2498–2504, Nov 2003.

25 Vasily Tcherepanov, Angelika Ehlers, and Chris Upton. Genome annotation transfer utility (gatu): rapid annotation of viral genomes using a closely related reference genome. *BMC Genomics*, 7:150, 2006.

26 Tobias Wittkop, Jan Baumbach, Francisco P Lobo, and Sven Rahmann. Large scale clustering of protein sequences with force -a layout based heuristic for weighted cluster editing. *BMC Bioinformatics*, 8:396, 2007.

27 Tobias Wittkop, Dorothea Emig, Sita Lange, Sven Rahmann, Mario Albrecht, John H Morris, Sebastian Böcker, Jens Stoye, and Jan Baumbach. Partitioning biological data with transitivity clustering. *Nat Methods*, 7(6):419–20, Jun 2010.

28 Tobias Wittkop, Dorothea Emig, Anke Truss, Mario Albrecht, Sebastian Böcker, and Jan Baumbach. Comprehensive cluster analysis with transitivity clustering. *Nat Protoc*, 6(3):285–95, Mar 2011.

29 Tobias Wittkop, Sven Rahmann, Richard Röttger, Sebastian Böcker, and Jan Baumbach. Extension and robustness of transitivity clustering for protein-protein interaction network analysis. *Internet Mathematics*, 7(4):255–273, 2011.

ConReg: Analysis and Visualization of Conserved Regulatory Networks in Eukaryotes

Robert Pesch[1], Matthias Böck[2], and Ralf Zimmer[1]

1 **Institut für Informatik, Ludwig-Maximilians-Universität München, Amalienstr. 17, 80333 München,** `{Robert.Pesch,Ralf.Zimmer}@bio.ifi.lmu.de`

2 **Institut für Informatik / I12, Technische Universität München, Boltzmannstr. 3, 85748 Garching,** `Matthias.Boeck@in.tum.de`

Abstract

Transcription factors (TFs) play a fundamental role in cellular regulation by binding to promoter regions of target genes (TGs) in order to control their gene expression. TF-TG networks are widely used as representations of regulatory mechanisms, e.g. for modeling the cellular response to input signals and perturbations.

As the experimental identification of regulatory interactions is time consuming and expensive, one tries to use knowledge from related species when studying an organism of interest. Here, we present ConReg, an interactive web application to store regulatory relations for various species and to investigate their level of conservation in related species. Currently, ConReg contains data for eight model organisms. The regulatory relations stored in publicly available databases cover only a small fraction both of the actual interactions and also of the regulatory relations described in the scientific literature. Therefore, we included regulatory relations extracted from PubMed and PubMedCentral using sophisticated text-mining approaches and from binding site predictions into ConReg.

We applied ConReg for the investigation of conserved regulatory motifs in *D. melanogaster*. From the 471 regulatory relations in *REDfly* our system was able to identify 66 confirmed conserved regulations in at least one vertebrate model organism (*H. sapiens*, *M. musculus*, *R. norvegicus*, *D. rerio*). The conserved network consists among others of the well studied motifs for eye-development and the pan-bilaterian kernel for heart specification, which are well-known examples for conserved regulatory relations between different organisms.

ConReg is available at `http://services.bio.ifi.lmu.de/ConReg/` and can be used to analyze and visualize regulatory networks and their conservation among eight model organisms. It also provides direct links to annotations including literature references to potentially conserved regulatory relations.

1998 ACM Subject Classification J.3 Life and Medical Sciences, H.3.5 Online Information Services, I.2.7 Natural Language Processing

Keywords and phrases web application, evolutionary biology, regulatory networks, text-mining

Digital Object Identifier 10.4230/OASIcs.GCB.2012.69

1 Introduction

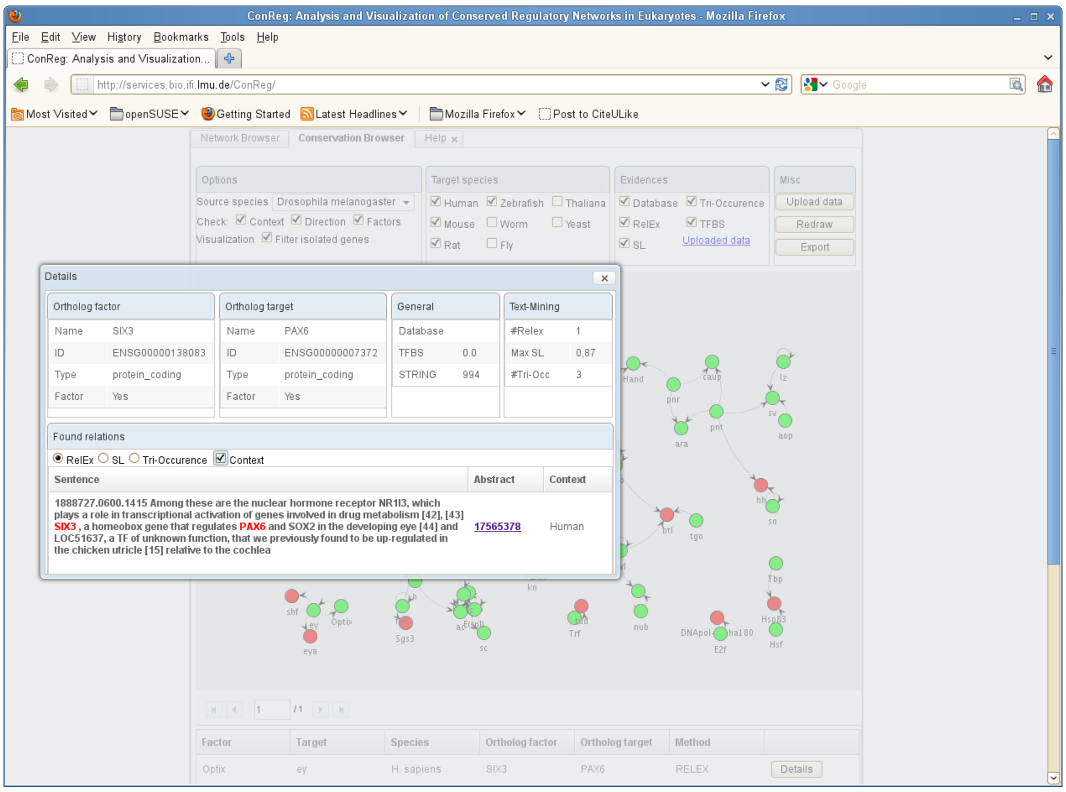
The physical regulatory relationships of an organism can be described by gene regulatory networks (GRNs). Transcription factors (TFs) and their respective targets (TGs) define the majority of these regulations. GRNs can describe systems on the scale of a few genes, a particular pathway or even on the whole gene complement of an organism. The inference of these GRNs is generally done from experimental data sets, like gene expression data and additional prior information from available databases [11]. Even though more and more high-throughput methods for the identification of TF-TG relations have been developed, data of experimentally validated relations is still sparse for higher multi-cellular organisms [22]. Therefore, transferring knowledge using orthologs from related species is typically done when studying an organism of interest. Several approaches have been proposed which are capable of transferring physical protein-protein interactions even between phylogenetically distant species, e.g. from *S. cerevisiae* to *A. thaliana* or *C. elegans* [38]. The conservation of a regulatory relation requires that at least the involved TF and the TG have to be conserved and that the TF binds to the promoter region of the TG in two or more organisms. Depending on the number of organisms in which the regulatory relation is conserved and the evolutionary distance, relations between different organisms can be transferred with a certain confidence [2, 27, 30].

Different methods have been proposed and used for the transfer of regulatory networks from one organism to another. A well-known example is KEGG, which transfers confirmed regulatory relations to (non-model) organisms based on ortholog definitions [13]. For bacteria more advanced approaches have successfully been applied, which additionally incorporate conserved information of the binding site [2] and subfamily classifications [27]. Similar approaches exist for eukaryotes, which also make use of conserved transcription factor binding sites [30]. Taher *et al.* claimed that 88 % of the orthologs between *H. sapiens* and *D. rerio* retain their regulatory mechanisms [30]. Nevertheless, the extend of regulatory relations that can be directly transferred between organisms remains controversial [2]. There are some well-known regulatory motifs, which appear to be conserved among a group of quite distant species, supporting the transfer of conserved regulatory relations. A famous example is the conservation of regulatory relations for the development of the eye in *D. melanogaster* and vertebrates. It was shown that in *M. musculus* and other vertebrates *Pax-6*, the ortholog of the *eyeless (ey)* gene - one of the central TFs controlling the eye development in *D. melanogaster* - shares an extensive sequence identity and is even capable of inducing ectopic eyes in *D. melanogaster* [36]. Also other motifs, like the pan-bilaterian kernel for heart specification [6] or regulation of apoptosis regulation in *D. melanogaster* and vertebrates [39] appear to be conserved. Studies revealing the similarity and the conservation of regulatory subnetworks have been conducted for different species as well, like *MAP kinase expression* in *C. elegans* and *H. sapiens* [15] or *Toll-like receptor 4* regulated genes [26].

In the following we present ConReg, an interactive web application to investigate regulatory relations. ConReg collects and visualizes evidences for the conservation of regulatory relations in other eukaryotic model organisms. For that purpose, we collected regulatory data for eight model organisms (*H. sapiens*, *M. musculus*, *R. norvegicus*, *D. rerio*, *D. melanogaster*, *C. elegans*, *A. thaliana* and *S. cerevisiae*). The data was obtained from general and species-specific regulatory databases, from text-mining approaches applied to PubMed abstracts and PubMedCentral full text publications and from transcription factor binding site predictions (TFBS).

2 Discovery of Conserved Regulatory Relations with ConReg

With ConReg conserved regulatory relations for a source species can be discovered in a target species if these relations can be found between the respective orthologs in both species (by default we do not require conservation of the transcription factor binding site in the two species). ConReg searches regulatory relations of a source species which were extracted from regulatory databases, in the specified target species. Several types of evidences for regulatory relations of the target species can be considered based on the user's selections.


ConReg: Analysis and Visualization of Conserved Regulatory Networks in Eukaryotes - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://services.bio.ifi.lmu.de/ConReg/
Network Browser
Conservation Browser
Help
Upload data
Redraw
Export
Uploaded data

Options
Source species: Drosophila melanogaster
Check: ✓ Context ✓ Direction ✓ Factors
Visualization: ✓ Filter isolated genes

Target species: ✓ Human ✓ Zebrafish ☐ Thaliana ✓ Mouse ☐ Worm ☐ Yeast ✓ Rat ☐ Fly

Evidences: ✓ Database ✓ Tri-Occurence ✓ RelEx ✓ TFBS ✓ SL

Details

Ortholog factor	
Name	SIX3
ID	ENSG00000138083
Type	protein_coding
Factor	Yes

Ortholog target	
Name	PAX6
ID	ENSG00000007372
Type	protein_coding
Factor	Yes

General	
Database	
TFBS	0.0
STRING	994

Text-Mining	
#Relex	1
Max SL	0,87
#Tri-Occ	3

Found relations
● RelEx ○ SL ○ Tri-Occurence ✓ Context

Sentence	Abstract	Context
1888727.0600.1415 Among these are the nuclear hormone receptor NR1I3, which plays a role in transcriptional activation of genes involved in drug metabolism [42], [43] SIX3 , a homeobox gene that regulates PAX6 and SOX2 in the developing eye [44] and LOC51637, a TF of unknown function, that we previously found to be up-regulated in the chicken utricle [15] relative to the cochlea	17565378	Human

Factor	Target	Species	Ortholog factor	Ortholog target	Method	
Optix	ey	H. sapiens	SIX3	PAX6	RELEX	Details

Figure 1 Screenshot of ConReg for the interactive discovery of conserved regulatory relations for a source species in user selected target species. Conservation of regulatory relations for a species can be interactively discovered using ConReg. The system allows searching for conservation in all provided species and with different prediction methods for the target species, whereas for the source species only reliable relations from databases are considered. For an identified conservation of a regulatory relation details such as text-mining results and binding site predictions can be visualized.

Currently, regulatory information from publicly available databases like TRANSFAC [18] or REDfly [10], relations found with text-mining approaches (RelEx [8], SL [9], Tri-occurrence) in PubMed and PubMedCentral and TFBS predictions can be selected (see Materials and Methods for details).

The *Conservation Browser*, where the entire predicted conserved network is shown and the *Motif Finder* with which the user can search for conservations in a defined subset of genes are the two main features of ConReg. For both features, the user can select the source species, regulatory data sources for the target species and further constraints for the text-mining approaches. Our system shows the conserved regulatory network as well as annotations for each found conserved relation. This includes information about the orthologs, the textual positions where our text-mining approach found regulatory relations in the literature, the TFBS predictions and the protein-protein interaction score from the STRING database (version 9) [29]. For further analysis the networks can be exported as tab separated files for use in advanced network analysis tools. An example can be seen in Figure 1, which shows a screenshot of the *Conservation Browser* with conserved relations for *D. melanogaster* as source species. The detail view window in the front shows information about the regulatory relations in the target species including an example of a regulatory relation which was discovered by RelEx between *Pax6* and *Six3* in *H. sapiens*.

Table 1 Overview of the model organisms from our database with the number of genes, number of predicted and known transcription factors and the number of factors with position weight matrices (PWMs). In addition, the number of regulatory relations collected from databases and relations which were extracted from the scientific literature by using different text-mining approaches (RelEx [8], SL [9] and Tri-occurrence) and from transcription factor binding site predictions (TFBS) are shown. Most regulatory relations could be found for *S. cerevisiae* and *A. thaliana* which originate mostly from genome-wide chromatin immunoprecipitation experiments. The numbers of found text-mining relations and of predicted binding sites is quite different for the model organisms.

Species	#Genes	#TF	#PWM	$\frac{100\times\#\text{TF}}{\#\text{Gene}}$	#Data-base	#RelEx	#SL	#Tri-Occ.	#TFBS
H. sapiens	21,673	1,416	300	6.5	3,230	20,391	29,422	103,511	220,245
M. musculus	23,497	1,431	276	6.1	932	10,682	15,616	51,729	130,456
R. norvegicus	22,503	1,181	20	5.2	321	5.950	8,905	33,857	3,050
D. rerio	21,322	1,081	0	5.9	0	2,930	4,322	16,219	0
D. melanogaster	14,076	570	139	4.0	471	2,433	3,802	11,635	6,054
C. elegans	19,992	688	6	3.2	128	102	149	385	1,308
A. thaliana	26,207	1,235	32	3.4	11,284	926	1,460	5,073	8,282
S. cerevisiae	5,884	233	170	4	29,716	812	1,446	4,036	6,075

2.1 Regulatory Data

For the source and target species, data from different data sources is available in our system. Table 1 gives an overview of the collected data in ConReg. For *S. cerevisiae* and *A. thaliana*, processed data from genome-wide chromatin immunoprecipitation (ChIP) experiments for some TFs was additionally available. This explains why quite many regulatory relations were found for these two species. For the other species only very few relations could be found which emphasizes the need of text-mining approaches to get a more complete view on the currently discovered regulatory networks. For instance, for *D. rerio* no relations were found in the databases, but 16,219 putative relations were found using text-mining. Nevertheless, we assume that data extracted from databases is reliable and use this data as source data for the discovery of conservations, whereas also the predicted relations are considered for the conservation search in the target species.

For our prediction methods, most relations were found with the Tri-occurrence text-mining approach and the TFBS predictions, but likely with a large number of false positives. Unfortunately, for some organisms the number of position weight matrices (PWM) for the search of TFBS is very limited. For *D. rerio* no PWMs were available and for *C. elegans* and *R. norvegicus* only six and 20 PWMs could be found in the public domain. This explains the comparably low number of binding site predictions for these three species. The Tri-occurrence approach was used as pre-filter for the more sophisticated relation extraction approaches RelEx and SL. By comparing the relations found with RelEx to known relations extracted from databases a small overlap can be observed. For example for *H. sapiens* 22 % of the database relations could also be found with RelEx. A similar consistency could be observed for *R. norvegicus*, *M. musculus* and *D. melanogaster*. For SL 21 % of the known relations from *H. sapiens* could be detected. By combining the two state-of-the-art relation extraction approaches RelEx and SL, this rate could be increased to 28 % for *H. sapiens*. For those species with regulatory data from ChIP experiments (*S. cerevisiae* and *A. thaliana*), this fraction is much lower as can be seen on the number of found regulatory text-mining relations. Furthermore, for *A. thaliana* and *C. elegans* only 34,729 and 31,325 species relevant abstracts could be found, whereas for *H. sapiens* and *M. musculus* 13,053,996 and 1,121,698 abstracts could be used. This explains the quite small number of relations for *A. thaliana* and *C. elegans* extracted with our text-mining approaches.

Most of the TFBS predictions could neither be confirmed with databases knowledge nor with the text-mining results. For example for *S. cerevisiae* 84 % of the predictions were unique for this method. The number and accuracy of TFBS predictions strongly depends on the available PWMs and their quality. Short PWMs for example produce many hits, but only with low scores which were not considered for the predicted relations in ConReg.

The currently available regulatory data is distributed in many different databases and stems from different data sources such as manual literature curations or genome-wide ChIP experiments. In general, the overlaps and consistency between different sources are still quite small. The collection and integration of data from different sources and organisms is a difficult task and needs to be continued to make the most out of the available knowledge.

2.2 ConReg for the Discovery of Conserved Relations in *D. melanogaster*

We used *D. melanogaster* as source species to outline the usability of our system to find conserved regulatory relations for the 471 documented regulatory relations in REDfly[10]. We selected as target species the vertebrates *H. sapiens*, *M. musculus*, *R. norvegicus* and *D. rerio* and used all available data sources for these species. *D. melanogaster* is phylogenetically distant from the other species, but several conserved motifs are described in the literature as already mentioned in the introduction. We checked all conserved relations predicted by ConReg. We assume that relations extracted from databases are correct and manually checked the relations found with our Tri-occurrence approach by reading the literature reported as evidence for each found relation. The Tri-occurrence relations are a super set of the relations extracted with our other text-mining approaches so that the performance for these approaches could also be checked, whereas the TFBS predictions were compared to the relations found in the databases and with the text-mining approaches.

The entire predicted conserved network is shown in Figure 2. Manual annotations where we could confirm a conserved regulatory relation between the orthologs in at least one vertebrate are shown as red edges. The conserved *D. melanogaster* network also contains the well-studied motifs for eye-development (*Optix*, *ey*, *eya* and *shf*) and conservations for the pan-bilaterian kernel for heart specification, including the genes *Tin*, *Mef2* and *Mad*.

Only seven conserved relations, involving nine different genes could be identified with target relations extracted from databases. From these seven relations four were auto-regulations and the others were isolated edges. By using only the knowledge from databases, not even the well-studied conserved motifs between *D. melanogaster* and the other organisms could be rediscovered. With our Tri-occurrence approach 132 possible conservations could be found from which we could confirm 66 relations (50 %) in at least one species. We compared the different methods to each other with respect to the number of predicted and confirmed relations. Furthermore, we compared the intersections of the predicted conserved relations from the different approaches (see Figure 3). All of the 67 found conservations found with RelEx could be confirmed or were also found by SL or the binding site predictions. With SL six additional validated conservations could be found. In addition, 124 possible conserved relations were discovered with the TFBS predictions. 33 of these relations could be found with a different method including 25 confirmed Tri-occurrence relations. We note, that with RelEx the best relation extraction performance could be achieved with 57 out of 67 confirmed

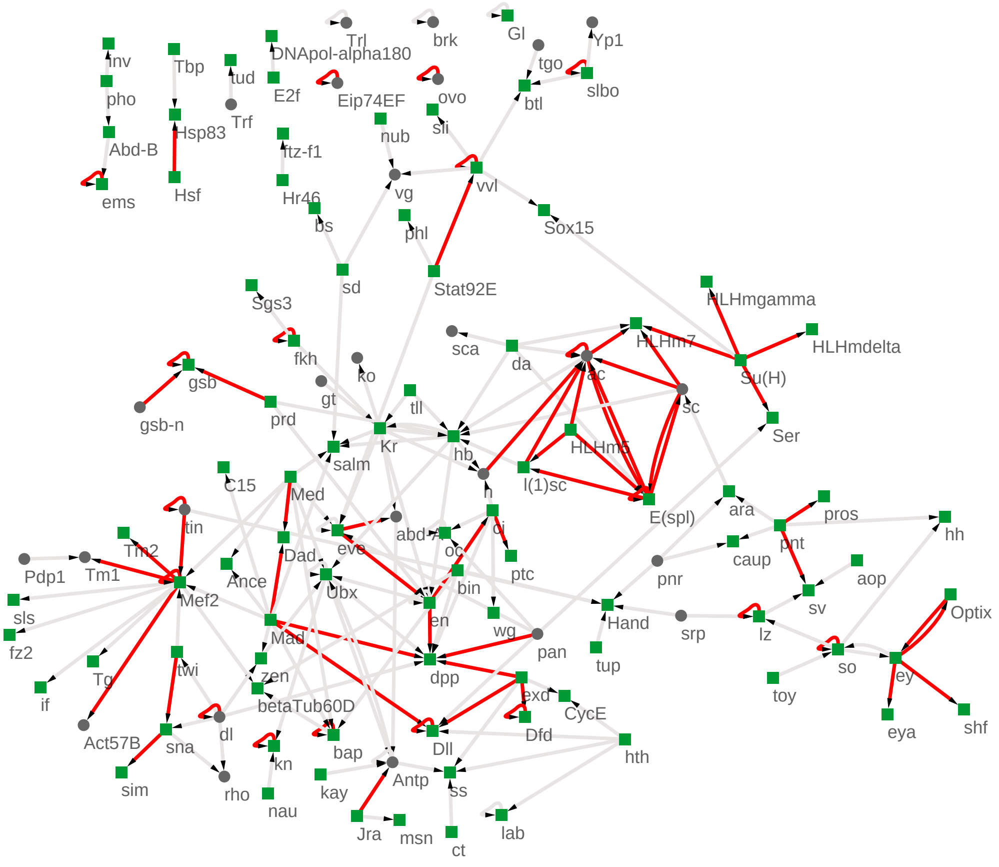


Figure 2 Network of conserved regulatory relations from the 471 documented regulatory relations in REDfly for *D. melanogaster* and in at least another vertebrate. The gray edges represent all relations where we could find a possible conservation. Red edges represent edges where we could confirm the relations between the orthologs in vertebrates using the literature references provided by ConReg. In green, we highlighted the nodes where at least two ortholog identification approaches found an ortholog mapping to another vertebrate for the respective gene. The conserved network contains among others, the well studied motifs for eye-development and the pan-bilaterian kernel for heart specification.

conserved relations (85 %). With SL a similar performance could be reached with 59 out of 76 confirmed conserved relations (78 %).

2.3 Comparison to alternative tools

There are several other tools which support the identification of conserved relations in eukaryotes. For example, with the UCSC Genome Browser [7] one can map TFBS (e.g. from ORegAnno [19]) and genome-wide chromatin immunoprecipitation experiments to the genome of interest together with the conservation of DNA sequences for different species. Another tool, the Genomatix suite[1], allows for uploading experimental data and for searching for conservations.

[1] http://www.genomatix.de

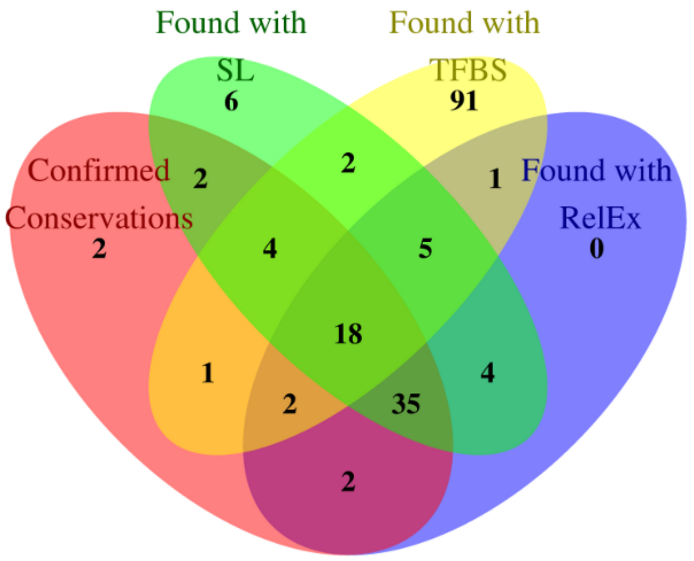


Figure 3 Venn-Diagram of found regulatory conservations between the 471 documented regulatory relations in REDfly for *D. melanogaster* and vertebrates. Confirmed conservations are regulatory relations which could be transferred from databases or which are correctly identified with our Tri-occurrence approach for at least one vertebrate (relations were manually checked by reading the corresponding literature). All relations found with RelEx could also be found with another method, whereas most of the TFBS predictions were not reported with our text-mining approaches.

Compared to prokaryotic genomes, eukaryotic genomes are rich in non-coding sequences of unknown functions and promoters can be several kilobases upstream from the transcription start site. Nevertheless, different approaches have been introduced to predict conserved binding sites [16, 4].

Furthermore, for microbial gene regulatory networks different platforms exist for the storage and web based analysis as reviewed by Baumbach *et al.* [3].

In comparison to these tools, ConReg focuses on eukaryotes only. ConReg provides detailed information of possible conservations. The main contribution of ConReg is the addition of conserved relations mined from the publicly available literature, only a small fraction of which is currently represented in databases. Moreover, TFBS predictions are also integrated. All these data can easily be selected via the web-interface of ConReg, via a Cytoscape [28] plug-in or downloaded from our server.

3 Conclusion

ConReg is a novel interactive online system for the discovery of conserved regulatory relations in currently eight eukaryotic model organisms. Our system allows searching for regulatory conservations among arbitrary user-definable sets of target species and outputs several annotations for predicted conserved relations.

We collected regulatory relations from databases, via text-mining from scientific text descriptions and from binding site predictions. We observed a severe incompleteness of regulatory relations in databases which are not even sufficient for the discovery of well-known conserved motifs. This slightly improves via the integration of information from state-of-the-art text-mining approaches and binding site predictions. E.g., several conserved motifs could be found using *D. melanogaster* as source species. For this showcase ConReg could identify

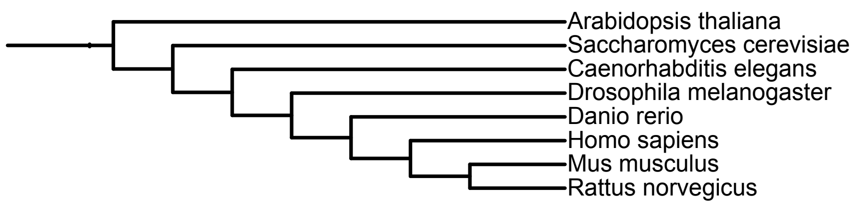


Figure 4 The eight species considered in our study and the associated phylogenetic tree as extracted from the NCBI taxonomy tree. Currently, ConReg contains six animal model organisms (*H. sapiens*, *M. musculus*, *R. norvegicus*, *D. rerio*, *D. melanogaster*, *C. elegans*) as well as *S. cerevisiae* and the model plant *A. thaliana*.

conserved regulations for 14 % (66 out of 461) of the relations from *REDfly* in at least one vertebrate. Still, it remains unknown to which extent regulatory relations are conserved since only few regulatory relations are experimentally confirmed for eukaryotes.

For our selected showcase we noticed that even with the simple Tri-occurrence text-mining approach 50 % of the identified regulatory relations are correctly identified if also experimentally validated regulatory relations between orthologs are known. Thus, with the integration of additional background knowledge the relation extraction could be significantly increased. We designed our system in such a manner, that other information sources can easily be added. In particular, we are planning to incorporate information from the increasing number of available ChIP experiments into ConReg.

Availability

The ConReg web interface is publicly available at `http://services.bio.ifi.lmu.de/ConReg/` and an interface is provided as Cytoscape plug-in to access the data for follow-up analyses. Furthermore, the extracted text-mining relations are provided for download on our website.

4 Materials and Methods

4.1 Data Sources

We collected regulatory relations for *H. sapiens*, *M. musculus*, *R. norvegicus*, *D. rerio*, *D. melanogaster*, *C. elegans*, *A. thaliana* and *S. cerevisiae* (see Figure 4 for a phylogenetic tree of these species). Regulatory relations were extracted from the multi-species curated databases TRANSFAC (Version 9.3) [18] and ORegAnno [19]. Species-specific relations were extracted from YEASTRACT [31], REDfly [10] and AtRegNet [20] and from curated pathways from Biocarta and NCI-Pathway [24]. Transcription factors were collected from these databases and the transcription factor prediction database [14]. For the transfer of relations, we used orthologs from InParanoid [21], EnsemblCompara [35] and OMA [25]. These databases were used due to the evaluation results in [1, 12] and the coverage of ortholog mappings for all considered species. All genes were mapped to Ensembl to obtain unique genomic locations and the associated annotations. Relations involving genes which could not be mapped were not considered.

4.2 Regulatory Relation Extraction from the Scientific Literature

Abstracts from PubMed (20,766,340 abstracts) and the corresponding full text publications from the PubMedCentral open access subset (389,322 documents) were used to search for regulatory relations in textual descriptions. In order to find relations in unstructured descriptions two tasks have to be accomplished: the named entity recognition (NER) of gene names and the correct identification of relations between genes. For example, consider the following sentence from [17]: *"There is evidence that the expression of Six3 is regulated by Pax6."* To infer a regulatory relation, the gene names *Six3* and *Pax6* need to be found and the regulatory relation between *Pax6* → *Six3* has to be identified. We used syngrep [5], a dictionary based NER tool, for the gene name recognition and the mapping of gene names to identifiers. Dictionaries were compiled by combining gene names, aliases and synonyms from UniProt, Ensembl, HGNC, MGI, RGD, Tair and FlyBase. Regulatory relations between genes were initially identified with a simple Tri-occurrence approach. For this approach, a relation was assumed between all pairs of genes which were found in a sentence, if a keyword indicating a regulatory relation was found and at least one gene is annotated as a TF. For this task, we defined a list of keywords, which are supposed to indicate regulatory interactions, like *regulates*, *represses*, or *downregulates*. Such a Tri-occurrence approach provides a good recall, but also implies many false positives. Therefore, we also used the following more sophisticated relation extraction approaches to filter the discovered relations found with the Tri-occurrence approach:

RelEx [8]: RelEx is a rule based relation extraction tool using dependency parse trees to find relations.

SL [9]: SL is a shallow linguistics SVM kernel for the identification of relations. Since no model was available for the identification of regulatory relations with this kernel, we used the simple margin active learning [34] approach to train a SVM. A set of 175 positive and negative relations was used to learn an initial model. This model was refined by applying the learned predictor to 10,000 randomly selected relations found by the Tri-occurrence approach. The 100 instances which were closest to the separating margin of the SVM were manually annotated and used for the next round of SVM training. The model was iteratively refined with this approach until no further performance improvement on a control set of 100 examples (including 33 positive regulatory relations) could be observed. An area under the receiver operating characteristic (ROC) of 0.85 and an area under the precision-recall curve of 0.72 could be achieved with the final model on the control set. Furthermore, with a standard probability threshold of 0.5 for the SVM, a precision of 0.56 and a recall of 0.75 were reached (see Figure 5 for the ROC curve of the final predictor).

We decided to use RelEx and SL due to their good performance on the task of identifying undirected protein-protein interactions on different corpora [33]. The Tri-occurrence and the SL kernel approach predict only undirected relations. We used our list of TFs to derive the direction from the transcription factor to the non-factor. In the case that both genes are non-factors or both are factors the relations were omitted by default in our system. Gene names between closely related species can highly overlap. Therefore, we identified the species context in each abstract using a pre-defined set of possible names for the different species.

4.3 Transcription Factor Binding Site Predictions

The promoter sequence for each gene was extracted using RSAT [32]. The same promoter size of 1 kilo base pairs upstream of the transcription start site was chosen for all species.

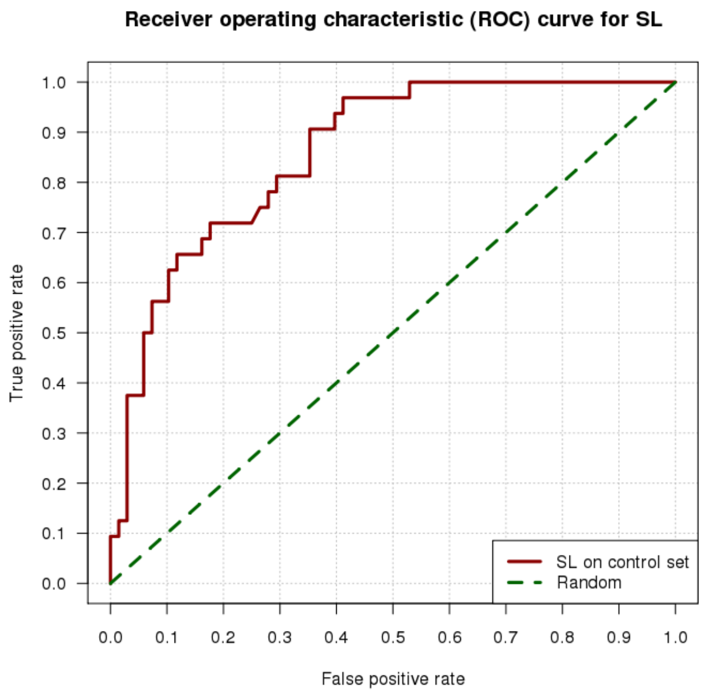


Figure 5 Receiver operating characteristic (ROC) curve for the final shallow linguistics (SL) SVM model which was used for the identification of regulatory relations. The evaluation set consists of 100 examples including 33 positive regulatory relations and 67 negative regulatory relations. On this control set the model reached an area under the ROC of 0.85 and an area under the precision-recall curve of 0.72. With a standard probability threshold of 0.5 for the SVM, a precision of 0.56 and a recall of 0.75 were reached.

Binding motifs for the different TFs were taken from TRANSFAC [18] and JASPAR [23]. The matching of these motifs to the promoter sequences was predicted with the R package cureos [37]. We used an empirically chosen threshold of 16 on the TFBS scores to filter out insignificant binding sites.

4.4 ConReg System Design

ConReg was developed as object oriented Java application using the open-source Ajax Web application framework ZK. The underlying data was unified in a structured MySQL database.

Funding

This work was supported by the DFG international research training group (IRTG) RECESS 1563 via scholarships to RP and MB.

Authors' contributions

RP designed and implemented ConReg and drafted the paper. MB provided TFBS predictions. RZ designed the study and helped with the evaluations. All authors contributed in reading and editing the paper.

References

1 Adrian M. Altenhoff and Christophe Dessimoz. Phylogenetic and Functional Assessment of Orthologs Inference Projects and Methods. *PLoS Comput Biol*, 5(1):e1000262, 2009.

2 Jan Baumbach. On the power and limits of evolutionary conservation–unraveling bacterial gene regulatory networks. *Nucleic Acids Res*, 38(22):7877–7884, Dec 2010.

3 Jan Baumbach, Andreas Tauch, and Sven Rahmann. Towards the integrated analysis, visualization and reconstruction of microbial gene regulatory networks. *Brief Bioinform*, 10(1):75–83, Jan 2009.

4 Eugene Berezikov, Victor Guryev, and Edwin Cuppen. CONREAL web server: identification and visualization of conserved transcription factor binding sites. *Nucleic Acids Res*, 33(Web Server issue):W447–W450, Jul 2005.

5 Gergely Csaba. syngrep - Fast synonym-based named entity recognition. Master's thesis, LMU-Munich, 2008.

6 Eric H. Davidson. *The Regulatory Genome. Gene Regulatory Networks In Development and Evolution.* Academic Press, Amsterdam, 2006.

7 Pauline A. Fujita, Brooke Rhead, Ann S. Zweig, Angie S. Hinrichs, Donna Karolchik, Melissa S. Cline, Mary Goldman, Galt P. Barber, Hiram Clawson, Antonio Coelho, Mark Diekhans, Timothy R. Dreszer, Belinda M. Giardine, Rachel A. Harte, Jennifer Hillman-Jackson, Fan Hsu, Vanessa Kirkup, Robert M. Kuhn, Katrina Learned, Chin H. Li, Laurence R. Meyer, Andy Pohl, Brian J. Raney, Kate R. Rosenbloom, Kayla E. Smith, David Haussler, and W James Kent. The UCSC Genome Browser database: update 2011. *Nucleic Acids Res*, 39(Database issue):D876–D882, Jan 2011.

8 Katrin Fundel, Robert Küffner, and Ralf Zimmer. RelEx–relation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371, 2007.

9 Claudio Giuliano, Alberto Lavelli, and Lorenza Romano. Exploiting Shallow Linguistic Information for Relation Extraction from Biomedical Literature. In *In Proc. EACL 2006*, 2006.

10 Marc S. Halfon, Steven M. Gallo, and Casey M. Bergman. REDfly 2.0: an integrated database of cis-regulatory modules and transcription factor binding sites in Drosophila. *Nucleic Acids Res*, 36(Database issue):D594–D598, 2008.

11 Michael Hecker, Sandro Lambeck, Susanne Toepfer, Eugene van Someren, and Reinhard Guthke. Gene regulatory network inference: data integration in dynamic models – a review. *Biosystems*, 96(1):86–103, Apr 2009.

12 Tim Hulsen, Martijn A. Huynen, Jacob de Vlieg, and Peter M A. Groenen. Benchmarking ortholog identification methods using functional genomics data. *Genome Biol*, 7(4):R31, 2006.

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

14 Sarah K. Kummerfeld and Sarah A. Teichmann. DBD: a transcription factor prediction database. *Nucleic Acids Res*, 34(Database issue):D74–D81, 2006.

15 Myon-Hee Lee, Brad Hook, Guangjin Pan, Aaron M. Kershner, Christopher Merritt, Geraldine Seydoux, James A. Thomson, Marvin Wickens, and Judith Kimble. Conserved regulation of MAP kinase expression by PUF RNA-binding proteins. *PLoS Genet*, 3(12):e233, 2007.

16 Gabriela G. Loots and Ivan Ovcharenko. rvista 2.0: evolutionary analysis of transcription factor binding sites. *Nucleic Acids Res*, 32(Web Server issue):W217–W221, Jul 2004.

17 Martine Manuel, Thomas Pratt, Min Liu, Glen Jeffery, and David J. Price. Overexpression of Pax6 results in microphthalmia, retinal dysplasia and defective retinal ganglion cell axon guidance. *BMC Dev Biol*, 8:59, 2008.

18 V. Matys, O. V. Kel-Margoulis, E. Fricke, I. Liebich, S. Land, A. Barre-Dirrie, I. Reuter, D. Chekmenev, M. Krull, K. Hornischer, N. Voss, P. Stegmaier, B. Lewicki-Potapov, H. Saxel, A. E. Kel, and E. Wingender. TRANSFAC and its module TRANSCompel: transcriptional gene regulation in eukaryotes. *Nucleic Acids Res*, 34(Database issue):D108–D110, 2006.

19 S. B. Montgomery, O. L. Griffith, M. C. Sleumer, C. M. Bergman, M. Bilenky, E. D. Pleasance, Y. Prychyna, X. Zhang, and S J M. Jones. ORegAnno: an open access database and curation system for literature-derive promoters, transcription factor binding sites and regulatory variation. *Bioinformatics*, 22(5):637–640, 2006.

20 Saranyan K. Palaniswamy, Stephen James, Hao Sun, Rebecca S. Lamb, Ramana V. Davuluri, and Erich Grotewold. AGRIS and AtRegNet. a platform to link cis-regulatory elements and transcription factors into regulatory networks. *Plant Physiol*, 140(3):818–829, 2006.

21 M. Remm, C. E. Storm, and E. L. Sonnhammer. Automatic clustering of orthologs and in-paralogs from pairwise species comparisons. *J Mol Biol*, 314(5):1041–1052, 2001.

22 Richard Röttger, Ulrich Rückert, Jan Taubert, and Jan Baumbach. How Little Do We Actually Know? – On the Size of Gene Regulatory Networks. *IEEE/ACM Trans Comput Biol Bioinform*, 2012.

23 Albin Sandelin, Wynand Alkema, Pär Engström, Wyeth W. Wasserman, and Boris Lenhard. JASPAR: an open-access database for eukaryotic transcription factor binding profiles. *Nucleic Acids Res*, 32(Database issue):D91–D94, 2004.

24 Carl F. Schaefer, Kira Anthony, Shiva Krupa, Jeffrey Buchoff, Matthew Day, Timo Hannay, and Kenneth H. Buetow. Pathway Interaction Database. *Nucleic Acids Res*, 37:D674–9, 2009.

25 Adrian Schneider, Christophe Dessimoz, and Gaston H. Gonnet. OMA browser–exploring orthologous relations across 352 complete genomes. *Bioinformatics*, 23(16):2180–2182, 2007.

26 Kate Schroder, Katharine M. Irvine, Martin S. Taylor, Nilesh J. Bokil, Kim-Anh Le Cao, Kelly-Anne Masterman, Larisa I. Labzin, Colin A. Semple, Ronan Kapetanovic, Lynsey Fairbairn, Altuna Akalin, Geoffrey J. Faulkner, John Kenneth Baillie, Milena Gongora, Carsten O. Daub, Hideya Kawaji, Geoffrey J. McLachlan, Nick Goldman, Sean M. Grimmond, Piero Carninci, Harukazu Suzuki, Yoshihide Hayashizaki, Boris Lenhard, David A. Hume, and Matthew J. Sweet. Conservation and divergence in Toll-like receptor 4-regulated gene expression in primary human versus mouse macrophages. *Proc Natl Acad Sci U S A*, 109(16):E944–E953, 2012.

27 Rachita Sharma, Patricia A. Evans, and Virendrakumar C. Bhavsar. Regulatory link mapping between organisms. *BMC Syst Biol*, 5 Suppl 1:S4, 2011.

28 Michael E. Smoot, Keiichiro Ono, Johannes Ruscheinski, Peng-Liang Wang, and Trey Ideker. Cytoscape 2.8: new features for data integration and network visualization. *Bioinformatics*, 27(3):431–432, Feb 2011.

29 Damian Szklarczyk, Andrea Franceschini, Michael Kuhn, Milan Simonovic, Alexander Roth, Pablo Minguez, Tobias Doerks, Manuel Stark, Jean Muller, Peer Bork, Lars J. Jensen, and Christian von Mering. The STRING database in 2011: functional interaction networks of proteins, globally integrated and scored. *Nucleic Acids Res*, 39(Database issue):D561–D568, 2011.

30 Leila Taher, David M. McGaughey, Samantha Maragh, Ivy Aneas, Seneca L. Bessling, Webb Miller, Marcelo A. Nobrega, Andrew S. McCallion, and Ivan Ovcharenko. Genome-wide identification of conserved regulatory function in diverged sequences. *Genome Res*, 21(7):1139–1149, 2011.

31 Miguel C. Teixeira, Pedro Monteiro, Pooja Jain, Sandra Tenreiro, Alexandra R. Fernandes, Nuno P. Mira, Marta Alenquer, Ana T. Freitas, Arlindo L. Oliveira, and Isabel Sá-Correia.

The YEASTRACT database: a tool for the analysis of transcription regulatory associations in Saccharomyces cerevisiae. *Nucleic Acids Res*, 34(Database issue):D446–D451, 2006.

32 Morgane Thomas-Chollier, Matthieu Defrance, Alejandra Medina-Rivera, Olivier Sand, Carl Herrmann, Denis Thieffry, and Jacques van Helden. RSAT 2011: regulatory sequence analysis tools. *Nucleic Acids Res*, 39(Web Server issue):W86–W91, 2011.

33 Domonkos Tikk, Philippe Thomas, Peter Palaga, Jörg Hakenberg, and Ulf Leser. A comprehensive benchmark of kernel methods to extract protein-protein interactions from literature. *PLoS Comput Biol*, 6:e1000837, 2010.

34 Simon Tong, Daphne Koller, and Pack Kaelbling. Support Vector Machine Active Learning with Applications to Text Classification. In *Journal of Machine Learning Research*, pages 999–1006, 2001.

35 Albert J. Vilella, Jessica Severin, Abel Ureta-Vidal, Li Heng, Richard Durbin, and Ewan Birney. EnsemblCompara GeneTrees: Complete, duplication-aware phylogenetic trees in vertebrates. *Genome Res*, 19(2):327–335, Feb 2009.

36 S. Wawersik and R. L. Maas. Vertebrate eye development as modeled in Drosophila. *Hum Mol Genet*, 9(6):917–925, Apr 2000.

37 Frank Westermann, Daniel Muth, Axel Benner, Tobias Bauer, Kai-Oliver Henrich, André Oberthuer, Benedikt Brors, Tim Beissbarth, Jo Vandesompele, Filip Pattyn, Barbara Hero, Rainer König, Matthias Fischer, and Manfred Schwab. Distinct transcriptional MYCN/c-MYC activities are associated with spontaneous regression or malignant progression in neuroblastomas. *Genome Biol*, 9(10):R150, 2008.

38 Haiyuan Yu, Nicholas M. Luscombe, Hao Xin Lu, Xiaowei Zhu, Yu Xia, Jing-Dong J. Han, Nicolas Bertin, Sambath Chung, Marc Vidal, and Mark Gerstein. Annotation transfer between genomes: protein-protein interologs and protein-DNA regulogs. *Genome Res*, 14(6):1107–1118, Jun 2004.

39 Zongzhao Zhai, Nati Ha, Fani Papagiannouli, Anne Hamacher-Brady, Nathan Brady, Sebastian Sorge, Daniela Bezdan, and Ingrid Lohmann. Antagonistic regulation of apoptosis and differentiation by the Cut transcription factor represents a tumor-suppressing mechanism in Drosophila. *PLoS Genet*, 8(3):e1002582, Mar 2012.

Designing q-Unique DNA Sequences with Integer Linear Programs and Euler Tours in De Bruijn Graphs

Marianna D'Addario[1,2], Nils Kriege[2], and Sven Rahmann[1,3]

1 Collaborative Research Center (Sonderforschungsbereich, SFB) 876, TU Dortmund, Germany
2 Computer Science XI, TU Dortmund, Germany
{Marianna.Daddario,Nils.Kriege}@tu-dortmund.de
3 Genome Informatics, Institute of Human Genetics, Faculty of Medicine, University of Duisburg-Essen, Germany
Sven.Rahmann@uni-due.de

Abstract

DNA nanoarchitechtures require carefully designed oligonucleotides with certain non-hybridization guarantees, which can be formalized as the q-uniqueness property on the sequence level. We study the optimization problem of finding a longest q-unique DNA sequence. We first present a convenient formulation as an integer linear program on the underlying De Bruijn graph that allows to flexibly incorporate a variety of constraints; solution times for practically relevant values of q are short. We then provide additional insights into the problem structure using the quotient graph of the De Bruijn graph with respect to the equivalence relation induced by reverse complementarity. Specifically, for odd q the quotient graph is Eulerian, so finding a longest q-unique sequence is equivalent to finding an Euler tour and solved in linear time with respect to the output string length. For even q, self-complementary edges complicate the problem, and the graph has to be Eulerized by deleting a minimum number of edges. Two sub-cases arise, for one of which we present a complete solution, while the other one remains open.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.2.2 Graph Theory, G.2.3 Applications

Keywords and phrases DNA sequence design, De Bruijn graph, quotient graph, reverse complement, Euler graph, Euler tour

Digital Object Identifier 10.4230/OASIcs.GCB.2012.82

1 Introduction

DNA synthesis technology allows to synthesize custom oligonucleotides of up to 80–100 basepairs (bp). Such oligonucleotides may assemble into larger regular structures, such as grids of 4x4 tiles [6]. These structures in turn provide a basis for attaching other molecules to protruding DNA arms, such as antibodies to capture specific proteins, which is why the research field of DNA nanoarchitechtures offers exciting future prospects for molecular biology and medicine.

Single-stranded DNA molecules form stable double-stranded helices according to the canonical Watson-Crick base pairing rules (A-T, G-C). The stability of a DNA double helix is given by a fine balance of interactions, such as hydrogen bonds between bases, stacking interactions of parallel bonds, and thermodynamic properties, such as the melting temperature or the free Gibbs Energy of a DNA sequence [9].

The problem of designing appropriate DNA oligonucleotide sequences for assembly into a desired structure has been pioneered by Seeman, who in cooperation with Kallenbach developed the semi-automated SEQUIN software [10] for this purpose. Later, Feldkamp developed the DNA sequence compiler (`dsc` [7]) that allows to specify constraints between oligonucleotides as a mini-language and automates the search process by randomized construction and backtracking on sequences that violate the specifications.

For both approaches, a central aspect of the design problem is that the chosen oligonucleotides must be *q-unique* (see Section 2). Of course, besides q-uniqueness, other factors play an important role as well, such as uniform thermodynamic stability of all hybridizing regions in the desired structure. Here, however, we focus on the theoretical and combinatorial aspects of q-uniqueness.

We present a novel view on the computational problem of finding a longest q-unique DNA sequence. This is of practical relevance: A long q-unique sequence can be cut at arbitrary positions to obtain sets of oligonucleotides of any length that do not interact with each other. For practically relevant values of q, the problem is straightforwardly solved as an integer linear program (ILP) defined on a (standard directed) De Bruijn graph (Section 3). This approach has the advantage that it allows to incorporate quite general constraints (exclusion of certain substrings such as long homopolymers, inclusion of mandatory substrings, etc.) and can be solved with standard technology.

Interestingly, as we show in Section 4, the basic design problem can be cast as an Euler tour problem in an undirected variant of the De Bruijn graph, essentially the original De Bruijn graph modulo the equivalence relation induced by reverse complementarity. This graph is Eulerian for odd values of q, and the maximization problem has a closed-form solution. For even values of q, the graph is not Eulerian, and the problem arises to convert it into an Eulerian graph with a minimum number of edge (q-gram) deletions. An explicit solution is obtained when self-complementary edges are allowed in the graph, but when they are forbidden, a closed-form solution remains open.

2 Preliminaries

The *reverse complement* $\overline{s}$ of a sequence $s = \sigma_1\sigma_2\cdots\sigma_n$ over the DNA alphabet $\Sigma = \{\text{A,C,T,G}\}$ is defined as $\overline{s} := \overline{\sigma}_n\overline{\sigma}_{n-1}\cdots\overline{\sigma}_1$, where $\overline{\text{A}} = \text{T}$, $\overline{\text{T}} = \text{A}$, $\overline{\text{C}} = \text{G}$ and $\overline{\text{G}} = \text{C}$. A length-$q$ string is an element of Σ^q and called a *q-gram*. A q-gram that is a substring a given sequence s is called a *q-gram of s*. In the following, for a given sequence s and a given (small) q, we consider the sequence of overlapping q-grams of s. For $s = \text{GATTACA}$ and $q = 4$, we obtain the q-gram sequence (GATT, ATTA, TTAC, TACA).

A sequence s is said to be *q-unique* [7] if the following requirements are fulfilled.

1. Every q-gram occurs at most once in s.
2. If a q-gram occurs in s, then its reverse complement does not.

Note that the second requirement implies that (for even q) no self-complementary q-gram may occur in s. For odd q, self-complementary q-grams do not exist.

The q-uniqueness requirements (for small values of q) ensure that the designed sequence does not hybridize to itself in a stable way, except at exact reverse complementary counterparts, which are not part of the designed sequence, but introduced deliberately at a later stage. The first requirement ensures that each q-gram of s has exactly one reverse complementary counterpart and hence binds at a well-defined location to the reverse complement of s. If this requirement is violated, switching may occur, as shown in Figures 1a, 1b, which leads to unstable DNA structures. The second requirement ensures that s does not bind to itself; the

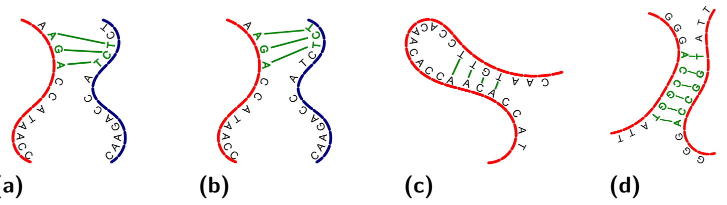


Figure 1 Sequences violating the q-uniqueness requirements. (a), (b): The q-gram TCT occurs twice, which leads to two alternative hybridizations. (c), (d): Existence of (self-)complementary q-grams results in the possibility of the sequence folding unto itself or binding to another molecule of the same type.

adverse case is shown in Figures 1c, 1d.

We note that there exist 4^q DNA q-grams. The number of self-complementary q-grams is zero if q is odd, and $4^{q/2}$ if q is even. Thus, an upper bound on the number of q-grams that can be used in a q-unique sequence is $M_q := 4^q/2$ if q is odd, and $M_q := (4^q - 4^{q/2})/2$ if q is even. If self-complementary q-grams are allowed (a variant of the problem that is irrelevant in practice, but combinatorially interesting), the M_q-bound for even q is increased by $4^{q/2}$. Thus $M'_q = 4^2/2$ if q is odd and $M'_q = (4^q + 4^{q/2})/2$ if q is even.

The optimization problem at hand is to find a q-unique sequence of maximal length. In the following, we study the question whether the upper bound M_q can be achieved. For this, the M_q selected q-grams would have to be ordered such that consecutive q-grams overlap by $(q-1)$ characters. We use a De Bruijn graph as a convenient data structure to study the optimization problem.

3 An Integer Linear Program on the De Bruijn Graph

We briefly recall the definition of a (directed) De Bruijn graph (DBG).

▶ **Definition 1** (De Bruijn graph). The *De Bruijn graph* of order $q \geq 2$ over the alphabet Σ has vertices and edges

$$V := \Sigma^{q-1},$$
$$E := \{(\sigma_1 s, s\sigma_2) : \sigma_1, \sigma_2 \in \Sigma,\ s \in \Sigma^{q-2}\}.$$

Thus, the vertices are $(q-1)$-grams and the edges represent q-grams by connecting two vertices such that the source's $(q-2)$-suffix equals the sink's $(q-2)$-prefix.

Traversing a DBG using every edge at most once and avoiding edges labeled with self-complementary q-grams and complementary q-grams of used edges results in a path that describes a q-unique sequence. The optimization problem of finding a longest q-unique sequence can now be cast as an edge selection problem in the DBG: Find an edge sequence of maximal length that forms a path (or cycle), such that each edge is traversed at most once, traversal of an edge implies non-traversal of its reverse complement, and (for even q) no self-complementary edges are traversed.

This problem in turn can be conveniently cast as an integer linear program (ILP); see Table 1. Three sets of binary indicator variables are defined: $(x_e)_{e\in E}$, $(a_v)_{v\in V}$, and $(z_v)_{v\in V}$. The variable x_e indicates whether edge e is selected. Variables a_v and z_v indicate the start vertex and end vertex of a path, respectively. The objective function naturally maximizes the number of selected edges.

Table 1 ILP for selecting a maximal number of edges under q-uniqueness constraints. Selected edges may form cycles and/or one path. All variables are binary indicator variables: edge selection indicators x_e, path start indicators a_v, and path end indicators z_v. Functions γ, In and Out are defined in the text.

$$\begin{aligned}
&\text{Maximize } \textstyle\sum_{e\in E} x_e \\
&\text{subject to} && 0 \le x_e \le 1,\quad x_e \in \mathbb{Z} && \forall e \in E,\\
& && 0 \le a_v \le 1,\quad a_v \in \mathbb{Z} && \forall v \in V,\\
& && 0 \le z_v \le 1,\quad z_v \in \mathbb{Z} && \forall v \in V,\\
& && x_e + x_{\gamma(e)} \le 1 && \forall e \in E,\\
& && \sum_{e\in In(v)} x_e + a_v = \sum_{e\in Out(v)} x_e + z_v && \forall v \in V,\\
& && \sum_v a_v \le 1, && \\
& && \sum_v z_v \le 1, && \\
& && a_v + z_v \le 1 && \forall v \in V.
\end{aligned}$$

Let $\gamma(e)$ denote the edge labeled with the reverse complement of e's label. To ensure q-uniqueness, we must not select both edge e and its complement, thus $x_e + x_{\gamma(e)} \le 1$ for all edges $e \in E$. If $e = \gamma(e)$, this becomes $2x_e \le 1$, which implies $x_e = 0$ because of the integer constraints.

Let $In(v)$ and $Out(v)$, respectively, be the set of incoming and outgoing edges of vertex v. To ensure the chosen edges form a path or cycle, we require that for every vertex the number of incoming edges equals the number of outgoing edges. If the path is not a cycle, the start vertex has one incoming edge less than outgoing edges, and the end vertex has one outgoing edge less than incoming ones. Thus we require $\sum_{e\in In(v)} x_e + a_v = \sum_{e\in Out(v)} x_e + z_v$ for all $v \in V$. To ensure that we obtain at most one start vertex and one end vertex and that they are not identical, we require $\sum_v a_v \le 1$, $\sum_v z_v \le 1$ and for all $v \in V : a_v + z_v \le 1$.

Table 1 summarizes the ILP. Its conditions do not ensure that the solution is a single path. Instead, the solution may consist of one or more cycles and at most one path. To ensure a single path or cycle, further constraints may be added: Assume that the optimal solution of the ILP in Table 1 consists of two vertex-disjoint components, such as two cycles on vertex sets V' and V'', respectively. Then we add a constraint requiring the existence of an edge between V' and V'' and solve the ILP again. This technique of adding violated constraints on demand is called cutting-plane method. However, we found that the solution of the ILP from Table 1 already produced a single cycle or path, i.e., adding cutting planes was not necessary.

Table 2 shows the ILP's solution for $2 \le q \le 8$. The ILP was implemented in Python with the `python-zibopt` library [8] and solved with the SCIP solver [1]. The obtained solutions were single cycles for odd q and single paths for even q. The ILP's optimal solution agrees with the upper bound M_q for odd q (cf. Section 2), but differs for even $q \ge 4$. These findings suggest that for odd q, the problem is trivial and that the solution agrees with the upper bound. However, self-complementary q-grams appear to complicate the problem. We now present a different approach that proves that for odd q, the above conjecture holds, and that the solution can be obtained in linear time (without resorting to an ILP).

Table 2 ILP results on DNA De Bruijn graphs of order $2 \le q \le 8$. $|E_q|$: number of edges, $|E_q^{\mathrm{sc}}|$: number of self-complementary edges, M_q: Upper bound on the number of selectable q-grams (see Section 2), opt: optimal ILP solution value ($\le M_q$), diff: $M_q -$ opt.

q	$\lvert E_q\rvert$	$\lvert E_q^{\mathrm{sc}}\rvert$	M_q	opt	diff
2	16	4	6	6	0
3	64	0	32	32	0
4	256	16	120	115	5
5	1024	0	512	512	0
6	4096	64	2016	1959	57
7	16384	0	8192	8192	0
8	65536	256	32640	32279	361

4 Euler Tours of the De Bruijn Quotient Graph

To represent the double-stranded nature of DNA directly in the graph, the central idea is to collapse vertices and edges whose label is the reverse complement of each other into a single vertex or edge. Formally, we define an equivalence relation $\equiv$ by $s \equiv t := (s = t \text{ or } s = \overline{t})$. Each equivalence class has size 1 (for self-complementary sequences) or 2 (all others). The De Bruijn graph modulo $\equiv$ on edges and vertices is called *De Bruijn Quotient Graph* (DBQG). It was introduced by Anderson et al. [2] in a different context (to construct universal DNA footprints). Each vertex of the quotient graph is jointly labeled with a $(q-1)$-gram and its reverse complement; each edge represents a q-gram and its reverse complement.

▶ **Definition 2** (De Bruijn quotient graph, DBQG). The *De Bruijn quotient graph* of order $q \ge 2$ over the DNA alphabet Σ has vertices and edges

$$\tilde{V}_q := \{\{s, \overline{s}\} : s \in \Sigma^{q-1}\},$$
$$\tilde{E}_q := \left\{\left\{\{\sigma_1 s,\, \overline{s}\,\overline{\sigma_1}\}, \{s\sigma_2,\, \overline{\sigma_2}\,\overline{s}\}\right\} : \sigma_1, \sigma_2 \in \Sigma,\ s \in \Sigma^{q-2}\right\},$$

where $\overline{s}$ denotes the reverse complement of a sequence s.

We observe that in the DBQG of order q,

$$|\tilde{V}_q| = [4^{(q-1)} + 4^{(q-1)/2}]/2, \qquad |\tilde{E}_q| = 4^q/2 \qquad \text{if } q \text{ is odd},$$
$$|\tilde{V}_q| = 4^{(q-1)}/2, \qquad |\tilde{E}_q| = [4^q + 4^{q/2}]/2 \qquad \text{if } q \text{ is even}.$$

A key characteristic of the quotient graph is that it contains paths not corresponding to valid sequences. Although the edges of the quotient graph are undirected, choosing one of the edge labels determines a direction. As a consequence, a vertex v which is reached via an edge e_1 must be left via an edge e_2, such that the label of v, the $(q-1)$-suffix of a label of e_1 and the $(q-1)$-prefix of a label of e_2 are identical. Since in this case, we may also enter v via e_2 and leave via e_1, we refer to e_1 and e_2 as *admissible edge pair*. Consider the vertex {ACT, AGT} in Figure 2. By entering it via CACT, the vertex label ACT is selected. Now only the edges starting with ACT can be followed to leave the vertex, here {ACTC, GAGT}, {ACTG, CAGT}, {ACTT, AAGT}, or {ACTA, TAGT}.

An *admissible path* in the DBQG is a path whose internal vertices are entered and left via admissible edge pairs. We consider only admissible paths from now on and "path" always refers to admissible path.

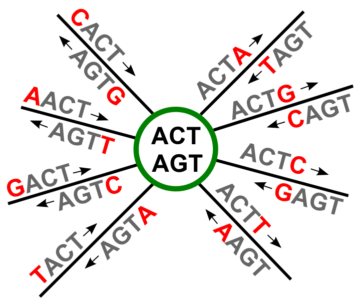
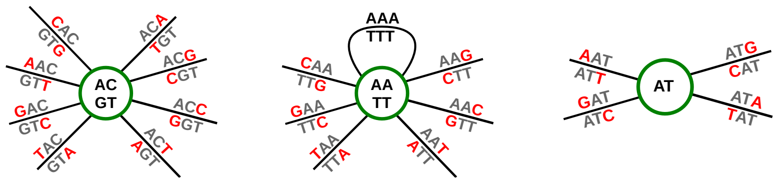


Figure 2 Admissible edge pairs (see text).



(a) Standard vertex **(b)** Homopolymer **(c)** Self-complementary

Figure 3 Vertex types of a DBQG with odd q.

The structure of the DBQG is more complex than the structure of the original DBG due to the presence of additional self-loops. In the DBG, an edge e is a self-loop from v to v if and only if e is labeled with a homopolymer, e.g., AA...A $= \mathrm{A}^q$ and v is labeled with the corresponding homopolymer of length $q-1$. However, in the DBQG, an edge e is a self-loop if its label is either a homopolymer or self-complementary (this new case only occurs for even q). In the following sections, we precisely discuss this fact's implications on the graph structure and the maximization problem.

Each self-loop is part of two admissible edge pairs: First the entering edge and the loop are one admissible edge pair and then the loop and a leaving edge are the second one.

We define a vertex to be *balanced* if all incident edges can be arranged in admissible edge pairs, including self-loops. A DBQG is *Eulerian* if there exists a path or cycle that uses each edge exactly once. The following lemma is an adaptation of the same lemma for standard undirected graphs and proved in exactly the same way.

▶ **Lemma 3.** *A DBQG is Eulerian if and only if each vertex is balanced.*

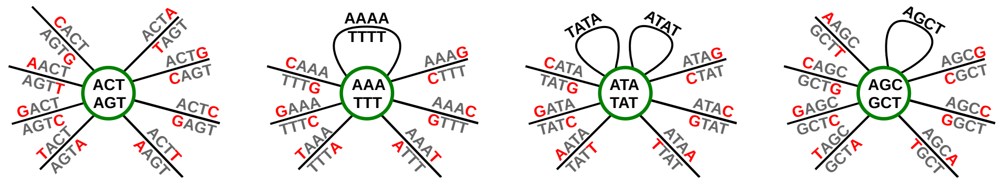
If the DBQG is Eulerian, an Euler tour may be constructed in linear time with a slight modification of the Hierholzer algorithm [4].

Note that a path that traverses each edge at most once and that does not traverse any self-complementary edge corresponds by definition to a q-unique double-stranded DNA sequence.

4.1 The Case of Odd q

Using the DBQG, we can understand why finding a longest q-unique sequence is easy when q is odd. In particular, the optimization problem can be solved in linear time with respect to the resulting q-unique sequence.

▶ **Theorem 4.** *For odd q, the DBQG is an Eulerian graph [2]. Each Euler tour is a cycle whose overlapping concatenated edge labels specify a q-unique sequence of maximum length.*



(a) Standard vertex **(b)** Homopolymer **(c)** Two self-complementary edges **(d)** One self-complementary edge

Figure 4 Vertex types of a DBQG with even q.

Proof. Recall from Section 1 that for odd q, the upper bound on the number of selectable q-grams is $M_q = 4^q/2 = |\tilde{E}_q|$. Thus if an Euler tour exists, each possible q-gram is used exactly once, and the upper bound is attained.

To prove that an Euler tour exists (this was already noted in [2]), we need to show that each vertex is balanced according to Lemma 3. Thus we consider the possible vertex types in a DBQG of odd order; see Figure 3. There are *self-complementary* vertices (the vertex label is self-complementary), *homopolymer* vertices (the vertex label is the repetition of a single nucleotide) and *standard* vertices. Note that the self-loop of a homopolymer vertex can be traversed when entering and leaving the vertex via an admissible edge pair. While the incident edges differ in number and type, each vertex is balanced. ◀

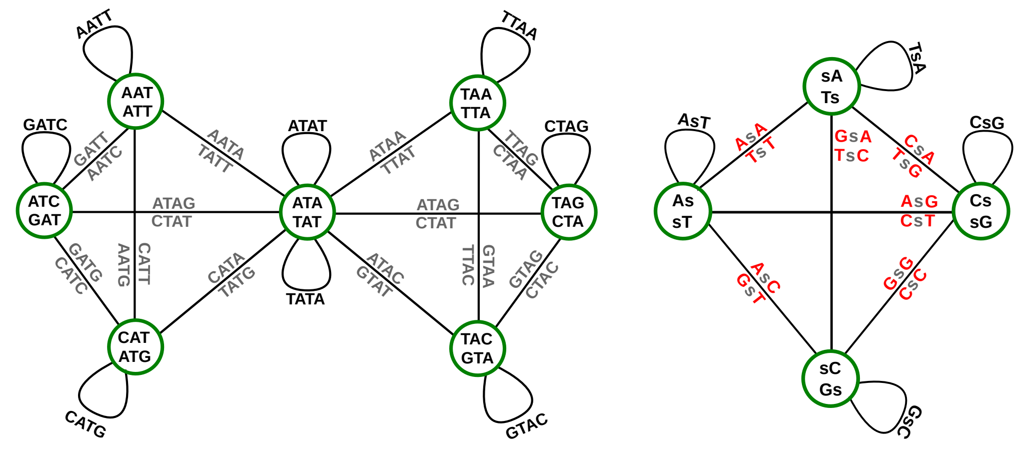
4.2 The Case of Even q

As in the odd case, we consider all types of vertices in a DBQG for even q, see Figure 4 and note that the graph is not Eulerian due to unbalanced vertices with one self-complementary edge (Figure 4d). It follows that an Euler tour cannot exist, and the bound M_q cannot be attained, explaining the differences in Table 2.

Finding a longest q-unique sequence now becomes the following problem: Given an undirected graph $G = (V, E)$, find an Eulerian graph $G^* = (V, E^*)$ with $E^* \subseteq E$ such that $|E \setminus E^*|$ is minimal. We call this process Eulerization-by-deletion. (This use of the term Eulerization is non-standard, as the literature reserves the term for Eulerization by duplication of existing edges.) In standard undirected graphs, Eulerization-by-deletion is polynomially solvable via minimum-weight perfect matchings [3, 5], where in our case the weight corresponds to the path length between the matched odd-degree vertices. Unfortunately this approach cannot be directly applied to the DBQG because of the restrictions imposed by admissible edge pairs.

We consider two cases for even q. The first one is of theoretical interest and assumes that the self-complementary edges still exist in the DBQG (i.e., self-complementary q-grams are allowed). We present a simple construction to optimally Eulerize the DBQG, so finding a maximal q-unique sequence is polynomially solvable. The second case is the original problem, where self-complementary edges are removed. For this case, we also present a construction to Eulerize the graph such that an Eulerian path exists; however, this construction is not optimal. Therefore, we only obtain a lower bound on the number of selectable edges, and the complexity remains open.

DBQG with self-complementary edges. Four types of vertices exist in the graph (Figure 4); only the vertices with one self-complementary edge (Figure 4d) are unbalanced. We investigate the the odd-degree vertices and their relations in the graph assuming $q \geq 4$.



(a) Bat-group: To obtain an Euler tour in the graph, four edges of this group must be deleted. Two of these must be incident to the central vertex. Every other vertex is incident to exactly one deleted edge. Non-deletable edges between these vertices are not shown.

(b) Kite-group: The substring s of length $q-2$ is self-complementary. To obtain an Euler tour in the graph, two edges of this group must be deleted and each vertex is incident to exactly one deleted edge.

Figure 5 Substructures in DBQG with even q.

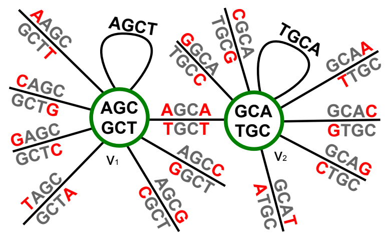


Figure 6 After deleting the central edge {AGCA, TGCT}, the vertices are balanced. Possible sets of paths through the vertex are {AAGC, GCTT}→{AGCC, GGCT}, {CAGC, GCTG}→{AGCG, CGCT} and {GAGC, GCTC}→{AGCT}→{GCTA, TAGC}. Using the self-complementary edge {AGCT} is essential for the balance.

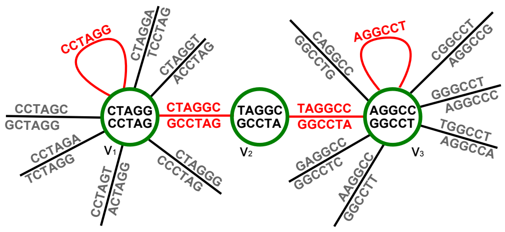


Figure 7 Path of length $q/2-1$ obtained by cyclic permutation of self-complementary edge CCTAGG into AGGCCT. When the self-complementary edges and the path are removed, all vertices on the path are balanced.

A *bat-group* is a set of 7 vertices centered around a vertex labeled $\{\sigma(\overline{\sigma}\sigma)^{q/2-1}, \overline{\sigma}(\sigma\overline{\sigma})^{q/2-1}\}$, $\sigma \in \Sigma$ (Figure 5a). There are only 2 bat-groups (the central vertex is labeled with ATAT. . . A or CGCG. . . C); each bat-group contains 8 self-complementary edges. The central vertex is balanced, but the other six are not. At least four edges must be deleted to balance each vertex in such a group, and Figure 5a shows several ways to achieve this.

Let s be a self-complementary $(q-2)$-gram. Then the *kite-group* of s is the set of 4 vertices that are not part of a bat-group and are labeled with $\{\sigma s, \overline{s}\,\overline{\sigma}\}$, for all $\sigma \in \Sigma$ (Figure 5b). Each kite-group contains four self-complementary edges. Because of the two bat-groups, there are $(4^{q/2}-16)/4$ kite-groups in the graph (for $q = 4$, there are none). At least two edges must be deleted to balance each vertex in a kite-group, and Figure 5b shows all possibilities to achieve this. In more detail, Figure 6 shows that a single edge (the central edge in the figure) between two problematic vertices can be deleted, leaving both vertices balanced. For the balance, the self-complementary edge in each of the vertices is essential.

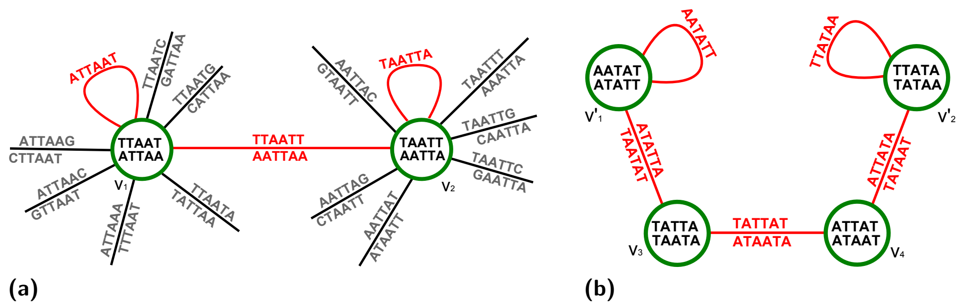


Figure 8 Alternative permutation path to delete: Vertex v_1 in (a) and vertex v_1' in (b) are the cyclic permutation pair described in Section 4; so are v_2 in (a) and v_2' in (b). Now, (a) shows a shorter path from v_1 to v_2 and (b) a longer one from v_1' to v_2'.

Summarizing, to Eulerize the graph, it is both necessary and possible to delete $\delta := 2 \cdot 4 + (4^{q/2} - 16)/4 \cdot 2 = 4^{q/2}/2$ edges. As we are equally satisfied with an Eulerian path instead of a cycle, we allow two unbalanced vertices and re-add two edges.

▶ **Theorem 5.** *In the DBQG for even q with self-complementary edges, an Eulerian subgraph with $N_q' := |\tilde{E}_q| - \delta + 2 \;=\; 4^q/2 + 2$ edges exists; larger Eulerian subgraphs do not exist.*

DBQG without self-complementary edges. When self-complementary edges are omitted, deleting an edge connecting two odd-degree vertices may leave them unbalanced (consider Figure 6 without the self-loops and with the central edge removed). Identifying an edge-minimal set of paths between pairs of these vertices, whose deletion leaves all vertices balanced, appears difficult and remains an open problem.

Here we present a systematic but sub-optimal construction. Consider a self-complementary q-gram of the form st, where s and t have length $q/2$ and are reverse complements of each other, and consider the vertices incident to st and its cyclic permutation ts. There is a path of length $q/2 - 1$ between these vertices (see Figure 7).

As there are two vertices with two self-complementary edges (Figure 4c), there are $4^{q/2}/2 - 2$ such (s,t) pairs. Since an Eulerian path instead of a cycle suffices, one of these paths may remain, resulting in $\Delta := (4^{q/2}/2 - 3)(q/2 - 1)$ edge deletions.

▶ **Theorem 6.** *In the DBQG for even q without self-complementary edges, an Eulerian subgraph with*

$$N_q \;:=\; (|\tilde{E}_q| - 4^{q/2}) - \Delta \;=\; (4^q - 4^{q/2})/2 - (4^{q/2}/2 - 3)(q/2 - 1)$$

edges exists. Larger Eulerian subgraphs may exist.

For even $q \geq 6$, the value of N_q does not match the optimal ILP solution and is therefore suboptimal (e.g., $q = 6$: $N_q = 1958$, optimum 1959; $q = 8$: $N_q = 32265$, optimum 32279). For $q = 6$, we can manually improve the Eulerization to match the ILP solution: Due to the symmetry of some self-complementary edges it is possible to find an alternative shorter path between two self-complementary edges. Figure 8 shows this alternative pairing. Deleting only the path in Figure 8a and preserving that in Figure 8b saves exactly one edge. In general, the problem of optimal Eulerization in this case remains open.

5 Discussion and Conclusion

We presented two approaches to generate q-unique sequences, via an ILP and an Euler tour in a De Bruijn quotient graph.

For odd q, the DBQG is Eulerian and an adaptation of a standard algorithm for finding Euler tours yields an optimal algorithm. Additionally, the number of different longest q-unique sequences is related to the number of Euler tours in the DBQG, which can be computed by a small modification (accounting for admissible pairs instead of all edges) of the BEST Theorem (de Bruijn, van Aardenne-Ehrenfest, Smith and Tutte, [11]) for standard undirected graphs.

For even q, the DBQG turns out not to be Eulerian, and edges must be removed to Eulerize it. In the variant where self-complementary edges are allowed, we presented an optimal Eulerization by deleting edges, proving this problem variant to be as easy as the case of odd q. When self-complementary edges are forbidden, the cyclic permutation construction gives suboptimal results compared to the optimal ILP solution for even $q \geq 6$.

In practice, the design problem of q-unique sequences is generally restricted to $q \in \{3, 4, 5\}$, but augmented by additional constraints: Some substrings might be prescribed (existing DNA libraries that must be used) and others forbidden (homopolymers of certain length, q-grams with too extreme GC-content). The ILP formulation supports such constraints by fixing certain variables and provides reasonable performance for relevant values of q.

Respecting constraints with the approach based on the DBQG corresponds to the removal of certain edges. Clearly, conducting a thorough structural analysis for individual constraints is prohibitive. Therefore, this work poses the general problem of Eulerizing a given quotient graph by removing a minimum number of edges, such that each vertex has even degree and allows admissible edge pairings.

Acknowledgments. We gratefully acknowledge funding by the DFG Collaborative Research Center (Sonderforschungsbereich, SFB) 876 "Providing Information by Resource-Constrained Data Analysis" within project TB1 (`http://sfb876.tu-dortmund.de`). We thank C. Niemeyer, B. Sacca and J. Ksienczyk for fruitful discussions, and P. Mutzel and T. Schwentick for very helpful pointers to the Eulerization-by-deletion problem. We also thank the participants of Dagstuhl seminar 12291 "Structure Discovery in Biology: Motifs, Networks & Phylogenies" organized by A. Dress, L. Parida and A. Apostolico for their valuable comments.

References

1 T. Achterberg. SCIP: solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009.
2 J.W. Anderson, K.R. Fox, and G.A. Niblo. A fast algorithm for the construction of universal footprinting templates in DNA. *Journal of mathematical biology*, 52(3):307–342, 2006.
3 W. Cook and A. Rohe. Computing minimum-weight perfect matchings. *INFORMS Journal on Computing*, 11(2):138–148, 1999.
4 R. Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer Verlag, 2005.
5 J. Edmonds and E.L. Johnson. Matching, euler tours and the chinese postman. *Math. Programming*, 5:88–124, 1973.
6 U. Feldkamp and C.M. Niemeyer. Rational design of DNA nanoarchitectures. *Angewandte Chemie International Edition*, 45(12):1856–1876, 2006.

7 U. Feldkamp, H. Rauhe, and W. Banzhaf. Software tools for DNA sequence design. *Genetic Programming and Evolvable Machines*, 4(2):153–171, 2003.

8 Ryan J. O'Neil. Python interfaces to the ZIB Optimization Suite. `http://code.google.com/p/python-zibopt/`. Accessed June 2012.

9 J. SantaLucia Jr, H.T. Allawi, and P.A. Seneviratne. Improved Nearest-Neighbor Parameters for Predicting DNA Duplex Stability. *Biochemistry*, 35(11):3555–3562, 1996.

10 N.C. Seeman and N.R. Kallenbach. Design of immobile nucleic acid junctions. *Biophysical journal*, 44(2):201–209, 1983.

11 T. van Aardenne-Ehrenfest and N.G. de Bruijn. Circuits and trees in oriented linear graphs. *Simon Stevin*, 28:203–217, 1951.

Polyglutamine and Polyalanine Tracts Are Enriched in Transcription Factors of Plants

Nina Kottenhagen[1*], Lydia Gramzow[1*], Fabian Horn[2], Martin Pohl[3], and Günter Theißen[1]

1 Friedrich-Schiller-Universität Jena, Lehrstuhl für Genetik, Philosophenweg 12, D–07743 Jena, {nina.kottenhagen,lydia.gramzow,guenter.theissen}@uni-jena.de
2 Leibniz-Institut für Naturstoffforschung und Infektionsbiologie e.V. - Hans-Knöll-Institut, Systembiologie/ Bioinformatik, Beutenbergstr. 8, D–07745 Jena, fabian.horn@hki-jena.de
3 Friedrich-Schiller-Universität Jena, Lehrstuhl für Bioinformatik, Ernst-Abbe-Platz 2, D–07743 Jena, m.pohl@uni-jena.de

Abstract

Polyglutamine (polyQ) tracts have been studied extensively for their roles in a number of human diseases such as Huntington's or different Ataxias. However, it has also been recognized that polyQ tracts are abundant and may have important functional and evolutionary roles. Especially the association of polyQ and also polyalanine (polyA) tracts with transcription factors and their activation activity has been noted. While a number of examples for this association have been found for proteins from opisthokonts (animals and fungi), only a few studies exist for polyQ and polyA stretches in plants, and systematic investigations of the significance of these repeats in plant transcription factors are scarce. Here, we analyze the abundance and length of polyQ and polyA stretches in the conceptual proteomes of six plant species and examine the connection between polyQ and polyA tracts and transcription factors of the repeat-containing proteins. We show that there is an association of polyQ stretches with transcription factors in plants. In grasses, transcription factors are also significantly enriched in polyA stretches. While there is variation in the abundance, length, and association with certain functions of polyQ and polyA stretches between different species, no general differences in the evolution of these repeats could be observed between plants and opisthokonts.

1998 ACM Subject Classification J.3 Life and Medical Sciences

Keywords and phrases tandem repeats, molecular evolution, GO annotation

Digital Object Identifier 10.4230/OASIcs.GCB.2012.93

1 Introduction

In general, amino acid repeats (AARs) in a protein can have very different consequences, ranging from causing severe diseases over neutral polymorphisms being suitable as genetic markers to "tuning knobs" of evolution [27, 28]. Polyglutamine (polyQ) tracts have found special interest because they are found in various severe human neurodegenerative or neuromuscular hereditary diseases [55]. In the case of Huntington's disease, aberrantly extended polyQ tracts in the HUNTINGTIN protein cause abnormal folding, subsequent protein aggregation and neuronal loss (reviewed in [41]). The repeat length and the severity of the disease are positively correlated: The longer the repeat, the earlier the age of onset and also

* These authors contributed equally to this work.

the more severe the symptoms. However, apart from their roles in diseases, the functional and evolutionary importance of polyQ stretches has also already been recognized, especially when they occur in transcription factors (TFs) [4, 17, 18, 19, 45].

Concerning the functional significance of polyQ stretches in TFs, their role in breeding and evolution has found special interest. Fondon and Garner [17], for example, studied protein repeats in RUNT-RELATED TRANSCRIPTION FACTOR 2 (RUNX-2), which is a main regulator of osteoblast differentiation, in various dog breeds. The authors found different lengths of adjacent polyalanine (polyA) and polyQ tracts in RUNX-2 in the different dog breeds, where the length ratio of these sequences correlates with the dorsoventral nose bend and midface length across breeds. As the repeat lengths evolve fast, Fondon and Garner concluded that repeats contributed to an acceleration of the morphological evolution of limbs and skulls in different dog breeds [17].

Other striking examples, where polyA or polyQ stretches in TFs may lead to phenotypic variability, were found in a broad range of species. In insects, the acquisition of a polyA transcription repression domain in the HOX protein ULTRABITHORAX may have contributed to the suppression of abdominal limbs during arthropod evolution and hence to the macroevolution of a body plan [45, 19]. A polymorphism at a polyA stretch in the protein Hoxd-13 may contribute to variation in sesamoid bone formation in amniotes [4]. The longer the polyQ tract in White Collar-1 (WC-1), a protein which influences the circadian clock of the fungus *Neurospora crassa*, the shorter are the circadian periods [18].

One important molecular mechanism by which differences in repeat lengths lead to phenotypic variation is likely due to the capacity of polyA and polyQ tracts to modulate transcription factor activity. Specifically, polyA tracts are thought to decrease and polyQ stretches are thought to increase transcriptional activation [20]. Hence, changes in the lengths of polyA and polyQ stretches of TFs may alter the transcription rate and implicate changes in the expression of a set of target genes. This could generate variation upon which selection acts and thus contribute to morphological changes [17].

Based on this exemplary evidence for the importance of polyA and polyQ tracts in TFs, it was hypothesized that, in general, polyA and polyQ stretches predominantly occur in TFs [20]. Subsequently, several systematic studies have been conducted as well. For a number of proteomes, the entirety of polyA and polyQ tracts was analyzed and the association of the repeat-containing proteins with functions in transcriptional activation was studied [29]. In yeast, polyQ stretches belong to the most abundant amino acid repeats found beside repeats of asparagine (N), aspartic acid (D), glutamic acid (E), and serine (S) [2]. Furthermore, repeats of acidic and polar amino acids, to which Q belongs, were found to be significantly associated with TFs and protein kinases. Similar results were obtained for other species. In rodents, humans, fruit flies, and nematodes, a functional bias of proteins with repeats, including A and Q repeats, was observed, where TFs were consistently overrepresented [1, 2, 23, 51].

In plants, however, systematic studies on polyA and polyQ tracts and their association with certain functions of the repeat-containing proteins are scarce [49, 59]. Plants, including Glaucophytes, Red algae, Green algae and Embryophytes (land plants), are one of the major eukaryotic groups whereas animals and fungi, for which association studies between amino-acid repeats and the function of the containing proteins are common, belong to another major group, the opisthokonts [58]. The most recent common ancestor of opisthokonts and plants was recently estimated to have lived at least one billion years ago [12]. Considering this long period of independent evolution between plants and opisthokonts, it is questionable whether polyA and polyQ tracts show the same abundance and variability and the same

functional bias of the repeat-containing proteins in these two major eukaryotic groups.
Amino-acid repeats are thought to expand or contract mainly by replication slippage or recombination of the corresponding protein-coding DNA [32, 43]. These mechanisms result in a rapid evolution of repeats. However, selection may act on repeat length as has been shown for opisthokonts [36]. For polyA and polyQ stretches in TFs, a correlation between repeat length and transcriptional activation has been shown [20] and hence the existence of repeats may be favored in some cases. On the other hand, long polyA and polyQ tracts may have negative effects, such as aggregation of the containing proteins, and thus there may be selection against polyA and polyQ stretches that are too long [10, 35]. All these factors, i.e., the mutational mechanism leading to the expansion and contraction of repeats, the capacity of polyQ and polyA to modulate transcription, and the negative effects of long repeats, may be different in plants. Hence, whether polyA and polyQ tracts are as important for phenotypic variation and morphological evolution in plants as in opisthokonts remains to be clarified.
Here, we analyze A and Q repeats in six proteomes of diverse species which span the phylogeny of land plants. We obtain the total number of polyA and polyQ tracts and their lengths in these proteomes and study the association between the containing proteins and a function in transcriptional regulation. We compare our findings to those found for opisthokonts and hypothesize on similarities and differences of the importance of polyA and polyQ stretches in opisthokonts and plants.

2 Materials and methods

To study the evolution of polyQ and polyA stretches in plants, six species spanning the phylogeny of land plants were selected with respect to their taxonomic placement and the availability of their proteome and corresponding GO annotations. These species are the moss *Physcomitrella patens*, the lycophyte *Selaginella moellendorffii*, the eudikotyledonous angiosperms *Arabidopsis thaliana* and *Populus trichocarpa* and the monocotyledonous angiosperms *Sorghum bicolor* and *Oryza sativa*. As an animal reference set, *Homo sapiens*, *Danio rerio*, and *Anopheles gambiae* were selected. We decided not to take our data from databases like COPASAAR [13], GENPEPT [14], TRIPS [29], ProtRepeatsDB [25], or ProRepeat [34], as they proved not to be customizable for our research. Therefore, conceptual proteome datasets for the selected species were obtained from the Ensembl, EnsemblMetazoa and EnsemblPlants project [16] (versions: May, 22nd 2012, `http://www.ensembl.org/index.html`, `http://metazoa.ensembl.org/index.html`, `http://plants.ensembl.org/index.html`). The datasets chosen contained all *ab initio* predicted protein sequences as well as manually curated sequences. Ensembl distinguishes between „known“, „novel“, and „putative“ proteins. For our analyses, all three classes were used. We selected only the first splice variant to avoid a bias due to overrepresentation of proteins originating from genes with multiple splice variants. The number of protein sequences included in our analyses for each species is given in Table 1.
Here, we define an AAR as a stretch of at least five identical amino acids in a row which is significant according to Karlin *et al.* (2002) [26]. The number and length of polyQ and polyA stretches within each species was determined using a custom Perl-script (all scripts are available upon request). The length of an amino acid repeat is defined as the number of residues forming the repeat. For comparison, the number and length of polyasparagine (polyN) stretches were also determined.
Gene Ontology (GO) annotation files [5] were downloaded with the help of Ensembl BioMart

[30] (version: May, 22nd 2012, v0.7, `http://www.biomart.org/biomart/martview/`). To check whether polyQ, polyA or polyN stretches show a specific enrichment pattern, a GO enrichment analysis was performed using a custom R-script based on the topGO package available from the Bioconductor website [3]. An exact Fisher test [15] was used to investigate whether certain GO categories are enriched in sequences containing a polyA, polyQ, or polyN stretch. The obtained p-values were corrected for multiple testing using the Benjamini-Hochberg method [8]. We focused on the GO category GO:0003700 (sequence-specific DNA binding transcription factor activity) as it has been shown for fungi and animal species that polyQ as well as polyA stretches are associated with this GO category [1, 2, 22, 51].
The significant results were assigned a rank according to their p-value. The lower the p-value was, the lower was the assigned rank. We use the p-value as a sign of the strength of the association. To compare the magnitude of the association between the studied repeats and the GO category GO:0003700, the average ranking of this GO category was compared between plants and animals. If the category was not identified as an enriched GO category, a penalty score was assigned to be able to calculate the average ranking. This penalty score was chosen to be the number of the highest rank of the category GO:0003700 across all studied plant and animal species plus one.

Table 1 Overview on source data. Species are in taxonomic order, animals are highlighted in gray.

Species	Number of protein sequences	Number of first splice variants	Number of annotated first splice variants	Reference
Arabidopsis thaliana	35386	27416	27155	[24]
Populus trichocarpa	45778	41377	26091	[56]
Oryza sativa	68619	57995	16994	[37]
Sorghum bicolor	36338	34496	21744	[38]
Selaginella moellendorffii	34825	34799	22650	[7]
Physcomitrella patens	38354	32273	8931	[42]
Homo sapiens	100354	22400	22400	[33]
Danio rerio	42171	26212	26212	[54]
Anopheles gambiae	14324	12670	8873	[21]

3 Results

3.1 Number and length of polyQ, polyA and polyN stretches

We studied amino acid repeats which had a length of at least five residues. On average, polyQ stretches were shorter in plants than in animals (Table 2). Also the maximum number of residues in a polyQ stretch was lower in plants than in animals. In *P. trichocarpa*, the longest polyQ stretch contained 33 residues, in *A. gambiae*, there were an exceptional 132 residues in the longest repeat. The total number of polyQ stretches was on average lower in plant proteins than in animal proteins, which is due to the exceptionally high number in *A. gambiae*.

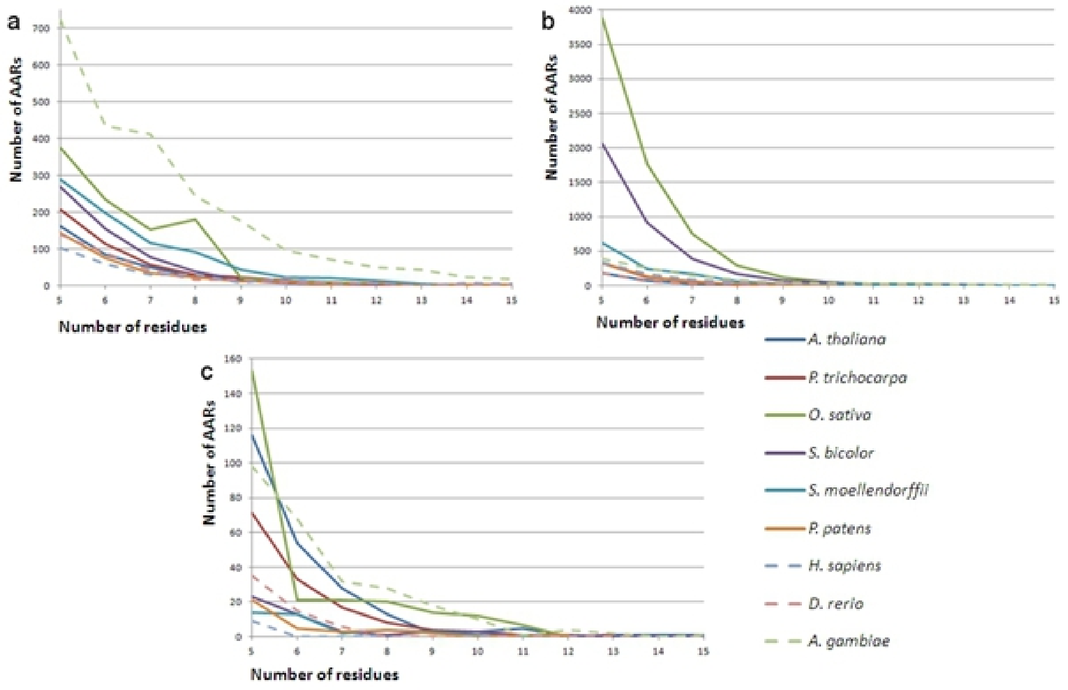


Figure 1 Length distribution of polyQ (a), polyA (b) and polyN (c) stretches in the conceptual proteomes of plant and animal species. Plant species are indicated by solid lines, broken lines represent animal species.

Similar to polyQ stretches, the average polyA stretch in plant proteins was a bit shorter than in animal proteins (Table 2). Also the maximum number of residues in a polyA stretch was lower in plants than in animals. Both numbers were lower as compared to polyQ stretches. The total number of polyA stretches varied widely within both kingdoms with plants harboring a higher number of them. PolyA stretches were also more commonly found than polyQ stretches in all species except *A. gambiae*.
PolyN was used as a control for polyQ because both amino acids are encoded by two codons and are chemically quite similar; their side chains differ from one another only by one methyl group. Unlike polyQ and polyA stretches, a higher average length was found in plant proteins than in animal proteins for polyN stretches (Table 2). The maximum number of residues in a polyN stretch was also higher in proteins of plants than of animals as well as the total number of polyN stretches. Across all species, fewer polyN stretches were found than polyQ and polyA stretches. However, the trends for polyN stretches were less clear than for polyQ and polyA stretches in both, plants and animals.
The length distributions for polyQ, polyA and polyN stretches followed the same trend across all species: While there are many short repeats the number of long repeats is low (Figure 1). *A. gambiae* harbored more polyQ stretches of every length than any other species. Remarkebly, however, also *O. sativa* showed an increased number of polyQ stretches of eight residues. PolyA stretches were most abundant in the grasses *O. sativa* and *S. bicolor*. For polyN stretches, no clear pattern could be found. In *O. sativa*, a plateau for polyN stretches of six to eleven residues was observed.

Table 2 Number, average repeat length and maximum length of polyQ, polyA, and polyN stretches in the conceptual proteomes of plant and animal species. Species are in taxonomic order, animals are highlighted in gray.

Amino acid	Species	Number of stretches	Average length of stretches	Longest stretch
Q	*A. thaliana*	376	6.61	24
	P. trichocarpa	462	6.44	33
	O. sativa	1005	6.49	27
	S. bicolor	571	6.08	24
	S. moellendorffii	811	6.74	25
	P. patens	322	6.63	22
	H. sapiens	307	8.79	40
	D. rerio	334	6.79	41
	A. gambiae	2373	7.68	132
A	*A. thaliana*	296	5.65	17
	P. trichocarpa	545	5.7	17
	O. sativa	6926	5.77	19
	S. bicolor	3705	5.83	18
	S. moellendorffii	1193	6.09	19
	P. patens	540	5.67	10
	H. sapiens	800	6.86	21
	D. rerio	335	6.12	18
	A. gambiae	1068	6.92	23
N	*A. thaliana*	226	6.13	20
	P. trichocarpa	138	5.99	13
	O. sativa	249	6.18	19
	S. bicolor	53	10.02	61
	S. moellendorffii	38	6.39	12
	P. patens	37	6.19	12
	H. sapiens	10	5.3	8
	D. rerio	57	5.58	11
	A. gambiae	263	6.56	16

3.2 Abundance of amino acid repeats within sequences annotated as transcription factors

The overall percentage of proteins related to transcription factor activity (GO:0003700) was quite low in plants as well as in animals, except in *A. thaliana* for which 10% of its proteins were annotated as TFs (Table 3). The percentage of polyQ-containing sequences in plants and animals was also quite low. Only in *A. gambiae* an exceptional 9% of its proteins contained polyQ stretches. The percentage of proteins annotated as TFs and containing polyQ stretches ranged from 4% to 10% in plants and animals. However, an exception was again *A. gambiae* with 26% of its putative TFs containing polyQ stretches.
Very few proteins harbored polyA stretches, not even *A. gambiae* proteins. In contrast, of the proteins annotated as TFs between 2% and 35% contained polyA stretches. They were most commonly found in proteins of *O. sativa*, *S. bicolor*, *H. sapiens* and *A. gambiae*.
Except for in *A. gambiae* with 2%, polyN stretches were hardly found. Proteins annotated as TFs contained to a slightly higher percentage polyN stretches (~1%). *A. thaliana* and *A. gambiae* constituted the exceptions with 4% and 13%, respectively, of their proteins annotated as TFs and containing polyN stretches. Hence, the control amino acid asparagine

Table 3 Percentages of sequences annotated as transcription factors (TFs) and sequences containing polyQ, polyA or polyN stretches. Species are ordered according to taxonomy, animals are highlighted in gray.

Species	% of proteins annotated as TFs	% of proteins containing polyQ stretches	% of TFs containing polyQ stretches	% of proteins containing polyA stretches	% of TFs containing polyA stretches	% of proteins containing polyN stretches	% of TFs containing polyN stretches
A. thaliana	10.06	1.02	3.77	0.21	2.07	0.80	3.59
P. trichocarpa	2.10	0.89	4.26	0.06	2.65	0.32	1.27
O. sativa	0.88	1.49	7.48	0.31	35.24	0.43	0.39
S. bicolor	1.95	1.41	8.2	0.46	23.85	0.15	0.45
S. moellendorffii	0.97	0.75	4.79	0.07	7.35	0.11	<0.01
P. patens	1.15	1.81	9.52	0.10	8.52	0.10	<0.01
H. sapiens	4.14	1.00	4.20	0.59	14.22	0.04	<0.01
D. rerio	2.88	1.01	3.58	0.13	4.64	0.22	0.80
A. gambiae	2.15	8.72	26.47	0.49	22.79	1.74	12.87

was less prevalent in both, proteins annotated as TFs and in stretches, than glutamine and alanine.
Thus, even though only a low percentage of proteins contained polyQ or polyA stretches and only a low percentage of proteins were annotated as TFs, a high percentage of proteins annotated as TFs contained polyQ or polyA stretches in both plants and animals (Table 3).

3.3 PolyQ and polyA stretches significantly associated with transcription factors

To find out whether TFs are significantly overrepresented in plant proteins containing polyQ and polyA stretches, we carried out GO enrichment analyses. The significant results were assigned a rank according to their p-value. Thereby, the rank was lower when the p-value was lower.
In all plant species examined, protein sequences containing polyQ tracts were found to be significantly associated with the GO category GO:0003700 "sequence-specific DNA binding transcription factor activity" (Table 4 in appendix). The mean rank value equaled 4.13.
In animals, proteins annotated as having transcription factor activity were also significantly enriched in polyQ stretches. However, other transcription-related categories were found at even lower p-values except in *A. gambiae* where categories related to different binding activities were found at the lowest ranks. In comparison to plants, the category GO:0003700 was found at lower ranks in animals. However, other more specific GO categories related to transcription regulation like GO:0044212 "transcription regulatory region DNA binding" exhibited lower p-values in animals.
Protein sequences containing polyA stretches also showed a significant association with the GO category GO:0003700 in all species except in *P. trichocarpa*. Instead, the association of proteins annotated with GO categories related to catalytic activity was found on the first ranks in this species, while in the other plants and in animals proteins belonging to various

categories connected to binding activity were found on the first ranks. In humans, protein sequences annotated with more specific GO categories such as “transcription regulatory region DNA transcription activity” appeared at ranks shortly after the more general category GO:0003700. Thus, except for *P. trichocarpa*, plant and animal proteins belonging to GO categories related to transcription factor activity were enriched in polyA stretches.
Unlike the two other types of amino acid repeats, polyN-containing proteins were hardly associated with GO categories related to transcription factor activity. Only in *A. thaliana* and *P. trichocarpa* such an association was found. In the other species, either no significant association with any category was found or with proteins belonging to GO categories related to different kinds of binding and catalytic activities.

4 Discussion

PolyQ and polyA tracts in the conceptual proteomes of different plant and animal species were investigated and their association with TFs was analyzed to determine whether there are differences in the evolution of these repeats between plants and animals. As a control, we also studied the occurrence of polyN stretches.

4.1 No major difference in the abundance and length of polyQ and polyA stretches between plants and animals

On average, the length distributions of polyQ, polyA and polyN stretches varied between the different species but no major difference between plants and animals could be observed (Figure 1). The abundance of repeats generally decreased with increasing repeat lengths for all species. The length distributions were different for the different amino acids. PolyQ and polyA stretches were found far more often, with longer repeat lengths and a higher average repeat length than polyN stretches. These findings correspond well with the findings of Faux *et al.* [14]. who found more polyQ and polyA stretches than polyN stretches in proteins of *H. sapiens*, *D. rerio*, *A. thaliana*, and *O. sativa*. Siwach *et al.* [51] also found length distributions of many shorter and few longer amino acid repeats. Of the proteins they analyzed, 84% contained repeats consisting of 10 residues or less. At least in part, this may simply be due to the fact that the DNA sequences encoding long repeats have a higher likelihood of being split into shorter repeats by non-synonymous point mutations. As we have only analyzed pure repeats, those long impure repeats escaped our statistics. On the other hand, long polyQ and polyA stretches are probably also selected against, possibly because the containing proteins have a tendency of aggregation [10, 35].
Even though Q is encoded by only two codons and found less often than A, an amino acid encoded by four codons [31, 47], it occurs in quite long repeats and a high number of repeats in both, plants and animals. N, also encoded by two codons and approximately as frequent as Q in the proteomes of *A. thaliana* and *O. sativa* [31, 47], forms fewer and shorter repeats than the other two types of amino acids in both kingdoms. Furthermore, polyQ stretches were found to a higher percentage in proteins than polyA and polyN stretches. This indicates that there are differences in the rate at which the different repeats are elongated and contracted and/ or in the selection forces acting on the different repeats. There are two main mechanisms which are thought to contribute to the expansion of repeats, replication slippage and unequal recombination [50]. While replication slippage is supposed to be the major mechanism for the extension of polyQ stretches, polyA stretches are thought to be mainly extended by unequal recombination, at least in animals [9]. Selection has also been shown to contribute to the generation and extension of amino acid repeats in animals [36, 52] and in simulation studies

[46]. Analyses of the codons contributing to the different repeats will provide insights into the relative contributions of mutation and selection to the observed frequencies of polyQ, polyA and polyN stretches. Replication slippage and unequal recombination generally lead to amino acid repeats which are encoded by the same codon [57]. In contrast, if selection has contributed to the conservation of a polyQ or polyA stretch, one would expect that the amino acid repeat may be encoded by different synonymous codons rather than by the same codon [46].

A clear trend was that polyQ and polyA stretches are longer on average in animals than in plants (Table 2). Also the maximum number of residues in a polyQ stretch is markedly higher in animals than in plants. In contrast, a higher abundance of polyQ, polyA and polyN stretches is observed in plants (excluding the outlier *A. gambiae*). These differences may be caused by differences in the underlying mutation rates and/ or repair mechanisms or by differences in the selection patterns between plants and animals. Selection against long polyQ and polyA stretches may be stronger while a higher number of amino acid repeats is tolerated by or selected for in plants. It is possible that, instead of having long repeats, plant proteins may have a number of consecutive shorter repeats to fulfill the same function (if any). However, we did not study the distribution of the repeats within the proteins and additional studies are required to test this hypothesis.

Further studies, taking into account the underlying codons of the polyQ and polyA stretches and the distribution of these repeats within the proteins will permit a more thorough characterization of the relative contributions of the mutational mechanisms and selection regimes to the generation and maintenance of repeats. Moreover, studies including, for example, additional plant, animal, fungal and possibly also bacterial genomes will allow more general conclusions as to which of the observed abundances and length distributions of polyQ and polyA stretches are characteristic for certain species and which patterns are observed in a broader taxonomic range.

In other words, some differences in the number and length distributions of polyQ, polyA and polyN stretches were found but they currently cannot be extrapolated to major differences distinguishing plants and animals.

4.2 PolyQ and polyA stretches are enriched in transcription factors also in plants

The genomes of all analyzed species were published several years ago [7, 21, 24, 33, 38, 40, 42, 54, 56] and have been studied for some time now [37, 48, 53] resulting in a good quality of the sequence assembly and annotation. Many TFs belong to transcription factor families which each are defined by highly conserved DNA-binding domains (DBD). These DBD are used to predict TFs in newly sequenced species. Thus, many common TFs are reasonably predictable and are likely annotated even in newly released proteomes. In animals, more proteins annotated as TFs (GO:0003700) were found than in plants. This may be ascribed to the fact that the animal proteins are completely (*H. sapiens* and *D. rerio*) or to a high percentage annotated [16]. Annotation of proteins of *A. thaliana* is also nearly complete (99%). *A. thaliana* has previously been shown to have a higher amount of TFs than investigated animal and fungal species [44]. When the annotations of the other species become more elaborated, the number of TFs may change. Most annotations have been assigned automatically and may improve with manual curation. However, the quality of automatic annotations has been shown to be quite high for *H. sapiens* and several other model organisms [52]. Hence, despite possible differences in the progress of the assembly, the amount of TFs and the completeness of protein annotation in the different species, we do not assume that our findings are considerably biased.

PolyQ as well as polyA stretches are found more often in TFs than one would expect from their individual occurrence rates in both, plants and animals. An exceptionally high percentage of TFs of *O. sativa* and *S. bicolor* contain polyA stretches. The high GC content of grass genomes [11] and the fact that A is encoded by GC-rich codons (GCN) may contribute to this phenomenon.

Our GO enrichment analyses confirm that polyQ and polyA tracts are associated with proteins annotated as TFs (GO category GO:0003700). Hence, our findings corroborate that the correlation between polyQ and polyA stretches and TFs found in animals and fungi [20] also holds true for plants. The ranks of this category are a bit lower in plants than in animals. However, this probably does not indicate a major difference between plants and animals because the other associations at low ranks in animals are also categories involved in transcription activity. Thus, differences are rather species-specific than kingdom-specific.

Unlike polyQ and polyA, polyN, used as a control here, was not found to be associated with TFs in several plant and animal species. Thus, the overrepresentation of polyQ and polyA hints at a function of these stretches in TFs. Selection may often favor polyQ and polyA stretches, at least up to a certain length, whereas polyN stretches may be neutral or even deleterious except for *A. thaliana* and *P. trichocarpa*. In these two species, polyN-rich regions, just like polyQ-rich regions, may have a role in mediating protein-protein interactions [39]. For vertebrates, selection increasing the retention of amino-acid repeats including polyQ and polyA has been found recently [36]. It has also been shown before that polyQ tracts enhance transcriptional activation in animals in a length dependent manner [20]. This result was recently extended to fungi [6], indicating a role for polyQ tracts in the modulation of transcription in opisthokonts. Our findings now make it appear likely that polyQ stretches also have such a function in plants. PolyA stretches have been hypothesized to repress transcriptional activity in animals [17]. The overrepresentation of polyA stretches also in plant TFs again hints at a conservation of this function between plants and opisthokonts. Further support for the role of these repeats in transactivation could be gained from an analysis of the position of the repeats within the corresponding proteins. The repeats would be expected to occur outside of the DNA-binding domain where transcriptional activity can mainly be modulated. In our analyses, we observed some species-specific differences in the abundance and length distributions of polyQ and polyA stretches. However, there seem to be no major differences in the evolution of polyQ and polyA stretches between plants and opisthokonts. Hence, the mutational mechanisms for the generation, expansion and contraction as well as the selection pressure on polyQ and polyA stretches seem to be similar in respective species, or differences in mutation and selection compensate each other. Furthermore, the association between TFs and polyQ and polyA stretches was also found for the plant species examined. These stretches may have similar roles in plants and opisthokonts. Hence, we provide data suggesting that polyQ and polyA tracts act as "evolutionary tuning knobs" [28, 27] not only for opisthokonts but also for land plants.

Acknowledgements We would like to thank all members of the Theißen lab for their kind support during the preparation of this publication.

References

1 M. M. Albà and R. Guigo. Comparative analysis of amino acid repeats in rodents and humans. *Genome Res.*, 14(4):549–54, 2004.

2 M. M. Albà, M. F. Santibáñez-Koref, and J. M. Hancock. Amino acid reiterations in yeast are overrepresented in particular classes of proteins and show evidence of a slippage-like mutational process. *J Mol Evol*, 49(6):789–797, 1999.

3 A. Alexa and J. Rahnenfuhrer. topGO: Enrichment analysis for Gene Ontology. *R package version 2.4.0*, 2010.

4 K. Anan, N. Yoshida, Y. Kataoka, M. Sato, H. Ichise, M. Nasu, and S. Ueda. Morphological change caused by loss of the taxon-specific polyalanine tract in Hoxd-13. *Mol Biol Evol*, 24(1):281–287, 2007.

5 M. Ashburner et al. Gene Ontology: tool for the unification of biology. *Nat Genet.*, 25(1):25–29, 2000.

6 L. Atanesyan, V. Gunther, B. Dichtl, O. Georgiev, and W. Schaffner. Polyglutamine tracts as modulators of transcriptional activation from yeast to mammals. *Biol Chem.*, 393(1-2):63–70, 2012.

7 J. A. Banks et al. The *Selaginella* genome identifies genetic changes associated with the evolution of vascular plants. *Science*, 332(6032):960–3, 2011.

8 Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(1):289–300, 1995.

9 L. Y. Brown and S. A. Brown. Alanine tracts: the expanding story of human illness and trinucleotide repeats. *Trends Genet.*, 20(1):51–8, 2004.

10 S. Caburet, A. Demarez, L. Moumné, M. Fellous, E. De Baere, and R. A. Veitia. A recurrent polyalanine expansion in the transcription factor FOXL2 induces extensive nuclear and cytoplasmic protein aggregation. *J Med Genet*, 41(12):932–936, 2004.

11 N. Carels and G. Bernardi. The compositional organization and the expression of the *Arabidopsis* genome. *FEBS Lett.*, 472(2-3):302–6, 2000.

12 D. Chernikova, S. Motamedi, M. Csürös, E. V. Koonin, and I. B. Rogozin. A late origin of the extant eukaryotic diversity: divergence time estimates using rare genomic changes. *Biol Direct*, 6:26, 2011.

13 D. P. Depledge and A. R. Dalby. COPASAAR–a database for proteomic analysis of single amino acid repeats. *BMC Bioinformatics*, 6:196, 2005.

14 N. G. Faux, S. P. Bottomley, A. M. Lesk, J. A. Irving, J. R. Morrison, M. G. de la Banda, and J. C. Whisstock. Functional insights from the distribution and role of homopeptide repeat-containing proteins. *Genome Res*, 15(4):537–551, 2005.

15 R. A. Fisher. On the interpretation of χ^2 from contingency tables, and the calculation of p. *Journal of the Royal Statistical Society.*, 85(1):87–94, 1922.

16 P. Flicek. Ensembl 2012. *Nucleic Acids Res.*, 40(D1):D84–D90, 2012.

17 J. W. Fondon and H. R. Garner. Molecular origins of rapid and continuous morphological evolution. *Proc Natl Acad Sci U S A*, 101(52):18058–18063, 2004.

18 A. C. Froehlich, Y. Liu, J. J. Loros, and J. C. Dunlap. White Collar-1, a circadian blue light photoreceptor, binding to the frequency promoter. *Science*, 297(5582):815–819, 2002.

19 R. Galant and S. B. Carroll. Evolution of a transcriptional repression domain in an insect Hox protein. *Nature*, 415(6874):910–913, 2002.

20 H. P. Gerber, K. Seipel, O. Georgiev, M. Höfferer, M. Hug, S. Rusconi, and W. Schaffner. Transcriptional activation modulated by homopolymeric glutamine and proline stretches. *Science*, 263(5148):808–811, 1994.

21 R. A. Holt et al. The genome sequence of the malaria mosquito *Anopheles gambiae*. *Science*, 298(5591):129–49, 2002.

22 M. Huntley and G. B. Golding. Evolution of simple sequence in proteins. *J Mol Evol*, 51(2):131–140, 2000.

23 M. A. Huntley and A. G. Clark. Evolutionary analysis of amino acid repeats across the genomes of 12 *Drosophila* species. *Mol Biol Evol*, 24(12):2598–2609, 2007.

24 The Arabidopsis Genome Initiative. Analysis of the genome sequence of the flowering plant *Arabidopsis thaliana*. *Nature*, 408(6814):796–815, 2000.

25 M. K. Kalita, G. Ramasamy, S. Duraisamy, V. S. Chauhan, and D. Gupta. Protrepeatsdb: a database of amino acid repeats in genomes. *BMC Bioinformatics*, 7:336, 2006.

26 S. Karlin, L. Brocchieri, A. Bergman, J. Mrazek, and A. J. Gentles. Amino acid runs in eukaryotic proteomes and disease associations. *Proc Natl Acad Sci U S A*, 99(1):333–8, 2002.

27 Y. Kashi, D. King, and M. Soller. Simple sequence repeats as a source of quantitative genetic variation. *Trends Genet*, 13(2):74–78, 1997.

28 Yechezkel Kashi and David G King. Simple sequence repeats as advantageous mutators in evolution. *Trends Genet*, 22(5):253–259, 2006.

29 M. V. Katti, R. Sami-Subbu, P. K. Ranjekar, and V. S. Gupta. Amino acid repeat patterns in protein sequences: their diversity and structural-functional implications. *Protein Sci*, 9(6):1203–1209, 2000.

30 R.J. Kinsella. Ensembl BioMarts: a hub for data retrieval across taxonomic space. *Database (Oxford)*, 2011, 2011.

31 N. Kottenhagen. *Frequency and distribution of amino acid repeats in the proteomes of Arabidopsis thaliana and Oryza sativa*. Diploma thesis, Friedrich-Schiller-Universität Jena, 2009.

32 S. Kruglyak, R. T. Durrett, M. D. Schug, and C. F. Aquadro. Equilibrium distributions of microsatellite repeat length resulting from a balance between slippage events and point mutations. *Proc Natl Acad Sci U S A*, 95(18):10774–10778, 1998.

33 E. S. Lander et al. Initial sequencing and analysis of the human genome. *Nature*, 409(6822):860–921, 2001.

34 H. Luo, K. Lin, A. David, H. Nijveen, and J. A. Leunissen. ProRepeat: an integrated repository for studying amino acid tandem repeats in proteins. *Nucleic Acids Res.*, 40(Database issue):D394–9, 2012.

35 J. F. Morley, H. R. Brignull, J. J. Weyers, and R. I. Morimoto. The threshold for polyglutamine-expansion protein aggregation and cellular toxicity is dynamic and influenced by aging in *Caenorhabditis elegans*. *Proc Natl Acad Sci U S A*, 99(16):10417–10422, 2002.

36 L. Mularoni, A. Ledda, M. Toll-Riera, and M. M. Albà. Natural selection drives the accumulation of amino acid tandem repeats in human proteins. *Genome Res.*, 20(6):745–54, 2010.

37 S. Ouyang et al. The TIGR Rice Genome Annotation Resource: improvements and new features. *Nucleic Acids Res.*, 35(Database issue):D883–7, 2007.

38 A. H. Paterson et al. The *Sorghum bicolor* genome and the diversification of grasses. *Nature*, 457(7229):551–6, 2009.

39 M. F. Perutz, B. J. Pope, D. Owen, E. E. Wanker, and E. Scherzinger. Aggregation of proteins with expanded glutamine and alanine repeats of the glutamine-rich and asparagine-rich domains of Sup35 and of the amyloid beta-peptide of amyloid plaques. *Proc Natl Acad Sci U S A.*, 99(8):5596–600, 2002.

40 International Rice Genome Sequencing Project. The map-based sequence of the rice genome. *Nature*, 436(7052):793–800, 2005.

41 A. Reiner, I. Dragatsis, and P. Dietrich. Genetics and neuropathology of Huntington's disease. *Int Rev Neurobiol*, 98:325–372, 2011.

42 S. A. Rensing et al. The *Physcomitrella* genome reveals evolutionary insights into the conquest of land by plants. *Science*, 319(5859):64–9, 2008.

43 G. F. Richard and F. Paques. Mini- and microsatellite expansions: the recombination connection. *EMBO Rep.*, 1(2):122–6, 2000.

44 J. L. Riechmann, J. Heard, G. Martin, L. Reuber, C. Jiang, J. Keddie, L. Adam, O. Pineda, O. J. Ratcliffe, R. R. Samaha, R. Creelman, M. Pilgrim, P. Broun, J. Z. Zhang, D. Ghandehari, B. K. Sherman, and G. Yu. *Arabidopsis* transcription factors: genome-wide comparative analysis among eukaryotes. *Science*, 290(5499):2105–10, 2000.

45 M. Ronshaugen, N. McGinnis, and W. McGinnis. Hox protein mutation and macroevolution of the insect body plan. *Nature*, 415(6874):914–917, 2002.

46 M. M. Rorick and G. P. Wagner. The origin of conserved protein domains and amino acid repeats via adaptive competition for control over amino acid residues. *J Mol Evol.*, 70(1):29–43, 2010.

47 P. Seeber. *Frequency and distribution of amino acid repeats in transcription factors.* Diploma thesis, Friedrich-Schiller-Universität Jena, 2010.

48 M. V. Sharakhova, M. P. Hammond, N. F. Lobo, J. Krzywinski, M. F. Unger, M. E. Hillenmeyer, R. V. Bruggner, E. Birney, and F. H. Collins. Update of the *Anopheles gambiae* PEST genome assembly. *Genome Biol.*, 8(1):R5, 2007.

49 N. Sharopova. Plant simple sequence repeats: distribution, variation, and effects on gene expression. *Genome*, 51(2):79–90, 2008.

50 R. R. Sinden, V. N. Potman, E. A. Oussatcheva, C. E . Pearson, Y. L. Lyubchenko, and L. S. Shlyakhtenko. Triplet repeat DNA structures and human genetic disease: dynamic mutations from dynamic DNA. *J Biosci.*, 27(1):53–65, 2002.

51 P. Siwach, S. D. Pophaly, and S. Ganesh. Genomic and evolutionary insights into genes encoding proteins with single amino acid repeats. *Mol Biol Evol*, 23(7):1357–1369, 2006.

52 N. Skunca, A. Altenhoff, and C. Dessimoz. Quality of computationally inferred gene ontology annotations. *PLoS Comput Biol.*, 8(5), May 2012.

53 D. Swarbreck et al. The Arabidopsis Information Resource (TAIR): gene structure and function annotation. *Nucleic Acids Res.*, 36(Database issue):D1009–14, 2008.

54 The *Danio rerio* Sequencing Project. (http://www.sanger.ac.uk/projects/d_rerio/).

55 Y. Trottier, Y. Lutz, G. Stevanin, G. Imbert, D. Devys, G. Cancel, F. Saudou, C. Weber, G. David, and L. Tora. Polyglutamine expansion as a pathological epitope in Huntington's disease and four dominant cerebellar ataxias. *Nature*, 378(6555):403–406, 1995.

56 G. A. Tuskan et al. The genome of black cottonwood, *Populus trichocarpa* (Tor. & Gray). *Science*, 313(5793):1596–604, 2006.

57 E. Viguera, D. Canceill, and S. D. Ehrlich. Replication slippage involves DNA polymerase pausing and dissociation. *EMBO J.*, 20(10):2587–95, 2001.

58 Hwan Su Yoon, Jessica Grant, Yonas Tekle, Min Wu, Benjamin Chaon, Jeffrey Cole, John Logsdon, David Patterson, Debashish Bhattacharya, and Laura Katz. Broadly sampled multigene trees of eukaryotes. *BMC Evolutionary Biology*, 8(1):14, 2008.

59 Y. Zhou, J. Liu, L. Han, Z. G. Li, and Z. Zhang. Comprehensive analysis of tandem amino acid repeats from ten angiosperm genomes. *BMC Genomics.*, 12:632, 2011.

A Appendix

Table 4 GO enrichment analysis. Shown are the three significant results with the lowest p-values of the analysis for polyA, polyN, and polyQ tracts for all species. If the GO category related to transcription factor activity (GO:0003700) did not belong to the first three entries, this category with its corresponding rank is shown in addition, if found significant. Species are in taxonomic order, animals are highlighted in gray. T – transcription, A – activity, TFA – transcription factor activity, if no significant results were found, this is indicated by "–".

Repeat	Species	Rank	GO-ID	GO category	P-value
polyQ	*A. thaliana*	1	GO:0001071	nucleic acid binding TFA	1E-28
		2	GO:0003700	sequence-specific DNA binding TFA	1E-28
		3	GO:0003677	DNA binding	1.8E-25
	P. trichocarpa	1	GO:0003676	nucleic acid binding	1E-28
		2	GO:0003677	DNA binding	1E-28
		3	GO:0001071	nucleic acid binding TFA	7.8E-11
		4	GO:0003700	sequence-specific DNA binding TFA	7.8E-11
	O. sativa	1	GO:0003677	DNA binding	2.5E-28
		2	GO:0003676	nucleic acid binding	2.5E-24
		3	GO:0001071	nucleic acid binding TFA	8.4E-12
		4	GO:0003700	sequence-specific DNA binding TFA	8.4E-12
	S. bicolor	1	GO:0003677	DNA binding	1E-28
		2	GO:0003676	nucleic acid binding	1E-28
		3	GO:0001071	nucleic acid binding TFA	2.3E-21
		4	GO:0003700	sequence-specific DNA binding TFA	2.3E-21
	S. moellendorffii	1	GO:0003677	DNA binding	1E-28
		2	GO:0003676	nucleic acid binding	1.3E-26
		3	GO:0005515	protein binding	3.7E-20
		5	GO:0003700	sequence-specific DNA binding TFA	3.4E-16
	P. patens	1	GO:0003676	nucleic acid binding	1.7E-14
		2	GO:0046983	protein dimerization A	2.6E-10
		3	GO:0003677	DNA binding	8.7E-09
		6	GO:0003700	sequence-specific DNA binding TFA	6.1E-06
	H. sapiens	1	GO:0003676	nucleic acid binding	1.2E-14
		2	GO:0003677	DNA binding	1.3E-14
		3	GO:0044212	T regulatory region DNA binding	6.7E-12
		9	GO:0003700	sequence-specific DNA binding TFA	2.1E-11
	D. rerio	1	GO:0003676	nucleic acid binding	3,00E-19
		2	GO:0003677	DNA binding	6.3E-17
		3	GO:0003712	transcription cofactor A	3.4E-07
		6	GO:0003700	sequence-specific DNA binding TFA	1.3E-06
	A. gambiae	1	GO:0008270	zinc ion binding	7.1E-26
		2	GO:0005488	binding	7.6E-24
		3	GO:0005515	protein binding	1.1E-23
		8	GO:0003700	sequence-specific DNA binding TFA	8.1E-16
polyA	*A. thaliana*	1	GO:0004124	cysteine synthase A	3.6E-05
		2	GO:0001071	nucleic acid binding TFA	6.7E-05
		3	GO:0003700	sequence-specific DNA binding TFA	6.7E-05
	P. trichocarpa	1	GO:0004124	cysteine synthase A	9.00E-09
		2	GO:0004970	ionotropic glutamate receptor A	1.3E-06
		3	GO:0005231	excitatory extracellular ligand-	1.3E-06
				gated ion channel A	1.3E-06
	O. sativa	1	GO:0050824	water binding	1E-28
		2	GO:0050825	ice binding	1E-28
		3	GO:0001071	nucleic acid binding TFA	1E-28
		4	GO:0003700	sequence-specific DNA binding TFA	1E-28
	S. bicolor	1	GO:0001071	nucleic acid binding TFA	7.1E-21
		2	GO:0003700	sequence-specific DNA binding TFA	7.1E-21
		3	GO:0003677	DNA binding	2.9E-19
	S. moellendorffii	1	GO:0003676	nucleic acid binding	7.5E-13
		2	GO:0003677	DNA binding	3.2E-09
		3	GO:0005488	binding	1.6E-07
		6	GO:0003700	sequence-specific DNA binding TFA	3.2E-06

continued on next page

	P. patens	1	GO:0001071	nucleic acid binding TFA	5.2 E-06
		2	GO:0003700	sequence-specific DNA binding TFA	5.2 E-06
		3	GO:0003844	1,4-alpha-glucan branching enzyme A	7.8E-04
	H. sapiens	1	GO:0003676	nucleic acid binding	1E-28
		2	GO:0043565	sequence-specific DNA binding	1E-28
		3	GO:0003677	DNA binding	1E-28
		5	GO:0003700	sequence-specific DNA binding TFA	1E-28
	D. rerio	1	GO:0003676	nucleic acid binding	1.5E-23
		2	GO:0003677	DNA binding	6.1E-09
		3	GO:0003700	sequence-specific DNA binding TFA	1.8E-08
	A. gambiae	1	GO:0003676	nucleic acid binding	1E-28
		2	GO:0003677	DNA binding	3.4E-26
		3	GO:0005488	binding	6.5E-26
		7	GO:0003700	sequence-specific DNA binding TFA	4.9E-16
polyN	*A. thaliana*	1	GO:0001071	nucleic acid binding TFA	1E-28
		2	GO:0003700	sequence-specific DNA binding TFA	1E-28
		3	GO:0003677	DNA binding	4.1E-19
	P. trichocarpa	1	GO:0003676	nucleic acid binding	1.7E-07
		2	GO:0043565	sequence-specific DNA binding	1.4E-07
		3	GO:0003677	DNA binding	5.6E-04
		7	GO:0003700	sequence-specific DNA binding TFA	1.0E-02
	O. sativa	—	—	—	—
	S. bicolor	—	—	—	—
	S. moellendorffii	1	GO:0015018	galactosylgalactosylxylosylprotein 3-beta-glucuronosyltransferase A	5.7E-04
		2	GO:0015020	glucuronosyltransferase A	1.7E-03
	P. patens	—	—	—	—
	H. sapiens	—	—	—	—
	D. rerio	1	GO:0003676	nucleic acid binding	1.6E-02
	A. gambiae	1	GO:0003677	DNA binding	1.6E-18
		2	GO:0001071	nucleic acid binding TFA	5,00E-17

Computation and Visualization of Protein Topology Graphs Including Ligand Information

Tim Schäfer[1], Patrick May[2], and Ina Koch[1]

1 **Institute of Computer Science, Department of Molecular Bioinformatics, Johann Wolfgang Goethe-University Frankfurt (Main), Robert-Mayer-Straße 11–15, 60325 Frankfurt (Main), Germany,**
`ina.koch@bioinformatik.uni-frankfurt.de`

2 **Luxembourg Centre for Systems Biomedicine, University of Luxembourg, Campus Belval, 7 Avenue des Hauts-Fourneaux, L–4362 Esch-sur-Alzette, Luxembourg**

Abstract

Motivation: Ligand information is of great interest to understand protein function. Protein structure topology can be modeled as a graph with secondary structure elements as vertices and spatial contacts between them as edges. Meaningful representations of such graphs in 2D are required for the visual inspection, comparison and analysis of protein folds, but their automatic visualization is still challenging. We present an approach which solves this task, supports different graph types and can optionally include ligand contacts.
Results: Our method extends the field of protein structure description and visualization by including ligand information. It generates a mathematically unique representation and high-quality 2D plots of the secondary structure of a protein based on a protein-ligand graph. This graph is computed from 3D atom coordinates in PDB files and the corresponding SSE assignments of the DSSP algorithm. The related software supports different notations and allows a rapid visualization of protein structures. It can also export graphs in various standard file formats so they can be used with other software. Our approach visualizes ligands in relationship to protein structure topology and thus represents a useful tool for exploring protein structures.
Availability: The software is released under an open source license and available at `http://www.bioinformatik.uni-frankfurt.de/` in the *Software* section under *Visualization of Protein Ligand Graphs.*

1998 ACM Subject Classification J.3 Life and Medical Sciences

Keywords and phrases protein structure, graph theory, ligand, secondary structure, protein ligang graph

Digital Object Identifier 10.4230/OASIcs.GCB.2012.108

1 Introduction

Our knowledge of proteins expands rapidly with the high-throughput technologies and more and more 3D structures are available in databases like the RCSB Protein Data Bank (PDB, [1]). Thus, computational tools to automatically search, compare and classify protein structures are needed. Proteins are complex macromolecules and representing their structure in a way which allows for fast visual analysis requires significant simplification. This can be achieved by choosing the secondary structure level as an abstraction level and then drawing a 2D cartoon image of the protein. After the pioneering work of Jane Richardson, who first defined protein topology cartoons [23] and the first visualization methods, e.g. based on hydrogen bonds [10], it is not surprising that many secondary structure databases and software tools to perform these tasks on all levels of protein structure exist today. The databases CATH [22], SCOP [21] and TOPS [19] all operate on the secondary structure level, i.e., abstracting from the atomic level they consider secondary structure elements (SSEs) and their relations in proteins to describe protein topology. Both CATH and SCOP split proteins into structural domains and then use a combination of automated and manual techniques to classify them into a hierarchic system that reflects structural as well as evolutionary relationships. In contrast, GRAST [7] and TOPS are fully automated methods. They all use graphs to represent protein topologies. GRATH abstracts SSEs as vectors and uses geometric relationships between all pairs of vectors to construct a graph of the protein structure. VAST [6, 16] also uses graph algorithms to detect similar spatial orientation and connectivity between SSEs.

Protein structure databases usually compare protein topologies based on atom coordinates from PDB files, but all methods listed above ignore the ligand atoms stored in those files. TOPS+ [28, 29] is the first method which includes ligand information in the comparison of proteins on the secondary structure level. It works on the domain level, uses a string-based description of protein secondary structure and includes additional biochemical and structural features like length of SSEs. String alignment methods are used to find similarities between domains.

The Protein Topology Graph Library (PTGL, [17, 18]) is a database of protein topologies that provides a web interface to compare and visualize protein folds based on SSEs. It uses a graph-based protein model related to earlier work by Koch *et al.* [12, 15, 14] and finds structural similarities between proteins by detecting maximum common substructures in their graph representations, allowing abstraction from the order of the SSEs in the amino acid sequence. This approach has recently been used to explore super-secondary structure patterns in proteins [13].

Images of protein topologies should contain information on the SSEs, their contacts and relative orientations and should still be clear and unequivocal. Automatically arranging the SSE symbols in the plane and fulfilling these constraints is not trivial, but some programs which address this task exist. TOPS cartoons [30] use circles and triangles to represent helices and strands. Pro-Origami [25] also draws protein cartoons and finds a balance between structural information and a clear layout. Additionally, it provides interactive diagrams and a search interface via a website. With the exception of TOPS+, all these programs consider α-helices and β-strands, but ignore ligands.

2 Approach

Our method works on the level of the secondary structure for protein chains. The 3D atom coordinates of all protein residues of a chain as well as the ligand molecules associated with the chain are parsed from PDB files. Each amino acid of the protein chain is assigned a SSE type by the DSSP algorithm [11]. After a pre-processing step which includes filtering of certain residues, the contacts between all remaining residues and ligand molecules are calculated. These residue level contacts are analyzed and SSE level contacts are defined according to specific rules. For each SSE pair which is in contact, we compute the spatial orientation. This leads to a contact matrix which can be used to create a *protein graph* (see Figure 1). These protein graphs are not necessarily connected and can thus be split into one or more connected components, which we call *folding graphs*. Both protein graphs and folding graphs can be visualized as 2D cartoons which allow for a rapid visual inspection and comparison.

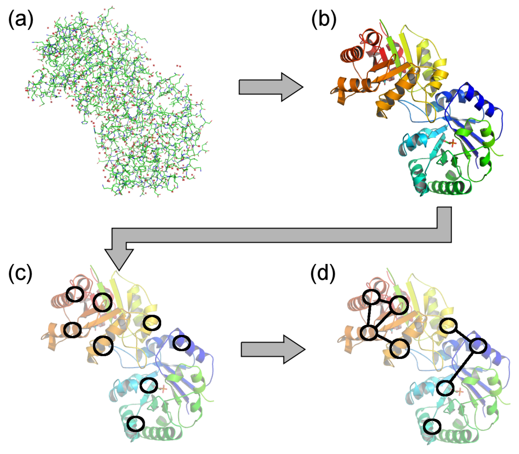


Figure 1 Computation of the protein graph from 3D atom data. Contacts are calculated on atom level from the 3D data in a PDB file (a). All residues of the considered protein chain are assigned to SSEs (b), which become the vertices of the protein graph (c). The atom contact information is used to determine the spatial relationships between the SSEs represented by edges in the graph (d).

3 Methods

3.1 Preprocessing and secondary structure assignment

Meta data, data on protein residues and atom coordinates is first parsed from the PDB file. During this step, PDB files which contain only very short protein chains with less than 30 residues or no protein chains at all are ignored. A list of all helices and strands of a protein chain in sequential order is created from the DSSP output according to the rules described below. Then all protein residues and ligands are assigned to one of the classes listed in Table 1.

Table 1 Description of the SSE classes used in protein ligand graphs.

SSE class	DSSP types	Description
H	H, G, I	residue which is part of an α-helix
E	E, B	residue which is part of a β-strand
C	S, T, none	residue which is not part of an α-helix or β-strand
L	n/a	ligand molecule
O	n/a	other non-protein molecules (solvent, polymers)

Solvent molecules like H_2O as well as polymers like DNA and RNA are filtered at this stage. Protein residues which are not part of a helix or strand are also ignored by default. This leads to a list of SSEs which is ordered by appearance in the primary structure from the N- to the C-terminus. An example for the residues with DSSP numbers 212–247 of chain A of PDB ID 7TIM is given below.

```
DSSP pos. | 212     220       230       240    247
Residue   | NGSNAVTFKDKADVDGFLVGGASLKPEFVDIINSRN
DSSP SSE  |  TTTGGGGTT TT  EEEESGGGGSTTHHHHHHTT
SSE class | CCCCHHHHCCCCCCCEEEECHHHHCCCHHHHHHCCC
```

Consecutive residues of the classes E or H form an SSE, which means that the example above contains four SSEs: a β-strand (227–230) and three α-helices (216–219, 232–235, 239–244). Each ligand is treated as a single SSE of type L and appended to the SSE list at the end of the chain. The contacts between the residues and ligands of the classes H, E and L are computed in the following step.

3.2 Computation of atom level and residue level contacts

A hard-sphere model is used to determine atom contacts: atoms are treated as hard spheres, and the atom coordinates defined in a PDB file are assigned to *collision spheres*. The default collision sphere radius r_{atom} is 2 Å for protein atoms and 3 Å for ligand atoms. Depending on the experiment which was used to generate the data, a PDB file may or may not contain data on hydrogen atoms, so we ignore them if they are listed. By definition, a contact between two atoms exists if their collision spheres overlap and a contact between two residues exists if the collision spheres of any of their atoms overlap.

According to its type, each atom can be assigned to one of the following classes: protein backbone atom, protein side chain atom or ligand atom. Thus, a contact between a pair of residues can be the result of multiple contacts of different types on the atom level. We distinguish the following contact types:

- backbone–backbone (BB) contacts
- backbone–side chain (BC) contacts
- side chain–side chain (CC) contacts
- ligand–backbone (LB) contacts
- ligand–side chain (LC) contacts
- ligand–ligand (LL) contacts
- ligand–non-ligand (LX) contacts, i.e., LX = LB || LC

For each residue pair which is in contact, the number of contacts of all these classes is saved for SSE contact determination.

To speed up the calculation, collision spheres are also assigned to residues by determining the central atom c and the maximal distance d_{max} between c and any other atom of the same residue. For protein residues, the C_{α}-atom is considered to be the residue center. This rule cannot be applied to ligand molecules though, so we choose the atom with minimal maximal distance to all other atoms of the molecule in this case. Thus, the collision sphere for a residue has the radius $r_{res} = d_{max} + r_{atom}$. If the collision spheres of two residues do not overlap, no contact is possible and the atom level comparison can be omitted.

Furthermore, multiple consecutive residues may be skipped entirely under certain circumstances (see Figure 2): if two residues with DSSP numbers n and m are very far from each other, the residues n and $m+1$ are far from each other as well, and it may be possible to skip the residues $m+1, m+2, \ldots, m+k$. The maximum distance between the central atoms of consecutive residues in the chain, $dist_{max}$, the maximum collision sphere radius of all residues, and the distance $dist_{n,m}$ between the central atoms of the residues n and m are required to determine k, the number of residues that can be skipped. Integer division[1] is used to compute $k = dist_{n,m} \backslash dist_{max}$ and skip residues during the pairwise contact calculation.

[1] Integer division can be defined as $a \backslash b \equiv \lfloor a/b \rfloor$.

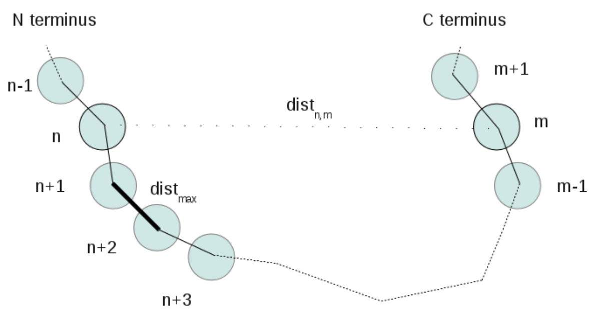


Figure 2 Residue skipping during the determination of residue level contacts. The maximum distance between the central atoms of consecutive residues in the chain is $dist_{max}$, and $dist_{n,m}$ is the distance between the central atoms of the residues n and m. Then, $dist_{n,m+1}$ is at least $dist_{n,m} - dist_{max}$.

3.3 Computation of SSE level contacts and relative orientation

We apply a rule set to determine whether two SSEs are in contact. This is the case if enough atom level contacts exist between residues of the considered pair. Determining the number of atom level contacts that is considered to be sufficient for an SSE contact is a difficult and time-consuming task though, because different settings have to be tested on a number of proteins and their results have to be verified manually by visual inspection using a viewer program like PyMol [24]. The rules depend on the type of both SSEs, see Table 2.

Table 2 Rules for the determination of SSE contacts. The class X represents an arbitrary SSE class, i.e., X = E || H || L.

SSE 1	SSE 2	Required contacts
E	E	$BB > 1 \,\|\|\, CC > 2$
H	E	$(BB > 1 \,\&\&\, BC > 3) \,\|\|\, CC > 3$
H	H	$BC > 3 \,\|\|\, CC > 3$
L	X	$LX >= 1$

Since a polypeptide chain has a direction, multiple spatial relationships can be defined between two adjacent sections: they can be *parallel*, *anti-parallel* or something in between, which we call *mixed* orientation. Each edge in the protein graph is labeled with the spatial relation of the SSEs represented by its vertices, which is computed as follows:

Let u and v be two spatially adjacent, non-ligand SSEs of a protein chain. Let A and B be the sets of the DSSP residue numbers of the SSEs u and v respectively. Let S be the set of all sums of the DSSP residue numbers of the residue pairs (a_i, b_j) with $a_i \in A$ and $b_j \in B$ which form a contact and D the set of all differences of these pairs. The double difference is then defined as $DD = (S_{max} - S_{min}) - (D_{max} - D_{min})$. The spatial relation between the SSEs u and v is determined from DD using four thresholds for which $T_{antip} < T_{mixedLower} < 0 < T_{mixedUpper} < T_{parallel}$ holds. By definition, the SSEs are (1) mixed, if $T_{mixedLower} < DD < T_{mixedUpper}$, (2) parallel, if $DD > T_{parallel}$ and (3) anti-parallel, if $DD < T_{antip}$. If one of them is a ligand and thus has no direction, the contact type is defined as (4) *ligand*.

During the computation of contacts between the n SSEs of a protein chain and the relative orientation of all SSE pairs which are in contact, an orientation matrix M of size n^2

is generated. At position (i, j), the matrix is 0 if the SSEs i and j are not in contact, 1 if they are anti-parallel, 2 if they are mixed, 3 if they are parallel and 4 if one of them is a ligand.

3.4 Computation of the different protein graph types

The orientation matrix M can be interpreted as the adjacency matrix of an undirected protein graph $G = (V, E)$ with the vertex set V and the edge set E. Each vertex v of the graph represents an SSE and each edge $e = (v_i, v_j)$ a contact between the vertices v_i and v_j it connects. An edge $e = (v_i, v_j)$ is added to E if and only if $M_{i,j} > 0$.

Depending on what the user is interested in, a subset of the SSE types can be ignored when creating the graph. For example, a protein graph which only consists of helices and the contacts between them is called an *alpha-graph*. We support the following graph types:

- the **alpha-graph** based on the α-helices (H),
- the **beta-graph** consisting of all β-strands (E),
- the **albe-graph** with all SSEs of types E and H,
- the **alphalig-graph** consisting of all SSEs of types H and L (ligands),
- the **betalig-graph** with all SSEs of types E and L,
- the **albelig-graph** based on all β-strands, helices, and ligands.

Note that two consecutive SSEs in the SSE list may be separated by a variable region in the amino acid sequence and thus an SSE is not necessarily spatially adjacent to its primary structure predecessor and successor in the protein graph. This means that protein graphs are not necessarily connected in the graph-theoretic sense, i.e., protein graphs may consist of several connected components. We call these connected components the *folding graphs* of a protein graph. They can be computed using the algorithm described in [9] and drawn separately. Optionally, folding graphs under a certain size can be ignored during this process. It turned out that many protein graphs consist of one large connected component and several isolated vertices which often represent helices on the protein surface.

3.5 Visualization of protein graphs and folding graphs

Both protein and folding graphs can be visualized in the same way. The vertices are ordered and labeled by the position of the SSE they represent in the amino acid sequence (S) and by their position in the graph (G). Helices are shown as red, filled circles, strands are represented by black, filled squares, and ligands by magenta rings. The contacts are drawn as arcs, and their color encodes the spatial relation between the two SSEs (red $\rightarrow$ parallel; blue $\rightarrow$ anti-parallel; green $\rightarrow$ mixed; magenta $\rightarrow$ ligand contact). The protein graph of triosephosphate isomerase is depicted in Figure 3 as an example.
To visualize a graph $G = (V, E)$ consisting of the n vertices $v_0, v_1, \ldots, v_{n-1}$, the visualization function first computes the required image dimensions from the number of vertices. Ignoring the page margins of fixed size, the drawing area can be split into three parts vertically:

1. The **header section** is used to print information on the depicted graph, including the graph type, PDB ID and chain ID. Its width w_h is determined from the font size and the number of characters of the printed string. The header section is not shown in the figures of this paper.

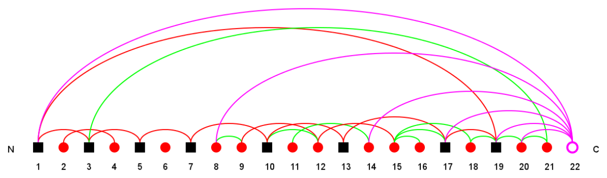


Figure 3 The albelig-graph of the β-chain of triosephosphate isomerase. The α-helices are shown as red, filled circles and the β-strands as black, filled squares. Ligands are represented by magenta circles. From left to right, the vertices are labeled by their position in the amino acid sequence. The arcs mark spatial contacts (red for parallel; blue for anti-parallel; green for mixed; magenta for ligand contact).

2. The **graph section** contains the visualization of the protein graph. Its width w_g is $|V| * dist_v + 2 * r$, where $dist_v$ is the distance of adjacent vertices in the visualization ($dist_v$ is fixed and set to 50 pixels by default) and r is the vertex radius. Its height h_g is $h_{max} + r$, where h_{max} is the height of the largest arc that could possibly occur in the image, the one between vertices v_0 and v_{n-1}.
3. The **footer section** is used to label the vertices from N- to C-terminus. The labels are written directly underneath the vertices, so the width of the footer also is w_g.

The total width of the image thus is $margin_{left} + max(w_h, w_g) + margin_{right}$ and its height is $margin_{top} + h_h + h_g + h_f + margin_{bottom}$.

Once the canvas has been prepared, the header and footer are printed and the graph is depicted. Before drawing a vertex or edge, the color is set based on the vertex or edge type. The position (v_x^1, v_y^1) of the first vertex in the list is determined and all other vertices are drawn relative to it, i.e., the n^{th} vertex is drawn at position $(v_x^n, v_y^n) = (v_x^1 + (n-1) * dist_v, v_y^1)$.

An arc is then used to visualize contacts between a pair of SSEs. The required values for an arc connecting a vertex pair (n, m) can be computed from the position of these vertices. All edges are drawn and the resulting image is written to the output directory in in pixel-based (PNG or JPG) and vector-based (SVG) formats. Additionally, a text file representing the protein graph is saved to the same directory in our own PLG file format, GML format [8], GraphML format [26], and DOT language format [27]. These text files can be used to get more detailed information on the SSEs shown in the image, i.e., their length and amino acid sequence. They can also be opened in other graph editing and visualization software like GraphViz [3]. Furthermore, our software can read PLG files and visualize them directly without having to re-compute any SSE data. In that case, no PDB and DSSP files are required.

The software comes with a graphical user interface but can also be used on the command line, which is useful in batch mode. Options exist to write the results to a database and to include coiled regions in the protein graphs. Our software also includes an option to filter ligands based on their atom count.

4 Discussion

We have applied our method to a copy of the PDB retrieved on May 2011 and saved the results in a database. The database contains entries on 139,923 protein chains stored in 60,880 PDB files, the average PDB file in the database thus contains 2.3 chains. These proteins contain a total of 1,165,616 α-helices, 1,322,639 β-strands and 357,311 ligands. The average SSE consists of about 7 residues, and in general helices are longer than β-strands.

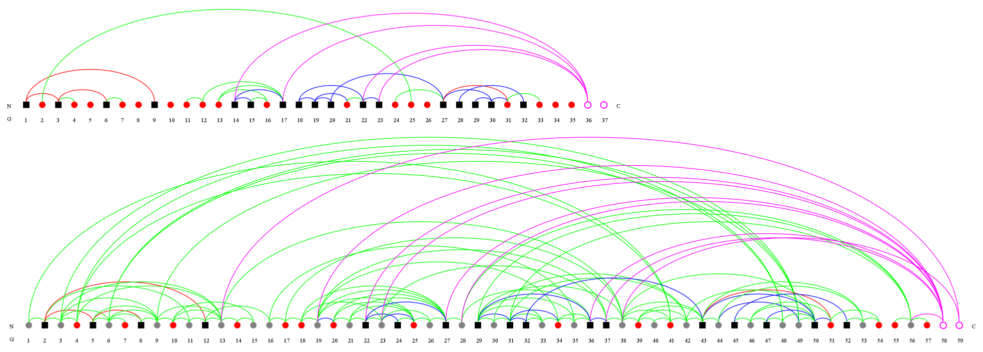


Figure 4 Coiled regions in protein graphs. **(a)** The protein graph of chain A of biotin carboxylase, PDB identifier 2W70. The chloride ion (SSE #37) is isolated because it only has contacts with residues in coiled regions. **(b)** A modified version of the protein graph which includes coiled regions as a new SSE type (shown as gray, filled circles). The contacts between the chloride ion (SSE #59) and protein residues in two coiled regions of the chain (SSE #13 and #38), become visible in this representation.

Note that not all currently available PDB files are included in the database. Many of the ignored files contain only RNA or DNA, others contain very short polypeptides or cannot be processed properly by DSSP.

More than 2,500,000 contacts between SSEs and about 400,000 ligand contacts were detected in the data set. This means that the average α-helix or β-strand has 1.0 contacts to other SSEs of the protein while the average ligand is in contact with 1.1 SSEs or other ligands. These numbers include 319,962 SSEs and 133,216 ligands that do not have any SSE contacts at all, i.e., they are isolated vertices in the protein graph. The higher percentage of isolated ligands compared to SSEs may be related to the fact that ligands usually are small molecules bound to the surface area of a protein. Note that some ligands like Mg^{2+} only consist of a single atom. This means they cannot be surrounded by a large number of SSEs like an α-helix or β-strand deep within the protein core. Additionally, the majority of contacts to coiled regions are ignored by our method, because coiled regions are not made up of regular spatial patterns.

The phosphate ion PO4-147, which is associated with chain D of hemoglobin in the PDB structure 2HHB [5], is such an example: with a collision sphere setting of 2 Å, it is in contact only with VAL-1, but this residue is part of a coiled region. A situation like this is not directly related to the contact computation algorithm, but rather is a consequence of the protein model. A possible solution could be to increase the collision sphere radius of ligand atoms. Visual inspection did indeed show that in many cases, only a subset of the residues forming the binding pocket overlaps with the collision sphere of small ligands. Therefore the default collision sphere radius of ligand atoms was increased to 3 Å. An alternative solution could be to include coiled regions as a new SSE type and to consider all protein residues which are neither an α-helix nor a β-sheet to be part of a coiled region. This would mean that *all* protein residues are represented by an SSE in the protein graph and ligands which only have contacts to coiled regions are not longer isolated.

The large protein graph for chain A of PDB entry 2W70 [20] is depicted in Figure 4, with and without consideration of coiled regions. In the graph which includes coiled regions, the last ligand, the chloride ion CL-1448 is not longer represented by an isolated vertex, but the number of vertices and edges in the graph is dramatically increased. A problem is that allowing coiled regions as SSEs may split other SSEs into multiple parts. Another approach would be to determine the SSE which is spatially closest to such ligands and introduce a new contact type with a new spatial relation ”close to“.

5 Conclusion

In the following we demonstrate the results of our approach using the structure of triosephosphate isomerase (TIM) with PDB identifier 7TIM [2]. TIM is a glycolytic enzyme which catalyzes the interconversion of the three-carbon sugars dihydroxyacetone phosphate and D-glyceraldehyde 3-phosphate. Each of its two chains consists of a central parallel eight-strand β-barrel surrounded by α-helices. The albelig-graph consists of 22 SSEs and is depicted in Figure 3. The betalig-graph makes it easier to spot the β-barrel (see Figure 5). The ligand, phosphoglycolohydroxamic acid, has contacts to several β-strands and α-helices.

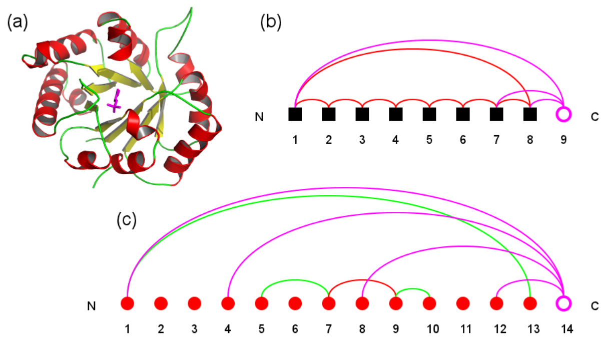


Figure 5 Examples for different types of protein graphs and the structure of triosephosphate isomerase (a). The betalig-graph (b) and the alphalig-graph (c) of the β-chain of triosephosphate isomerase, PDB identifier 7TIM. The α-helices are shown as red, filled circles and the β-strands as black, filled squares. Ligands are represented by magenta circles. From left to right, the vertices are ordered by their position in the amino acid sequence. The arcs mark spatial contacts (red, parallel; blue, anti-parallel; green, mixed; magenta, ligand contact). Note that the β-barrel is clearly visible in the betalig-graph as a series of parallel β-strands.

We have presented a new method to describe and visualize the structure of proteins on the secondary structure level including ligand information. This method is based on a unique graph-theoretic description of protein structure topology. In this paper, we explain the description and how ligand information is considered in the graph representation as well as in the topology cartoons. The visualization of these cartoons is described in detail using examples for illustration.

The related software is baded on PDB structures and DSSP files and produces high quality cartoons which include ligand data. These cartoons can be used for exploring protein structure topology with ligand information. Additionally, the graphs can be stored in a database or exported in various standard graph file formats for usage in other software tools.

The protein model defined by the software has been evaluated on the atom and SSE levels. Limits of both the input data and our method have been discussed. We demonstrate that our approach produces a clear, unique and meaningful representation of protein structure topology. The description as well as the visualization can be applied in proteomics, drug design, medicine and biology. Our software can also be used to extend the existing PTGL [18] database by adding ligand information. The protein graphs could be used for similarity searching within the database using graph-based or string-based methods. A web server with links to other databases, including sequence and pathway databases as well as ligand databases like the PDB Ligand Expo [4], is planned.

References

1 H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, and P.E. Bourne. The protein data bank. *Nucleic Acid Research*, 28:235–242, 2000.

2 R.C. Davenport, P.A. Bash, B.A. Seaton, M. Karplus, G.A. Petsko, and D. Ringe. Structure of the triosephosphate isomerase-phosphoglycolohydroxamate complex: an analogue of the intermediate on the reaction pathway. *Biochemistry*, 30:5821–6, 1991.

3 J. Ellson, E.R. Gansner, E. Koutsofios, S.C.> North, and G. Woodhull. Graphviz and dynagraph – static and dynamic graph drawing tools. In M. Junger and P. Mutzel, editors, *Graph Drawing Software*, pages 127–148. Springer-Verlag, 2004.

4 Z. Feng, L. Chen, H. Maddula, O. Akcan, R. Oughtred, H.M. Berman, and J. Westbrook. Ligand depot: a data warehouse for ligands bound to macromolecules. *Bioinformatics*, 20:2153–5, 2004.

5 G. Fermi, M.F. Perutz, B. Shaanan, and R. Fourme. The crystal structure of human deoxyhaemoglobin at 1.74 Å resolution. *Journal of Molecular Biology*, 175:159–174, 1988.

6 J.F. Gibrat, T. Madej, and S.H. Bryant. Surprising similarities in structure comparison. *Curr Opin Struct Biol.*, 6:377–85, 1996.

7 A. Harrison, F. Pearl, I. Silitoe, T. Slidel, and R. Mott. Recognizing the fold of a protein structure. *Bioinformatics*, 19:1748–1759, 2003.

8 Michael Himsolt. GML: A portable graph file format. Technical report, 1996.

9 John Hopcroft and Robert Tarjan. Algorithm 447: efficient algorithms for graph manipulation. *Communications of the ACM*, 16:372–378, 1973.

10 E.G. Hutchinson and J.M. Thornton. HERA - a program to draw schematic diagrams of protein secondary structures. *PROTEINS: Structure, Function, and Genetics*, 8:203–212, 1990.

11 W. Kabsch and C. Sander. Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22:2577–2637, 1983.

12 I. Koch, F. Kaden, and J. Selbig. Analysis of protein sheet topologies by graph-theoretical methods. *PROTEINS: Structure, Function, and Genetics*, 12:314–323, 1992.

13 I. Koch, A. Kreuchwig, and P. May. *Protein supersecondary structure*, chapter Hierarchical representation of super-secondary structures using a graph-theoretical approach. Humana Press, New York, 2012. In press.

14 I. Koch and T. Lengauer. Detection of distant structural similarities in a set of proteins using a fast graph-based method. In T. Gaasterland, P. Karp, K. Karplus, C. Ouzounis, C. Sander, and A. Valencia, editors, *Proceedings of the Fifth International Conference on Intelligent Systems for Molecular Biology*, pages 167–178, 1997.

15 I. Koch, T. Lengauer, and E. Wanke. An algorithm for finding maximal common subtopologies in a set of protein structures. *Journal of Computational Biology*, 3:289–306, 1996.

16 T. Madej, J.-F. Gibrat, and S.H. Bryant. Threading a database of protein cores. *PROTEINS: Structure, Function, and Genetics*, 23:356–369, 1995.

17 P. May, S. Barthel, and I. Koch. PTGL - protein topology graph library. *Bioinformatics*, 20:3277–3279, 2004.

18 P. May, A. Kreuchwig, T. Steinke, and I. Koch. PTGL: a database for secondary structure-based protein topologies. *Nucleic Acid Research*, 38:D326–D330, 2010.

19 I. Michalopoulos, G.M. Torrance, D.R. Gilbert, and D.R. Westhead. Tops: an enhanced database of protein structural topology. *Nucleic Acids Research*, 32:D251–D254, 2004.

20 I. Mochalkin, J.R. Miller, L.S. Narasimhan, V. Thanabal, P. Erdman, P. Cox, J.V. Prasad, S. Lightle, M. Huband, and K. Stover. Discovery of antibacterial biotin carboxylase inhibitors by virtual screening and fragment-based approaches. *Acs Chem.Biol.*, 4:473–483, 2009.

21 A.G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia. SCOP: a structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 247:536–540, 1995.

22 C.A. Orengo, A.D. Michie, S. Jones, D.T. Jones, M.B. Swindells, and J.M. Thornton. CATH – a hierarchical classification of protein domain structures. *Structure*, 5:1093–1108, 1997.

23 J.S. Richardson. β-sheet topology and the relatedness of proteins. *Nature*, 268:495–500, 1977.

24 Schrödinger, LLC. The PyMOL molecular graphics system, version 1.3r1. August 2010.

25 A. Stivala, M. Wybrow, A. Wirth, J. Whisstock, and P. Stuckey. Automatic generation of protein structure cartoons with Pro-origami. *Bioinformatics*, 27:3315–3316, 2011.

26 The GraphML Team. The GraphML file format. `http://graphml.graphdrawing.org/`.

27 The GraphViz Team. The DOT language. `http://www.graphviz.org/content/dot-language/`.

28 M. Veeramalai. *A Novel Method for Comparing Topological Models of Protein Structures Enhanced with Ligand Information.* PhD thesis, University of Glasgow, 2005.

29 M. Veeramalai and D. Gilbert. A novel method for comparing topological models of protein structures enhanced with ligand information. *Bioinformatics*, 24:2698–2705, 2008.

30 D.R. Westhead, T.W.F. Slidel, T.P.J. Flores, and J.M. Thornton. Protein structural topology: Automated analysis and diagrammatic representation. *Protein Science*, 8:897–904, 1999.

Unbiased Protein Interface Prediction Based on Ligand Diversity Quantification*

Reyhaneh Esmaielbeiki and Jean-Christophe Nebel

Faculty of Science, Engineering and Computing, Kingston University, London
Kingston-Upon-Thames, Surrey, KT1 2EE, UK
`{r.esmaielbeiki,J.Nebel}@kingston.ac.uk`

Abstract

Proteins interact with each other to perform essential functions in cells. Consequently, identification of their binding interfaces can provide key information for drug design. Here, we introduce Weighted Protein Interface Prediction (WePIP), an original framework which predicts protein interfaces from homologous complexes. WePIP takes advantage of a novel weighted score which is not only based on structural neighbours' information but, unlike current state-of-the-art methods, also takes into consideration the nature of their interaction partners. Experimental validation demonstrates that our weighted schema significantly improves prediction performance. In particular, we have established a major contribution to ligand diversity quantification. Moreover, application of our framework on a standard dataset shows WePIP performance compares favourably with other state of the art methods.

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

Keywords and phrases Protein-protein interaction, protein interface prediction, homology modeling

Digital Object Identifier 10.4230/OASIcs.GCB.2012.119

1 Introduction

Protein-protein interaction (PPIs) is essential for the functionality of living cells. Alterations of these interactions affect biochemical processes which may lead to critical diseases such as cancer [31]. Therefore, knowledge about protein interactions and their resulting 3D complexes can provide key information for drug design. A number of experimental techniques are available to identify residues involved in PPIs [31]. Although they provide valuable contribution to PPI knowledge, their cost in terms of time and expense limits their practical use [10]. Consequently, computational methods have been proposed to identify protein interfaces. They can be broadly divided in sequence only and structure based approaches.

Sequence based methodologies usually rely on a sliding window which allows calculating specific features associated to each amino acid according to its neighbours [24, 28, 35, 8]. Then, a classifier discriminates between interface and non-interface residues according to residue scores. Those approaches differ mainly in their selection of amino acid properties, such as physico-chemical properties [8, 7], residues distribution [24] or conservation degree [34, 25], machine learning algorithm and scoring functions [38].

When the 3D structure of the query protein (QP) is available, integration of structural information, e.g. residues secondary structure or solvent-accessible surface area, allows

* This work was in part supported by grant 6435/B/T02/2011/40 of the Polish National Centre for Science.

better predictions [25, 32]. Three approaches taking advantage of these properties are seen as state of the art [38]. ProMate combines 13 different properties, such as chemical component, geometric properties and information from relevant crystal structures, to generate a quantitative measure [23]. Protein interface residues are then predicted using a clustering process relying on mutual information. Alternatively, Cons-PPISP discriminates between residues using neural networks trained with protein's surface sequence profiles and solvent accessibility of neighbouring residues [6]. Finally, PINUP addresses the problem using an empirical energy function which is based on a linear combination of side chain energy score, interface propensity and residue conservation [22]. Despite fundamental differences, these three approaches display very similar performance [38]. However, each of them seems to capture different important aspects of residue interactions. As a result, a meta-predictor, Meta-PPISP, combining their scores using a linear regression analysis, is able to outperform each of these individual methods in terms of accuracy [27].

With the increasing number of experimentally determined protein 3D structures, they have become the main source of interface prediction methods. First, structurally homologous proteins tend to display similar interaction sites [1, 9]. Secondly, protein's binding sites are evolutionarily conserved among structurally similar proteins (or structural neighbours) [37, 18, 19, 4, 36, 5]. Even remote structural neighbours have been shown to display a significant level of interface conservation [37]. Consequently, structure based methods for interface prediction rely on analysing proteins which are structurally similar to the query protein.

Initial approaches focused on detecting conserved areas among homologous structural neighbours. Carl et al. use graph representation of surface residues [18, 19] to describe homologous binding sites [4]. They then refine their technique by using local structural similarity instead of global similarities of the query protein to detect the structural neighbours [5]. A more general method, PredUs, maps interacting residues from structural neighbours onto QP even if they do not display any homology [36]. Whereas PredUs still dependents on the existence of structural neighbours of QP, PrISE proposes to deal with this limitation by predicting interface residues from local structural similarity only [15]. This is achieved using a repository of structural elements (SE) generated from the Protein Data Bank (PDB) [3]. For each SE of the query protein (consisting of a central residues and it neighbours) a set of similar SEs are extracted from the repository and a weight is assigned to them based on their similarity to QP. The central residues of the query protein's SE are predicted as interface if a weighted majority of its similar SEs are interface residues. Although more general, PrISE displays comparable performance to PredUs [15].

Those two approaches have significantly improved the ability of predicting interface residues; see Table 3. However, they do not deal satisfactorily with the very heterogeneous nature of the PDB. First, the presence of complex duplicates, or homologs, biases predictions towards specific configurations, which can affect negatively performance. Secondly, confidence in the information provided by the interface of a structural neighbour should depend on its degree of homology with QP. Although PrISE acknowledges both issues, it does not address the first one [15]. PredUs deals with these matters in a binary fashion. Complexes involving structural neighbours are clustered and a 40% similarity cut-off is used to choose the representatives which will inform interface prediction. Here, we address those limitations of structure based methods by quantifying, first, homology between QP and its structural neighbours and,second, ligand diversity between the partners, or ligands, of the structural neighbours.

In this study we introduce Weighted Protein Interface Prediction (WePIP) framework, a novel PIP approach based on structural neighbours' information. Its main contribution

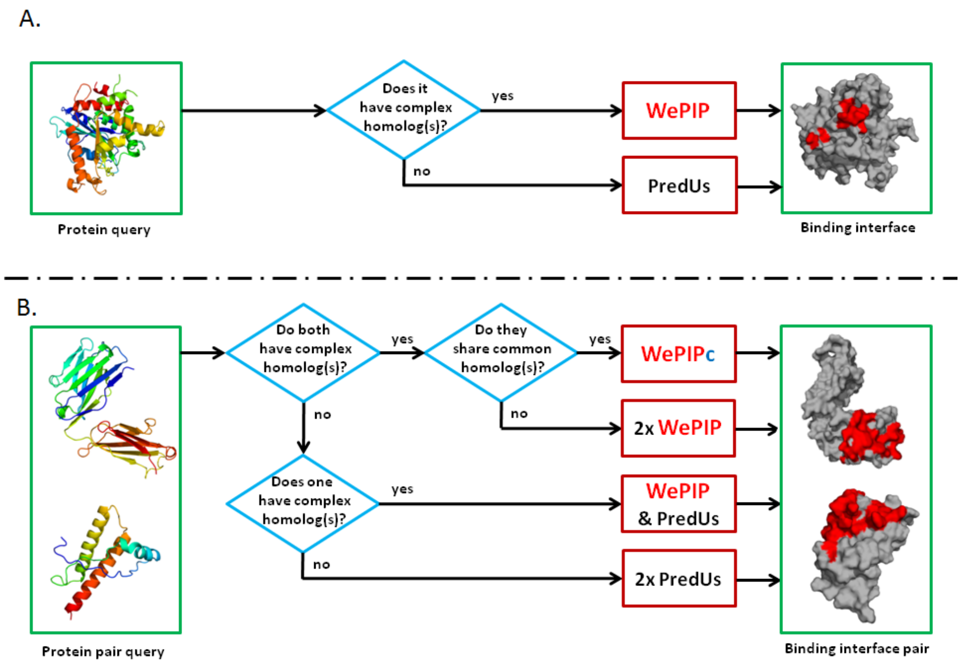


Figure 1 Interface prediction framework for single (A) and pair protein queries (B). A) For a single query, if at least one homologous complex exists, interfaces are predicted using WePIP otherwise PredUs is used. B) For a pair of proteins, depending on the existence of homologous complexes, interfaces are predicted by one or a combination of the WePIPc, WePIP and PredUs methods.

is the weighted score assigned to each residue of QP, which takes into account not only the degree of homology of structural neighbours, but also the nature of their interacting partners. After description of the methodology and validation of the proposed weighted schema, WePIP is evaluated against state of art protein interface prediction methods using a standard benchmark dataset.

2 Methods

2.1 Interface Prediction Principles

When two protein chains form a dimer, they bind through their interaction interfaces. We propose a novel methodology predicting the amino acids which are involved in binding interactions based on the 3D structures of the dimer partners. In this study dimer refers to any two protein chains involved in interaction. Not only does our approach predict the locations of interfaces when both binding partners are known, but it also infers the most likely binding interface of a single protein. Figure 1.A and 1.B describe those interface prediction pipelines for single and pair protein queries respectively.

Our methodology relies on discovering interaction patterns from the analysis of the 3D structures of complexes involving homologs of the dimer partners, called 'homologous complexes'. In this work, proteins are defined as homologous if their sequence similarity is expressed by an E-value $\leq 10^{-2}$. First, Blast [2] is used to classify each query partner according to the availability of homologous complexes in PDB [3]. When both interaction partners are known and common homologous complexes exist, the WePIPc method exploits them as templates to predict both interfaces jointly (see Section 2.2). If both partners have homologous complexes, but none of them is common, or, if one deals with only one partner and this partner has homologous complexes, then interfaces are predicted independently from the interaction partner by the WePIP method (see Section 2.3). Finally, when no homologous complex is available, interface prediction is outside the scope of the WePIP/ WePIPc suite. Therefore, a third party PIP software, such as PredUs [36], is required. In this work, we use PredUs when homologous complexes are not available, because not only it is one of the best performing methods, but also it has been implemented as a Web server which can be used free of charge.

2.2 WePIPc

With rapid increase of experimentally determined structures, homology modelling of the whole 3D structure of a dimer is becoming more and more possible. It has been reported [11] that high quality homologous models could be found for 62% of the protein complexes present in the standard Protein Docking Benchmark 4.0 [13]. Consequently, template-based docking methods have been proposed based on common dimer complexes, i.e. dimers where each chain is homologous to a sequence of the query dimer [11, 21, 20]. For example, 66% of complexes generated by HOMBACOP were categorised as either acceptable or medium-quality models according to CAPRI assessment criteria [21]. Since these approaches have proved particularly accurate, we have included in our PIP framework a module, WePIPc, which infers interfaces based on common complex homologs.

Homologous complexes of each sequence of the protein query pairs are extracted from the PDB using Blast. Common homologous complexes are then selected and ranked by multiplying the E-values associated with the sequential alignments of each query chain with the homologous chain of the common complex. The common complex with the lower score is selected as the template from which the interfaces of the query chains are inferred. This is achieved by mapping the interface residues of the templates on the query chains according to their sequence alignments.

2.3 WePIP

WePIP relies on the observation that interface residues are usually structurally conserved between evolutionary related proteins [36]. Following extraction of homologous complexes from the PDB using Blast, the 3D structure of QP is structurally aligned with its homologs. In this study processing time is reduced by considering at most the 30 homologous complexes involving a chain whose E-value shows closest similarity to the QP. Alignment of multiple protein structures is performed by Multiprot [30], since it is a popular tool [16, 12, 33, 26] that has already be used successfully in interface residue prediction [16]. Using this information, a structure-based multiple sequence alignment (S-MSA) is produced. Then, known interface residues of the homolog complexes are highlighted on the S-MSA, see Figure 2. In agreement with the CAPRI definition [14], an interface residue is defined as an amino acid whose heavy atoms are within $5\mathring{A}$ from those of a residue in a separate chain. Using this multiple alignment, an interaction score is calculated for each residue of the query protein (see Section 2.3.1). Finally, the expected number of interface residues, n_{IA}, is predicted from known interfaces (see Section 2.3.2). The n_{IA} residues with the highest scores are then returned as defining the interaction interface.

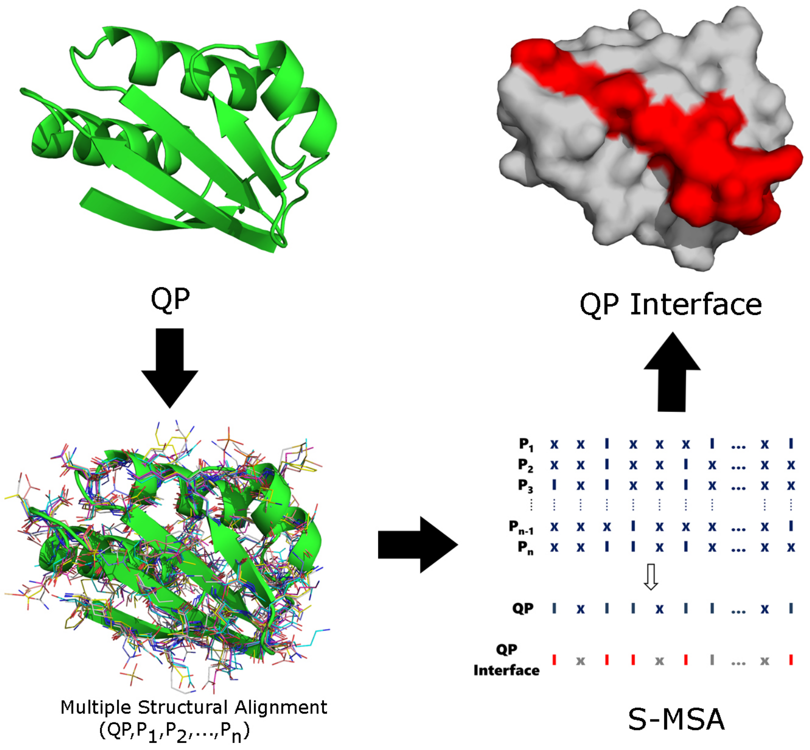


Figure 2 Application of the WePIP method on a query protein (green). First, it is structurally aligned with its homologous complexes. Then, an S-MSA is produced where X and I represent non-interface and interface residues, respectively. Finally, interaction residues (red) of QP are predicted according to interaction scores and the estimated size of the interface. Note that residues weights are not shown here.

2.3.1 Interaction Score

In order to identify residues likely to be involved in the dimer interaction, we propose a residue scoring function which relies on the S-MSA of QP and its complexed homologs. In principle, any QP amino acid aligned with a residue involved in a dimer interaction is a potential candidate. However, confidence in the association of interaction activity to a residue depends on three factors: the degree of homology between the QP sequence and that of the protein from which the interaction is inferred, the nature of the ligand involved in the interaction with the homologous protein and the number of homologous proteins suggesting interaction. First, the more homologous a complexed protein, k, is to QP, the more informative is that protein regarding which residues are likely to be involved in the dimer interaction. We express this information by the query weight, x_k, (1):

$$x_k = \begin{cases} 1 - 10^{-200}, & \text{if } E_k < 10^{-200} \\ 1 - E_k, & \text{if } 10^{-200} \leq E_k \leq 10^{-2} \\ 0, & \text{if } E_k > 10^{-2} \end{cases} \tag{1}$$

where E_k is the E-value of protein k against QP as estimated by Blast.

Secondly, since none of the homologous complexed proteins interacts with the query ligand, diversity of ligands has to be rewarded given that they increase generalisation of interaction patterns. A second weight conveys this requirement by penalising homologous proteins, whose ligands are similar to each others. This is estimated by the average distance between the sequence of a ligand and all the other as expressed by the arithmetic mean of the pair wise E-values. Given a homologous complex protein, k, interacting with a ligand, L_k, and the other $N-1$ homologous complexed proteins interacting with their respective ligand, L_j, the ligand weight, y_k, is formulated as (2):

$$y_k = \begin{cases} \frac{\sum_{j=1,j\neq k}^{N} E_{(L_k,L_j)}}{N-1}, & \text{if } N > 1 \\ 1, & \text{if } N = 1 \end{cases} \tag{2}$$

where $E_{(L_k,L_j)}$ is set to 1, if $E_{(L_k,L_j)} > 1$, and $E_{(L_k,L_j)}$ is set to 10^{-200}, if $E_{(L_k,L_j)} < 10^{-200}$.

The y_k score is designed so that the presence of complex duplicates does not bias predictions towards their configuration. For example, if a QP has 3 complex homologs, A, B and C where L_A is unrelated to L_B and L_C, but L_B and L_C are identical, $E_{(L_A,L_B)} = 1$, $E_{(L_A,L_C)} = 1$ and $E_{(L_B,L_C)} = 0$. Therefore, $y_A = y_B + y_C$, i.e. interface configurations of A and B/C will have the same weight.

The weighted score of the residue i of protein, k, is expressed by the product of these two weights (3):

$$w_{ik} = \begin{cases} x_k y_k, & \text{if } i \text{ interacts with } C_k \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

Finally, since it was shown that usage of non-interface information improves prediction performance [36, 15], the score for residues i of QP is calculated in (4) as the sum of the weights of the interface residues in the homologs over all the interface and non-interface residues which are 3D aligned with i:

$$S_i = \frac{\sum_{j=1}^{N} w_{ij}}{\sum_{j=1}^{N} x_j y_j} \tag{4}$$

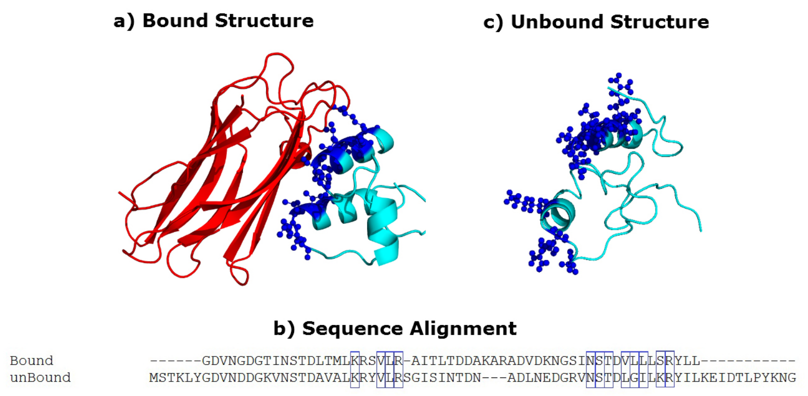
Note that for non-interface residues the ligand which is geometrically the closest is used to calculate their weight y_k.

2.3.2 Estimation of the number of interface residues

After calculating S_i for all the amino acids of QP, it is necessary to estimate the expected number of interface residues of its interface, n_{IA}. Studies have shown that despite variability in ligand structures, the binding location between homologous structures and their ligands is conserved [17]. This suggests that the number of interface residues between homologs should remain quite stable even when the binding partners vary. Therefore, WePIP uses the weighted average number of interacting amino acids of all QP's homologs (n_{IA}) to estimate the number of interface residues of QP (5):

$$n_{IA} = \sum_{i=1}^{R} S_i \tag{5}$$

where R is the number of residues in the QP sequence plus the number of gaps added to allow alignment with its homologs. Finally, once n_{IA} has been calculated, the predicted interface is defined as the n_{IA} residues with the highest scores.



```
Bound            ------GDVNGDGTINSTDLTMLKRSVLR-AITLTDDAKARADVDKNGSINSTDVLLLSRYLL-----------
unBound          MSTKLYGDVNDDGKVNSTDAVALKRYVLRSGISINTDN---ADLNEDGRVNSTDLGILKRYILKEIDTLPYKNG
```

Figure 3 Generation of ground truth interface residues. a) Interfaces residues (blue spheres) are identified on the bound structure (cyan). The interaction partner is in red. b) The unbound and bound sequences are aligned to infer interfaces of the unbound structure. Mapping is shown by blue rectangles. c) Inferred interfaces are shown as blue spheres on the unbound structure (cyan).

3 Experimental results

3.1 Dataset and ground truth

Our interface prediction framework has been evaluated on a standard benchmark dataset, Ds56unbound, to allow comparison of its performance with state of the art methods [37, 15, 38]. The dataset is comprised of 56 unbound chains generated from 27 targets, T01$\sim$T27, investigated during the communitywide experiment CAPRI [14]. The corresponding bound structures (Ds56bound) are used as ground truth. In total, DS56unbound contains 12173 residues including 2112 interacting ones. According to CAPRI's definition, interface residues are defined as amino acids on separate chains which have at least one heavy atom within a cut-off threshold of $5\AA$.

Since interface residues are not explicitly provided in DS56unbound, they were generated from the interface residues in their bound form. The process is illustrated in Figure 3. First, interface residues are detected on the bound complexes. Then, the unbound sequences are aligned with the bound sequences. Finally, the interfaces are mapped from the bound sequences onto the unbound ones.

3.2 WePIP Performance

All chains from Ds56unbound were processed by our interface prediction framework. Following initial homolog search where the Ds56bound complexes were excluded from the Blast results, homologous complexes were returned for 51 chains. Among them, 27 chains (Ds27unbound) displayed common complex(es) with their interacting partner and were further processed by WePIPc. Interfaces of the other 24 chains (Ds24unbound) were estimated by WePIP. Finally, the 5 chains (Ds5unbound) that could not be handled using a homology based approach were submitted to the PredUs server. Table 1 provides detailed performance of our system using standard measures, i.e. precision, recall, F-measure (F1), accuracy, Matthews correlation coefficient (MCC) and area under the receiver operating characteristic – ROC - curve (AUC). As expected, the more the method is able to exploit homology, the better is the interface prediction. Moreover, the table reveals that WePIP is quite conservative in its prediction: it

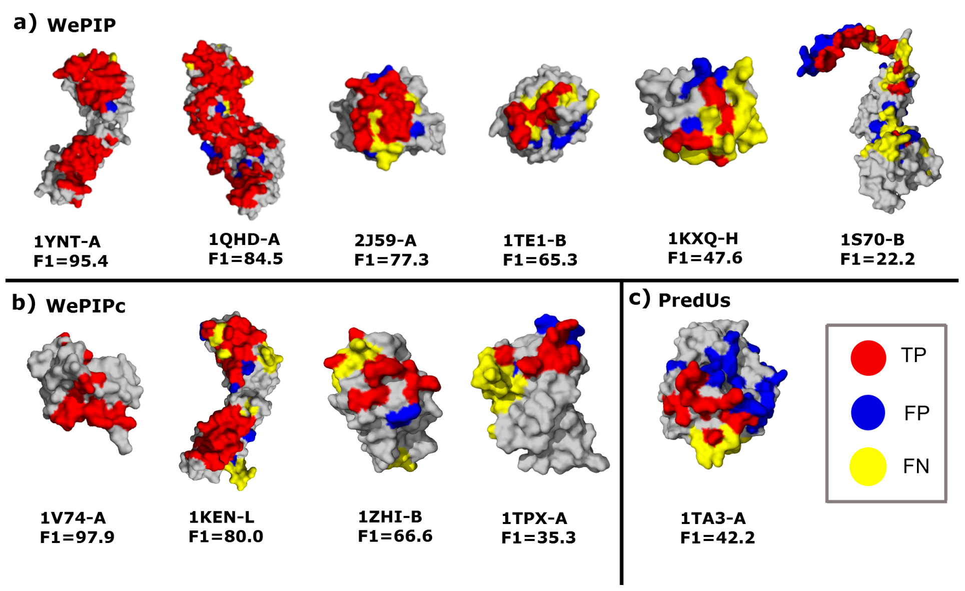


Figure 4 Interface predictions generated by the WePIP framework using either a) WePIP, b) WePIPc or c) PredUs. On each PDB target, true interface residues are coloured in red, whereas false positives and false negatives are shown in blue and yellow respectively. Corresponding F1 scores are also provided.

Table 1 Detailed performance of the WePIP framework. *DSxunbound: this means **x** chains out of the 56 unbound chains are solved by this specific predictor.

Predictor*	**Precision**	**Recall**	**F1**	**Accuracy**	**MCC**	**AUC**
WePIPc (DS27unbound)	68.8	60.5	63.2	87.0	56.0	77.1
WePIP (DS24unbound)	43.3	35.6	38.2	82.3	28.8	70.1
PredUs (DS5unbound)	23.6	45.5	30.1	75.8	19.4	63.2
WePIP + WePIPc (DS51unbound)	56.8	48.8	51.5	84.8	43.2	73.6
WePIP framework (DS56unbound)	**53.9**	**48.5**	**49.6**	**84.0**	**41.1**	**72.9**

displays relatively low recall value compared to precision. Figure 4 illustrates qualitatively those results by displaying representative predicted interfaces compared to ground truth.

In Figure 5, we provide the receiver operating characteristic curves of WePIP interface predictions for the six targets represented in Figure 4 a). First, curves are in agreements with model ranking based on F1 score. Second, accuracy regarding the number of estimated residues in the interface is highly correlated with the AUC. Finally, actual numbers of residues tend to be close to the curve's optimal cut-off point [29]. This point is located on the ROC curve where the distance is the largest to the random diagonal. This suggests there is scope for improving the estimation of expected interacting residues.

In order to validate the formulation of the weights used by WePIP, interface predictions for Ds24unbound were also estimated by setting those weights to 1. As shown in Table 2, results confirm the added value provided by our weighted schema. While usage of the query weight, x_k, only provides modest improvements when compared to a non weighted approach, the proposed ligand weight, y_k, offers a more significant increase of performance. Finally, when both weights are combined, most performance indicators improve further. These results

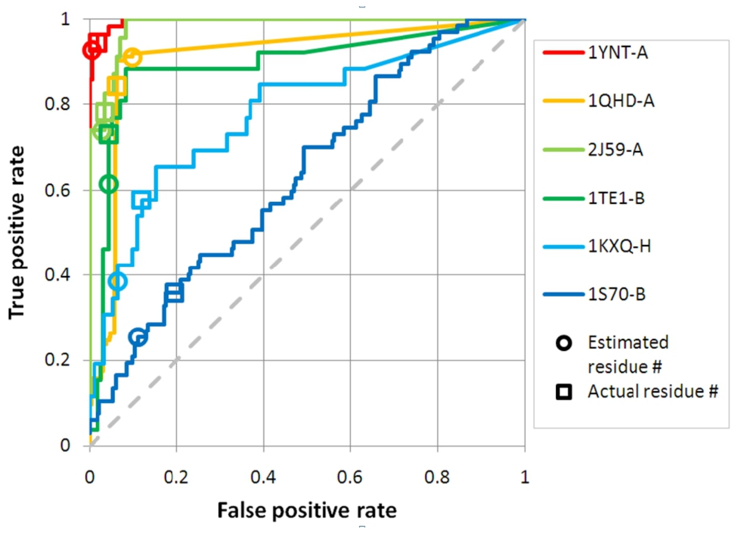


Figure 5 Receiver operating characteristic of WePIP interface predictions.

Table 2 Validation of weights used by WePIP (DS24unbound).

Query weight	Ligand weight	**Precision**	**Recall**	**F1**	**Accuracy**	**MCC**	**AUC**
1	1	40.6	32.5	37.0	81.6	25.5	70.3
x_k	1	40.8	32.7	38.7	82.4	26.2	70.4
1	y_k	42.4	34.6	37.3	**82.6**	28.1	**70.9**
x_k	y_k	**43.3**	**35.6**	**41.7**	82.3	**28.8**	70.1

confirm that taking into account both homology of QP and ligands leads to better interface predictions.

Finally, performance of our framework is compared with state of the art methods, see Table 3. Whatever measure is considered, the WePIP framework displays either best or second best performance competing with PrISE [15] and PredUS [36]. Moreover, the two aggregate measures, i.e. F1 and MCC, show that, globally, our system tends to produce better prediction than any other approach. In addition, since WePIP running time for a 300-residue protein with 30 homologous complexes is around 20 seconds on a standard PC, it can be used online. WePIP will be available as a web server in the near future.

Table 3 Comparison of WePIP framework with state of the art methods.

Predictor (DS56unbound)	**Precision**	**Recall**	**F1**	**Accuracy**	**MCC**	**AUC**
Promate [15]	28.7	27.3	28.0	76.6	14.0	62.7
PINUP [15]	30.4	30.1	30.2	76.9	16.4	60.0
Cons-PPISP [15]	37.4	34.5	35.9	79.5	23.8	71.2
Meta-PPISP [15]	38.9	24.0	29.7	81.1	20.2	71.5
PrISE [15]	43.7	44.0	43.8	81.2	32.6	**75.5**
PredUs [36]	43.3	**53.6**	47.9	73.2	30.4	72.9
WePIP framework	**53.9**	48.5	**49.6**	**84.0**	**41.1**	72.9

4 Conclusion

Although structure based methods perform best at predicting protein interfaces, they do not deal adequately with biases generated by the heterogeneous nature of the PDB. To address this issue, we have introduced the WePIP framework which associates to each putative interaction residue a confidence score taking into account both the degree of homology of structural neighbours and their ligands. Validation demonstrated that our novel weighted schema significantly improves prediction performance. In particular, we showed the major contribution of ligand diversity quantification. Moreover, application of our framework on a standard dataset shows WePIP performance compares favourably with other state of the art methods.

Despite the fact that our framework is state of the art, prediction of interface residues is still an unsolved problem: predictions remain unreliable for too many protein targets, even when homologous complexes are available. In future work, we intend to refine the proposed framework by integrating within our scoring schema the degree of spatial clustering of interface residues among structural neighbours.

Acknowledgments This work was in part supported by grant 6435/B/T02/2011/40 of the Polish National Centre for Science. The authors sincerely thank Rafael A Jordan at Iowa State University for providing the PrISE prediction results and his helpful assistance.

References

1 Patrick Aloy, Hugo Ceulemans, Alexander Stark, and Robert B Russell. The relationship between sequence and interaction divergence in proteins. *J Mol Biol*, 332(5):989–998, Oct 2003.

2 S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Res*, 25(17):3389–3402, Sep 1997.

3 H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The protein data bank. *Nucleic Acids Res*, 28(1):235–242, Jan 2000.

4 Nejc Carl, Janez Konc, and Dusanka Janezic. Protein surface conservation in binding sites. *J Chem Inf Model*, 48(6):1279–1286, Jun 2008.

5 Nejc Carl, Janez Konc, Blaz Vehar, and Dusanka Janezic. Protein-protein binding site prediction by local structural alignment. *J Chem Inf Model*, 50(10):1906–1913, Oct 2010.

6 Huiling Chen and Huan-Xiang Zhou. Prediction of interface residues in protein-protein complexes by a consensus neural network method: test against nmr data. *Proteins*, 61(1):21–35, Oct 2005.

7 Peng Chen and Jinyan Li. Sequence-based identification of interface residues by an integrative profile combining hydrophobic and evolutionary information. *BMC Bioinformatics*, 11:402, 2010.

8 Xue-wen Chen and Jong Cheol Jeong. Sequence-based prediction of protein interaction sites with an integrative method. *Bioinformatics*, 25(5):585–591, Mar 2009.

9 Reyhaneh Esmaielbeiki, Declan P Naughton, and Jean-Christophe Nebel. Structure prediction of ldlr-hnp1 complex based on docking enhanced by ldlr binding 3d motif. *Protein Pept Lett*, 19(4):458–467, Apr 2012.

10 Iakes Ezkurdia, Lisa Bartoli, Piero Fariselli, Rita Casadio, Alfonso Valencia, and Michael L Tress. Progress and challenges in predicting protein-protein interaction sites. *Brief Bioinform*, 10(3):233–246, May 2009.

11 Anisah W Ghoorah, Marie-Dominique Devignes, Malika Smaïl-Tabbone, and David W Ritchie. Spatial clustering of protein binding sites for template based protein docking. *Bioinformatics*, 27(20):2820–2827, Oct 2011.

12 Inbal Halperin, Haim Wolfson, and Ruth Nussinov. Protein-protein interactions; coupling of structurally conserved residues and of hot spots across interfaces. implications for docking. *Structure*, 12(6):1027–1038, Jun 2004.

13 Howook Hwang, Thom Vreven, Joël Janin, and Zhiping Weng. Protein-protein docking benchmark version 4.0. *Proteins*, 78(15):3111–3114, Nov 2010.

14 Joël Janin and Shoshana Wodak. The third capri assessment meeting toronto, canada, april 20-21, 2007. *Structure*, 15(7):755–759, Jul 2007.

15 Rafael A Jordan, Yasser El-Manzalawy, Drena Dobbs, and Vasant Honavar. Predicting protein-protein interface residues using local surface structural similarity. *BMC Bioinformatics*, 13:41, 2012.

16 Ozlem Keskin, Buyong Ma, and Ruth Nussinov. Hot regions in protein–protein interactions: the organization and contribution of structurally conserved hot spot residues. *J Mol Biol*, 345(5):1281–1294, Feb 2005.

17 Ozlem Keskin and Ruth Nussinov. Similar binding sites and different partners: implications to shared proteins in cellular pathways. *Structure*, 15(3):341–354, Mar 2007.

18 Janez Konc and Dusanka Janezic. Protein-protein binding-sites prediction by protein surface structure conservation. *J Chem Inf Model*, 47(3):940–944, 2007.

19 Janez Konc and Dusanka Janezic. Probis algorithm for detection of structurally similar protein binding sites by local structural alignment. *Bioinformatics*, 26(9):1160–1168, May 2010.

20 Petras J Kundrotas and Emil Alexov. Predicting 3d structures of transient protein-protein complexes by homology. *Biochim Biophys Acta*, 1764(9):1498–1511, Sep 2006.

21 Petras J Kundrotas, Marc F Lensink, and Emil Alexov. Homology-based modeling of 3d structures of protein-protein complexes using alignments of modified sequence profiles. *Int J Biol Macromol*, 43(2):198–208, Aug 2008.

22 Shide Liang, Chi Zhang, Song Liu, and Yaoqi Zhou. Protein binding site prediction using an empirical scoring function. *Nucleic Acids Res*, 34(13):3698–3707, 2006.

23 Hani Neuvirth, Ran Raz, and Gideon Schreiber. Promate: a structure based prediction program to identify the location of protein-protein binding sites. *J Mol Biol*, 338(1):181–199, Apr 2004.

24 Yanay Ofran and Burkhard Rost. Predicted protein-protein interaction sites from local sequence information. *FEBS Lett*, 544(1-3):236–239, Jun 2003.

25 Yanay Ofran and Burkhard Rost. Isis: interaction sites identified from sequence. *Bioinformatics*, 23(2):e13–e16, Jan 2007.

26 Utkan Ogmen, Ozlem Keskin, A. Selim Aytuna, Ruth Nussinov, and Attila Gursoy. Prism: protein interactions by structural matching. *Nucleic Acids Res*, 33(Web Server issue):W331–W336, Jul 2005.

27 Sanbo Qin and Huan-Xiang Zhou. meta-ppisp: a meta web server for protein-protein interaction site prediction. *Bioinformatics*, 23(24):3386–3387, Dec 2007.

28 I. Res, I. Mihalek, and O. Lichtarge. An evolution based classifier for prediction of protein interfaces without using protein structures. *Bioinformatics*, 21(10):2496–2501, May 2005.

29 Enrique F Schisterman, Neil J Perkins, Aiyi Liu, and Howard Bondell. Optimal cut-point and its corresponding youden index to discriminate individuals using pooled blood samples. *Epidemiology*, 16(1):73–81, Jan 2005.

30 Maxim Shatsky, Ruth Nussinov, and Haim J Wolfson. A method for simultaneous alignment of multiple protein structures. *Proteins*, 56(1):143–156, Jul 2004.

31 Benjamin A Shoemaker and Anna R Panchenko. Deciphering protein-protein interactions. part i. experimental techniques and databases. *PLoS Comput Biol*, 3(3):e42, Mar 2007.

32 Mile Sikić, Sanja Tomić, and Kristian Vlahovicek. Prediction of protein-protein interaction sites in sequences and 3d structures by random forests. *PLoS Comput Biol*, 5(1):e1000278, Jan 2009.

33 Christof Winter, Andreas Henschel, Wan Kyu Kim, and Michael Schroeder. Scoppi: a structural classification of protein-protein interfaces. *Nucleic Acids Res*, 34(Database issue):D310–D314, Jan 2006.

34 Li C Xue, Drena Dobbs, and Vasant Honavar. Homppi: a class of sequence homology based protein-protein interface prediction methods. *BMC Bioinformatics*, 12:244, 2011.

35 Changhui Yan, Drena Dobbs, and Vasant Honavar. A two-stage classifier for identification of protein-protein interface residues. *Bioinformatics*, 20 Suppl 1:i371–i378, Aug 2004.

36 Qiangfeng Cliff Zhang, Lei Deng, Markus Fisher, Jihong Guan, Barry Honig, and Donald Petrey. Predus: a web server for predicting protein interfaces using structural neighbors. *Nucleic Acids Res*, 39(Web Server issue):W283–W287, Jul 2011.

37 Qiangfeng Cliff Zhang, Donald Petrey, Raquel Norel, and Barry H Honig. Protein interface conservation across structure space. *Proc Natl Acad Sci U S A*, 107(24):10896–10901, Jun 2010.

38 Huan-Xiang Zhou and Sanbo Qin. Interaction-site prediction for protein complexes: a critical assessment. *Bioinformatics*, 23(17):2203–2209, Sep 2007.