Proceedings of WCB 2010

Workshop on Constraint Based Methods for Bioinformatics

July 21st, 2010 Edinburgh

Table Of Contents

Preface i
Alessandro Dal Palù, Agostino Dovier, Sebastian Will

Constraint-Based Modeling in Systems Biology *(invited talk)* 1
Alexander Bockmayr

Minimizing enzymes to diferenciate between species 2
David Buezas, Joao Almeida, Pedro Barahona

Maximum likelihood pedigree reconstruction using integer programming . 9
James Cussens

Optimal haplotype reconstruction in half-sib families 20
Aurélie Favier, Jean-Michel Elsen, Simon de Givry, Andrés Legarra

Alignnment of RNA with Structures of Unlimited Complexity 31
Alessandro Dal Palù, Mathias Möhl, Sebastian Will

Geometric Constraints for the Phase Problem in X-Ray Crystallography . 36
Corinna Heldt, Alexander Bockmayr

Building Portfolios for the Protein Structure Prediction Problem 42
Alejandro Arbelaez, Youssef Hamadi, Michele Sebag

Lattice model refinement of protein structures 47
Martin Mann, Alessandro Dal Palù

Finding Minimal Reaction Sets in Large Metabolic Pathways 54
Takehide Soh, Katsumi Inoue

Perspectives on Constraints, Process Algebras, and Hybrid Systems 69
Luca Bortolussi, Alberto Policriti

Preface

Computation has become an indispensable method for today's bio-sciences. Many routine tasks of biological research are performed using well-established algorithms. However, the increasing complexity of studied systems demands for new computational solutions to complex and often NP-hard problems. By their ability to simplify problem modeling and to face computationally hard problems, constraint-based methods promise to contribute essentially to the Bioinformatics tool set. Many Bioinformatics problems are naturally formalized in constraint-systems over finite domains or reals. This trend emerged more than 10 years ago, witnessed by the workshops *Constraints and Bioinformatics/Biocomputing* associated to CP97 and CP98 and later, starting from 2005, the *Workshop on Constraint based methods for Bioinformatics* (briefly, WCB) series have been held regularly every year. These workshops provide an excellent overview over recent approaches for tackling Bioinformatics problems using constraint methodology. Furthermore it acts as a platform for discussion among experts in this area. Constraint techniques proved to be successful for a variety of problems. Some particular topics are sequence analysis, biological systems simulations, protein structure prediction and docking, pedigree analysis, haplotype inference.

This year, we accepted 9 strong workshop contributions out of high quality submissions. The topics comprise analysis of pedigrees, haplotypes and differentiation of species. Furthermore, alignment of RNA with complex structures, protein structure analysis and prediction, and finally, the analysis of metabolic pathways and the modeling of biological systems. Again, we observe a continuing high interest in the workshop. The diversity of topics and the reported results emphasize that constraints provide a valuable tool for bioinformatics and promise significant advance for a broad variety of applications.

We would like to thank here our colleagues Rolf Backofen, Pedro Barahona, Alexander Bockmayr, Mats Carlsson, Esra Erdem, Simon de Givry, Francois Fages, Inês Lynce, Neil Moore, and Enrico Pontelli, that accepted to be member of the Program Committee and dedicated their precious time in the reviewing phase, as well as the external reviewers Lars Kotthoff, Halit Erdogan, and Ozan Erdem, that helped us in the same stage. We also would like to thank the FLOC organization, and in particular, Veronica Dahl, Phil Scott, Bartek Klin, Nicole Schweikardt, and Andrei Voronkov, and of course all the authors that submitted a paper to this edition of the workshop. A particular thank to *Alexander Bockmayr* who contributed, by accepting to give the invited talk, to enrich the contents of the workshop.

Alessandro Dal Palù and Agostino Dovier are partially supported by the grants: GNCS-INdAM *Tecniche innovative per la programmazione con vincoli in applicazioni strategiche*, and PRIN 2008 *Innovative multi-disciplinary approaches for constraint and preference reasoning*.

May 2010

Alessandro Dal Palù
Agostino Dovier
Sebastian Will

Constraint-Based Modeling in Systems Biology

Alexander Bockmayr
DFG-Research Center MATHEON,
FB Mathematik und Informatik,
Freie Universität Berlin,
Arnimallee 6, 14195 Berlin, Germany
Alexander.Bockmayr@fu-berlin.de

Abstract

Systems biology is a new interdisciplinary research field that has received considerable attention in recent years. While traditional molecular biology studies the various components of a biological system (e.g. genes, RNAs, proteins) in isolation, systems biology aims to understand how these components interact in order to perform complex biological functions. A variety of mathematical and computational methods is currently being used to model and analyze biological systems, ranging from continuous, stochastic, and discrete to various hybrid approaches.

The idea of constraint-based modeling in systems biology is to describe a biological system by a set of constraints, i.e., by pieces of partial information about its structure and dynamics. Using constraint-based reasoning one may then draw conclusions about the possible system behaviors.

In this talk, we will focus on constraint-based modeling techniques for regulatory networks starting from the discrete logical formalism of René Thomas. In this framework, logic and constraints arise at two different levels. On the one hand, Boolean or multi-valued logic formulae provide a natural way to represent the structure of a regulatory network, which is given by positive and negative interactions (i.e., activation and inhibition) between the network components. On the other hand, temporal logic formulae (e.g. CTL) may be used to reason about the dynamics of the system, represented by a state transition graph or Kripke model.

Emphasis will be on non-deterministic modeling of the network dynamics and model checking techniques for network inference. We will also discuss how to include additional temporal constraints on time delays in a hybrid discrete-continuous modeling framework.

Minimizing sets of enzymes to differentiate between species

David Buezas*
CENTRIA †
FCT/UNL
david.buezas@gmail.com

João Almeida
CREM ‡
FCT/UNL
jmfa@fct.unl.pt

Pedro Barahona
CENTRIA †
FCT/UNL
pb@di.fct.unl.pt

Abstract

A large number of species cannot be distinguished via standard non genetic analysis in the lab. In this paper we address the problem of finding minimum sets of restriction enzymes that can be used to unequivocally identify the species of a yeast specimen by analyzing the size of digested DNA fragments in gel electrophoresis experiments. The problem is first mapped into set covering and then solved using Constraint Programming techniques. Although the data sets used are relatively small (23 yeast species and 331 enzymes), a similar approach might be applicable to larger ones and to a number of variants as discussed in the conclusion. The subject of this paper has already raised the interest of our biologist partners and may become a benchmark for the application of Constraint Programming techniques to Bioinformatics.

1 Introduction

The problem of yeast identification was historically addressed through the study of both morphological traits and physiological features [3, 5, 16], but alternative molecular methods have been adopted to obtain the sequence of particular genomic regions and thus identify a given species [6, 11].

Although sequencing nucleic acids is more accessible than ever, it is still an expensive technique, especially if applied to a high numbers of specimens. In contrast to less expensive techniques like RFLP, RAPD, MSP-PCR (which allow the formation of clusters among the specimens to be identified, with inherent result limitations in scope), ARDRA [12] was proposed to differentiate between species of a eubacterial family and it represents an approach that goes beyond the mere clustering operation. The amplified fragment and the digestion products sizes are reproducible, characteristic for the substrate sequence, and thus characteristic for the source taxon, generally enabling the identification of the organism.

ARDRA-ITS [10] was developed with the purpose of differentiating fungal species. The differences between the original technique and the ITS variant lay on the primers, ITS1 and ITS4 [15], that amplify the 5.8S-ITS region of the operon, and in the set of enzymes used: HaeIII and TaqI. Other authors [1] took a step further and proposed the use of a variant, ARDRA-ITS, as an identification method for yeasts. They used a different set of restriction enzymes (CfoI, HaeIII, HinfI, and several more to resolve occasional ambiguities), and the latter target region, the 5.8S-ITS region. This genomic region also happens to be one of the better represented in the public nucleotidic sequences databases (GenBank, EMBL Bank and DDBJ consortium). This approach has been used with considerable success to identify yeasts associated with food [2, 7, 13] and a commercial database is available for this purpose (www.yeast-id.com), but its usefulness has been hindered by the reduced set of yeast strains studied and the limitations of size resolution of classical electrophoresis apparatus.

Recent papers are acknowledging the power of *in silico* contributions in this field. One is limited to the forecast of electrophoretic patterns [8], the other presents a program to assess the utility of a fixed set

*The author David Buezas was supported by the European Master's Program in Computational Logic (EMCL).

†Centro de Inteligência Artificial, Dep. de Informática, Faculdade de Ciências e Tecnologia / Universidade Nova de Lisboa, Caparica, Portugal.

‡Centro de Recursos Microbiológicos, Dep. Ciências da Vida, Faculdade de Ciências e Tecnologia / Universidade Nova de Lisboa, Caparica, Portugal.

of endonucleases to distinguish between a given set of sequences [14]. However, the integration of all the available data in a comprehensive in silico approach, targeting optimality in identification by ARDRA is still to be proposed.

The purpose of this paper is twofold. Firstly, in the context of ARDRA-ITS, we propose to infer the minimum set of enzymes required to identify one, from a given set of yeasts. Secondly, we propose that this problem is used as benchmark for Constraint Programming methods applied to Bioinformatics. Although the instance presented in the paper has been solved, larger instances and variations of the problem may pose a relevant challenge to CP techniques.

The paper is organized as follows. Section 2 shows how the ARDRA-ITS technique can be cast into a minimum set covering problem. Section 3 presents different models to obtain both a single solution and all solutions to this problem, and briefly discusses the experimental results obtained with them. Finally, section 4 presents some initial conclusions and a discussion of further work.

2 Mapping ARDRA into a Minimum Set Covering problem

The ARDRA-ITS technique identifies one from a set of specimens through analysis of a specific DNA sub-sequence of its genome. Restriction enzymes (that, as is well known, cut double-stranded or single stranded DNA at specific recognition nucleotide sequences, known as *restriction sites*) play a central role in the ARDRA-ITS technique that proceeds as follows: First, a "standard" fragment of the test specimen DNA is obtained (in the case of yeasts, the 5.8S-ITS region of their operons), and many copies of it are produced. Secondly, a set of restriction enzymes are separately applied to these copies. The complete digestion of each enzyme yields several smaller nucleotide segments that, subject to gel-electrophoresis, originate bands of different lengths.

Each yeast - restriction enzyme pair generates a specific band pattern, but given the similarity of their DNA, several yeasts are likely to present similar patterns when digested by most restriction enzymes. Subject to some experimental error, there is a one-to-one correspondence between fragment sizes and the position of the respective band in the pattern, hence the sizes of the fragments obtained can be approximately calculated from the gel electrophoresis experiments. On the other hand, when its DNA sequence is known, the pattern produced by the digestion of yeast Y (or rather, the 5.8S-ITS region of its operon) by the restriction enzyme R can be computed by running a simulation of a gel electrophoresis experiment. A simple diagram of digestion in this context is shown in Figure 1.

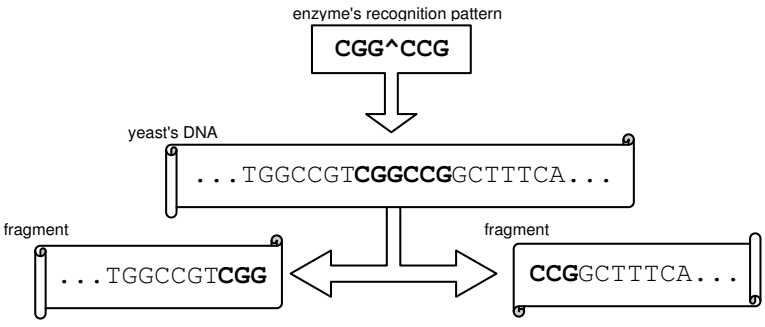


Figure 1: Diagram of digestion

A restriction enzyme R differentiates two yeast specimens Y_1 and Y_2 if the patterns it produces from them are distinguishable, i.e. at least one fragment in one of the digested yeasts is of a sufficiently

different size from any fragment in the other digested yeast.

It is thus possible to produce a Boolean coverage table D, where rows denote yeast pairs and columns represent restriction enzymes. In this table the cell in the row $Y_i - Y_j$ and the column E_k tells whether that yeast pair is differentiated by that restriction enzyme. The problem of identifying a yeast among a set of similar yeasts can be formulated as finding a set S of restriction enzymes that differentiate any pair of yeasts, i.e. for all rows there is at least one enzyme in S such that its corresponding column has a true value for that row.

More formally, given a set of yeasts $Y = \{Y_1, ..., Y_{Ny}\}$ and a set of enzymes $E = \{E_1, ..., E_{Ne}\}$, we denote as $P(i,k)$ the induced pattern for Y_i by E_k, i.e. the set of segment lengths produced by the digestion of yeast Y_i by enzyme E_k. Two patterns P and Q are distinct if there is a fragment length in one of them that is sufficiently different (depending on the experimental error and denoted by $\not\approx$) from any fragment of the other pattern i.e.

$$\mathsf{distinct}(P,Q) =_{def} (\exists u \in P)(\forall v \in Q)(u \not\approx v) \vee (\exists u \in Q)(\forall v \in P)(u \not\approx v)$$

Two yeasts are differentiated by a restriction enzyme if the patterns induced in them are distinct:

$$(\forall i < j \text{ in } 1..N_y)(\forall k \text{ in } 1..N_e)$$
$$\mathsf{differentiate}(i,j,k) =_{def} \mathsf{distinct}(P(i,k),P(j,k))$$

A discriminating set of enzymes S is a subset of the set E of enzymes that, for any pair of yeasts in the set Y, has an element that differentiates them, i.e.

$$\mathsf{disc}(S,Y) =_{def} \forall (i\text{-}j) \in Y \ \exists k \in S : \mathsf{differentiate}(i,j,k)$$

A minimal (optimal) discriminating set of enzymes S is a discriminating set with minimal cardinality:

$$\mathsf{min\_disc}(S,Y) =_{def} \mathsf{disc}(S,Y) \wedge (\forall R \, \mathsf{disc}(R,Y) \rightarrow \#S \leq \#R)$$

Hence, given a set Y of yeast specimens, the ARDRA-ITS problem can be regarded as the problem of finding, from a set E of available restriction enzymes, a minimal discriminating set S for the set of yeast specimens.

Since some solutions might be preferred over others by the user (according to not yet fully formalized criteria such as reliability, availability and cost) it is also interesting to find not only one, but all minimal discriminating sets.

3 Alternative models

We tested a number of alternative models with a dataset of commercially available restriction enzymes, containing about 3500 elements [9]. As this set was redundant, meaning that many enzymes had the same recognition sequences, it was reduced to an equivalent one containing only 331 enzymes. The dataset of yeasts that we used is available in [http://www.cbs.knaw.nl/databases/], it includes the nucleotide sequences of the 5.8S-ITS region of the operons of 23 yeast specimens. All the tests presented below where run in a Intel(R) Core(TM)2 Duo T5670 @1.80GHz (2 CPUs) with 3 GB of RAM, with a SICStus 4 CLP system.

3.1 Greedy model

A simple and greedy approach to solve the problem was implemented by accumulating the best enzymes (i.e. those that differentiate more yeast pairs still to be covered) until all yeast pairs are covered. The pseudo code is shown in Algorithm 1.

Algorithm 1 Greedy model

$S \leftarrow \emptyset$
$Y \leftarrow \{i\text{-}j : i \in 1..N_y,\ j \in 1..N_y,\ i < j\}$ ▷ the set of all yeast pairs to cover
$E \leftarrow \{k : k \in 1..N_e\}$ ▷ the set of all enzymes available
while $Y \neq \emptyset$ **do**
    $e \leftarrow \underset{k \in E}{\text{argmax}} \left(\sum_{i\text{-}j \in Y} \text{differentiate}(i, j, k) \right)$ ▷ select the most covering enzyme
    $S \leftarrow S \cup \{e\}$
    $Y \leftarrow Y \setminus \{i\text{-}j : \text{differentiate}(i, j, e)\}$ ▷ subtract the covered yeast pairs
end while

Of course, this greedy approach does not guarantee that, upon termination, set S is an optimal discriminating set. In fact, notwithstanding the very fast execution time (125 ms), the solution found with our datasets contains nine enzymes, being far from minimal and therefore useless.

3.2 Backtrack model

This model guarantees optimality by finding differentiating sets of restriction enzymes with an increased size. The first set obtained is thus an optimal discriminating set. Alternative minimal differentiating sets can be obtained (with backtracking) by changing the condition in the while loop. The pseudo code is shown in Algorithm 2.

Algorithm 2 Backtrack model

$Y \leftarrow \{1, ..., N_y\}$ ▷ the set of all yeast identifiers
$p \leftarrow 0$ ▷ p stands for the size of the solution
$found \leftarrow$ **false**
while $\neg found$ **do**
    $p \leftarrow p + 1$ ▷ the size of the optimal solution is searched incrementally
    for all $k_1..k_p \in 1..N_e : e_1 < ... < e_p$ **do**
        $S \leftarrow \{k_1, ..., k_p\}$
        for all $i, j \in 1..N_y : i < j$ **do**
            if $\neg$ discriminate(i, j, S) **then**
                break
            end if
        end for
    end for
    $found \leftarrow$ differentiate(Y, S)
end while

The execution time was close to 1 minute, but is heavily dependent on the order in which the enzymes are considered. If all solutions were to be found by backtracking alone, a huge number (around 6 million) of triplets would have to be tested, requiring an unacceptably huge execution time.

3.3 Constraint Programming model with Boolean variables

The selection of the k^{th} enzyme in the discriminating set is modeled by a Boolean variable x_k, and a Constraint Programming system simply solves the problem of finding an assignment of these variables that covers all the yeast pairs, each covering being represented by a sum-product constraint of the variables x_k and the $0/1$ constants representing the differentiating features. Of course all solutions can be obtained by backtracking. The pseudo code is shown in Algorithm 3.

Algorithm 3 Boolean CP model

1: $X \leftarrow [x_1, ..., x_{N_e}]$ ▷ one boolean variable for each enzyme identifier
2: **for all** $x \in 1.N_e$ **do**
3: $x_k \in 0..1$
4: **for all** $i, j \in 1..N_y : i < j$ **do**
5: $\sum_{k \in 1..N_e} x_k * \text{differentiate}(i, j, k) \geq 1$ ▷ constraints are imposed
6: **end for**
7: **end for**
8: label(X): minimising$\left(\sum_{k \in 1..N_e} x_k\right)$

With this model, the first minimum solution was found 15 seconds, which includes the 5 seconds necessary to initialize the covering table. This model is sufficiently efficient to compute all solutions of this problem instance. After the initialization time, all solutions were found in 15 minutes (the timing for finding the next solution vary widely from a some milliseconds to a few minutes).

3.4 Constraint Programming model with Finite Domain variables

Now each variable x_{ij} in the X vector is associated to the yeast pair Y_i-Y_j and its domain is the set of enzymes that differentiate such pair. By labeling X minimizing the number of different elements it uses, minimum solutions are found. The pseudo code is shown in Algorithm 4.

Algorithm 4 Finite Domain CP model I

1: $X = [x_{1\text{-}2}, ..., x_{i\text{-}j}, ..., x_{(N_y-1)\text{-}N_y}] : i < j$ ▷ one Finite Domain variable for each yeast pair
2: list_to_set(X, S)
3: **for all** $i, j \in 1..N_y : i < j$ **do**
4: $x_{i\text{-}j} \in \{k \in 1..Ne : \text{differentiate}(i, j, k)\}$
▷ the domain of $x_{i\text{-}j}$ is the set of enzyme identifiers that cover the i-j pair
5: **end for**
6: label(X): minimizing(#S)

To be effective, this model requires the minimization of the number of distinct values in list X (or equivalently, the number of elements in set S). In CP systems this can be achieved using the Nvalue(K,L) global constraint, that maps into the finite domain variable K, the number of distinct values in list L, as proposed in [4].

With this model, finding the first minimal solution takes 1 second (after the 5 seconds for table initialization). Unfortunately, the model cannot be used to find all solutions since many repetitions are obtained. For example, let us assume we have three yeast specimens (Y_1, Y_2, Y_3) forming three distinct yeast pairs

$$P_1 =< Y_1, Y_2 >, P_2 =< Y_1, Y_3 >, P_3 =< Y_2, Y_3 >$$

and that P_1 is covered by enzymes 2 and 3, P_2 by enzymes 1 and 3, and P_3 by enzymes 1 and 2. The tree yeast pairs would be represented as the vector $X = [x_{1\text{-}2}, x_{1\text{-}3}, x_{2\text{-}3}]$ where $x_{1\text{-}2} \in \{2,3\}$, $x_{1\text{-}3} \in \{1,3\}$ and $x_{2\text{-}3} \in \{1,2\}$. This configuration allows six different labelings for X which use the least number of enzymes and therefore minimize the cardinality of S, namely:

$$X_1 = [2,1,1] \; X_2 = [3,1,1] \; X_3 = [3,3,2]$$
$$X_4 = [2,1,2] \; X_5 = [3,3,1] \; X_6 = [2,3,2]$$

but since S is a set, each pair of labelings in the same column represent the same solution. Here there are only two repetitions per solution, but when real data is used the number of repetitions is so big that it prevents the enumeration of all solutions.

3.5 Avoiding repetitions with a different Finite Domain model

The previous model could not be easily adapted finding all solutions because, as just discussed, the same solution can come in a wide variety of encodings. Hence we decided to use a somewhat dual model of the previous one by having variables associated to restriction enzymes instead of yeast pairs. Once we found a minimal solution using the previous model, we may fix the size of a list of enzyme variables that must distinguish all yeast pairs. The pseudo code is shown in Algorithm 5.

Algorithm 5 Finite Domain CP model II

1: $p \leftarrow \#S$ ▷ where #S is the minimum solution size
2: $S \leftarrow [k_1, ..., k_p]$
3: $k_1 < ... < k_p$ ▷ imposing an order avoids repeated solutions
4: **for all** $i, j \in 1..N_y : i < j$ **do**
5: $E \leftarrow \{e :$ differentiate$(i, j, e)\}$
6: $\bigvee_{i \in 1..p} (k_i \in S)$ ▷ a disjunctive constraint is posed
7: **end for**
8: label(X)

Note that this model can only be setup when the size of a minimal solution is known. Alternatively, we may start with a set of enzymes with cardinality 1 and increment this size, as with the second (backtrack) model. With this model all solutions were found in 50 seconds. Hence, Finite Domain models improve on the Boolean model by one order of magnitude, both to find the first solution (1 sec against 10 secs) as well as all solutions (50 secs against 15 mins).

4 Conclusions and further work

In this paper we explore several potential models to a Bioinformatics problem, raised by the ARDRA-ITS experimental technique, requiring the minimization of the number of enzymes that must be used in gel electrophoresis experiments to unequivocally tell a yeast within a set of alternative and related yeasts specimens. By and large the problem can be applied to other types of organisms (ARDRA-ITS is an adaptation of ARDRA, originally used for identification of eubacterial family members) so its practical application can be quite wide.

Species are the taxonomic level we dealing with in this paper, but this approach can be extended to handle any taxonomic level. This idea is worth pursuing since when higher taxonomic levels are considered the execution time is reduced (because the number of specimen pairs in the coverage table is smaller) and solutions are likely to require a smaller number of enzymes.

The technique we used mapped the problem into a set covering problem, whose complexity is proportional to number of available restriction enzymes and the square of the number of specimen to identify. The data sets we used (around 300 enzymes and 23 yeasts, i.e. 253 yeast pairs) show the advantage of using constraint programming techniques over backtracking or purely heuristic search techniques, which solve this problem somewhat naively. Incidentally, this also justifies why we did not compare our models with Integer Programming alternatives, although we plan to do so whenever larger data sets are be available.

A number of variants to deal with uncertainty can be considered for this problem. On the one hand, we arbitrarily assumed that bands in electrophoresis experiments are distinguishable if their lengths differ by a certain minimum ratio (we used $\pm$ 5%). This is hardwired in our models but it would be interesting to model such relative difference as a parameter that is to be maximised, so that the solutions found are not only minimal but also the most reliable ones. On the other hand, we may consider that the yeast databases are obtained by consensus, and some of their nucleotides may vary. A quantified version of the problem would be to find a minimal set of enzymes that unequivocally identify a yeast, whatever the nucleotides a yeast variant may present. We plan to address both variants of this problem and provide a more comprehensive set of benchmarks, as well as experimental results.

References

[1] Esteve-Zarzoso B, Belloch C, Uruburu F, and Querol A, *Identification of yeasts by RFLP analysis of the 5.8s rRNA gene and the two ribosomal internal transcribed spacers*, International Journal of Systematic Bacteriology **49** (1999), 329–337.

[2] Esteve-Zarzoso B, Fernandez-Espinar MT, and Querol A, *Authentication and identification of saccharomyces cerevisiae 'flor' yeast races involved in sherry ageing*, Antonie Van Leeuwenhoek International Journal of General and Molecular Microbiology **85** (2004), 151–158.

[3] J. Barnett, R. Payne, and D. Yarrow, *Yeasts, characteristics and identification*, 3rd ed., Cambridge University Press, 2000.

[4] Christian Bessiere, Emmanuel Hebrard, Brahim Hnich, Zeynep Kiziltan, and Toby Walsh, *Filtering algorithms for the NValue constraint*, Constraints **11** (2006), no. 4, 271–293.

[5] K. Boundy-Mills, *The yeast handbook, biodiversity and ecophysiology of yeasts* (2006), 67–100.

[6] C.P. Kurtzman and C.J. Robnett, *Identification and phylogeny of ascomycetous yeasts from analysis of nuclear large subunit (26s) ribosomal DNA partial sequences*, Antonie van Leeuwenhoek **73** (1998), 331–371.

[7] Elena S. Naumova, Nataliya N. Sukhotina, and Gennadi I. Naumov, *Molecular-genetic differentiation of the dairy yeast kluyveromyces lactis and its closest wild relatives*, FEMS Yeast Research **5** (2004), no. 3, 263–269.

[8] Raspor P, Zupan J, and Cadez N, *Validation of yeast identification by in silico RFLP*, Journal of Rapid Methods and Automation in Microbiology **15** (2007), 267–281.

[9] R.J. Roberts, T. Vincze, J. Posfai, and D. Macelis, *REBASE–a database for DNA restriction and modification: enzymes, genes and genomes*, Nucleic Acids Res **38** (2010), 234–236.

[10] O. Schmidt and U. Moreth, *Genetic studies on house rot fungi and a rapid diagnosis*, European Journal of Wood and Wood Products **56** (1998), no. 6, 421–425.

[11] G. Scorzetti, J.W. Fell, A. Fonseca, and A. Statzell-Tallman, *Systematics of basidiomycetous yeasts: a comparison of large subunit d1/d2 and internal transcribed spacer rDNA regions*, FEMS yeast research **2** (2002), 495–517.

[12] M. Vaneechoutte, R. Rossau, P. De Vos, M. Gillis, D. Janssens, et al., *Rapid identification of bacteria of the Comamonadaceae with amplified ribosomal DNA restriction analysis (ARDRA)*, FEMS Microbiol Lett **72** (1992), 227–233.

[13] Bockelmann W, Heller M, and Heller K, *Identification of yeasts of dairy origin by amplified ribosomal DNA restriction analysis (ARDRA)*, Int Dairy J **18** (2008), 1066–1071.

[14] Wei W, Lee IM, Davis RE, Suo X, and Zhao Y, *Automated RFLP pattern comparison and similarity coefficient calculation for rapid delineation of new and distinct phytoplasma 16sr subgroup lineages*, International Journal of Systematic and Evolutionary Microbiology **58** (2008), 2368–2377.

[15] T. White, T. Bruns, S. Lee, and J. Taylor, *Amplification and direct sequencing of fungal ribosomal RNA genes for phylogenetics* (1990), 315–322.

[16] D. Yarrow, *The yeasts, a taxonomic study. methods for the isolation, maintenance and identification of yeasts* (1998), 148–152.

Maximum likelihood pedigree reconstruction using integer programming

James Cussens
Dept of Computer Science & York Centre for Complex Systems Analysis
University of York, York, YO10 5DD, UK jc@cs.york.ac.uk

Abstract

Pedigrees are 'family trees' relating groups of individuals which can usefully be seen as Bayesian networks. The problem of finding a maximum likelihood pedigree from genotypic data is encoded as an integer linear programming problem. Two methods of ensuring that pedigrees are acyclic are considered. Results on obtaining maximum likelihood pedigrees relating 20, 46 and 59 individuals are presented. Running times for larger pedigrees depend strongly on the data used but generally compare well with those in the literature. Solving is particularly fast when allele frequency is uniform.

1 Introduction

The problem of finding the most probable pedigree ('family tree') for a group of related individuals, whether human or not, is often needed for paternity and family reunion cases [7]. Correctly specifying relationships is also needed for the proper application of genetic linkage analysis. In the literature the problem is often called *pedigree reconstruction* and we will also make use of this term.

A Bayesian approach is frequently taken where prior knowledge is combined with observed data to define a posterior probability for any given pedigree. Prior knowledge can include information such as known relationships, age and/or sex of some of the individuals and perhaps limits on the number of generations in the pedigree. In addition, probabilistic prior knowledge stating that, for example, very high levels of inbreeding are unlikely, can also be included [7, 13].

Data will be *genotypic* data for each individual under consideration. This data will be defined via a set of *marker loci* each specifying a position on a particular chromosome. The DNA sequence at such loci will vary between different individuals and so the marker can be thought of as a variable. The possible values of this variable are known as *alleles*. Chromosomes come in pairs, one inherited from the father and one from the mother, so there is a pair of allele values, called the *genotype*, for each locus. See [9] for further information. Data for pedigree reconstruction typically consists of genotypes for a number of marker loci: call this a *multi-locus genotype*. In the interests of brevity *multi-locus genotype* will often be abbreviated to just *genotype* in what follows.

As a result of its importance a number of computational techniques have been used for pedigree reconstruction including simple enumeration [7], simulated annealing [2, 11], MCMC [3] and dynamic programming [5]. However, it appears that constraint-based methods have yet to be used for pedigree reconstruction although [12, 10] apply weighted CSP and SAT techniques, respectively, to check the consistency of a given pedigree. Also a weighted MAX-SAT approach has been used for the problem of Bayesian network learning [6]; pedigree reconstruction can be seen as a special case of this (see Section 2).

In this paper pedigree reconstruction is cast as a instance of Bayesian network learning and integer linear programming (IP) is used to search for maximum likelihood Bayesian networks. The paper is structured as follows. In Section 2 a method for representing pedigrees as Bayesian networks (BNs) is given and the likelihood function for such BNs is analysed. Section 3 discusses IP encodings for pedigree reconstruction. Section 4 shows results for the most successful encoding found to date and the paper ends with conclusions and pointers to future work in Section 5.

2 Pedigrees as Bayesian networks

A Bayesian network (BN) is an acyclic directed graph whose nodes (V) represent random variables. (Such graphs are often called, somewhat imprecisely, directed acyclic graphs (DAGs).) If there is an arrow from node $u \in V$ to node $v \in V$ in the graph then u is said to be a *parent* of v. The parameters of a BN are conditional probability distributions for each node given its parents in the graph. Since the graph is acyclic the product of these conditional probability distributions defines a full joint probability distribution over all random variables represented in the graph.

There are a number of ways of representing pedigrees as BNs [9], but here, like [5], each node in the BN represents a known individual, or more precisely the multi-locus genotype of that individual. An arrow from u to v is a statement that u is the biological parent of v. It follows that no node may have more than two parents. A node with no parents represents a *founder*: an individual neither of whose parents are to be found amongst the individuals considered. A node with one parent represents an individual with exactly one known parent and a node with two parents represents an individual both of whose parents are known. Following [5] Hardy-Weinberg equilibrium will be assumed which implies that the multi-locus genotypes for founders will be probabilistically independent. In addition only complete genotypic data (for a given collection of markers) will be considered.

Following [5] let $\alpha_1(g_v|g_u)$ denote the probability that individual v has genotype g_v given that it has one known parent u with genotype g_u. Let $\alpha_2(g_v|g_u,g_w)$ be the probability that individual v has genotype g_v given that its has two known parents u and w with genotypes g_u and g_w. Let $\alpha_0(g_v)$ be the marginal probability that individual v has genotype g_v. Since for any particular pedigree reconstruction problem the observed genotype g_v for each individual v is fixed, the following notational abbreviation can be introduced.

$$\begin{aligned}\alpha(v,\{\}) &\stackrel{\text{def}}{=} \alpha_0(g_v)\\ \alpha(v,\{u\}) &\stackrel{\text{def}}{=} \alpha_1(g_v|g_u)\\ \alpha(v,\{u,w\}) &\stackrel{\text{def}}{=} \alpha_2(g_v|g_u,g_w)\end{aligned}$$

As noted by [5] due to the assumption of a complete sample, the likelihood of any candidate pedigree G decomposes into a product of conditional probabilities. (The likelihood of G is the probability of the observed data conditional on G being the true pedigree.) Letting $\mathrm{Pa}(v,G)$ denote the parents that $v \in V$ has in a pedigree G, this product can be represented as in (1) and so the log-likelihood, which is more convenient to work with, can be represented as in (2).

$$L(G) = \prod_{v \in V} \alpha(v, \mathrm{Pa}(v,G)) \tag{1}$$

$$l(G) = \log L(G) = \sum_{v \in V} \log \alpha(v, \mathrm{Pa}(v,G)) \tag{2}$$

The problem of maximum likelihood pedigree reconstruction is that of finding G such that $l(G)$ is maximised.

3 An integer programming encoding for ML pedigree reconstruction

In this section a method of encoding the maximum likelihood pedigree reconstruction problem as an *integer programming* problem is presented. A first step towards the encoding is the simple observation

that a pedigree specifies the parents, if any, of each individual. So given an individual v, a parent set W and a pedigree G it is determined whether v has parents W in G. This is formalised using the functions $I(W \to v)$ defined in (3), where W is implicitly restricted to be: $W \subseteq V \setminus \{v\}, |W| \leq 2$. This restriction on W will be assumed throughout to simplify the presentation.

$$I(W \to v)(G) = \begin{cases} 1 & \text{if } v \text{ has parents } W \text{ in } G \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

The log-likelihood (2) of any pedigree can now be rewritten as in (4).

$$l(G) = \sum_{v,W} \log \alpha(v,W) I(W \to v)(G) \tag{4}$$

Note that in (4), $I(W \to v)(G)$ only takes the value 1 when $W = \mathrm{Pa}(v,G)$. For any other value of W, $I(W \to v)(G) = 0$. The key to the integer programming encoding is to view the $I(W \to v)$ as *binary variables*. Any particular pedigree G determines a joint instantiation of these binary variables, setting exactly $|V| = n$ of these binary variables to 1 and all others to 0. However, most joint instantiations of the $I(W \to v)$ do not correspond to any pedigree. With this in mind the maximum likelihood pedigree reconstruction problem can be reformulated as in (5).

Find an instantiation of the $I(W \to v)$ which maximises:
$\sum_{v,W} \log \alpha(v,W) I(W \to v)$ (5)
subject to the $I(W \to v)$ representing a valid pedigree.

Because the variables in (5) are integer-valued and the *objective function* $\sum_{v,W} \log \alpha(v,W) I(W \to v)$ is *linear* in these variables this is an *integer linear programming* problem—as long as the necessary constraints on the $I(W \to v)$ can be expressed as linear equations and inequalities. In the following subsections it will be shown that this is indeed the case. ('Integer linear programming' will continue to be abbreviated to 'integer programming' throughout.)

3.1 Constraints

The most basic constraint on the $I(W \to v)$ is that each individual v has exactly one parentset. This can be expressed by n linear equations:

$$\forall v : \sum_{W} I(W \to v) = 1 \tag{6}$$

Any instantiation of the $I(W \to v)$ satisfying the equations given by (6) will represent a graph, but the graph may contain directed cycles. To rule out cycles auxiliary variables are required. There are many ways of ruling out cycles. The following sections present two possible approaches.

3.1.1 Ruling out cycles with a total order

For each distinct pair of individuals u, v a binary variable $I(u < v)$ is created. $I(u < v) = 1$ indicates that u is older than v (u's birth was before that of v). Without loss of generality it can be assumed that no two distinct individuals are of exactly the same age, so that exactly one is older than the other which makes the order on the ages of the individuals a *total order*. This is expressed using the following $n(n-1)/2$ equations:

$$\forall u, v : I(u < v) + I(v < u) = 1 \tag{7}$$

Note that (7) means that half of the $I(u < v)$ variables are effectively redundant. However, it is more convenient to work with the full complement of $I(u < v)$ variables. This does not introduce inefficiency since the IP solver will detect this redundancy and simplify the representation of constraints internally. Note also that in (7) the obvious requirement that $u \neq v$ has not been explicitly stated. This notational convenience will be used throughout: whenever a constraint depends on more than one individual these individuals will be distinct, but this will not be made explicit.

The total order must be transitive (if $u < v$, $v < w$ then $u < w$). This can be represented by the following $n(n-1)(n-2)/3$ constraints:

$$\forall u, v, w : 1 \leq I(u < v) + I(v < w) + I(w < u) \leq 2 \tag{8}$$

Finally, the constraint that parents are older than their children needs to be expressed:

$$\forall u, v : I(u < v) \geq \sum_{W : u \in W} I(W \rightarrow v) \tag{9}$$

To see that (9) expresses this relation, suppose that u is a parent of v. In that case exactly one of the $I(W \rightarrow v)$ on the RHS of (9) is 1 and thus the RHS is 1. To satisfy the inequality $I(u < v)$ must also be 1.

3.1.2 Ruling out cycles with generation variables

An alternative approach associates a *generation number* with each individual in a pedigree. For a founder this number is zero. For any other individual the generation number is the length of the longest path from a founder to the individual. Let m denote the maximum generation number which is a value set by the user to reflect any prior knowledge. In the absence of such knowledge m takes its maximal value $n-1$.

Let $\text{gen}(v)$ denote the generation number of individual v. It is not difficult to see that if u is a parent of v then $\text{gen}(v) \geq \text{gen}(u) + 1$. This leads to the following set of $n(n-1)$ constraints:

$$\forall u, v : \text{gen}(v) - \text{gen}(u) \geq -m + (m+1) \sum_{W : u \in W} I(W \rightarrow v) \tag{10}$$

To understand (10) observe that if u is not a parent of v then the sum on the RHS is zero and the constraint becomes vacuous. If u is a parent of v then the sum is 1 and so the entire RHS becomes 1, effecting the desired constraint. To see that (10) suffices to rule out cycles note that if w is an ancestor of v then $\text{gen}(w) < \text{gen}(v)$. If a cycle obtains, at least two individuals are their own ancestors and the obvious inconsistency arises. Thus as long as (10) is respected no cycles are possible.

Note that (10) does not fix the values of the $\text{gen}(v)$ variables to their correct values. To see this, suppose that n were, say, 10 and m set to 9, reflecting an absence of domain knowledge. Suppose that an optimal pedigree were found with the highest valued generation variable having value 5. It would be possible to increase each generation variable by one without violating (10). If our only concern is to rule out cycles this is not a problem, but if it is necessary to ensure that the $\text{gen}(v)$ variables take on their correct values then additional constraints placing upper bounds on $\text{gen}(v)$ are required.

3.1.3 Ensuring sex-consistency

Any instantiation of the $I(W \rightarrow v)$ variables satisfying constraints (6–9) or alternatively (10) will specify an *acyclic* directed graph where each vertex has at most two parents, but not all such graphs represent pedigrees. It is also necessary to ensure that a sex can be assigned to each individual in a consistent manner. An example of an acyclic directed graph where this is *not* the case can be seen in Fig 1. Note that this example was also given by [5]. Call such pedigrees *sex-inconsistent*.

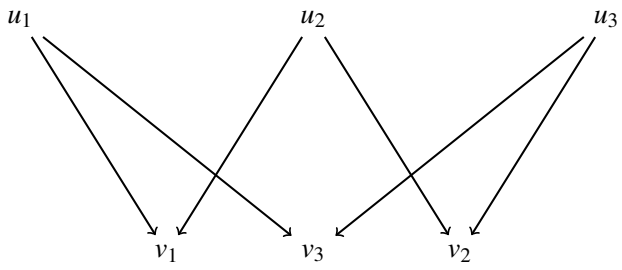


Figure 1: A sex-inconsistent pedigree. It is not possible to consistently assign a sex to each individual.

To rule out sex-inconsistent pedigrees another n auxiliary binary variables $I_f(v)$ are created. $I_f(v) = 1$ states that individual v is a female. Constraint (11) states that if an individual v has two parents at most one is female and constraint (12) states that at least one is female. Note that in both cases, if $I(\{u,w\} \to v) = 0$ then the constraints are vacuously satisfied.

$$\forall u,v,w : I(\{u,w\} \to v) + I_f(u) + I_f(w) \leq 2 \tag{11}$$

$$\forall u,v,w : I(\{u,w\} \to v) - I_f(u) - I_f(w) \leq 0 \tag{12}$$

With all these constraints in place, the maximum likelihood pedigree reconstruction problem can be restated as follows:

$$\begin{array}{l} \text{Maximise: } \sum_{v,W} \log \alpha(v,W) I(W \to v) \\ \text{subject either to (6–9,11,12) or (6,10,11,12).} \end{array} \tag{13}$$

4 Results

All results shown here were produced using a 3GHz dual-core Linux machine with the Gurobi IP solver [8]. A number of tests (not reported here) have also been done with SCIP [1] which produced respectable running times which were nonetheless clearly longer than those produced by Gurobi (which automatically parallelises solving on multi-core machines).

Only synthetic genotype data sampled from test pedigrees has been used. This sampling process mimics the inheritance of genotypes from parent to offspring, which is probabilistic, and thus different datasets will be sampled from a given pedigree depending on which random seed is being used. Data was created in this way using the the C++ program `pedsim` used to produce the results in [5]. `pedsim` was also used to compute the log conditional probabilities $\log \alpha(v,W)$ from each of these synthetic datasets, and then to remove $\log \alpha(v,W)$ scores where there exists a higher $\log \alpha(v,W')$ score with $W' \subset W$. Such scores and their accompanying $I(W \to v)$ variables are not needed since in such a case v would never have parents W in the maximum likelihood pedigree. A Python script (available on request from the author) was used to read in the (filtered) $\log \alpha(v,W)$ scores from `pedsim`. Gurobi's Python interface was then used to define and solve the optimisation problem (13). The following sections present results for a number of synthetic pedigree reconstruction problems.

4.1 Pedigree reconstruction for 20 individuals using a total order

The data for this experiment was chosen to be the same (modulo sampling variation) as that of one of Cowell's [5]. The same software (`pedsim`) was used to generate genotypic data from one of Cowell's

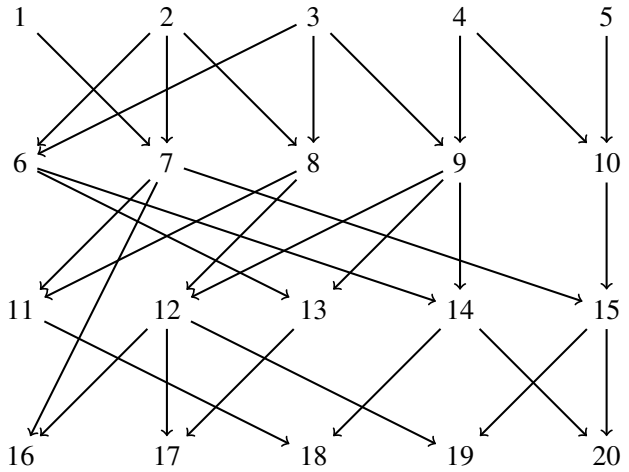


Figure 2: Pedigree of 20 individuals. This pedigree is [5, Fig 3].

test pedigrees (Fig 2). Ten marker loci were used. Although the data is synthetic, the markers correspond to real ones. Allele values and founder allele frequencies were taken from [4].

A thousand datasets were sampled and the time taken to find a maximum likelihood sex-consistent pedigree in each case was recorded. Total order constraints were used in each case. The mean solving time was 0.65 seconds with the slowest run taking 17.3 seconds. Only 8 runs took longer than 4 seconds.

4.2 Pedigree reconstruction for 46 individuals using a total order

An experiment identical to that described in Section 4.1 except using a test pedigree of 46 individuals was then carried out. This pedigree was created by editing the source of Cowell's `pedsim` program and is displayed in Fig 3. Only 10 runs were attempted since solving time is substantially higher than for the 20 individuals case. The solving times for the first 9 runs were, in decreasing order: 20,566s, 7,470s, 4,266s, 1,175s, 722s, 669s, 115s, 59s and 51s. The 10th run was abandoned after failing to identify the maximum likelihood pedigree after 44,135s. The very high variation in solving times is notable. Most datasets produced optimisation problems which could be solved in a reasonable time, but in some cases solving was unacceptably lengthy. Problems of this size are too large for the approach of [5].

Two further experiments were conducted for the 46-individual pedigree. In the first an extra constraint specifying that there must be at least 20 founders in the pedigree was added. 80 runs were done with this extra constraint. The mean solving time was 18 seconds with 75% of runs below 20 seconds and the slowest taking 128 seconds. In the second experiment a more reasonable constraint on founders was used: that the number of founders was between 10 and 20. 100 runs were done with this constraint. The mean solving time was 34s, with 75% within 30s and the slowest taking 613 seconds.

4.3 Pedigree reconstruction for 46 individuals using generation variables

Although using a total order to rule out cycles in pedigrees produced acceptable results for small numbers of individuals, it is clear that for bigger problems finding a maximum likelihood pedigree is unacceptably slow. Fortunately, switching to using generation variables to rule out cycles as described in Section 3.1.2 results in a significant speed up.

In an initial experiment 100 synthetic datasets were generated from the 46-individual pedigree shown in Fig 3. 19 runs completed successfully, taking a mean time of 432 seconds, but a median time of only

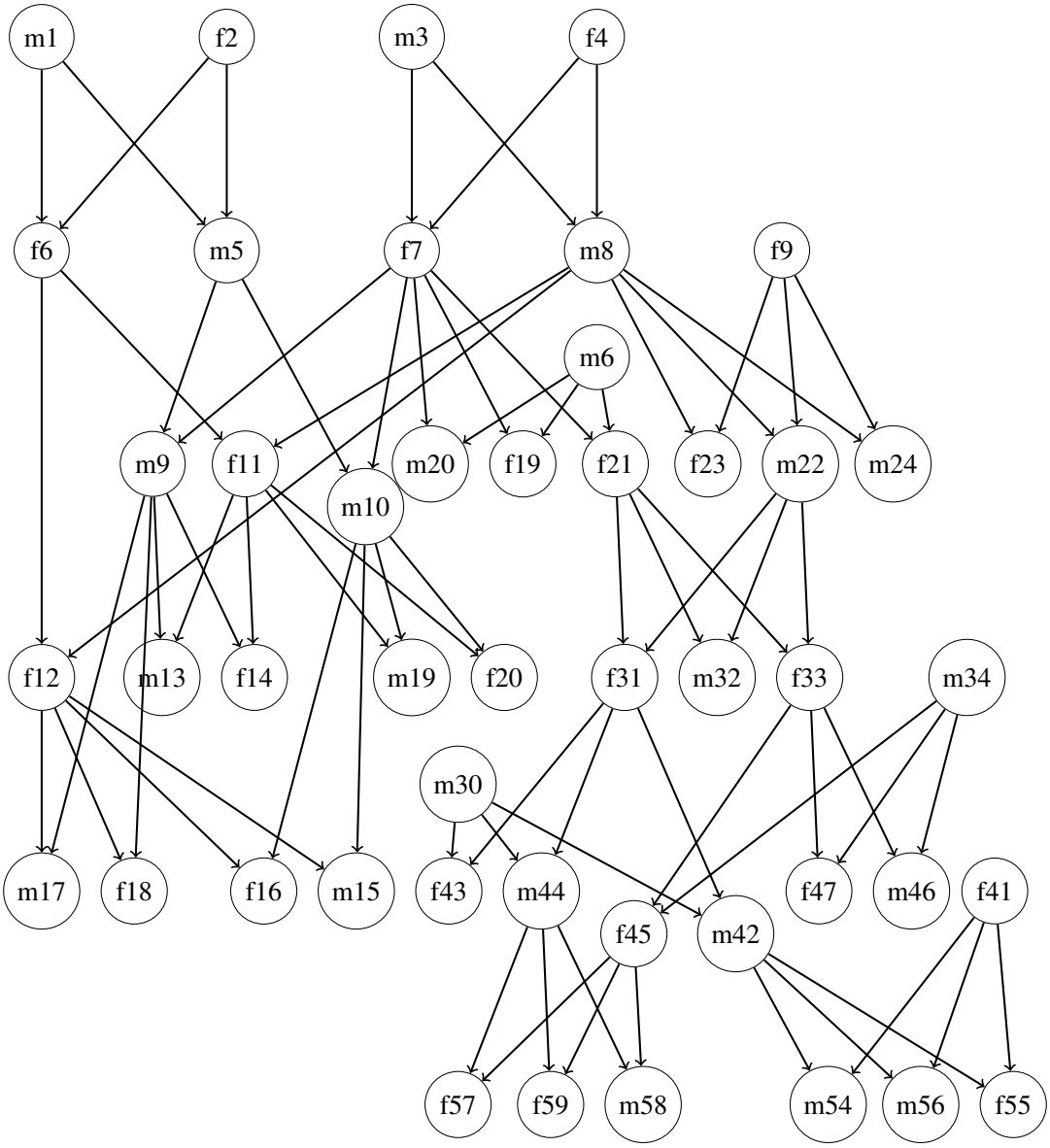


Figure 3: Test pedigree of 46 individuals.

2.2 seconds. The distribution of solving times for these 19 runs was thus highly skewed with the five longest runs taking 7349, 810, 20, 9 and 4 seconds and each of the 6 quickest taking less than a second. However, on the 20th run, Gurobi ran out of memory.

This problem could probably be addressed by instructing Gurobi to use the hard disk when (RAM) memory is exhausted, but this would lead to much slower solving. Instead an extra constraint was added in the hope of both speeding up solving and reducing memory consumption. This constraint stated that in each pedigree there is at least one founder. Since this is always true (due to the acyclicity of pedigrees) a maximum likelihood pedigree will still be returned, but hopefully more quickly. This *founder constraint*

LB	Time in seconds	Likelihood
0	3189.75858617	-6.3680054000e+02
1	1050.95497108	-6.3680054000e+02
2	1695.93844795	-6.3680054000e+02
3	2350.830338	-6.3680054000e+02
4	1220.98797202	-6.3680054000e+02
5	431.202931166	-6.3680054000e+02
6	246.708929062	-6.3680054000e+02
7	63.2229361534	-6.3680054000e+02
8	3.00327396393	-6.3680054000e+02
9	0.523219823837	-6.3933824000e+02
10	0.293282032013	-6.5144025000e+02

Table 1: Solving times for different lower bounds on the number of founders. 'Likelihood' is the likelihood of a maximum likelihood pedigree with the given bound.

is formally expressed as follows:

$$\sum_{v} I(\{\} \rightarrow v) \geq 1 \tag{14}$$

With the founder constraint added 100 runs were attempted (i.e. 100 datasets were simulated and maximum likelihood pedigrees were found for each) and all completed successfully. The mean solving time was 195 seconds and the median was 3.8 seconds. As usual there was therefore a highly skewed distribution of solving times with the ten slowest runs taking the following number of seconds: 8181, 7361, 1638, 830, 309, 249, 127, 118, 110 and 45.

To investigate the effect of increasing the lower bound on the number of founders above one, a particular dataset simulated from the pedigree in Fig 3 was used. This dataset was chosen since it is one of the 'harder' ones resulting in reasonably long solving times. A maximum likelihood pedigree for this data is shown in Fig 4.

As Table 1 makes clear, increasing the lower bound on the number of founders makes a big difference in the time it takes to find a maximum likelihood pedigree. Note, from Fig 4, that at least one maximum likelihood pedigree has 8 founders (m1, f2, m3, f4, f9, m6, f41 and m30). Using 8 as a lower bound on the number of founders solving takes only 3 seconds. Higher lower bounds reduce the solving time further but the pedigree returned is no longer of maximum likelihood as shown by the third column in Table 1. Importantly, raising the lower bound from 0 (which amounts to removing the constraint) to 1 reduces the solving time drastically. Also, interestingly, using lower bounds of 2, 3 or 4 actually increases the solving time compared to a lower bound of 1, but all are still quicker than using no lower bound.

4.4 Pedigree reconstruction for 59 individuals using generation variables

An experiment was done to provide as direct a comparison as possible with Almudevar's simulated annealing approach [2]. Datasets were simulated from Almudevar's 59 individual pedigree [2, Fig 2] and as in that paper ten marker loci were used each with 8 equally frequent alleles. Generation variables were used to rule out cycles and the (always admissible) constraint that there is at least one founder was used.

Maximum likelihood pedigrees were obtained from 1000 simulated datasets. The mean solving time was 0.44846 seconds, the median 0.2350 and 75% of runs completed within 0.43860 seconds. A few much longer runs occurred, with one of length 15.26 seconds. The distribution of the 1000 solving

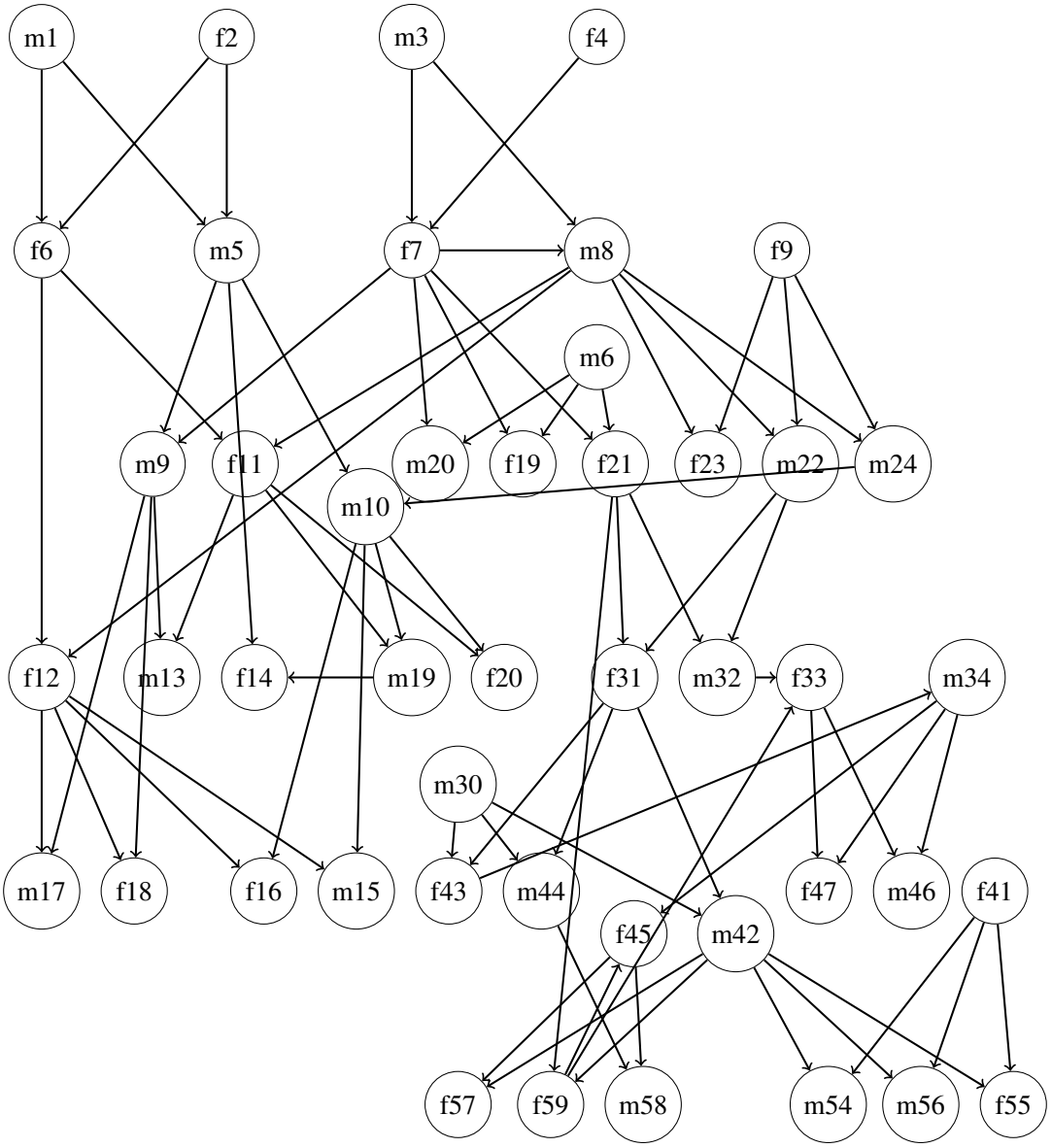


Figure 4: A maximum likelihood pedigree of 46 individuals for a particular dataset simulated from the pedigree in Fig 3

times is shown as a box plot in Fig 5 It is notable that solving times are substantially faster on this 59 individual pedigree than for the 46 individual pedigree previously discussed. This is most probably due to the assumption of *equally frequent* (i.e. equally probable) alleles for each of the ten marker loci. This means that genotypic data is more informative than is the case where the distribution is (realistically) skewed as is the case with Cowell's ten marker loci data (Markus Riester, personal communication). Comparing running times to Almudevar [2], there it is stated that for the quickest configuration the time taken for the simulated annealing algorithm to converge "was approximately 6.6 min using a standard personal computer". Note also that simulated annealing does not guarantee that the pedigree found has maximal likelihood.

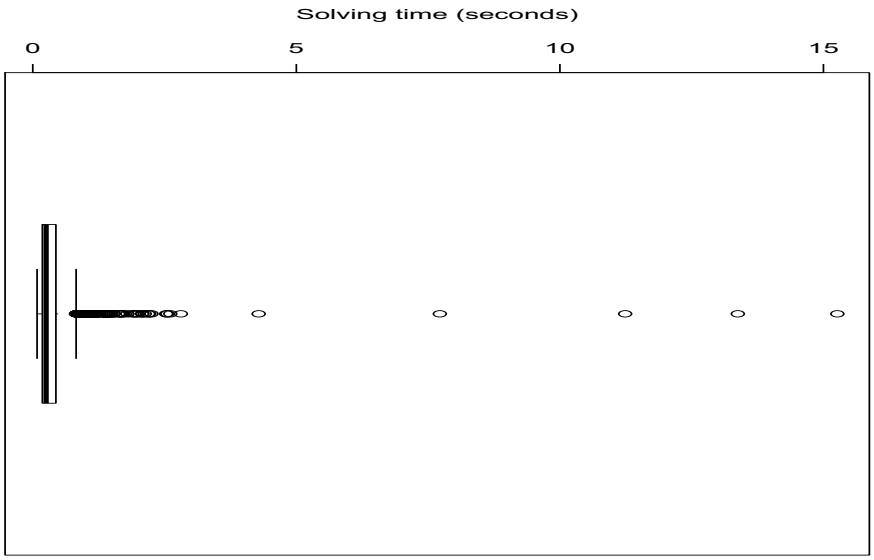


Figure 5: Boxplot representation of the distribution of 1000 maximum likelihood pedigree reconstruction solving times for datasets simulated from Almudevar's 59 individual pedigree.

5 Conclusions and future work

Results on finding pedigrees which are guaranteed to have maximal likelihood using IP have been presented in this paper. The results compare favourably with others in the literature as regards scalability, speed and ensuring sex-consistency (however Riester *et al* [11] report on reconstructing pedigrees from thousands of individuals using simulated annealing).

In this paper the focus is on how best to do maximum likelihood pedigree reconstruction, but there is also the entirely distinct question of whether maximising likelihood is the best way to reconstruct pedigrees. With large amounts of data maximum likelihood usually provides a reasonable estimate of the true pedigree. So, for example, comparing the 1000 maximum likelihood pedigrees discussed in Section 4.4 to the true data-generating pedigree [2, Fig 2] we find that 533 of them are exactly equal to the true pedigree. The full distribution of parent assignment errors is shown in Fig 6.

Nonetheless the alternative Bayesian approach allows the incorporation of domain knowledge and allows a principled way of quantifying the uncertainty inherent in pedigree reconstruction (*model uncertainty*). In an IP formulation the prior distribution is represented by incorporating extra terms in the objective function. Model uncertainty is addressed by finding many distinct high probability pedigrees rather than returning a single one. Work is currently underway on such a Bayesian approach.

Acknowledgements

Thanks to Robert Cowell, Nuala Sheehan and Markus Riester for useful expertise on pedigrees. Many thanks to Robert Cowell for supplying the `pedsim` software. Thanks also to the anonymous reviewers for their comments and suggestions.

References

[1] Tobias Achterberg. *Constraint Integer Programming*. PhD thesis, TU Berlin, July 2007.

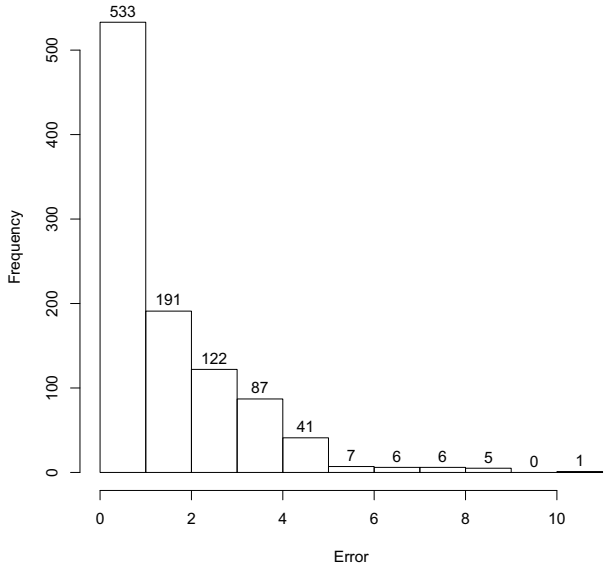


Figure 6: Distribution of parent assignment errors for 1000 pedigrees constructed from data generated from the 'true' pedigree [2, Fig 2]

[2] Anthony Almudevar. A simulated annealing algorithm for maximum likelihood pedigree reconstruction. *Theoretical Population Biology*, 63:63–75, 2003.

[3] Anthony Almudevar. A graphical approach to relatedness inference. *Theoretical Population Biology*, 71:213–229, 2007.

[4] J.M. Butler, R. Schoske, P.M. Vallone, J.W. Redman, and M.C. Kline. Allele frequencies for 15 autosomal STR loci on U.S. Caucasian, African American and Hispanic populations. *Journal of Forensic Sciences*, 48(4), 2003.

[5] Robert G. Cowell. Efficient maximum likelihood pedigree reconstruction. *Theoretical Popluation Biology*, 76(4):285–291, December 2009.

[6] James Cussens. Bayesian network learning by compiling to weighted MAX-SAT. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence (UAI 2008)*, pages 105–112, Helsinki, 2008. AUAI Press.

[7] T. Egeland, P. F. Mostad, B. Mevåg, and M. Stenersen. Beyond traditional paternity and identification cases: Selecting the most probable pedigree. *Forensic Science International*, 110:47–59, 2000.

[8] Gurobi Optimization Inc. *Gurobi Optimizer Reference Manual*, 2010. Version 3.0.

[9] Steffen L. Lauritzen and Nuala A. Sheehan. Graphical models for genetic analyses. *Statistical Science*, 18(4):489–514, 2003.

[10] Panagiotis Manolios, Marc Galceran Oms, and Sergi Oliva Valls. Checking pedigree consistency with PCS. In *Thirteenth International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2007)*, number 4424 in LNCS, pages 339–342. Springer, 2007.

[11] Markus Riester, Peter F. Stadler, and Konstantin Klemm. FRANz: reconstruction of wild multi-generation pedigrees. *Bioinformatics*, 25(16):2134–2139, 2009.

[12] Marti Sanchez, Simon de Givry, and Thomas Schiex. Mendelian error detection in complex pedigrees using weighted constraint satisfaction techniques. *Constraints*, 13:130–154, 2008.

[13] N A Sheehan and T Egeland. Structured incorporation of prior information in relationship indentification problems. *Annals of Human Genetics*, 71:501–518, 2007.

Optimal haplotype reconstruction in half-sib families

Aurélie Favier, Jean-Michel Elsen, Simon de Givry, Andrés Legarra
INRA, Toulouse, France
{afavier,jean-michel.elsen,degivry,andres.legarra}@toulouse.inra.fr

Abstract

In the goal of genetic improvement of livestock by marker assisted selection, we aim at reconstructing the haplotypes of sires from their offspring. We reformulated this problem into a binary weighted constraint satisfaction problem. Our results showed these problems have a small treewidth and can be solved optimally, improving haplotype reconstruction compared to previous approaches especially for medium-size half-sib families.

1 Introduction

Haplotype-based analysis plays an important role in genetics, including study of a population, association mapping, and linkage / association analysis. However haplotypes of diploid individuals cannot easily be acquired and only unphased genotype data can be obtained through application of experimental techniques. It is therefore necessary to propose efficient haplotype reconstruction methods from genotype data, able to cope with a large number of dense markers such as *single nucleotide polymorphisms* (SNPs). Di-allelic SNPs are mutations at single nucleotide positions taking two values (e.g., *allele A* or *B*), and are the most prevalent sequence variations between individuals of all species. The combination of marker alleles on a single chromosome is called a *haplotype*. The combination of unordered pairs of alleles on homologous chromosomes is called a *genotype*.

There are two main sources of genotype data for haplotype inference: coming either from a population of unrelated individuals, or from a *pedigree*, which gives the parental relationships between individuals [13]. We are interested in the latter case where large pedigrees of livestock are available. Two categories of methods exist: statistical methods [1, 12, 18, 8, 17, 7] and rule-based methods [10, 14, 6]. The latter often assume zero recombinants or are more appropriate for pedigree data with a small expected number of recombinations, such as high density marker data in a short chromosomal region. The problem is NP-hard, even in the case of tree pedigree and no missing data [6]. Exact (i.e., complete) methods [1, 6, 8] have their worst-case time complexity exponential in the minimum between the number of individuals and the number of markers. Another option is the use of approximate methods such as greedy and iterative search methods [10, 14, 18, 7] or Monte-Carlo methods [17].

There is a need in animal genetics for exact and fast methods for haplotype reconstruction: current data in cattle genetics consists of thousands of individuals and tens of thousands of markers. We propose a new statistical exact method for haplotype inference from genotypes on such large pedigree data under the Mendelian laws of inheritance and the probability of recombination events.

Mendel's laws involved are very simple: there is one marker allele coming from each of the parents, and, for a given marker, the copy that the parent transmits to its progeny is picked up at random. So, in some cases the determination of allele origin is very simple. For example, if a father/*sire* has an *homozygous* (i.e., same alleles) genotype AA at one marker, and his son has an *heterozygous* (i.e., different alleles) genotype AB, then with certainty allele A in the son came from the father.

The second law involved is the probability of *recombination events*. Recombination events are produced by meiosis, which is a complicated biological phenomenon. However, *genetic maps* have been built that condense the probability of recombination (or recombination fraction) between any two points in a chromosome, aka two *loci*, into a linear metric, usually the Haldane's mapping function, assuming no interference in the formation of crossing-overs. For instance, if the mother/*dam* haplotypes on

three loci are AA and BB, then the probability of the transmitted AB gamete to her son is simply the recombination fraction between loci 1 and 2 divided by 2.

Now, haplotype inference is explained on a simple example. Assume a family of two parents and one offspring. The genotype of the father for two SNPs is AB AB (recall that the allele order in a pair doesn't matter), and the genotype of the mother is BB AA. The mother haplotypes are trivial (both haplotypes are BA); however the father has two possible sets of haplotypes (AA and BB, or AB and BA). Assume that one son has genotype AB AB. For this son, B in the first locus and A in the second came from the mother (because there is no other possibility) and they form a first haplotype BA. Thus, AB constitutes the other haplotype that came from the father. Now, if the recombination fraction between the two loci is less than 0.5 (roughly if they are on the same chromosome), probably the father transmitted haplotype AB with no recombination; thus, its haplotypes are AB and BA.

In Section 2, we present an efficient method to reconstruct the haplotypes of the sire from the information of its genotype and of its offspring genotypes in a *half-sib family* (each son has a different dam). This particular pedigree is common in livestock genetics for marker assisted selection. Further, once the sire haplotypes are reconstructed, and conditionally to this configuration, the haplotypes of its sons are easy to compute [7]. Assuming *linkage equilibrium* (i.e., random association of alleles at two or more loci) and equal allele frequencies at every locus, our method reformulates the likelihood of genotype data in a compact way, resulting in a binary weighted constraint satisfaction problem [11], which can be maximized later by a systematic search method or by a dynamic programming algorithm, exploiting the small treewidth of the resulting instances.

Section 3 gives experimental results on simulated and real datasets. In this study, we assumed no missing data (except the dams) and no erroneous genotypes. However, we could impute missing sire genotype data from its offspring, removing beforehand Mendelian errors [15].

2 Method

Assume a single half-sib family, the sire and its n descendants are genotyped at L loci but not the dams. Let $\mathbf{M}$ be a matrix such that $M^i_{l,1}, M^i_{l,2}$ are the observed genotype information of individual i ($i \in \{0,1\ldots,n\}$, with index 0 for the sire) at locus l ($l \in \{1,\ldots,L\}$) for its two alleles ($M^i_{l,j} \in \{A,B\}, j \in \{1,2\}$) with an arbitrary order. For convenience, in the following examples, the genotype of an individual i is given by a list of pairs of alleles, e.g., $\mathbf{M}^i =$ AB AA BA means $M^i_{1,1} = A, M^i_{1,2} = B, M^i_{2,1} = A, M^i_{2,2} = A, M^i_{3,1} = B, M^i_{3,2} = A$.

Let now define vector $\mathbf{h}$ ($h_l \in \{-1,1\}, l \in \{1,\ldots,L\}$) as the indicator of allele origin for the sire haplotypes. h_l has two possible states: $h_l = 1$ (resp. $h_l = -1$) if the first haplotype has allele $M^0_{l,1}$ (resp. $M^0_{l,2}$) and the second haplotype has allele $M^0_{l,2}$ (resp. $M^0_{l,1}$) at locus l. For instance, a sire genotype observed at three loci such that $\mathbf{M}^0 =$ AB AA BA and $\mathbf{h} = (1,1,-1)$ implies that the first sire haplotype is AAA and the second one is BAB. The problem to solve is to find the most probable assignment of $\mathbf{h}$ given the observed genotypes. Note that the assignment of h_l with homozygous sire locus l does not matter, it is set to 1.

Instead of using the observed genotypes in our probabilistic model directly, we will use an intermediate data that is sufficient to model meiosis events. Let $\mathbf{T}$ be a matrix such that indicator variable T^i_l, called the *transmission value*, defines the origin of the paternal allele at locus l ($l \in \{1,\ldots,L\}$) in the i-th descendant ($i \in \{1\ldots,n\}$). This origin is referred to the genotype information in the sire ($M^0_{l,1}, M^0_{l,2}$), not to its haplotypes. T^i_l has three possible states: $T^i_l = 1$ (resp. $T^i_l = -1$) if the paternal allele of the i-th descendant comes from the first allele $M^0_{l,1}$ (resp. the second allele $M^0_{l,2}$), or $T^i_l = \star$ if the origin of the paternal allele is unknown. Let $\mathbf{T}^i$ be the transmission vector $(T^i_1,\ldots,T^i_L)$.

Variable T_l^i is known with certainty (i.e., T_l^i is -1 or 1) if and only if the i-th descendant is homozygous and the sire is heterozygous at locus l. Otherwise, T_l^i is unknown ($T_l^i = \star$). Remember that the dam genotypes are assumed to be unknown. For example, if the descendant is AA and the sire BA, it is necessary that A in the descendant came from the second allele of the sire, so $T_l^i = -1$.

A locus l such that $T_l^i \neq \star$ is called an *informative locus* for the i-th descendant. A *preceding informative locus k of l* is the first informative locus found in the order from $l-1$ to 1. The set of pairs of consecutive informative loci is composed of all the pairs of informative loci with their corresponding preceding informative locus.

Example 1. *Consider a sire with three sons from three different dams. Only the sire and the sons are genotyped on seven loci such that* $\boldsymbol{M}$ *is given by:*

$\boldsymbol{M}^0$:	*AB*	*BB*	*AA*	*BA*	*BA*	*AA*	*AB*
$\boldsymbol{M}^1$:	*BB*	*BA*	*AA*	*AA*	*BB*	*AB*	*BB*
$\boldsymbol{M}^2$:	*BA*	*BB*	*AB*	*AA*	*BB*	*AA*	*AA*
$\boldsymbol{M}^3$:	*AA*	*BB*	*AA*	*AB*	*AA*	*AB*	*AA*

Construction of transmission vectors. *The sire is homozygous at loci* 2,3 *and* 6*, so for these loci the transmission value is* $\star$ *for each son. For the other loci, the sire is heterozygous, so we study the genotypes of the sons to complete the transmission vectors. We detail for son* 2*. At locus* 1*, the son is heterozygous as the sire so we do not know with certainty which of the alleles (*$M^0_{1,1}$ *or* $M^0_{1,2}$*) was transmitted:* $T_1^2 = \star$*. At locus* 4*, the son is AA and the sire is BA, so it is certain that the sire transmits the second allele (*$M^0_{4,2}$*) :* $T_4^2 = -1$*. At locus* 5*, the son is homozygous BB and the sire is BA, so it is certain that the sire transmits the first allele (*$M^0_{5,1}$*) :* $T_5^2 = 1$*. It is the same reasoning at locus 7.*
Finally matrix $\boldsymbol{T}$ *is equal to:*

$\boldsymbol{T}^1$:	*-1*	$\star$	$\star$	*-1*	*1*	$\star$	*-1*
$\boldsymbol{T}^2$:	$\star$	$\star$	$\star$	*-1*	*1*	$\star$	*1*
$\boldsymbol{T}^3$:	*1*	$\star$	$\star$	$\star$	*-1*	$\star$	*1*

The set of informative loci of son 1 *is* $\{1,4,5,7\}$ *and its set of pairs of consecutive informative loci is* $\{(1,4),(4,5),(5,7)\}$*. It is* $\{4,5,7\}$ *(resp.* $\{1,5,7\}$*) and* $\{(4,5),(5,7)\}$ *(reps.* $\{(1,5),(5,7)\}$*) for son* 2 *(resp. son* 3*).*

To summarize, $\mathbf{h}$ are the decision variables and $\mathbf{T}$ the observations used in our model. The posterior probability of the haplotypes is given by

$$p(\mathbf{h}|\mathbf{T}) = \frac{p(\mathbf{T}|\mathbf{h}).p(\mathbf{h})}{\sum_{\mathbf{h}'} p(\mathbf{T}|\mathbf{h}').p(\mathbf{h}')}$$

In the absence of prior information for $\mathbf{h}$, i.e., assuming *linkage equilibrium*, $p(\mathbf{h}|\mathbf{T}) \propto p(\mathbf{T}|\mathbf{h})$ and the most likely haplotype configuration is the one that maximizes $p(\mathbf{T}|\mathbf{h})$.

Because meiosis events producing each descendant are independent,

$$p(\mathbf{T}|\mathbf{h}) = \prod_{i=1}^{n} p(\mathbf{T}^i|\mathbf{h})$$

Applying the chain rule, we obtain also

$$p(\mathbf{T}^i|\mathbf{h}) = \prod_{i=1}^{n} p(T_1^i|\mathbf{h}).p(T_2^i|\mathbf{h},T_1^i).p(T_3^i|\mathbf{h},T_1^i,T_2^i)\ldots p(T_L^i|\mathbf{h},T_1^i,\ldots,T_{L-1}^i)$$

These probabilities are defined in an iterative way starting from $l=1$. For the first position, $p(T_1^i|\mathbf{h}) = 0.5$ if T_1^i equals to either -1 or 1 (i.e., $p(\mathbf{T}|\mathbf{h}) = p(\mathbf{T}|-\mathbf{h})$), and $p(T_1^i|\mathbf{h}) = 1$ if $T_1^i = \star$. The same applies for a series of $\star$'s up to the first $T_l^i \neq \star$. For any next position, two cases can be distinguished. For $T_l^i = \star$, $p(T_l^i|\mathbf{h}, T_1^i, \ldots, T_{l-1}^i) = 1$ because it is a complete set of events. For $T_l^i \neq \star$, assuming *no interference in the formation of crossing-overs* and *equal allele frequencies at each locus*, only the current informative locus l and the preceding informative locus k of l are used, $p(T_l^i|\mathbf{h}, T_1^i, \ldots, T_{l-1}^i) = p(T_l^i|h_k, h_l, T_k^i)$ (if l is the first informative locus, $p(T_l^i|\mathbf{h}, T_1^i, \ldots, T_{l-1}^i) = 0.5$). This is so because, assuming independence of crossing-over, the probability of recombination between k and l does not depend on the presence or not of previous recombinations between 1 and k. And because any $\star$ between $k+1$ and $l-1$ does not modify the likelihood, assuming all SNPs have equal allele frequencies, and so, transmitted alleles from the dams to their sons do not matter. Thus, only informative loci (transmission values) in $\mathbf{T}^i$ are used.

Let r_{kl} denote the recombination fraction between k and l, obtained by the Haldane mapping function from the known marker map ($r_{kl} \in [0, 0.5]$). A pair of alleles placed on the same chromosome in the sire at locus k and l will be transmitted together (no recombination) with a probability $1 - r_{kl}$; the opposite (transmitted alleles come from a recombination between homologous chromosomes) occurs with frequency r_{kl}.

Thus, $p(T_l^i|h_k, h_l, T_k^i) = (1 - r_{kl})$ in two cases: if $T_l^i = T_k^i$ and $h_l = h_k$, or if $T_l^i \neq T_k^i$ and $h_l \neq h_k$. Both indicate the same sire haplotype origins for these two loci in the i-th descendant. In any other case (different origins), $p(T_l^i|h_k, h_l, T_k^i) = r_{kl}$. An algebraic form of $p(T_l^i|h_k, h_l, T_k^i)$ is $r_{kl}^{1-a} \times (1 - r_{kl})^a$, where a measures the same origin ($a = 1$) or not ($a = 0$). We have $a = a_{kl}^i(\mathbf{h}) = \frac{1}{2} + \frac{1}{2}\frac{h_k h_l}{T_k^i T_l^i}$.

The log-likelihood of $\mathbf{h}$ can be expressed as

$$\begin{aligned} V =& \log\left[p(\mathbf{T}|\mathbf{h})\right] = \sum_{i=1}^{n} \log\left[p(\mathbf{T}^i|\mathbf{h})\right] = \sum_{i=1}^{n}\sum_{l=1}^{L} \log\left[p(\mathbf{T}_l^i|\mathbf{h}, T_1^i, \ldots, T_{l-1}^i)\right] \\ =& n\log\left(\frac{1}{2}\right) + \sum_{i=1}^{n}\sum_{l\in I_i}\left[\left(1 - a_{kl}^i(\mathbf{h})\right)\log(r_{kl}) + a_{kl}^i(\mathbf{h})\log(1 - r_{kl})\right] \end{aligned} \tag{1}$$

where I_i is the set of informative loci for the i-th descendant (except the first informative locus the contribution of which is $\log(\frac{1}{2})$), and k the preceding informative locus of l. A rewriting of equation 1 as a quadratic form in $\mathbf{h}$ allows a sparse representation, which is computationally easier to manipulate:

$$V = K + \sum_{l=1}^{L}\sum_{k<l} \frac{1}{2} h_k h_l \log\left(\frac{1 - r_{kl}}{r_{kl}}\right) \sum_{i\in\{1,n\} \text{ s.t. } (k,l)\in F_i} \frac{1}{T_k^i T_l^i} \tag{2}$$

where $K = n\log\left(\frac{1}{2}\right) + \sum_{i=1}^{n}\sum_{l\in I_i}\frac{1}{2}\log\left[(1 - r_{kl})r_{kl}\right]$ and F_i is the set of pairs of consecutive informative loci in the i-th descendant.

Therefore V can be expressed as a quadratic form: $V = K + \mathbf{h}'\mathbf{W}\mathbf{h}$ with a symmetric $L \times L$ matrix $\mathbf{W}$ such that

$$W_{ll} = 0 \text{ and } W_{kl} = W_{lk} = \frac{1}{4}\log\left(\frac{1 - r_{kl}}{r_{kl}}\right) \sum_{i\in\{1,n\} \text{ s.t. } (k,l)\in F_i} \frac{1}{T_k^i T_l^i}$$

Let N_{kl}^+ (respectively N_{kl}^-) be the number of descendants such that each descendant i has $T_l^i = T_k^i$ (resp. $T_l^i \neq T_k^i$) and (k,l) is a pair of consecutive informative loci for this descendant.
Finally,

$$W_{kl} = \frac{1}{4}(N_{kl}^+ - N_{kl}^-)\log\left(\frac{1 - r_{kl}}{r_{kl}}\right) \tag{3}$$

Example 2. *Consider Example 1, we now compute N_{kl}^+ and N_{kl}^- for every pair of consecutive informative loci occuring in at least one descendant. We obtain $N_{1,4}^+ = 1$ due to son 1 ($T_1^1 = T_4^1$), $N_{1,4}^- = 0$, $N_{1,5}^+ = 0$,*

$N^-_{1,5} = 1$ *due to son* 3 ($T^3_1 \neq T^3_5$), $N^+_{4,5} = 0$, $N^-_{4,5} = 2$ *due to sons* 1 ($T^1_4 \neq T^1_5$) *and* 2 ($T^2_4 \neq T^2_5$), $N^+_{5,7} = 1$ *due to son* 2 ($T^2_5 = T^2_7$), *and finally,* $N^-_{5,7} = 2$ *due to sons* 1 ($T^1_5 \neq T^1_7$) *and* 3 ($T^3_5 \neq T^3_7$). *Others* N^+_{kl} *and* N^-_{kl} *are all equal to zero.*

2.1 Weighted constraint satisfaction formulation

This quadratic form can be directly translated into a *binary Weighted Constraint Satisfaction Problem* (WCSP) [11].

A binary WCSP is a pair (X,F) where $X = \{1,\ldots,m\}$ is a set of m variables and F a set of binary cost functions. Each variable $i \in X$ has a finite domain D_i of values than can be assigned to it. A binary cost function $f_{ij} \in F$ is a function $f_{ij} : D_i \times D_j \mapsto \mathcal{N}$ where $\mathcal{N}$ is the set of non-negative integers. The *constraint graph* of a binary WCSP is a graph $G = (X,E)$ with one vertex for each variable and one edge $(i,j) \in E$ for every cost function $f_{ij} \in F$.

The weighted constraint satisfaction problem is to find a complete assignment t of all the variables minimizing the total cost function $\sum_{f_{ij}\in F} f_{ij}(t[i],t[j])$ where $t[i]$ is the value assigned to variable i in t. This problem is NP-hard.

State-of-the-art WCSP exact (i.e., complete) solving methods are either *Depth-First Branch and Bound* (DFBB) exploiting local consistency techniques [11] or dynamic programming algorithms such as *bucket elimination*, aka *Variable Elimination* (VE) [5] or a combination of both approaches such as *Backtrack with Tree Decomposition* (BTD) [9, 3].

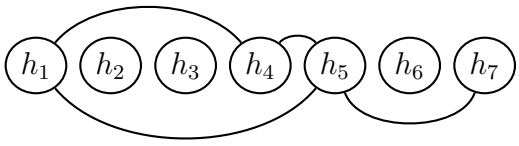
We have the following WCSP formulation of our haplotyping problem. We define $X = \{1,\ldots,L\}$ the set of $m = L$ variables with domain $D_i = \{-1,1\}, i \in \{1,\ldots,L\}$. Each WCSP variable i corresponds to a decision variable h_i of our problem. The set of binary cost functions is defined by $F = \{f_{kl} | W_{kl} \neq 0, k < l\}$. Each cost function f_{kl} represents two terms $-W_{kl}$ and $-W_{lk}$ of the symmetric matrix $\mathbf{W}$ ($W_{kl} = W_{lk}$) in the quadratic form $\mathbf{h}'\mathbf{W}\mathbf{h}$ (we use opposite terms for minimization). Because cost functions must be positive, where as $-W_{kl}$ may be negative, a constant term $2|W_{kl}|$ is added to each cost function[1]. Thus, $f_{kl}(h_k,h_l) = -2W_{kl}h_kh_l + 2|W_{kl}|$, or equivalently,

$$f_{kl}(-1,-1) = f_{kl}(1,1) = \begin{cases} -4W_{kl} & \text{if } W_{kl} < 0 \\ 0 & \text{otherwise} \end{cases} \qquad f_{kl}(-1,1) = f_{kl}(1,-1) = \begin{cases} 4W_{kl} & \text{if } W_{kl} > 0 \\ 0 & \text{otherwise} \end{cases}$$

Notice that these functions are soft versions of disequality (if $W_{kl} < 0$) and equality (if $W_{kl} > 0$) constraints.

For any complete assignment $\mathbf{h}$, we have $\sum_{f_{kl}\in F} f_{kl}(h_k,h_l) \propto V$ (see Equation 2). Thus, an optimal solution of WCSP (X,F) corresponds to the most likely haplotype configuration.

Consider Example 1 again, the constraint graph is given below.



2.2 Extension to the case of genotyped dams

In the case we know the genotypes of the dams, then we can take into account this extra information, by extending our definition of transmission values (without changing anything else). Variable T^i_l is known

[1] In order to get integer costs, we also multiply each cost function by a sufficiently large number and take the smallest following integer, such that it does not change the set of optimal solutions.

with certainty if and only if the i-th descendant is homozygous and the sire is heterozygous at locus l *or* the i-th descendant and the sire are heterozygous and the dam is homozygous at locus l (i.e., T_l^i is -1 or 1); otherwise T_l^i is unknown ($T_l^i = \star$). For example, if the descendant is AB, the sire BA and the dam AA, it is necessary that B in the descendant came from the first allele of the sire, and thus $T_l^i = 1$.

Example 3. *Consider Example 1 again, we add the information of genotyped dams. Let* $\boldsymbol{M}^{di}$ *be the genotypes of the i-th dam of the i-th descendant.*

$\boldsymbol{M}^{d1}$:	*AA*	*AB*	*AB*	*BB*	*AA*	*BA*	*AB*
$\boldsymbol{M}^{d2}$:	*AB*	*BA*	*BB*	*AB*	*AB*	*AA*	*AB*
$\boldsymbol{M}^{d3}$:	*AA*	*BA*	*AB*	*AA*	*AB*	*BA*	*AA*

$\boldsymbol{T}^1$:	*-1*	$\star$	$\star$	*-1*	*1*	$\star$	*-1*
$\boldsymbol{T}^2$:	$\star$	$\star$	$\star$	*-1*	*1*	$\star$	*1*
$\boldsymbol{T}^3$:	*1*	$\star$	$\star$	***1***	*-1*	$\star$	*1*

For son 2 *at locus* 1*, the transmitted allele from the sire is not identifiable (the son, sire, and dam are heterozygous). So,* T_1^2 *is still equal to* $\star$*. For son* 3 *at locus* 4*, the transmitted allele from the sire can be identified: the son and sire are heterozygous AB and BA respectively, and the dam is homozygous AA, so it is certain that the dam transmitted allele A and the sire transmitted allele B, and thus* $T_4^3 = 1$*. For this example, it is the only modification of the transmission values with respect to Example 1.* N^+ *and* N^- *are kept unchanged, except for* $N_{1,4}^+ = 2$*, due to sons* 1 *(*$T_1^1 = T_4^1$*) and* 3 *(*$T_1^3 = T_4^3$*). Finally, the constraint graph is the same as in Example 1.*

3 Experimental Results

3.1 Datasets and methods

A first dataset[2] consists of half-sib families which were simulated by considering either linkage disequilibrium at the sire/dams or not. In the former case, disequilibrium was generated first by simulating a Wright-Fisher scenario with 100 individuals mating at random during 100 generations; the sires and the dams haplotypes were sampled from the last generation. In both cases, the founders were simulated in linkage equilibrium and using a Beta distribution ($\alpha = 2, \beta = 2$) of allele frequency similar to the one observed in bovine livestock. Recombination events on a single chromosome of $S \in \{1,2\}$ Morgan were simulated using Haldane's mapping function, producing sons haplotypes. Genotypes were obtained by randomly permuting the two alleles at every locus of every pair of haplotypes. The number of SNPs L varied from 100 to 10000. These markers were evenly-spaced on the chromosome. The number of descendants n varied from 1 to 1000. 50 families were simulated for each set of parameters.

A second dataset[2] was built in the same way, but taking 44 real haplotypes of the father chromosome X in 44 trios of CEU population (see HAPMAP phase 3 release 2 project at `www.hapmap.org`) as initial sire/dams haplotypes. Because only 1 copy of chromosome X is present in males, its haplotype is known with certainty. This dataset provides a real pattern of linkage disequilibrium, contrary to simulated datasets. We selected $L = 36000$ SNPs such that for each locus the two alleles occured in our data. These markers were evenly-spaced on the chromosome of $S = 1.64$ Morgan.

Five haplotyping methods/softwares were studied. Exact methods are `Merlin` [1] version 1.1.2 ; `Superlink` [8] version 1.6 ; and our approach implemented in WCSP solvers `toulbar2` version 0.9.2 (for DFBB [4] used by default, and BTD [3]) and `toolbar` version 3.1 (for VE [5])[3]. Approximate methods are `W&M` [18] (implemented by us in R language) and `LinkPhase` [7] with parameters recommended by the authors and with unreconstructed loci fixed arbitrarily in a post-processing step. All the tested

[2] `carlit.toulouse.inra.fr/cgi-bin/awki.cgi/HaplotypeInference`

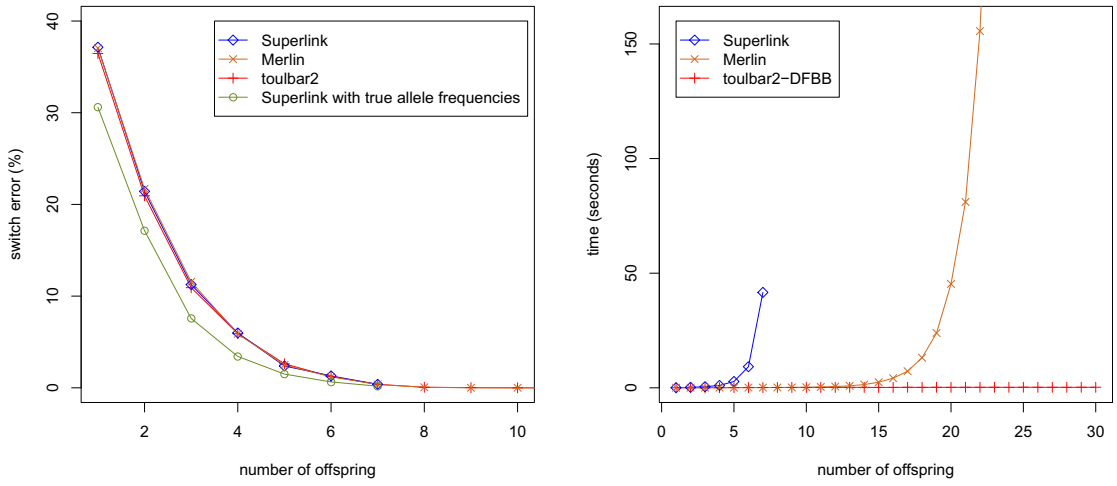
[3] `carlit.toulouse.inra.fr/cgi-bin/awki.cgi/ToolBarIntro`

methods except ours reconstruct all the individuals haplotypes. However, knowing the sire haplotypes, it is easy to find the most probable haplotypes for each son and its dam in linear time $O(L)$.

The experimentations were performed on a 2.6GHz Intel Xeon computer running Linux 2.627-11-server with 64 GB. These methods were compared in terms of the percentage of switch error [16], which measures the proportion of heterozygous loci whose allele origin (first or second sire haplotype) is wrongly inferred relative to the previous heterozygous locus ; and the CPU solving time in seconds. Reported results are mean over 50 families.

3.2 Comparison with exact methods

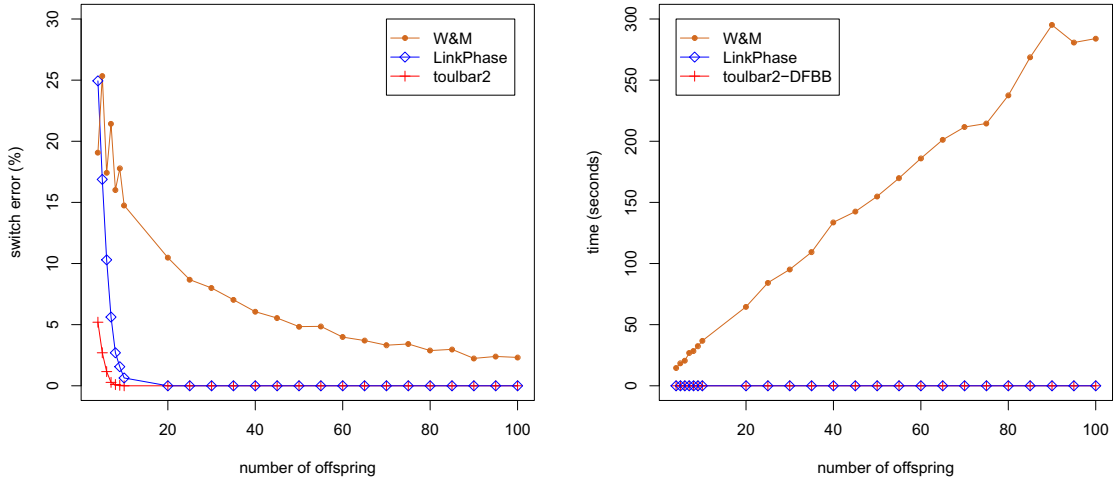
We compared our approach `toulbar2` with two exact methods, `Merlin` [1] and `Superlink` [8], varying the number of descendants in the first dataset without linkage disequilibrium and without genotyped dams. Figure 1(a) shows experimentally that these three methods find the same optimal sire haplotype configuration if we assumed all SNPs have equal allele frequencies for all the methods (option *-fe* in `Merlin` and given as input in .DAT file for `Superlink`)[4]. The switch error (Fig. 1(a)) decreases rapidly with the number of descendants. It is less than 1% (resp. 6%) for $n = 7$ (resp. 4) descendants. If we provide the true allele frequencies, `Superlink` found better sire haplotypes for small families and `Merlin` did not improve its results (because it does not fully reconstruct ungenotyped dam haplotypes). `Merlin` and `Superlink` are dynamic programming algorithms which have their time and space complexity increasing exponentially with the number of descendants. `Superlink` ran out of memory for more than 7 descendants. `Merlin` took more than 150 seconds for 22 descendants whereas `toulbar2` using default depth-first branch and bound (DFBB) took less than a second (Fig. 1(b)).



(a) (b)

Figure 1: Comparison with exact methods for $S = 1, L = 1500, n \in [1,30]$.

[4]In fact, there may be several optimal solutions and each method can find a different one resulting in small differences in terms of switch error (`Merlin` may output two solutions and we took the first one in our results).



(a) (b)

Figure 2: Comparison with approximate methods for $S = 1, L = 1500, n \in [4, 100]$.

3.3 Comparison with approximate methods

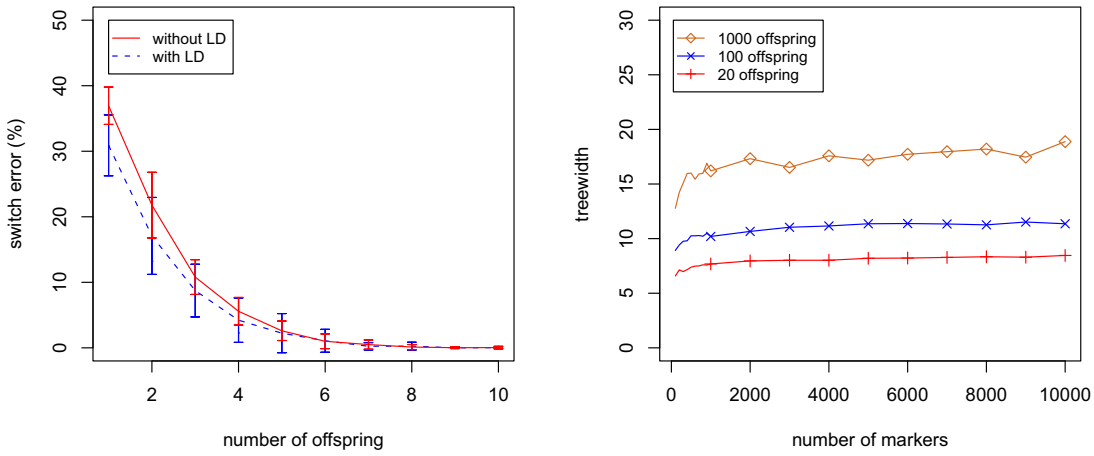
We compared `toulbar2` with two approximate methods, `W&M` [18] and `LinkPhase` [7], on the same dataset as in the previous section. `LinkPhase` required families of about twenty individuals to reconstruct entirely the sire haplotypes and to find the true haplotypes (Fig. 2(a)). For instance, with 4 descendants, `LinkPhase` did not reconstruct one third of the heterozygous loci; instead, `toulbar2` reconstructed all the sire haplotypes with 73% less of switch errors compared to `LinkPhase`. Moreover, `toulbar2` is guaranteed to find an optimal haplotype configuration. The convergence of `W&M` towards the true haplotypes was much slower compared to the two other methods. Furthermore, while `LinkPhase` and `toulbar2` (DFBB) solved every family within one second, `W&M` computing time grew linearly with the number of descendants (Fig. 2(b)).

3.4 Comparison with and without linkage disequilibrium

If we consider linkage disequilibrium, the mean switch error (Fig. 3(a)) is slightly better than without linkage disequilibrium, but the variance is much higher. This phenomenon may be due to the reduced number of different (sire and dams) haplotypes, resulting in less heterozygous markers ($\approx 40\%$ less than wout LD).

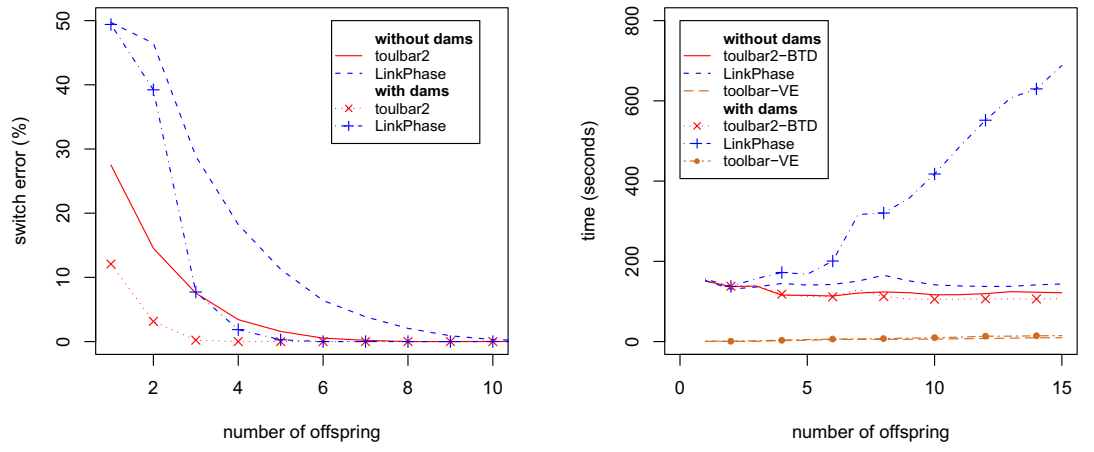
3.5 Study of the treewidth of our WCSP formulation

In order to assess the difficulty of the resulted WCSP instances of our first dataset (without linkage disequilibrium), we measured the treewidth of their constraint graph [2]. The treewidth of a graph gives an idea of its acyclicity (a tree as a treewidth of 1). Dynamic programming algorithms (VE [5] and BTD [3], but not DFBB) exploiting the WCSP formulation have their time and space complexity exponential in the treewidth (DFBB being exponential in the number of variables). Figure 3(b) shows the average treewidth obtained by a variable elimination order following the chromosome order. We noticed the



(a) (b)

Figure 3: (a) Comparison with/w. out linkage disequilibrium for $S = 1, L = 1500, n \in [1, 10]$ using `toulbar2`. (b) Constraint graph analysis of our WCSP formulation ($S = 2$).



(a) (b)

Figure 4: Human chr. X dataset w/wout genotyped dams ($L = 36000, n \in [1, 15]$).

treewidth remains relatively small (the maximal treewidth found in all our simulations was 30) and it seems to increase logarithmically with the number of individuals and the number of markers. We can conclude that the resulting WCSP instances are easy to solve by any dynamic programming algorithm. Therefore, we used VE and BTD instead of DFBB on very large datasets as done in the next section.

3.6 Comparison with and without genotyped dams on human chromosome X dataset

Using our second real dataset, we found the switch error was less than 1% (resp. 4%) for $n = 6$ (resp. 4) descendants (Fig. 4(a)), which is similar to our first dataset. By exploiting the additional information of genotyped dams, only 3 descendants are needed to reconstruct the sire haplotypes with less than 1% of switch error. Concerning performance (Fig. 4(b)), `toolbar` VE and `toulbar2` BTD performed similarly with or without the genotyped dams, although VE was much faster but needed more space than BTD. On the contrary, `LinkPhase` time increased linearly with the number of descendants in the case of genotyped dams. The treewidth was 11 in average.

4 Conclusion

In this paper, we have proposed a sparse representation (with a small treewidth) of sire haplotype reconstruction in half-sib families and a method which finds an optimal haplotype configuration. This method obtained good results, in terms of accuracy and time, on simulated and real datasets.

In the future, we will improve our results for small families with linkage disequilibrium and study other kinds of pedigrees.

References

[1] G. Abecasis, S. Cherny, W. Cookson, and L. Cardon. Merlin – rapid analysis of dense genetic maps using sparse gene flow trees. *Nature Genetics*, 30:97–101, 2002.

[2] H. Bodlaender. Discovering treewidth. In *Theory and Practive of Computer Science - SOFSEM'2005*, pages 1–16, 2005.

[3] S. de Givry, T. Schiex, and G. Verfaillie. Exploiting Tree Decomposition and Soft Local Consistency in Weighted CSP. In *Proc. of AAAI-06*, Boston, MA, 2006.

[4] S. de Givry, M. Zytnicki, F. Heras, and J. Larrosa. Existential arc consistency: Getting closer to full arc consistency in weighted CSPs. In *Proc. of IJCAI-05*, pages 84–89, Edinburgh, Scotland, 2005.

[5] Rina Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113(1–2):41–85, 1999.

[6] K. Doi, J. Li, and T. Jiang. Minimum recombinant haplotype configuration on tree pedigrees. In *WABI'03*, pages 339–353, 2003.

[7] T. Druet and M. Georges. A Hidden Markov Model Combining Linkage and Linkage Disequilibrium Information for Haplotype Reconstruction and Quantitative Trait Locus Fine Mapping. *Genetics*, 184, 2010.

[8] M. Fichelson, N. Dovgolevsky, and D. Geiger. Maximum Likelihood Haplotyping for General Pedigrees. *Human Heredity*, 59(1):41–60, 2005.

[9] P. Jégou and C. Terrioux. Hybrid backtracking bounded by tree-decomposition of constraint networks. *Artificial Intelligence*, 146:43–75, 2003.

[10] S. Knott, J.-M. Elsen, and C. Haley. Methods for multiple-marker mapping of quantitative trait loci in half-sib populations. *Theoretical and Applied Genetics*, 93(1-2):71–80, 1996.

[11] J. Larrosa and T. Schiex. Solving weighted CSP by maintaining arc consistency. *Artificial Intelligence*, 159(1-2):1–26, 2004.

[12] S. Lauritzen and N. Sheehan. Graphical Models for Genetic Analyses. *Statistical Science*, 18(4):489–514, 2003.

[13] T. Niu. Algorithms for inferring haplotypes. *Genetic Epidemio.*, 27:334–347, 2004.

[14] D. Qian and L. Beckmann. Minimum-Recombinant Haplotyping in pedigrees. *American journal of human genetics*, 70(6):1434–1445, 2002.

[15] M. Sanchez, S. de Givry, and T. Schiex. Mendelian error detection in complex pedigrees using weighted constraint satisfaction. *Constraints*, 13(1):130–154, 2008.

[16] M. Stephens and P. Donnelly. A comparison of bayesian methods for haplotype reconstr. from population genotype data. *Am J Hum Genet*, 73:1162–1169, 2003.

[17] E. Wijsman, J. Rothstein, and E. Thompson. Multipoint linkage analysis with many multiallelic or dense diallelic markers: MCMC provides practical approaches for genome scans on general pedigrees. *Am J Hum Genet*, 79:846–858, 2006.

[18] J. Winding and T. Meuwissen. Rapid haplotype reconstruction in pedigrees with dense marker maps. *J. of Animal Breeding and Genetics*, 121:26–39, 2004.

Alignment of RNA with Structures of Unlimited Complexity

Alessandro Dal Palù
Dipartimento di Matematica
Università Parma, Parma, Italy
alessandro.dalpalu@unipr.it

Mathias Möhl
Bioinformatics, Institute of Computer Science
Albert-Ludwigs-Universität, Freiburg, Germany
mmohl@informatik.uni-freiburg.de

Sebastian Will
Biology Lab, CSAIL
MIT, Cambridge MA, USA
swill@csail.mit.edu

Abstract

Sequence-structure alignment of RNA with arbitrary secondary structure is Max-SNP-hard. Therefore, the problem of RNA alignment is commonly restricted to nested structure, where dynamic programming yields efficient solutions. However, nested structure cannot model pseudoknots or even more complex structural dependencies. Nevertheless those dependencies are essential and conserved features of many RNAs. Only a few existing approaches deal with crossing structures. Here, we present a constraint approach for alignment of structures in the even more general class of unlimited structures. Our central contribution is a new RNA alignment constraint propagator. It is based on an efficient $O(n^2)$ relaxation of the RNA alignment problem. Our constraint-based approach Carna solves the alignment problem for sequences with given input structures of unlimited complexity. Carna is implemented using Gecode.

In the post-genomic era, biologists get more and more interested in studying non-coding RNA molecules with catalytic and regulatory activity as central players in biological systems. The computational analysis of non-coding RNA requires to take structural information into account. Whereas RNAs form three-dimensional structures, structural analysis of RNA is usually concerned with the secondary structure of an RNA, i.e. the set of RNA base pairs (i, j) that form contacts (H-bonds) between the bases i and j. The RNA alignment problem is to align two RNA sequences A and B with given secondary structure for each RNA such that a score based on sequence and structure similarity is optimized. The difficulty of this problem depends on the complexity of the RNA structures. Therefore, a complexity hierarchy of RNA structures was introduced. Most RNA analysis is performed for the class of nested structures P, where base-pairs do not cross, because for this class one can find efficient dynamic programming algorithms for structure prediction and alignment under reasonable scoring schemes [12, 5]. The more general class of crossing RNA structures P restricts the degree of base pairing to at most one, as is commonly assumed for single RNA structure. Prediction and alignment in this class is NP-hard in general [2]. However, one can devise a number of algorithms that efficiently predict or align RNAs with structures from classes in between non-crossing and arbitrary crossing [9, 8, 7]. However these algorithms have complexities that limit their application range. Other approaches for RNA alignment handle crossing structures with parametrized complexity, were the parameter captures the complexity of the structures [6]. Finally, the ILP approach Lara [1] computes alignments of arbitrarily complex crossing structures and appears to be more effective than dynamic programming based approaches. The success of this AI technique was a strong motivation for this work, where we study the alignment of RNAs with structures of unlimited complexity using constraint programming.

Contribution We devise a constraint algorithm for the problem of aligning two RNA molecules with given sequences and unlimited secondary structures. By modeling and propagating constraints on integers, the method goes beyond rephrasing the ILP approach [1] in CP. We describe the constraint model, develop a new RNA alignment propagator, and present a specific search strategy. It is implemented using the Gecode constraint programming system. Finally, we apply our method to align both RNA molecules with given fixed structures and RNA molecules with associated base pair probability matrices.

1 Methods

1.1 Preliminaries

An *RNA sequence* S is a string over the set of bases $\{A,U,C,G\}$ and an *RNA structure* P is a set of *base pairs* (also called *arcs*) (i,j) with $1 \leq i < j \leq |S|$. We define an *arc-annotated sequence* as pair of RNA sequence and RNA structure and denote the i-th symbol of S by $S[i]$.

One constructs a hierarchy of RNA structure classes based on the following properties. Two arcs (i,j) and (i',j') are called *nested* iff $i < i' < j' < j$ or $i' < i < j < j'$, they are *independent* iff $i < j < i' < j'$ or $i' < j' < i < j$. A RNA structure P is called *nested* if all differing base pairs $(i,j),(i',j') \in P$ are either nested or independent. In a *crossing* RNA structure P each base is involved in at most one base pair, i.e. $\forall (i,j) \neq (i',j') \in P : i \neq i' \wedge j \neq j' \wedge i \neq j \wedge i' \neq j'$. We use the term *unlimited* to refer to an arbitrary RNA structure. Note that by definition each nested structure is crossing, and each crossing structure is unlimited, such that these classes form a class hierarchy.

An *alignment* A of two *arc-annotated sequences* (S_a,P_a) and (S_b,P_b) is a set $A_m \cup A_g$, where $A_m \subseteq [1..|S_a|] \times [1..|S_b|]$ is a set of *match edges* such that for all $(i,j),(i',j') \in A_m$ it holds that 1.) $i > i'$ implies $j > j'$ and 2.) $i = i'$ if and only if $j = j'$ and A_g is the set of *gap edges* $\{(x,-) \mid x \in [1..|S_a|] \wedge \nexists y : (x,y) \in A_m\} \cup \{(-,y) \mid y \in [1..|S_b|] \wedge \nexists x : (x,y) \in A_m\}$. We define the (i,i')-*prefix of* A as $A \cap (\{(j,j') \mid j \leq i, j' \leq i'\} \cup \{(j,-) \mid j \leq i\} \cup \{(-,j') \mid j' \leq i'\})$ and the (i,i')-*suffix of* A as $A \cap (\{(j,j') \mid j > i, j' > i'\} \cup \{(j,-) \mid j > i\} \cup \{(-,j') \mid j' > i'\})$.

Fix two arc-annotated sequences (S_a,P_a) and (S_b,P_b) with unlimited structures P_a and P_b. Define the score of alignment A of (S_a,P_a) and (S_b,P_b) as

$$\text{score}(A_m \cup A_g) := \sum_{(i,i') \in A_m} \sigma(i,i') + \sum_{\substack{(i,j) \in P_a, (i',j') \in P_b, \\ (i,i') \in A_m, (j,j') \in A_m}} \tau(i,j,i',j') \quad + \quad \gamma |A_g|,$$

where $\sigma(i,j)$ is the similarity of bases $S_a[i]$ and $S_b[j]$, $\tau(i,j,i',j')$ is the similarity of base pairs $(i,j) \in P_a$ and $(i',j') \in P_b$ and γ is the gap cost. Commonly, scores for sequence-structure alignment penalize the base match of different bases but don't penalize the same match if it occurs as part of a base pair match. We emphasize that our scoring function can express such scores, in the same way as scoring functions that don't add base similarity in case of a structural base match. For example, if bases A_i and B'_i differ, the negative contribution by $\sigma(i,i')$ can be compensated by $\tau(i,i',j,j')$ in a structural match.

The *alignment problem* is to determine $\underset{A \text{ alignment of } (S_a,P_a) \text{ and } (S_b,P_b)}{\text{argmax}} \text{score}(A)$.

Please note that we score the matches of all base pairs that are matched by the alignment. Given unlimited structures P_a and P_b, our approach is thus able to simultaneously take into account several biologically relevant RNA structures per sequence. In contrast, Lara [1] would select a single, best crossing RNA structure for each sequence and score the match of only those structures. This assumes that there is only one conserved crossing structure for each RNA. The potential advantages of our scoring for aligning RNAs with conserved unlimited structure have still to be explored (see Discussion). For the special case of crossing structures with positive weights there is no difference between the scoring by our approach and Lara, because in this case Lara scores the matches of all base pairs matched by the alignment. This justifies our direct comparison of the two approaches for this case.

1.2 Constraint Model

We model an alignment of arc-annotated sequences (S_a,P_a) and (S_b,P_b) by variables $\texttt{MD}_i$ and $\texttt{M}_i$ for $1 \leq i \leq |S_a|$ with initial domains $D(\texttt{MD}_i) = \{1,\ldots,|S_b|\}$ and $D(\texttt{M}_i) = \{0,1\}$. We write $\vec{MD}$ and $\vec{M}$ to denote the vectors of respective variables $\texttt{MD}_i$ and $\texttt{M}_i$.

A valuation V of these variables corresponds to a class $\mathscr{A}(V)$ of alignments A of (S_a,P_a) and (S_b,P_b) as defined by

$$\begin{aligned} &V(\texttt{MD}_i) = j \wedge V(\texttt{M}_i) = 1 \text{ iff } (i,j) \in A \\ &V(\texttt{MD}_i) = j \wedge V(\texttt{M}_i) = 0 \text{ iff } (i,-) \in A \\ &\qquad\qquad \wedge \forall (i',j') \in A : i' < i \rightarrow j' \leq j \wedge i' > i \rightarrow j' > j. \end{aligned}$$

In this way, M_i tells whether i is matched or deleted and the value j of MD_i tells that i is matched to j or deleted after j. One can show that $\mathscr{A}(V)$ has at most one element and that for each alignment A of (S_a, P_a) and (S_b, P_b) there is a corresponding valuation.

For example, the following alignment and valuation correspond to each other:

$$A = \{(1,1),(-,2),(-,3),(2,4),(3,-),(4,5)\}$$

, which is often written as $\begin{matrix}\texttt{A--CUG}\\ \texttt{ACAC-G}\end{matrix}$, corresponds to the valuation $\vec{MD} = (1,4,4,5)$ and $\vec{M} = (1,1,0,1)$.

Notably, alignments corresponding to a valuation that assigns $MD_i = j$ can be composed from an alignment of prefixes $S_a[1..i]$ and $S_b[1..j]$ and an alignment of suffixes $S_a[i+1..|S_a|]$ and $S_b[j+1..|S_b|]$ regardless of $\mathtt{M}_i$.

We introduce a constraint $\texttt{Alignment}(\vec{MD}, \vec{M})$ that is satisfied by any valuation with a corresponding alignment. Furthermore, we model the score of the alignment. Therefore, we introduce a variable `Score` and a constraint $\texttt{AlignmentScore}(\vec{MD}, \vec{M}, \texttt{Score})$. This constraint relates a valuation of `MD` and `M` to the score of its corresponding alignment.

Both constraints are propagated by the propagator of the next subsection. For finding optimal alignments we perform a depth-first branch-and-bound search enumerating `MD` and `M` according to a specific search strategy described at the end of the next section. Successfully applying branch-and-bound requires good upper bounds for the alignment score, such that large parts of the search tree can be pruned. Computing such bounds efficiently is the central job of the alignment propagator.

1.3 The Alignment Propagator

The alignment propagator computes hyper-arc consistency for the constraint $\texttt{Alignment}(\vec{MD}, \vec{M})$ and propagates $\texttt{AlignmentScore}(\vec{MD}, \vec{M}, \texttt{Score})$.

It prunes $\vec{MD}$ and $\vec{M}$ due to the score by computing upper score bounds for single variable assignments and furthermore computes lower and upper bounds for `Score` based on $\vec{MD}$ and $\vec{M}$.

Define the class $\mathscr{A}(D)$ as union of $\mathscr{A}(V)$ over all valuations V that satisfy D. The computation of bounds is based on a relaxation of the alignment problem. In this relaxation the two ends of each base pair match are decoupled. Thus in the *relaxed optimization problem for D*, we maximize a relaxed score

$$\text{score}_{\text{relaxed}}(A_m \cup A_g) := \sum_{(i,i') \in A_m} \left[\sigma(i,i') + \frac{1}{2}\text{ub}_D(i,i')\right] \quad + \quad \gamma|A_g|,$$

over all alignments in $\mathscr{A}(D)$, where

$$\text{ub}_D(i,i') := \max_{A_m \cup A_g \in \mathscr{A}(D)} \sum_{\substack{(i,j)\in P_a,(i',j')\in P_b,\\ (i,i')\in A_m,(j,j')\in A_m}} \tau(i,j,i',j') \; + \sum_{\substack{(j,i)\in P_a,(j',i')\in P_b,\\ (i,i')\in A_m,(j,j')\in A_m}} \tau(j,i,j',i').$$

Here, ub_D works as an upper bound for the score contributions by arc matches involving (i,i') and consequently $\text{score}_{\text{relaxed}}(A) \geq \text{score}(A)$ for $A \in \mathscr{A}(D)$. Thus, solving the relaxed problem yields an upper bound of `Score`.

For a moment, postpone how to efficiently compute $\text{ub}_D(i,i')$. Then, because the relaxed score has the form of a sequence similarity score, one can apply the Smith-Waterman algorithm [10] to maximize the relaxed score in $O(n^2)$ by dynamic programming, where $n = \max(|S_a|,|S_b|)$. The optimization problem is easily constrained due to domain D, because domains directly restrict the valid cases in the dynamic programming recursion.

Tracing back through the dynamic programming matrix yields an alignment A^l such that $\text{score}\, A^l$ is a lower bound of `Score`. Furthermore, we compute upper bounds for each single variable valuation. This requires to complement the above "forward algorithm" that computes the matrix entries

$$\text{Prefix}\,(i,i') := \max_{(i,i')\text{-prefix } A^p_{ii'} \text{ of } A \in \mathscr{A}(D)} \text{score}_{\text{relaxed}}(A^p_{ii'})$$

by a symmetric "backward algorithm" that computes the entries

$$\text{Suffix}\,(i,i') := \max_{(i,i')\text{-suffix } A^s_{ii'} \text{ of } A \in \mathscr{A}(D)} \text{score}_{\text{relaxed}}(A^s_{ii'}).$$

Now the variables $\vec{MD}$ can be pruned efficiently, because $\text{Prefix}(i,i') + \text{Suffix}(i,i')$ is an upper bound for the assignment $\texttt{MD}_i = j$. Similarly, we prune $\vec{M}$ using the two matrices.

It remains to describe the efficient computation of $\text{ub}_D(i,i')$. It suffices to describe the maximization of $\sum_{\substack{(i,j)\in P_a,(i',j')\in P_b,\\ (i,i')\in A_m,(j,j')\in A_m}} \tau(i,j,i',j')$ over alignments in $\mathscr{A}(D)$. A single match (j,j') can occur in an alignment in $\mathscr{A}(D)$ if $j' \in D(\texttt{MD}_j)$ and $1 \in D(\texttt{M}_j)$. However, we look for the best set of simultaneously valid matches (j,j'). The structure of this subproblem is analogous to sequence alignment. Thus, it is solved efficiently by dynamic programming. Therefore, $\text{ub}(i,i')$ is computed in $O(kk')$ time, where k and k' are the respective number of base pairs incident to i and i'. For many applications k and k' can be constantly bounded such that the propagator runs in $O(n^2)$ time and space.

Incrementality The propagator profits from reduced domain sizes of the variables $\vec{MD}$, because $\text{Prefix}(i,i')$ is finite only if $i' \in D(\texttt{MD}_i)$ and the Suffix-matrix is analogously restricted. The complexity of the propagator is therefore given more precisely as $O(\sum_{i=1}^{|S_a|} |D(\texttt{MD}_i)|)$. We postponed the idea of incrementally updating the matrices according to domain changes, because we expect large domain changes due to our propagator. Large domain changes would likely counteract the benefits of matrix updates.

Affine gap cost The method is straightforwardly extended to affine gap cost by using a Gotoh-like forward and backward algorithm in the propagator without increasing its complexity. It appears that this modification comes more natural in our approach than the corresponding extension in ILP, because it does not require any change of the model.

Propagator-guided search strategy Our search strategy guides the search to disprove overestimated bounds fast and to find valid good alignments quickly. Because information for achieving both goals is computed during propagation and is expensive to recompute, we reuse propagation results for guiding the search. We select a variable with large domain size that yields a high undecided contribution to the upper bound. We split the domain of this variable to select the 20% highest relaxed scores first.

2 Results

The method, called Carna, is implemented in C++ using the constraint programming system Gecode. For handling input and output as well as for special datastructures we reused code of LocARNA [11].

We run tests for two application scenarios. First, we explore Carna's behavior on crossing input structure using instances from all 16 Rfam families with crossing structure. Table 1 compares our results to Lara [1]. The table omits all 8 instances where both approaches run in less than 0.1 seconds. In all but one of the omitted cases, Carna solves the problem without backtracking. In terms of performance, with the single exception of tmRNA, both programs are on a par.

In our second scenario, we align dot-plot matrices as computed by RNAfold[4]. We obtain unlimited input structures by base pair filtering as e.g. done in LocARNA. As in LocARNA and PMcomp [3] base pair similarities are weighted according to the base pair probabilities. This results in alignments that are guided by the common structural potential of both RNAs and not only a single common structure. We align two tRNAs closing the search tree after nine nodes. Two TPP riboswitches of sizes 108 and 111 are aligned in 0.24 seconds closing the tree after 100 nodes.

3 Discussion

We showed that a constraint-based approach to RNA alignment can be competitive with the ILP based method Lara for crossing structures. Moreover, the approach is the first such method that scores unlimited structure. In this way, it differs from simultaneous alignment and folding approaches like Lara, which score only crossing (or even more restricted) substructures of unlimited input structures. The full potential of scoring unlimited structure and its biological applications, e.g. for aligning dot-plots of riboswitches and RNAs with conserved

Family	Instance Size				Run-time (s)		Carna Search Tree		
	$\|S_a\|$	$\|S_b\|$	$\|P_a\|$	$\|P_b\|$	Carna	Lara	Depth	Fails	Size
Entero_OriR	126	130	35	41	0.03	0.18	38	13	50
Intron_gpI	443	436	60	60	0.1	0.2	0	0	1
IRES_Cripavirus	202	199	59	57	0.2	0.04	157	127	296
RNaseP_arch	303	367	88	110	0.46	1.4	63	8	64
RNaseP_bact_b	408	401	125	125	3.0	2.3	370	677	1463
RNaseP_nuc	317	346	65	66	0.07	2.9	14	4	16
Telomerase-vert	448	451	112	116	0.47	2.3	146	32	161
tmRNA	384	367	110	110	63	3.7	433	14347	28785

Table 1: Results for the eight hardest instances of the benchmark set with crossing structures. We omit details for 8 instances where both programs run in less than 0.1 seconds.

folding dynamics have still to be explored. A constraint-based method promises flexibility for further extensions and improvements. Solving relaxed problems in propagators for handling crossing and unlimited RNA structure was shown to be a viable approach and appears to be generalizable to related problems.

Acknowledgments This work is partially supported by DFG grants WI 3628/1-1, BA 2168/3-1, PRIN08 *Innovative multi-disciplinary approaches for constraint and preference reasoning* and GNCS-INdAM *Tecniche innovative per la programmazione con vincoli in applicazioni strategiche*.

References

[1] Markus Bauer, Gunnar W. Klau, and Knut Reinert. Accurate multiple sequence-structure alignment of RNA sequences using combinatorial optimization. *BMC Bioinformatics*, 8:271, 2007.

[2] Guillaume Blin, Guillaume Fertin, Irena Rusu, and Christine Sinoquet. Extending the hardness of RNA secondary structure comparison. In *Combinatorics, Algorithms, Probabilistic and Experimental Methodologies, First International Symposium, ESCAPE 2007, Hangzhou, China, April 7-9, 2007, Revised Selected Papers*, volume 4614 of *Lecture Notes in Computer Science*, pages 140–151. Springer, 2007.

[3] I. L. Hofacker, S. H. Bernhart, and P. F. Stadler. Alignment of RNA base pairing probability matrices. *Bioinformatics*, 20(14):2222–7, 2004.

[4] Ivo L. Hofacker, Walter Fontana, Peter F. Stadler, Sebastian Bonhoeffer, Manfred Tacker, and Peter Schuster. Fast folding and comparison of RNA secondary structures. *Monatshefte Chemie*, 125:167–188, 1994.

[5] Tao Jiang, Guohui Lin, Bin Ma, and Kaizhong Zhang. A general edit distance between RNA structures. *Journal of Computational Biology*, 9(2):371–88, 2002.

[6] Mathias Möhl, Sebastian Will, and Rolf Backofen. Fixed parameter tractable alignment of RNA structures including arbitrary pseudoknots. In *Proceedings of the 19th Annual Symposium on Combinatorial Pattern Matching (CPM 2008)*, LNCS, pages 69–81. Springer-Verlag, 2008.

[7] Mathias Möhl, Sebastian Will, and Rolf Backofen. Lifting prediction to alignment of RNA pseudoknots. *Journal of Computational Biology*, 2010. Accepted.

[8] Jens Reeder and Robert Giegerich. Design, implementation and evaluation of a practical pseudoknot folding algorithm based on thermodynamics. *BMC Bioinformatics*, 5:104, 2004.

[9] E. Rivas and S. R. Eddy. A dynamic programming algorithm for RNA structure prediction including pseudoknots. *Journal of Molecular Biology*, 285(5):2053–68, 1999.

[10] T.F. Smith and M.S. Waterman. Comparison of biosequences. *Adv. appl. Math.*, 2:482–489, 1981.

[11] Sebastian Will, Kristin Reiche, Ivo L. Hofacker, Peter F. Stadler, and Rolf Backofen. Inferring non-coding RNA families and classes by means of genome-scale structure-based clustering. *PLOS Computational Biology*, 3(4):e65, 2007.

[12] M. Zuker and P. Stiegler. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Research*, 9(1):133–48, 1981.

Geometric Constraints for the Phase Problem in X-Ray Crystallography

Corinna Heldt, Alexander Bockmayr
Freie Universität Berlin, FB Mathematik und Informatik,
Arnimallee 6, 14195 Berlin, Germany
Corinna.Heldt@fu-berlin.de,
Alexander.Bockmayr@fu-berlin.de

Abstract

X-ray crystallography is one of the main methods to establish the three-dimensional structure of biological macromolecules. In an X-ray experiment, one can measure only the magnitudes of the complex Fourier coefficients of the electron density distribution under study, but not their phases. The problem of recovering the lost phases is called the phase problem. Building on earlier work by Lunin/Urzhumtsev/Bockmayr, we extend their constraint-based approach to the phase problem by adding further 0-1 linear programming constraints. These constraints describe geometric properties of proteins and increase the quality of the solutions. The approach has been implemented using SCIP and CPLEX, first computational results are presented here.

1 Introduction

Knowledge about the three-dimensional structure of biological macromolecules is an essential foundation of structural biology and biotechnology. In X-ray crystallography the arrangement of atoms within a crystal is determined from a three-dimensional representation of the electron density. From X-ray experiments one gets *diffraction data* depending on the molecular structure, i.e., the intensities of reflections of X-rays diffracted by the crystal. X-rays are scattered exclusively by the electrons in the atoms, so one is searching for a relation between the measured intensities of the beams diffracted at the object in question and the crystal structure, which can be described by the electron density distribution. The electron density represents probabilistically where electrons can be found in the molecule. With the help of diffraction data and the usage of mathematical as well as experimental methods, an electron density map can be derived. *Direct methods* use mathematical techniques to compute an electron density map from the diffraction data without any further experiments. The main problem here is the *phase problem*: experiments provide only the intensities of the X-rays diffracted in different directions and so the electron density magnitudes can be calculated, whereas the information about the phase shift is lost.

Lunin, Urzhumtsev and Bockmayr [8] proposed a 0-1 linear programming approach to direct phasing. This approach yields a set of solutions. In order to increase the quality of this solution set, we formulate some geometric properties of proteins as additional 0-1 linear programming constraints. In [3], we described the basic ideas of the 0-1 linear programming approach by Lunin, Urzhumtsev and Bockmayr [8], now we derive the new geometric constraints and present first computational results.

2 The phase problem

Every crystal consists of identical molecules, resp. complexes of molecules strictly ordered in all three dimensions. This means that we can find a parallelepiped called *unit cell* containing such a complex of molecules which builds up the whole crystal if it is repeatedly stacked together in all three dimensions. We will denote the unit cell's volume with V_{cell}. Let $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3 \in \mathbb{R}^3$ span the unit cell. Then we can write every vector $\mathbf{r} \in \mathbb{R}^3$ in this basis, i.e., $\mathbf{r} = x_1\mathbf{b}_1 + x_2\mathbf{b}_2 + x_3\mathbf{b}_3$, where $\mathbf{x} = (x_1, x_2, x_3)^T \in [0,1]^3$ is the

vector of coordinates of $\mathbf{r}$ with respect to the basis $\{\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3\}$. We are searching for the electron density distribution $\rho(\mathbf{x})$ over the crystal. Due to the crystal structure, ρ is a periodic function and therefore can be developed into a *Fourier series* [6]

$$\rho(\mathbf{x}) = \frac{1}{V_{cell}} \sum_{\mathbf{h}\in\mathbb{Z}^3} \mathbf{F}(\mathbf{h}) \exp(-2\pi i(\mathbf{h}^T\mathbf{x})),\ \mathbf{x} \in V. \tag{1}$$

The Fourier coefficients $\mathbf{F}(\mathbf{h}), \mathbf{h} \in \mathbb{Z}^3$, which are called *structure factors* in crystallography, are given by the formula

$$\mathbf{F}(\mathbf{h}) = \int_V \rho(\mathbf{x}) \exp(2\pi i(\mathbf{h}^T\mathbf{x}))d\mathbf{x}. \tag{2}$$

Since these are complex numbers, the structure factors can be written in the form $\mathbf{F}(\mathbf{h}) = F(\mathbf{h})\exp(i\varphi(\mathbf{h}))$, where $F(\mathbf{h}) = |\mathbf{F}(\mathbf{h})|$ is the *magnitude* and $\varphi(\mathbf{h}) \in [0, 2\pi[$ the *phase*.

The only experimental data we get in X-ray-crystallography are the reflection intensities. The intensity $I(\mathbf{h})$ of a reflection is proportional to the magnitude of the squared structure factors, with a known constant of proportionality, i.e., $C \cdot I(\mathbf{h}) = |\mathbf{F}(\mathbf{h})|^2, C \in \mathbb{R}$. Thus, all we can calculate from our experimental data are the structure factor magnitudes. The phase information is lost and must be restored by other means. This is called the *phase problem*.

3 0-1 linear programming approach

Now, the main ideas of the approach proposed in [8] are presented. Instead of calculating the electron density distribution in the whole unit cell, we will work on a grid. Using discrete Fourier transforms, we calculate electron densities at the grid points. Consider a grid $\Pi = [0, M_1 - 1] \times [0, M_2 - 1] \times [0, M_3 - 1] \subseteq \mathbb{Z}^3$, where $M = M_1M_2M_3$ is the total number of grid points. Denote by $\mathbf{M}$ the diagonal matrix $\mathrm{diag}(M_1, M_2, M_3)$, with diagonal elements $M_1, M_2, M_3 \in \mathbb{N}$. The values of the electron density function $\rho(\mathbf{x}),\ \mathbf{x} \in V$ at the grid points are described by the *grid electron density function* $\rho_g(\mathbf{j}) = \rho(\mathbf{M}^{-1}\mathbf{j})$, $\forall \mathbf{j} \in \Pi$. We define the *grid structure factor* $\mathbf{F}_g(\mathbf{h})$ by the discrete Fourier transform

$$\mathbf{F}_g(\mathbf{h}) = \frac{1}{M} \sum_{\mathbf{j}\in\Pi} \rho_g(\mathbf{j}) \exp(2\pi i(\mathbf{h}^T\mathbf{M}^{-1}\mathbf{j})),\ \forall \mathbf{h} \in \Pi. \tag{3}$$

If we know the grid structure factors, we can restore the grid electron densities

$$\rho_g(\mathbf{j}) = \sum_{\mathbf{h}\in\Pi} \mathbf{F}_g(\mathbf{h}) \exp(-2\pi i(\mathbf{h}^T\mathbf{M}^{-1}\mathbf{j})),\ \forall \mathbf{j} \in \Pi, \tag{4}$$

using the inverse discrete Fourier transform.

In the context of direct phasing, it may be sufficient to find a binary *envelope* of the regarded molecules, i.e., a binary function representing areas where the electron density is above a certain cut-off level κ [8]. Using this idea, we may replace the unknowns $\rho_g(\mathbf{j})$ by binary variables $z_\mathbf{j} \in \{0, 1\}$, for each grid point $\mathbf{j} \in \Pi$, satisfying $z_\mathbf{j} = 0$, if $\rho(\mathbf{j}) \leq \kappa$ and $z_\mathbf{j} = 1$ otherwise.

By restricting the possible phase values $\varphi(\mathbf{h}) \in [0, 2\pi[, \forall \mathbf{h} \in \Pi$ to four ones, i.e., $\varphi(\mathbf{h}) \in \{\pm\frac{\pi}{4}, \pm\frac{3}{4}\pi\}$, $\forall \mathbf{h} \in \Pi$, the phase problem can be stated as a system of linear inequalities in 0-1 variables for representing the electron density values at grid points and for representing the phases. By penalizing the amount of violation, a suitable objective function can be introduced [8, 3].

In general, the resulting 0-1 linear program for solving the phase problem does not have a unique optimal solution, but a set of different optimal solutions. In order to reduce the number of those and at the same time increase the quality of the remaining ones, additional constraints can be added.

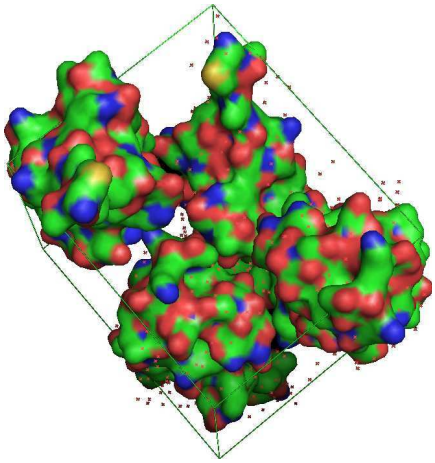


Figure 1: Unit cell of Protein G

4 Additional constraints

In the electron density distribution of a protein, no peaks of very high or very low electron density occur, if an appropriate resolution is used. This means, in the grid electron density distribution, no isolated points of high electron density surrounded by low electron density values as well as no isolated points of low electron density surrounded by high electron density values are expected to occur.

Definition 1 (Neighbour relation)**.** *Two grid points* $\mathbf{j}_1 \in \Pi$ *and* $\mathbf{j}_2 \in \Pi$ *are neighbours, denoted by* $\mathbf{j}_1 \mathfrak{n} \mathbf{j}_2$*, if and only if* $\| \mathbf{j}_1 - \mathbf{j}_2 \|_2 = 1$.

Definition 2 (Isolated point)**.** *A binary grid point* $z_\mathbf{j} \in \Pi$ *is called isolated if and only if* $z_\mathbf{j} = 0 \Rightarrow z_\mathbf{i} = 1,\ \forall\ \mathbf{i} \mathfrak{n} \mathbf{j}$ *and* $z_\mathbf{j} = 1 \Rightarrow z_\mathbf{i} = 0,\ \forall\ \mathbf{i} \mathfrak{n} \mathbf{j}$.

Every interior grid point has six neighbours, thus the condition $-5 \leq z_\mathbf{j} - \sum_{\mathbf{i} \mathfrak{n} \mathbf{j}} z_\mathbf{i} \leq 0$, for all $\mathbf{j} \in \Pi$ states the exclusion of isolated interior grid points.

5 Connectivity

At low resolution and a high enough cut-off level κ, the high-level region $\Omega_\kappa \stackrel{def}{=} \{\mathbf{j} : \rho(\mathbf{j}) > \kappa\}$ is expected to consist of a small number of connected components, which should be equal to the number of molecules inside the unit cell [9]. At lower cut-off level these components merge into fewer regions. So it is possible to give an upper bound for the number of molecules in advance.

We define a graph representing properties of the binary grid electron density maps. Let $G_\Pi = (V_\Pi, E_\Pi)$ be an undirected graph with $M = M_1 \cdot M_2 \cdot M_3$ vertices denoted by $v_\mathbf{j}$, $\mathbf{j} \in \Pi$. Vertices $v_\mathbf{j} \in V_\Pi$ and $v_\mathbf{i} \in V_\Pi$ with $\mathbf{j}$ and $\mathbf{i}$ being neighbours are connected by edges, i.e., $E_\Pi = \left\{e = (v_\mathbf{j}, v_\mathbf{i}) \mid \mathbf{j} \mathfrak{n} \mathbf{i}\right\}$. Let $V^*_\Pi \subseteq V_\Pi$ be the set of vertices with a corresponding electron density above the cut-off level, i.e., the set of vertices satisfying $V^*_\Pi = \{v_\mathbf{j} \mid z_\mathbf{j} = 1,\ \mathbf{j} \in \Pi\}$. With $E^*_\Pi \subseteq E_\Pi$ we denote the set of edges in the subgraph $G^*_\Pi = (V^*_\Pi, E^*_\Pi)$ induced by V^*_Π.

The binary grid electron density distribution contains $K \in \mathbb{N}$ components, if and only if the corresponding graph $G^*_\Pi = (V^*_\Pi, E^*_\Pi)$ contains K connected components. Figure 2 shows the graph representing

the binary grid electron density distribution. Black filled vertices represent grid electron density values above the cut-off level, neighboured black vertices are connected by solid edges.

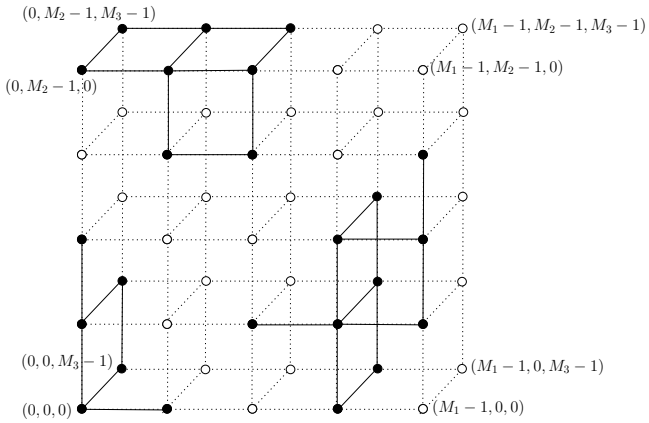


Figure 2: The graph $G^*_\Pi = (V^*_\Pi, E^*_\Pi)$

We introduce 0-1 variables $e_{\mathbf{j_1},\mathbf{j_2}}$ for $\mathbf{j_1},\mathbf{j_2} \in \Pi$ with $\mathbf{j_1} \pi \mathbf{j_2}$. These variables should take the value 1, if the corresponding edge connects two neighbouring nodes $\mathbf{j_1},\mathbf{j_2} \in \Pi$ with $z_{\mathbf{j_1}} = z_{\mathbf{j_2}} = 1$, and 0 otherwise. The constraint $-1 \leq 2e_{\mathbf{j_1}\mathbf{j_2}} - z_{\mathbf{j_1}} - z_{\mathbf{j_2}} \leq 0$, for all $\mathbf{j},\mathbf{j_1},\mathbf{j_2} \in \Pi$ with $\mathbf{j_1} \pi \mathbf{j_2}$, ensures this condition.

Now, a 0-1 linear programming approach will be presented to model that a binary grid electron density distribution satisfies the '*K-component-constraint*', i.e., it contains at most $K \in \mathbb{N}$ components. For any subset $\emptyset \neq T \subsetneq \Pi$ we introduce a binary variable u_T indicating whether T contains grid points $\mathbf{j} \in \Pi$ where the variable $z_\mathbf{j}$ takes the value 1.

$$u_T \overset{def}{=} \begin{cases} 1, & \text{if } \sum\limits_{\mathbf{j}\in T} z_\mathbf{j} \geq 1 \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

If this is the case for more than K disjoint subsets, there have to be edges connecting some of these components, otherwise the 'K-component-constraint' would be violated.

Theorem 1 ([7]). *A binary grid electron density distribution $z^* \in \{0,1\}^{M_1\times M_2\times M_3}$ contains at most K components if it satisfies the following constraints:*

$$-1 \;\leq\; 2e_{\mathbf{j_1}\mathbf{j_2}} - z_{\mathbf{j_1}} - z_{\mathbf{j_2}} \;\leq\; 0, \tag{6}$$

$$\frac{1}{|T_i|}\sum_{\mathbf{j}\in T_i} z_\mathbf{j} \;\leq\; u_{T_i} \;\leq\; \sum_{\mathbf{j}\in T_i} z_\mathbf{j}, \tag{7}$$

$$\sum_{i=1}^{K+1} u_{T_i} - K \;\leq\; \sum_{(\mathbf{j_1},\mathbf{j_2})\in\delta(T_1,\ldots,T_{K+1})} e_{\mathbf{j_1}\mathbf{j_2}} \tag{8}$$

$$u_{T_i},\, z_\mathbf{j}, e_{\mathbf{j_1}\mathbf{j_2}} \in \{0,1\}, \tag{9}$$

$$\forall \emptyset \neq T_1,\ldots,T_{K+1} \subsetneq \Pi,\; \bigcup_{i=1}^{K+1} T_i = \Pi,\; T_i \cap T_j = \emptyset,\; \forall i \neq j,\; i,j \in \{1,\ldots K+1\},$$
$$\forall \mathbf{j},\mathbf{j_1},\mathbf{j_2} \in \Pi,\; \textit{with}\; \mathbf{j_1} \pi \mathbf{j_2}.$$

Here $\delta(T_1,\ldots,T_{K+1})$ denotes the set of all edges connecting two different components T_i, T_j, with $i \neq j \in \{1,\ldots K+1\}$.

The number of constraints in (8) grows exponentially in the number of nodes. Using a separation algorithm within a branch-and-cut framework [7], only certain violated inequalities will be added to the formulation.

In the constraint programming literature, global constraints for restricting the number of connected components have been studied in [5].

6 Computational results

In order to evaluate the approach, real protein data from the Protein Data Bank [1] was taken. For the implementation, we used SCIP Version 1.2.0 [2] together with CPLEX 11.0 [4] as IP-solver. SCIP can solve mixed-integer as well as constraint integer programming problems. The running time to calculate a solution on a $6 \times 6 \times 6$-grid (216 independent grid points) on a i686 with 4 processors, a 3GHz CPU and 3GB RAM was about 10 minutes CPU time without additional constraints, and about 50 minutes CPU time with all constraints added. In the latter case, about 900 search nodes and 23MB of memory were needed, without the additional constraints 250 search nodes and 28MB of memory.

Once a set of solutions has been calculated, we evaluate the quality of those solutions. Using the minimal molecular volume that has been defined in the solution process to specify the number of non-zero grid values, the grid electron density distribution of the original protein is binarised. The distance $D(z_{exact}, z^i_{calc})$ between the resulting binary electron density z_{exact} and the calculated ones z^i_{calc}, $i \in \{1,\ldots,N\}$, where $N \in \mathbb{N}$ is the number of computed solutions, is defined by

$$D(z_{exact}, z^i_{calc}) \stackrel{def}{=} \sum_{\mathbf{j}\in\Pi} \left| z_{exact}(\mathbf{j}) - z^i_{calc}(\mathbf{j}) \right|. \tag{10}$$

The smaller the distance value, the better the quality of the considered solution. The smallest distance reached in the test run is $D_{min} \stackrel{def}{=} \min_{i=1,\ldots,N} D(z_{exact}, z^i_{calc})$.

As the exact solution normally is not known in advance, we use a method to get an average solution from the set of computed solutions. One possibility to calculate such an average solution for a set of $N \in \mathbb{N}$ solutions is the following one:

$$z_{av}(\mathbf{j}) \stackrel{def}{=} \frac{1}{N} \sum_{i=1}^{N} z^i_{calc}(\mathbf{j}),\ \forall \mathbf{j} \in \Pi, \quad D_{av} \stackrel{def}{=} D(z_{exact}, z_{av}). \tag{11}$$

Obviously, in general z_{av} is not a binary function. Using the defined molecular volume value, it can be binarised and compared to the exact solution.

Another possibility would be to choose the solution with a minimum distance from all other solutions. For every solution, the distances to all others are summed up, the solution for which this sum is minimal is chosen as reference solution z_{ref}:

$$D_{sum}(i) \stackrel{def}{=} \sum_{j=1}^{N} \sum_{\mathbf{j}\in\Pi} \left| z^i_{calc}(\mathbf{j}) - z^j_{calc}(\mathbf{j}) \right|,\ \forall i \in \{1,\ldots,N\}, \tag{12}$$

$$z_{ref} \stackrel{def}{=} z^i_{calc},\ \text{with } D_{sum}(i) = \min_{j=1,\ldots,N} D_{sum}(j), \quad D_{ref} \stackrel{def}{=} D(z_{exact}, z_{ref}). \tag{13}$$

In the table below some test results on $6 \times 6 \times 6$-grids are shown, based on the data for Protein G [1]. In order to get reasonable running times, a small grid size was chosen. For real applications it would be

desirable to handle bigger grid sizes. A covering of 30% is forced, the original binary electron density distribution then consists of 1 component. The 70 best solutions (with respect to the objective function specified in [3]) were considered. Only 28 of them also consisted of 1 component, 49 of them consisted of at most 2. The maximum number of components in one of these 70 solutions was 9.

Constraints	# sol	p_{min}	p_{av}	p_{ref}
none	70	72%	56%	54%
iso	67	72%	62%	54%
connected (2)	49	72%	66%	63%
connected (1)	28	72%	74%	65%
iso, connected (2)	49	72%	69%	68%
iso, connected (1)	28	72%	74%	70%

In the first column, the used additional constraints are specified: either only the constraint excluding isolated points (iso), or the constraint excluding isolated points and the 'K-component-constraint' (connected). In brackets the maximum number of components allowed is specified. The second column shows the number of solutions from the original solution set satisfying these constraints.

In the other columns, the percentage of correct solution values is given for the different distance measures, i.e.,

$$p_{min} = \frac{|\Pi| - D_{min}}{|\Pi|},\ p_{av} = \frac{|\Pi| - D_{av}}{|\Pi|},\ p_{ref} = \frac{|\Pi| - D_{ref}}{|\Pi|}. \tag{14}$$

Obviously, the values of p_{av} as well as p_{ref} increase by adding stricter constraints, showing the increasing quality of the regarded solutions.

7 Conclusions and further work

Based on the 0-1 linear programming approach to model the phase problem presented at WCB 2008 [3], we derived a way to model additional 0-1 linear programming constraints representing geometric properties of proteins. First results show that adding those to the original 0-1 program results in a higher quality of the set of solutions. Now, this approach will be tested on more data and also on bigger grids. Concerning future work, one could think of better ways to create a solution from the resulting solution set or of including further constraints.

References

[1] Protein Data Bank, 2010. `http://www.rcsb.org`.

[2] T. Achterberg. *Constraint Integer Programming*. PhD Thesis, TU Berlin, July 2007.

[3] C. Brinkmann and A. Bockmayr. A constraint-based approach to the phase problem in X-ray crystallography. In *WCB08 - Workshop on Constraint Based Methods for Bioinformatics, Paris*, 2008.

[4] CPLEX. *CPLEX*. ILOG, 2010. `http://www.ilog.com/products/cplex/`.

[5] G. Dooms. *The CP(Graph) Computation Domain in Constraint Programming*. PhD Thesis, Université Catholique de Louvain, Sept 2006.

[6] J. Drenth. *Principles of protein X-ray crystallography*. Springer-Verlag, 1994.

[7] C. Heldt. Constraint based methods for phasing in crystallography. PhD Thesis, FU Berlin, in preparation.

[8] V. Y. Lunin, A. Urzhumtsev, and A. Bockmayr. Direct phasing by binary integer programming. *Acta Crystallogr A*, 58(Pt 3):283–291, May 2002.

[9] N. Lunina, V. Y. Lunin, and A. Urzhumtsev. Connectivity-based ab initio phasing: from low resolution to a secondary structure. *Acta Crystallogr D Biol Crystallogr*, 59(Pt 10):1702–1715, Oct 2003.

Building Portfolios for the Protein Structure Prediction Problem

Alejandro Arbelaez
Microsoft-INRIA joint-lab,
Orsay, France
alejandro.arbelaez@inria.fr

Youssef Hamadi
Microsoft Research,
Cambridge, United Kingdom,
LIX École Polytechnique,
F-91128 Palaiseau, France
youssefh@microsoft.com

Michele Sebag
Project-team TAO,
INRIA Saclay Île-de-France,
LRI (UMR CNRS 8623),
Orsay, France
michele.sebag@inria.fr

Abstract

This paper, concerned with the protein structure prediction problem, aims at automatically selecting the Constraint Satisfaction algorithm best suited to the problem instance at hand. The contribution is twofold. Firstly, the selection criterion is the quality (minimal cost) in expectation of the solution found after a fixed amount of time, as opposed to the expected runtime. Secondly, the presented approach, based on supervised Machine Learning algorithms, considers the original description of the protein structure problem, as opposed to the features related to the SAT or CSP encoding of the problem.

1 Introduction

The protein structure prediction problem has been widely studied in the field of bioinformatics, because the 3D conformation of a given protein helps to determine its function. This problem is usually tackled using simplified models such as HP-models in [2] and a constraint logic programming approach in [5], however even considering these abstractions the problem is computationally very difficult and traditional strategies cannot reach a solution within a reasonable time. Also, there has been several attempts to predict the structure and proteins fold using well known machine learning techniques.

In this paper, we propose to use machine learning to automatically select the most promising Constraint Optimization algorithm for the protein structure prediction problem. In this context, proteins are represented as a feature vector in $\mathbf{R}^d$ and the algorithm selection process is based on a well known machine learning technique called *decision tree* which predict the most appropriate variable/value selection strategy used by a branch-and-bound algorithm in order to determine the three dimensional (3D) conformation of a given protein.

Unlike other portfolio-based selection approaches [7] which select the algorithm which minimizes the expected runtime, our work selects the strategy which minimizes the expected cost of the solution found after a fixed amount of time. To the best of our knowledge, this is the first work which performs algorithm selection in an optimization setting. Moreover, and unlike previous works which extract the features exploited during machine learning from the SAT or CSP encoding of the problem, our work uses features directly formulated in the application domain. Again, to the best of our knowledge it is the first time that domain-based features are used to predict the performance of a search algorithm.

The paper is organized as follows. Background material is presented in Section 2. Section 3 presents the general idea of algorithms portfolio. Section 4 shows the features or attributes used to describe proteins. Section 5 reports our experimental validation and Section 6 presents some concluding remarks and future research directions.

2 Background

2.1 Constraint Optimization Problems

A Constraint Optimization Problem (COP) is a tuple (X, D, C, f) where, X represents a set of variables, D a set of associated domains (i.e., possible values for the variables), C a finite set of constraints and $f(X)$ is a function to optimize. Solving a COP involves finding a value for each variable whose $f(X)$ is maximal (or minimal). A backtracking branch-and-bound algorithm is usually used to tackle COPs, at each node in the search tree a variable/value pair is used in cooperation with a look-ahead algorithm which narrows the domain of the variables.

In this paper, we consider six well known variable selection heuristics. The *lexico* heuristic selects the first unassigned variable (from left to right) in the list of decision variables, *mindom* selects the variable with minimal domain, *wdeg* [3] selects the variable which is involved in the highest number of failed constraints, *dom-wdeg* selects the variable which minimizes the ratio $\frac{dom}{wdeg}$, *impacts* [9] selects the variable/value pair which maximizes the reduction of the remaining search space and *domFD* [1] selects the variable that minimizes the ratio $\frac{dom}{FD}$ where *FD* represents the total number of weak functional dependencies of a given variable.

2.2 The protein structure prediction problem

The protein structure prediction problem is well known in computational biology and is currently considered as one of the "grand challenges" in this field. Broadly speaking the problem consists in finding the 3D conformation (so-called tertiary structure) of a protein defined by its primary structure or a sequence of residues $S = \{s_1, s_2, \ldots, s_n\}$ where each residue s_i of the sequence represents one of the 20 amino acids. The ternary structure is often defined by the minimal energy conformation.

This problem has been previously studied in [6] using a constraint programming based model. In this model, each amino acid is seen as a single atom unit and two consecutive amino acids in the sequence are separated by a fixed distance also known as a lattice unit. The energy is defined by minimizing the following formula:

$$E(w) = \sum_{1 \leq i < n} \sum_{i+2 \leq j \leq n} contact(w(i), w(j)) \times Pot(s_i, s_j)$$

where, $w(i)$ denotes the current position of the amino acid s_i in the three dimensional space of a given amino acid, *contact* is 1 iff two amino acids are immediate neighbors in the three dimensional cube (or lattice) and not sequential in the primary structure, otherwise contact is set to 0, and *pot* defines the energy contribution of two adjacent residues. It is also important to note that some other lattice models have been proposed such as [2] where each amino acid in the sequence is translated from the 20 symbols alphabet into a two symbols alphabet (i.e., hydrophobic (H) and polar (P)).

2.3 Supervised machine learning

Supervised Machine Learning exploits data labelled by the expert to automatically build hypothesis emulating the expert's decisions. Formally, a learning algorithm processes a training set $\mathscr{E} = \{(x_1, y_1), ..., (x_n, y_n)\}$ where x_i is the example description (e.g., a vector of features, $\Omega = \mathbf{R}^d$) and y_i is the associated output. The output can be a numerical value (i.e., regression) or a class label (i.e., classification).

The learning algorithm outputs a hypothesis $f : \Omega \mapsto Y$ associating to each example description x a desirable output y.

3 Algorithm portfolios

Many portfolio of algorithms have been proposed to select the best technique in order to process a given instance according to its features or descriptors. A classical portfolio is learned by taking into account the overall computational time to solve a set of problem instances. For instance SATzilla [10] a well known portfolio for SAT problems builds a regression model in order to learn the solving time of each constitutive SAT solver. In this way, once an unseen instance arrives SATzilla selects the algorithm with minimal expected run-time.

Solving a constraint optimization problem involves finding the best solution and prove that the solution is the optimal one. Unfortunately, in many cases this process cannot be completed within a reasonable amount of time and the system must provide to the user the best solution found so far. Following this idea, building the portfolio using algorithm's runtime is not an alternative. A solution would be building the portfolio taking into account the quality or cost of the solution found after some fixed amount of computational time (e.g., time-out parameter). In the following, we are going to use this technique to predict the cost of the solution found after 5 minutes for the protein structure prediction problem.

4 Features

The vector of feature was extracted from the extensive machine learning literature on protein fold prediction [8]. In order to build the feature set, every amino acid in the primary structure is replaced by the index 1, 2 or 3 according to the group it belongs to, i.e., Hydrophobicity, Volume, Polarity and Polarizability (see Table 1). For instance, the following sequence RSTVVH is encoded as 122332 based on the hydrophobicity attribute. This encoding is used to compute the following set of descriptors:

- Composition: 3 descriptors representing the percentage of each group in the sequence.
- Transition: 3 descriptors representing the frequency with which a residue from $group_i$ is followed by a residue from $group_j$ (or vice-versa).
- Distribution: 15 descriptors representing the fraction in the sequence where the first residue, 25%, 50%, 75% and 100% of the residues are contained for each encoding in Table 1.

Attribute	Group 1	Group 2	Group 3
Hydrophobicity	R,K,E,D,Q,N	G,A,S,T,P,H,Y	C,V,L,I,M,F,W
Volume	G,A,S,C,T,P,D	N,V,E,Q,I,L	M,H,K,F,R,Y,W
Polarity	L,I,F,W,C,M,V,Y	P,A,T,G,S	H,Q,R,K,N,E,D
Polarizability	G,A,S,D,T	C,P,N,V,E,Q,I,L	K,M,H,F,R,Y,W

Table 1: amino acid feature's group

In total the feature set is a composition of 105 descriptors: 84 ((15+3+3)×4) according to Table 1, 20 descriptors which represent the proportion of each amino acid in the sequence, and finally the size of the sequence.

5 Experiments

In this paper, we used the Gecode model proposed in [4]. All algorithms (see section 2.1) are home-made implementations integrated into the Gecode-2.1.1[1] constraint solver. We experimented with 400 random sequences of sizes ranging from 20 to 99, and performed 10-fold cross validation to evaluate the model. The timeout of each run was set to 5 minutes, which means that the objective of the machine learning part was to predict the strategy which would provide a solution of minimal cost after 5 minutes.

Initial experiments suggested that *lexico* is a powerful heuristic for the protein structure prediction problem, therefore we explored an extension of traditional variable selection algorithms, this novel version is presented as follows:

1. Select the first unassigned variable X_i if and only if X_{i+1} is assigned.
2. If the previous step cannot be satisfied, then select the variable according to a given heuristic criteria (e.g., *dom-wdeg*, *domFD*, etc)

The algorithms which follow the strategy mentioned above would be named as: *dom-wdeg*$^+$, *wdeg*$^+$, *domFD*$^+$ and *impacts*$^+$. Overall, we are considering a set of 10 variable selection heuristics $\mathscr{H}_{var} = \{$ *lexico*, *mindom*, *dom-wdeg*, *wdeg*, *domFD*, *impacts*,*dom-wdeg*$^+$, *wdeg*$^+$, *domFD*$^+$, *impacts*$^+$ $\}$ and 2 value selection algorithms $\mathscr{H}_{val} = \{$ *min-val*, *med-val* $\}$ would lead to 18 heuristics candidates ($8 \times 2 + 2$) (notice that *impacts* and *impacts*$^+$ are variable-value selection techniques). Nevertheless the majority of the candidates are low-quality heuristics that were almost always dominated by the top heuristics. In this way, after eliminating these weak ones, the portfolio is build on top of the following heuristic set $\mathscr{H} = \{$ $\langle$*lexico*,*min-val*$\rangle$, $\langle$*domFD*$^+$,*med-val*$\rangle$, $\langle$*wdeg*,*med-val*$\rangle$, $\langle$*wdeg*$^+$,*med-val*$\rangle$ $\}$. A fixed time limit of 5 minutes was used for each experiment.

In this paper, we experimented with the following learning schemes:

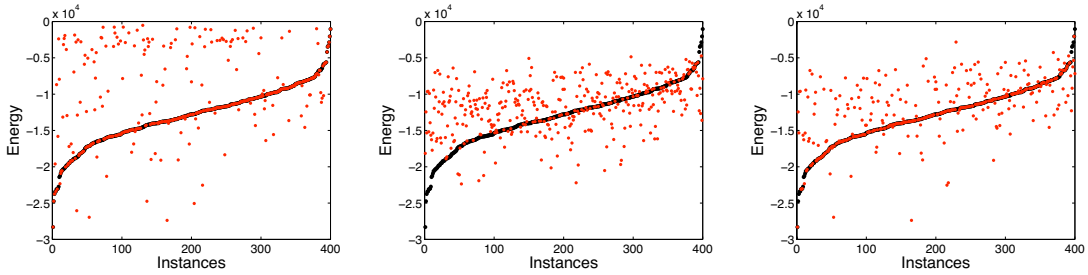
- J48[2]: weka implementation of the C4.5 algorithm. Although J48 supports continuous features values we experimentally found that including a feature discretization step improved the accuracy of the machine learning algorithm.
- SATzilla based approach: We used the code proposed in [7] to build the linear regression model, it includes an important feature pre-processing phase (i.e., pairwise feature composition and forward feature selection).

Fig 1 shows our overall experimental results. Each black point represents the performance of J48-portfolio for a given instance and each red point represents the performance of each comparative algorithm (i.e., *wdeg*$^+$, *domFD*$^+$ and SATzilla-based portfolio) for a given instance. For analysis purposes, data have been sorted according to the performance of J48-portfolio. Notice that since the optimization goal is to find the minimal energy configuration, red points above the black ones indicate that J48-portfolio is better.

Fig 1(a) shows the performance of *wdeg*$^+$ against the portfolio, in this figure J48-portfolio is better than *wdeg*$^+$ in 110 instances (resp. worse in 43 instances). Fig 1(b) shows the performance of *domFD*$^+$ against J48-portfolio, here J48-portfolio is better in 231 instances and worse in 127 instances, and finally Fig 1(c) shows the performance of J48-portfolio against the SATzilla based portfolio, in this case J48-portfolio is better than SATzilla-portfolio in 127 instances and worse in 66 instances.

[1] www.gecode.org

[2] www.cs.waikato.ac.nz/ml/weka



(a) wdeg$^+$ vs J48 (b) domFD$^+$ vs J48 (c) SATzilla vs J48

Figure 1: Overall evaluation

6 Conclusions and future work

In this paper, we have studied the application of Machine learning techniques to build algorithms portfolios in the context of the protein structure prediction problem, we have shown that using machine learning might help to select promising algorithms improving the overall performance of the system.

Currently, we manually select the algorithms of the portfolio. Part of our future work consists in automatically selecting algorithms using racing techniques (i.e., F-RACE) during the training phase, the idea would be to remove the algorithms that are statistically worse than the rest.

References

[1] Alejandro Arbelaez and Youssef Hamadi. Exploiting weak dependencies in tree-based search. In *ACM Symposium on Applied Computing (SAC)*, pages 1385–1391, Honolulu, Hawaii, USA, March 2009. ACM.

[2] Rolf Backofen and Sebastian Will. Fast, constraint-based threading of HP-sequences to hydrophobic cores. In *CP*, volume 2239 of *LNCS*, pages 494–508, Paphos, Cyprus, Nov 2001. Springer.

[3] Frederic Boussemart, Fred Hemery, Christophe Lecoutre, and Lakhdar Sais. Boosting systematic search by weighting constraints. In *ECAI*, pages 146–150, Valencia, Spain, Aug 2004. IOS Press.

[4] Raffaele Cipriano, Alessandro Dal Palù, and Agostino Dovier. A hybrid approach mixing local search and constraint programming applied to the protein structure prediction problem. In *Workshop on Constraint Based Methods for Bioinformatics (WCB)*, Paris, France, 2008.

[5] Alessandro Dal Palù, Agostino Dovier, and Federico Fogolari. Protein folding in clp(fd) with empirical contact energies. In *CSCLP*, volume 3010 of *LNCS*, pages 250–265, Budapest, Hungary, June 2003. Springer.

[6] Alessandro Dal Palù, Agostino Dovier, and Enrico Pontelli. A constraint solver for discrete lattices, its parallelization, and application to protein structure prediction. *Softw. Pract. Exper.*, 37(13):1405–1449, 2007.

[7] Frank Hutter, Youssef Hamadi, Holger H. Hoos, and Kevin Leyton-Brown. Performance prediction and automated tuning of randomized and parametric algorithms. In *CP*, volume 4204 of *LNCS*, pages 213–228, Nantes, France, Sept 2006. Springer.

[8] Nikhil R. Pal and Debrup Chakraborty. Some new features for protein fold prediction. In *ICANN*, volume 2714, pages 1176–1183, Istanbul, Turkey, June 2003. Springer.

[9] Philippe Refalo. Impact-based search strategies for constraint programming. In *CP*, volume 2004 of *LNCS*, pages 557–571, Toronto, Canada, Sept 2004. Springer.

[10] Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. The design and analysis of an algorithm portfolio for sat. In *CP*, volume 4741 of *LNCS*, pages 712–727, Providence, RI, USA, Sept 2007. Springer.

Lattice model refinement of protein structures

Martin Mann
Bioinformatics, University of Freiburg, Germany,
mmann@informatik.uni-freiburg.de

Alessandro Dal Palù
Dip. di Matematica, Università di Parma, Italy,
alessandro.dalpalu@unipr.it

Abstract

To find the best lattice model representation of a given full atom protein structure is a hard computational problem. Several greedy methods have been suggested where results are usually biased and leave room for improvement.

In this paper we formulate and implement a Constraint Programming method to refine such lattice structure models. We show that the approach is able to provide better quality solutions. The prototype is implemented in COLA and is based on limited discrepancy search. Finally, some promising extensions based on local search are discussed.

1 Introduction

Extensive structural protein studies are computationally not feasible using full atom protein representations. The challenge is to reduce complexity while maintaining detail [6, 11]. Lattice protein models are often used to achieve this but in general only the protein backbone or the amino acid center of mass is represented [1, 16, 18, 20, 26]. A huge variety of lattices and energy functions have previously been developed [5, 8, 28], while the lattices 2D-square, 3D-cubic and 3D face centered cubic (FCC) are most prominent.

In order to evaluate the applicability of different lattices and to enable the transformation of real protein structures into lattice models, a representative lattice protein structure has to be calculated. In detail, given a full atom protein structure one has to find the best structure representation within the lattice model that minimizes the applied distance measure. Maňuch and Gaur have shown the NP-completeness of this problem for backbone-only models in the 3D-cubic lattice when minimizing coordinate root mean square deviation (cRMSD) and named it the *protein chain lattice fitting (PCLF) problem* [19].

The PCLF problem has been widely studied for backbone-only models. Suggested approaches utilize quite different methods, ranging from full enumeration [4], greedy chain growth strategies [17, 20, 23], dynamic programming [10], simulated annealing [25], or the optimization of specialized force fields [13, 27]. The most important aspects in producing lattice protein models with a low root mean squared deviation (RMSD) are the lattice co-ordination number and the neighborhood vector angles [23, 24]. Lattices with intermediate co-ordination numbers, such as the face-centered cubic (FCC) lattice, can produce high resolution backbone models [23] and have been used in many protein structure studies (e.g. [11, 12, 29]).

Most of the PCFL methods introduced are heuristics to derive good solutions in reasonable time. Greedy methods as chain growth algorithms [17, 20, 23] enable low runtimes but the fitting quality depends on the chain growth direction and parameterization. Thus, resulting lattice models are biased by the method applied and have potential for refinement.

This paper has the goal to provide some evidence that greedy methods can be effectively improved by subsequent refinement steps that increase the fitting quality. We present a formalization and a simple working prototype. Moreover we briefly discuss some potential methodologies that we expect could be effectively employed.

2 Definitions and Preliminaries

In order to define the Constraint Programming approach we first introduce some preliminary formalisms.

Given a protein in full atom representation of length n (e.g. in Protein Data Base (PDB) format [2]), we denote the sequence of 3D-coordinates of its C_α-atoms (its *backbone trace*) by $P = (P_1, \ldots, P_n)$.

A regular *lattice* L is defined by a set of neighboring vectors $\vec{v} \in N_L$ of equal length $(\forall_{\vec{v}_i, \vec{v}_j \in N_L} : |\vec{v}_i| = |\vec{v}_j|)$, each with a reverse $(\forall_{\vec{v} \in N_L} : -\vec{v} \in N_L$, such that $L = \{\vec{x} \mid \vec{x} = \sum_{\vec{v}_i \in N_L} d_i \cdot \vec{v}_i \wedge d_i \in \mathbb{Z}_0^+\}$. $|N_L|$ gives the coordinate number of the lattice L, e.g. 6 for 3D-cubic or 12 for the FCC lattice. All neighboring vectors $\vec{v} \in N_L$ of the used lattice L are scaled to a length of $3.8\mathring{A}$, which is the mean distance between consecutive C_α-atoms in real protein structures.

A backbone-only *lattice protein structure* M of length n is defined by a sequence of lattice nodes $M = (M_1, \ldots, M_n) \in L^n$ representing the backbone (C_α) monomers of each amino acid. A valid structure ensures backbone connectivity $(\forall_{i<n} : M_i - M_{i+1} \in N_L)$ as well as selfavoidance $(\forall_{i \neq j} : M_i \neq M_j)$, i.e. it represents a selfavoiding walk (SAW) in the underlying lattice.

The *PCFL problem* is to find a lattice protein model M of a given protein's backbone P, such that a distance measure between M and P (dist(M,P)) is minimized [19].

In this contribution, we tackle the *PCFL refinement problem*. Here, a protein backbone P as well as a first lattice model M is given, e.g. derived by a greedy chain growth procedure [17, 20, 23]. The problem is to find a lattice model M', such that dist$(M',P) <$ dist(M,P), via a relaxation/refinement of the original model M.

In the following, we utilize distance RMSD (dRMSD, Eq. 1) as the distance measure dist(M,P). dRMSD is independent of the relative orientation of M and P since it captures the model's deviation from the pairwise distances of C_α-atoms in the original protein. Minimizing this measure optimizes the lattice model obtained.

$$\text{dRMSD}(M,P) \quad = \quad \sqrt{\frac{\sum_{i<j} \left(|M_j - M_i| - |P_j - P_i|\right)^2}{n(n-1)/2}} \tag{1}$$

3 Refinement of Lattice Models: a Constraint Model in COLA

In this section we formalize a Constraint Optimization Problem (COP) to solve the PCFL refinement problem (see Sec. 2), i.e. to refine a lattice model M of a protein P. The input is the original protein P and its lattice model M to be refined. The output is a lattice model M' derived from M via some relaxation that optimizes our distance measure dRMSD(M',P) (Eq. 1).

We first formalize the problem and show how to implement it in COLA, a COnstraint solver for LAttices [21]. This is followed by an altered formulation that utilizes limited discrepancy search [9].

3.1 The Constraint Optimization Problem

The COP can be formalized as follows:

$X_1 \dots X_n$	variables representing $M' = (M'_1, \dots, M'_n)$
$D(X_i)$	variable domains $= \{v \mid v \in L \wedge \lvert v - M_i \rvert \leq f_{\text{scale}} \cdot d_{\max}\}$, i.e. an M_i surrounding sphere with radius $f_{\text{scale}} \cdot d_{\max}$
$SAW(X_1 \dots X_n)$	self-avoiding walk constraint, e.g. split into a chain of binary `contiguous` and a global `alldifferent` constraint
O	objective function variable, implements dRMSD $= \sum_{i<j}(\lvert X_j - X_i \rvert - \lvert P_j - P_i \rvert)^2$ to be minimized

Note that $d_{\max}$ refers to the number of lattice units used and thus it is scaled to the correct distance of $f_{\text{scale}} = 3.8$Å. Thus, the domains for $d_{\max} = 0$ only contain the original lattice point M_i (domain size 1), while $d_{\max} = 1$ results in M_i as well as all neighbored lattice points (domain size $1 + 12 = 13$ in FCC). The domain size guided by $d_{\max}$ defines the allowed relaxation of the original lattice model M to be refined. For more details about global constraints for protein structures on lattices, the reader can refer to [1, 22].

The COLA implementation takes advantage of the availability of 3D lattice point domains and distance constraints. The implementation changes the original framework only in the input data handling and objective function definition. A working copy of COLA and the COP implemented for this paper are available at `http://www2.unipr.it/~dalpalu/COLA/`

3.2 Limited Discrepancy Search

A simple enumeration with $d_{\max} = 1$ and a protein of length 50, already shows that the search space of the COP from the previous section is not manageable. In this example, each point can be placed in 13 different positions in the FCC lattice, and even if the contiguous constraint among the amino acids is enforced, the number of different paths is still beyond the current computational limits.

We tried a simple branch and bound search an $X_1, \dots, X_n$, where the dRMSD bound is estimated by considering the possible placement of non labeled variables and the best dRMSD contribution provided by each amino acid. In detail, each amino acid s not yet labeled is compared to each other amino acid (s'). Each pair provides a range of different contributions to dRMSD measure, depending on the placement of s and the placement of the other amino acids (when not yet labeled). A closed formula computation (rather than a full enumeration of all combinations), based on bounding box of domain positions, is activated, in order to estimate the minimal contribution. Clearly, this estimation is not particularly suited, since we relax the estimation on $\mathbb{R}^3$, where the null (best) contribution can be easily found as soon as the bounds on $\lvert X_s - X_{s'} \rvert$ include the value $\lvert P_s - P_{s'} \rvert$. Unfortunately, the discrete version requires a more expensive evaluation that boils down to full pair checks. Therefore, the current bound is very loose and the pruning effects are modest.

A general impression is that the dRMSD measure presents a pathological distribution of local minima, depending on the placement of amino acids on the lattice. In general, due to the discrete nature of the lattice, the modification of a single amino acid's position can drastically vary its contributions to the measure.

Protein ID	8RXN	1CKA	2FCW
length	52	57	106

Table 1: Used proteins from the Protein Data Base (PDB) [2].

These considerations suggested us to focus on the identification of solutions that improve the dRMSD w.r.t. M rather than searching for the optimal one. In terms of approximated search we tried to capture the main characteristics of the COP and design efficient and effective heuristics.

A simple idea we tested is the *limited discrepancy* search [9]. This search compares the amino acid placements in the lattice models M and M'. Every time a corresponding amino acid is placed differently in the two conformations, we say that there is a *discrepancy*. We set a global constraint that limits the number of deviations to at most K. This allows to generate conformations that are rather similar to M, especially if $d_{\max}$ is greater than 1. The rational behind this heuristics is that we expect that potential conformations M' improve the dRMSD only when contained in a close neighborhood of the M structure.

The count of the number of discrepancies K is implemented directly in COLA at each labeling step.

3.3 Results

We summarize here the preliminary results coming from the COLA implementation of a K discrepancy search in 3D FCC lattice.

The initial lattice models to be refined were generated using the LatFit tool from the LatPack package [16, 17]. LatFit implements an efficient greedy dRMSD optimizing chain growth method and was parameterized to consider the best 100 structures from each elongation for further growth[1].

We test three proteins (Table 1) and for each of them we input the conformation M obtained from the greedy algorithm (LatFit). Table 2 reports the best dRMSD of our new model M' found depending on $d_{\max}$ and the number K of amino acids placed differently from the input conformation. Furthermore, time consumption for each parameterization is given.

Note that if either $K = 0$ or $d_{\max} = 0$ only the input structure resulting from the greedy LatFit run can be enumerated.

These results, yet preliminary, offer an interesting insight about the distribution of suboptimal solutions. It is interesting to note, e.g., that better solutions are found by allowing a rather large local neighborhood for a few amino acids ($d_{\max}$ parameter). On the other side, it seems that few modifications (K) are sufficient to alter the input sequence and obtain a better conformation.

In Figure 1 we exemplify the gain of model precision for the protein `8RNX`. Only the relaxation of $K = 4$ monomers enables the structural change that leads to a dRMSD drop from 1.2469 down to 1.0884, an improvement of about 13%. A movement of less monomers would not enable such a drastic change. This depicts the potential of a local search scheme that iteratively applies a series of such structural changes.

Investigating the time consumption (Table 2) one can see that the runtime increases drastically with K which governs the search tree size. The domain sizes implied by $d_{\max}$ do not show such an immense influence.

The behavior encountered is an indicator that a search based on exploring only the neighborhood should provide efficient and good suboptimal solutions. In the next section we briefly discuss some promising approaches that we plan to investigate.

[1] For details on the LatFit method see [17] and the freely available web interface at `http://cpsp.informatik.uni-freiburg.de`

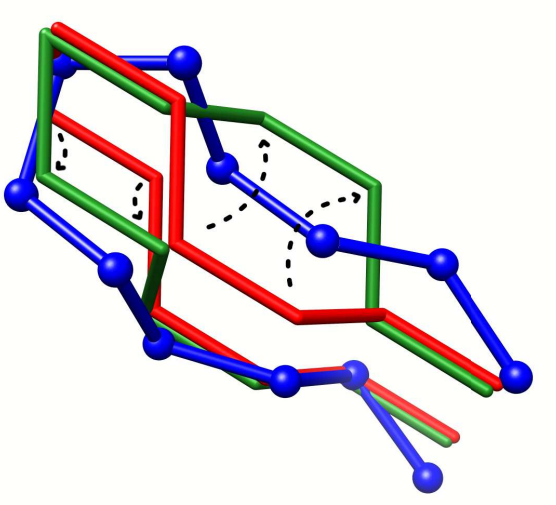


Figure 1: The initial lattice model M (red) of the protein chain P (blue, balls) and the final/refined lattice model M' (green) resulting from $d_{\max} = 2$ and $K = 4$ for protein `8RNX`. Note, only the altered loop regions (residue 2-14) are shown, but the whole structure models M and M' were superpositioned to P independently.

dRMSD

		K			
8RXN		1	2	3	4
$d_{\max}$	0	1.2469	1.2469	1.2469	1.2469
	1	1.2319	1.2172	1.1639	1.1189
	2	1.2319	1.1674	1.1596	1.0884
	3	1.2319	1.1674	1.1596	1.0884
		K			
1CKA		1	2	3	4
$d_{\max}$	0	1.2370	1.2370	1.2370	1.2370
	1	1.2226	1.2226	1.2226	1.2226
	2	1.2026	1.1887	1.1887	1.1887
	3	1.2026	1.1887	1.1887	1.1887
		K			
2FCW		1	2	3	4
$d_{\max}$	0	1.1353	1.1353	1.1353	1.1353
	1	1.1353	1.1324	1.1317	1.1309
	2	1.1321	1.1300	1.1254	1.1200
	3	1.1321	1.1300	1.1254	1.1200

time in seconds

		K			
8RXN		1	2	3	4
$d_{\max}$	0	0.048	0.081	0.040	0.039
	1	0.112	0.790	2.365	20.70
	2	0.068	0.983	6.500	106.6
	3	0.106	0.499	7.399	124.0
		K			
1CKA		1	2	3	4
$d_{\max}$	0	0.031	0.030	0.027	0.037
	1	0.402	0.615	3.442	39.27
	2	0.225	0.456	7.595	120.6
	3	0.421	0.616	8.573	140.2
		K			
2FCW		1	2	3	4
$d_{\max}$	0	0.043	0.050	0.058	0.078
	1	0.118	1.997	49.99	1128
	2	0.294	7.192	341.8	14235
	3	0.332	8.129	394.5	16140

Table 2: $d_{\max}$ and K influence on discrepancy search measured in dRMSD and time.

3.4 Future work

In our opinion, a framework that integrates CP and Local Search is particularly suited to generate fast suboptimal solutions, potentially very close to the optimal one. We identify some possible directions that we believe are excellent candidates to model and solve approximately the PCLF problem:

- **local neighboring search [3, 7]**: this technique allows to integrate Gecode and Local Search frameworks. The framework handles constraint specifications and local moves within C++ programming language;
- ***k*-local moves [25]:** the idea here is to apply structural changes on k consecutive amino acids and repeat the process in a Monte-Carlo and/or simulated annealing style.
- **side chain model [15]**: our model can be extended to include side chains and we could exploit a similar set of local moves.
- **the framework presented in [30]**: COLA is here extended and combined directly to a Local Search approach based on *pull moves* [14].

4 Conclusion

In this paper we presented a Constraint Programming based model for the refinement of lattice fitting of protein conformations. A simple branching was shown to be ineffective and a limited discrepancy search was modeled and shown to be beneficial to the identification of suboptimal solutions. A prototypical implementation in the framework COLA and some preliminary results have shown the feasibility of the method. We believe that an extension of the framework to Local Search is particularly suited for the PCLF problem at hand.

Acknowledgments This work is partially supported by PRIN08 *Innovative multi-disciplinary approaches for constraint and preference reasoning* and GNCS-INdAM *Tecniche innovative per la programmazione con vincoli in applicazioni strategiche*.

References

[1] R. Backofen and S. Will. A constraint-based approach to fast and exact structure prediction in three-dimensional protein models. *Constraints*, 11(1):5–30, 2006.

[2] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissigand I. N. Shindyalov, and P. E. Bourne. The Protein Data Bank. *Nucl. Acids Res.*, 28(1):235–242, 2000.

[3] R. Cipriano, L. Gaspero, and A. Dovier. A hybrid solver for large neighborhood search: Mixing Gecode and EasyLocal++. In *Proc. of HM'09*, pages 141–155, Berlin, Heidelberg, 2009. Springer-Verlag.

[4] D. G. Covell and R. L. Jernigan. Conformations of folded proteins in restricted spaces. *Biochemistry*, 29(13):3287–3294, April 1990.

[5] K. A. Dill. Theory for the folding and stability of globular proteins. *Biochemistry*, 24(6):1501–9, 1985.

[6] K. A. Dill, S. B. Ozkan, M.S. Shell, and T. R. Weikl. The protein folding problem. *Annual Review of Biophysics*, 37(1):289–316, 2008.

[7] I. Dotu, M. Cebrián, P. Van Hentenryck, and P. Clote. Protein structure prediction with large neighborhood constraint programming search. In *Proc of CP'08*, volume 5202 of *LNCS*, pages 82–96. Springer, 2008.

[8] A. Godzik, A. Kolinski, and J. Skolnick. Lattice representations of globular proteins: How good are they? *J Comp. Chem.*, 14(10):1194–1202, 1993.

[9] W. D. Harvey and M. L. Ginsberg. Limited discrepancy search. In *Proceedings of IJCAI'95*, pages 607–613, San Francisco, CA, USA, 1995.

[10] D. A. Hinds and M. Levitt. A lattice model for protein structure prediction at low resolution. *Proc Natl Acad Sci USA*, 89(7):2536–2540, 1992.

[11] S. Istrail and F. Lam. Combinatorial algorithms for protein folding in lattice models: A survey of mathematical results. *Commun Inf Syst*, 9(4):303–346, 2009.

[12] E. Jacob and R. Unger. A tale of two tails: Why are terminal residues of proteins exposed? *Bioinformatics*, 23(2):e225–30, 2007.

[13] P. Koehl and M. Delarue. Building protein lattice models using self-consistent mean field theory. *J. Chem. Phys.*, 108:9540–9549, June 1998.

[14] N. Lesh, M. Mitzenmacher, and S. Whitesides. A complete and effective move set for simplified protein folding. In *Proc. of RECOMB'03*, pages 188–195, 2003.

[15] M. Mann, M. A. Hamra, K. Steinhöfel, and R. Backofen. Constraint-based local move definitions for lattice protein models including side chains. In *Proc. of WCB'09*, 2009. arXiv:0910.3880.

[16] M. Mann, D. Maticzka, R. Saunders, and R. Backofen. Classifying protein-like sequences in arbitrary lattice protein models using LatPack. *HFSP J*, 2:396, 2008.

[17] M. Mann, R. Saunders, C. Smith, R. Backofen, and C. M. Deane. LatFit - producing high accuracy lattice models from protein atomic co-ordinates including side chains. (under review), 2010.

[18] M. Mann, S. Will, and R. Backofen. CPSP-tools - exact and complete algorithms for high-throughput 3D lattice protein studies. *BMC Bioinf*, 9:230, 2008.

[19] J. Maňuch and D. R. Gaur. Fitting protein chains to cubic lattice is NP-complete. *Journal of bioinformatics and computational biology*, 6(1):93–106, February 2008.

[20] J. Miao, J. Kleinseetharaman, and H. Meirovitch. The optimal fraction of hydrophobic residues required to ensure protein collapse. *J Mol. Bio.*, 344(3):797–811, 2004.

[21] A. Dal Palù, A. Dovier, and F. Fogolari. Constraint Logic Programming approach to protein structure prediction. *BMC Bioinformatics*, 5:186, 2004.

[22] A. Dal Palù, A. Dovier, and E. Pontelli. Computing approximate solutions of the protein structure determination problem using global constraints on discrete crystal lattices. *J of Data Mining and Bioinformatics*, 4(1):1 – 20, 2010.

[23] B. H. Park and M. Levitt. The complexity and accuracy of discrete state models of protein structure. *J Mol Biol*, 249:493–507, 1995.

[24] C. L. Pierri, A. Grassi, and A. Turi. Lattices for ab initio protein structure prediction. *Proteins*, 73(2):351–361, 2008.

[25] Y. Ponty, R. Istrate, E. Porcelli, and P. Clote. LocalMove: computing on-lattice fits for biopolymers. *Nucleic Acids Res*, 36(2):W216–W222, 2008.

[26] A. Renner and E. Bornberg-Bauer. Exploring the fitness landscapes of lattice proteins. In *Pac Symp Biocomput.*, pages 361–372, 1997.

[27] B. A. Reva, D. S. Rykunov, A. V. Finkelstein, and J. Skolnick. Optimization of protein structure on lattices using a self-consistent field approach. *Journal of Computational Biology*, 5(3):531–538, 1998.

[28] B. A. Reva, M. F. Sanner, A. J. Olson, and A. V. Finkelstein. Lattice modeling: Accuracy of energy calculations. *J Comp Chem*, 17(8):1025 – 1032, 1996.

[29] A. D. Ullah, L. Kapsokalivas, M. Mann, and K. Steinhöfel. Protein folding simulation by two-stage optimization. In *Proc. of ISICA'09*, volume 51 of *CCIS*, pages 138–145, 2009.

[30] A. D. Ullah and K. Steinhöfel. A hybrid approach to protein folding problem integrating constraint programming with local search. *BMC Bioinformatics*, 11(Suppl 1):S39, 2010.

Finding Minimal Reaction Sets in Large Metabolic Pathways

Takehide Soh[1,2] and Katsumi Inoue[3,2]

[1] Research Fellow of the Japan Society for the Promotion of Science
[2] Department of Informatics, The Graduate University for Advanced Studies, Chiyoda-ku, Tokyo, Japan
[3] Principles of Informatics Division, National Institute of Informatics, Chiyoda-ku, Tokyo, Japan

Abstract

In systems biology, identifying vital functions like glycolysis from a given metabolic pathway is important to understand living organisms. In this paper, we particularly focus on the problem of finding minimal sub-pathways producing target metabolites from source metabolites. We represent laws of biochemical reactions in propositional formulas and use a minimal model generator based on a state-of-the-art SAT solver. An advantage of our method is that it can treat reversible reactions represented in cycles. Moreover recent advances of SAT technologies enables us to obtain solutions for large pathways. We have applied our method to a whole *Escherichia coli* metabolic pathway. As a result, we found 5 sets of reactions including the conventional glycolysis sub-pathway described in a biological database EcoCyc.

1 Introduction

Living organisms are kept alive by a huge number of chemical reactions. In *systems biology*, interactions of such chemical reactions are represented in a network called *pathway*. Analyses of pathways have been active research field in the last decade and several methods have been proposed [7, 17]. A longstanding approach is to represent pathways as systems of differential equations. This method allows detailed analyses e.g. concentrations of each metabolite with time variation. However, it is not applicable to a large network due to its difficult parameter tuning. This is a problem because scalability is an important feature for macroscopical analyses of complex networks like cells, organisms and life, which is a fundamental goal in systems biology. Therefore other methods aiming for scalable and abstracted analyses have been proposed [2, 3, 12, 11, 15]. Although these methods are different from each others in these problem formalization and solving methods, their purpose is the same, that is, to identify biologically meaningful sets of reactions from a given pathway.

One of these methods proposed by Schuster *et al.* is called elementary mode analyses. It focuses on a flux distribution, which is computed by matrix calculus, corresponding to a set of reactions in metabolic pathways [15]. This method can treat multi-molecular reactions while taking into account stoichiometry, and its computational scalability is enough to analyze large pathways. However it tends to generate a large number of solutions without ordering e.g. over 20000 solutions for a pathway including 100 reactions. Even though found solutions are potentially interesting, analyzing all of them through biological experiments would be infeasible task. We thus need a method which generates lower number of solutions keeping its quality. Another approach relying on graphs is proposed by Croes *et al.* [3]. They formalized their problem using a weighted graph and ran a depth-first search algorithm to find the lightest paths from a source compound to a target compound. Planes and Beasley proposed to solve the same problem using a constraint-based method [11]. An advantage of these two methods is that an evaluation of the quality of the solution is provided. We can then choose an objective value to reduce the number of solutions that should be provided to biologists. However, this approach can only generate paths while sub-pathways would be a more natural representation. The approach thus sometimes generates not valid paths from a biological viewpoint because it can easily take non-meaningful shortcuts via common metabolites, such as water, hydrogen and adenosine triphosphate (ATP).

In this paper, we propose a new analysis method for metabolic pathways which finds sub-pathways producing a set of target metabolites from a set of source metabolites. In particular, we focus on finding minimal sub-pathways which has the property of not containing any other feasible sub-pathways, that is, intuitively, all elements in each minimal sub-pathway are qualitatively essential to produce target metabolites. We represent laws of biochemical multi-molecular reactions in propositional formulas and translate the problem into conjunctive normal form (CNF) formulas. We then use a model generator based on state-of-the-art SAT solver to solve the problem efficiently. Recent progresses in SAT domain now make it possible to apply our method to huge pathways. In realistic metabolic pathways, there are a lot of reversible reactions. Previous approaches thus needed pre-processing or post-processing, which is possibly costly, to deal with reversible reactions in a pathway [11, 16]. We also show how our method treats such reversible reactions by a minimal model generation. For evaluation, we compared our method with previously proposed approaches [1, 11] for a simplified pathways of *E. coli*, which consists of 880 reactions. As a result, our method identified 10 experimentally elucidated sub-pathways, while the previous methods identified at most 8 sub-pathways. We also tested our method with a whole *Escherichia coli* (*E. coli*) pathway consisting of 1777 reactions.

In the reminder of this paper, we explain propositional formulas and its minimal models in Section 2. In Section 3, we explain the sub-pathway finding problem and the difference from the previously proposed path finding problem. We show the translation from the sub-pathway finding problem into propositional formulas in Section 4. In Section 5, we show the experimental result. In Section 6 and 7 respectively discuss related work and future work.

2 Propositional Formulas and Minimal Model Generation

This section reviews propositional formulas and its *minimal models*. Let $V = \{v_1, v_2, \ldots, v_i\}$ be a set of propositional *variables*. A *literal* is a propositional variable v_i or its negation $\neg v_i$. A *clause* is a disjunction of literals. A *conjunctive normal form (CNF) formula* is a conjunction of clauses and is also identified with a set of clauses. The truth value of a propositional variable is either *true* (T) or $false$ (F). A *(partial) truth assignment* for V is a function $f : V \to \{T, F\}$. A literal v_i is said to be *satisfied* by a truth assignment f if its variable is mapped to T; a literal $\neg v_i$ is satisfied by a truth assignment f if its variable is mapped to F. A clause is satisfied if at least one of its literals is satisfied. A *model* for a CNF formula Ψ is a truth assignment f where all clauses are satisfied. Models can also be represented in the set of propositional variables to which it assigns true. For instance, the model assigning v_1 to *true*, v_2 to $false$, v_3 to *true* is presented by the set $\{v_1, v_3\}$. In this manner, we can compare two models by set inclusion. We here give the following two definitions [8]:

Definition 1. Let V_p, V_1 and V_2 be sets of propositional variables. Then, V_1 is said to be *smaller* than V_2 with respect to V_p if $V_1 \cap V_p \subset V_2 \cap V_p$ holds.

Definition 2. Let Ψ be a propositional formula, V_p a set of propositional variables, and I a model of Ψ. Then I is a *minimal model of* Ψ *with respect. to* V_p when there is no model smaller than I with respect to V_p.

Example. Suppose that Ψ is a propositional formula $(v_1 \vee v_2) \wedge (\neg v_1 \vee \neg v_2) \wedge (\neg v_2 \vee v_3)$. Then all models of Ψ are $\{v_1\}$, $\{v_2, v_3\}$, $\{v_1, v_3\}$ and the minimal models are $\{v_1\}$ and $\{v_2, v_3\}$. Please note that minimality does not depends on the number of the elements, however depends on the inclusion relation between models, in other words, we focus on subset minimal models rather than numerical minimal models. Niemelä reported a method to find the minimal models for propositional logic with ECLiPSe

Minimal Model Generation Procedure (Ψ, V_p)
begin
  $\Sigma := \emptyset$;
  loop
    (res, I) = **Solve**(Ψ) ;
    if res = `UNSAT` **then** return Σ ;
    else
      $V_x := I \cap V_p$;
      $V_y := \overline{I} \cap V_p$;
      $\Psi_c := \Psi \wedge \left(\bigvee_{x_i \in V_x} \neg x_i\right) \wedge \left(\bigwedge_{y_j \in V_y} \neg y_j\right)$;
      (res, V_c) = **Solve**(Ψ_c) ;
      if res = `UNSAT` **then** $\Sigma := \Sigma \cup \{I\}$;
      $\Psi := \Psi \wedge \left(\bigvee_{x_i \in V_x} \neg x_i\right)$;
end

Figure 1: Procedure of Generating Minimal Models

Prolog [10]. Koshimura *et al.* also reported a minimal model generator based on SAT solvers [8] which we use in this paper. They provided the following theorem [10, 8]:

Theorem 1. Let Ψ be a CNF formula, I be a model of Ψ, and V_p be a set of propositional variables. I is a minimal model of ψ with respect to V_p iff a formula $\Psi_c = \Psi \wedge \neg(x_1 \wedge x_2 \wedge \ldots \wedge x_i) \wedge \neg y_1 \wedge \neg y_2 \wedge \ldots \wedge \neg y_j$ is unsatisfiable, where $I \cap V_p = \{x_1, x_2, \ldots, x_i\}$, $\overline{I} \cap V_p = \{y_1, y_2, \ldots, y_j\}$.

We review a procedure of a minimal model generation [8] in Figure 1. The inputs of the procedure are a propositional formula Ψ and a set of propositional variables V_p. The output is a set Σ of minimal models. The function `Solve` corresponds to SAT solvers which return `SAT` and its model when a given formula is satisfiable. The function returns `UNSAT` otherwise.

3 Path Finding and Sub-pathway Finding

This section provides the detail of an existing problem *path finding problem* and formalises *sub-pathway finding problem* on which we are focusing. Let $M = \{m_1, m_2, \ldots, m_e\}$ be a set of metabolites, $R = \{r_1, r_2, \ldots, r_f\}$ a set of chemical reactions, and $A \subseteq (R \times M) \cup (M \times R)$ a set of arcs. A pathway is represented in a directed bipartite graph $G = (M, R, A)$ where M and R are two sets of nodes, A is a set of arcs. A metabolite $m \in M$ is called a *reactant* of a reaction $r \in R$ if there is an arc $(m, r) \in A$. On the other hand, a metabolite $m \in M$ is called a *product* of a reaction $r \in R$ if there is an arc $(r, m) \in A$. Let m_s be a source metabolite and m_t a target metabolite. Let $A_p = \{a_1, a_2, \ldots, a_n\}$ be a ordered set of arcs of a given pathway. In this paper, we call a simple and elementary path between m_s and m_t as a *path* which is represented in a ordered set A_p.

Example. Figure 2 shows a simple example of a directed bipartite graph representation of a pathway. Circle nodes and square nodes represent metabolites and reactions, respectively. For instance, `GLC-6-P` and `NADP` are reactants of the reaction `R1` and `PROTON` and `D-6-P-GLUCONO-DELTA-LACTONE` are products of the reaction `R1`. There are two paths between `RIBULOSE-5P` and `GAP`; one uses `R5` and `R6`, and the other uses `R4` and `R6`.

Path Finding Problem. The path finding problem which has been studied in the literature [3, 11, 12] is given as follows.

Definition 3. *Path Finding Problem*

Input Given by 6-tuple (M,R,A,w,m_s,m_t), where $M = \{m_1,m_2,...,m_e\}$ is a set of metabolites, $R = \{r_1,r_2,...,r_r\}$ is a set of reactions, $A \subseteq (R \times M) \cup (M \times R)$ is a set of arcs, $w : A \to Z^+$ is a mapping representing weights, $m_s \in M$ is a source compound and $m_t \in M$ is a target compound.

Output k-lightest paths such that $\sum_{a_i \in A_p} w(a_i) \leq k$

An important factor of the problem is a mapping w. The problem exactly corresponds to find k-shortest paths in the case of $w : A \to \{1\}$. However, solutions of the problem frequently include unexpected shortcuts [3]. To overcome this problem, Croes *et al.* proposed a mapping which is based on degree of the nodes of metabolites, that is, common compounds, such as water and hydrogen, are avoided because those have high degree. They had a comparison between three graphs. One is called raw graph which is original graph without any weights. Another one is filtered graph which omits the selected metabolites which have high degree from the raw graph. The other one is the weighted graph. Their method with the weighted graph successfully obtained better accuracy than former two graphs. An advantage of path finding approach is to be able to utilize existing graph-search algorithms. These algorithms are scalable enough to a whole pathway networks. However, there is still remaining problems of shortcuts. Figueiredo *et al.* summarised problems for path finding approach [3, 12] by a specific example [4]. For instance, graph-based approaches outputs two paths between `GLC-6-P` and `GAP` in the pathway. One is throughout `R1`, `R2`, `R3`, `R5` and `R6`. Another one uses `R4` and instead of `R5`. Although these paths give us one aspect of pathways, it is not easy to obtain reactions which are mainly occurred while target metabolites are produced. Considering a reaction `R6`, it needs `XYLULOSE-5-PHOSPHATE` and `RIBOSE-5P` as reactants. However, each obtained path cannot reflect this reaction law.

Sub-pathway Finding Problem. To overcome these problems, we propose and formalise a new problem called the *sub-pathway finding problem*. We here give additional terminology to define the problem.

Let $s : R \to 2^M$ be a mapping from a set of reactions to a set of metabolites such that $s(r) = \{m \in M | (m,r) \in A\}$ represents the set of metabolites which are needed for activating a reaction r. Let $p : R \to 2^M$ be a mapping from a set of reactions to a set of metabolites such that $p(r) = \{m \in M | (r,m) \in A\}$ represents the set of metabolites which are produced by a reaction r. Let s^{-1} and p^{-1} be inverse mappings of s and p, respectively. Let t be an integer variable representing a time and e be an integer value for a variable t. Let $M' \subseteq M$ be a subset of metabolites. A metabolite $m \in M$ is *producible* at a time $t = 0$ from M' if $m \in M'$ holds. A reaction $r \in R$ is *activatable* at a time $t = e\,(0 < e)$ from M' if $\forall m \in s(r)$ is producible at a time $t = e - 1$ from M'. A metabolite $m \in M$ is *producible* at a time $t = e\,(0 < e)$ from M' if $m \in p(r)$ holds for at least one reaction r which is activatable at a time $t = e$ from M'. If r is activatable at a time $t = e$ then r is activatable at a time $t = e + 1$. If m is producible at a time $t = e$ then m is producible at a time $t = e + 1$. Let $M_i \subseteq M$ be a subset of metabolites representing initial metabolites, $M_s \subseteq M$ a subset of metabolites representing source metabolites and $M_t \subset M$ a subset of metabolites representing target metabolites. Please note that we distinguish M_s from M_i. Every metabolite $m \in M_i$ represents universal metabolites which are always available in pathways, such that `WATER`, `ATP` and `PROTON`. On the other hand, M_s and M_t represent particular source metabolites and target metabolites in which we are interested, respectively.

Definition 4. Let π be a 6-tuple (M,R,A,M_i,M_s,M_t) and $G = \{M,R,A\}$ a graph. A sub-graph G' of G is a *sub-pathway* of π if $G' = (M',R',A')$ and it holds the following conditions (i), (ii) and (iii): (i) $M_s \subset M'$ and $M_t \subset M'$, (ii) For every $m \in M'$, m is producible from $M_i \cup M_s$ at a time $t \geq e$ for some $e \in Z^+$, (iii) For

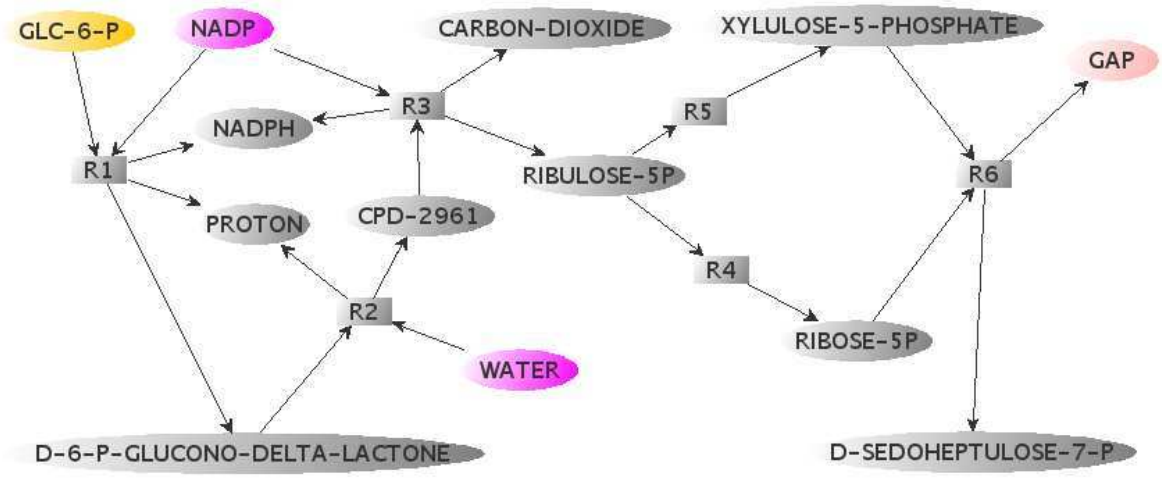


Figure 2: A Sub-pathway of the E. coli Pathway

every $r \in R'$, r is activatable from $M_i \cup M_s$ at a time $t \geq e$ for some $e \in Z^+$. In addition, a sub-pathway G' is called *minimal* if it holds that (vi) there is no sub-pathway G'' such that $G'' \subset G'$.

Definition 5. *Sub-pathway Finding Problem*

Input Given by a 6-tuple $\pi = (M, R, A, M_i, M_s, M_t)$, where $M = \{m_1, m_2, ..., m_x\}$ is a set of metabolites, $R = \{r_1, r_2, ..., r_y\}$, $A \subseteq (R \times M) \cup (M \times R)$ is a set of arcs, $M_i \subset M$ is a set of initial compounds, $M_s \subset M$ is a set of source compounds, $M_t \subset M$ is a set of target compounds.

Output All minimal sub-pathways of π.

In practice, we compute more restricted solutions of the problem since the number of all minimal sub-pathways tends to be large. We describe how to restrict solutions in the next session. The solution of the example shown in Figure 2 is different between the path finding problem and sub-pathway finding problem. While the solution of the path finding problem is two paths, the solution of the sub-pathway finding problem is the one sub-pathway: `R1`, `R2`, `R3`, `R4`, `R5` and `R6`. Please note that the reaction `R6`, `XYLULOSE-5-PHOSPHATE` and `RIBOSE-5P` are needed as reactants to produce `GAP` and `D-SEDOHEPTULOSE`. The solution correctly reflect the law of the reaction `R6`. However path finding approach returns the activation of `R6` without producing both reactants. Obviously, the output of the sub-pathway finding problem reflects a biological law of reactions and is natural representation. More details of the problem the path finding is discussed in the literature [4].

4 Translation into Propositional Formulas

4.1 Translation

This section provides a translation for a 6-tuple π. Let e be integer for a time t. Let $rt_{n,e}$ be a propositional variable which is *true* if a reaction $r_n \in R$ is activatable at a time $t = e$ and later. Let $mt_{i,e}$ be a propositional variable which is *true* if a metabolite $m_i \in M$ is producible at a time $t = e$ and later. For each reaction and metabolite, we have the following supplemental formulas ψ_s:

$$rt_{n,e-1} \rightarrow rt_{n,e}, \quad mt_{i,e-1} \rightarrow mt_{i,e}.$$

For each reaction r_n, we have the following formula representing that if a reaction r_n is activatable at a time $t = e$ then its reactants must be producible at a time $t = e - 1$.

$$rt_{n,e} \rightarrow \bigwedge_{m_i \in s(r_n)} mt_{i,e-1} \tag{1}$$

For each reaction r_n, we have the following formula representing that if a reaction r_n is activatable at a time $t = e$ then its products must be producible at a time $t = e$.

$$rt_{n,e} \rightarrow \bigwedge_{m_j \in p(r_n)} mt_{j,e} \tag{2}$$

In a naive way, above two formulas are generated by every time t for every reaction. However it results in the expansion of translated clauses. We thus need to reduce the size of the translated formulas. Let $n()$ be a mapping from a set to the number of elements of the set. A time $t = e$ is called the *earliest activatable time* of a reaction $r \in R$ if r cannot be activatable at a time $0 < t < e$ and can be activatable $e \leq t$. Let $M' = M_s \cup M_i$ be a set of metabolites. Let d and n be integers. Let R' be the set of reactions which are activatable from M'. Let T be a set of integers $\{1, \ldots, n(R')\}$. Let $f_e : R' \rightarrow T$ be a mapping from a set of reactions to a set of integers representing each reaction $r_i \in R$ and its earliest activatable time $e \in T$. The mapping f_e is also represented in a set of pairs (r_i, e) of $r_i \in R$ and $e \in T$. We here show a procedure to form the mapping f_e in Figure 3. Let d_{max} be a constant represent the output integer value d of the procedure. Please note that this procedure can be done in polynomial time. This procedure can be seen a filtering method for a given π, that is, it omits the reactions which are not activatable from M'. Moreover, the earliest activatable time is useful to reduce the size of translated formulas. If e is the earliest activatable time for a reaction r obviously do not need to consider a time $t < e$ for the reaction. However the size of translated formulas still tends to be large.

Let $f_u : R' \rightarrow T$ be a bijection from a set of reactions to a set of integers representing each reaction and its *unique time*. In Figure 4, we show a procedure to construct the bijection f_u. To complete the procedure, we need to consider how to sort a set $\{r_i \mid (r_i, d) \in f_e\}$. We use a mapping $f_s(r_i) = \Sigma_{m_j \in s(r_i)} deg(m_j)$ where $deg(m_j)$ represents the outdegree of the node m_j. We sort a set $\{r_i \mid (r_i, d) \in f_e\}$ according to increasing order of $f_s(r_i)$. This sorting procedure place reactions, which consume low outdegree metabolites, earlier. We assumed that such a reaction is easier to be activatable because it may have a less number of competitive reactions which possibly dominates the compound.

For each reaction r_n and its unique time $f_u(r_n)$, we have the third formula representing that if a reaction r_n is not activatable then metabolites $m_j \in p(r_n)$ keeps its state from a time $f_u(r_n) - 1$.

$$\neg rt_{n,f_u(r_n)} \rightarrow \bigwedge_{m_j \in p(r_n)} \left(\neg mt_{j,f_u(r_n)-1} \rightarrow \neg mt_{j,f_u(r_n)}\right) \tag{3}$$

Please note that this formula does not mean that if r_n is not activatable then metabolites $m_j \in p(r_n)$ is not producible for any time. Some of those metabolites can be made to producible at a different time by some reactions since each reaction has its unique time. We have the following formula D_{r_n}, which is given as conjunction of the formulas (1), (2) and (3):

$$\begin{aligned} D_{r_n} = & \left(rt_{n,f_u(r_n)} \rightarrow \bigwedge_{m_i \in s(r_n)} mt_{i,f_u(r_n)-1} \wedge \bigwedge_{m_j \in p(r_n)} mt_{j,f_u(r_n)} \right) \wedge \\ & \left(\neg rt_{n,f_u(r_n)} \rightarrow \bigwedge_{m_j \in p(r_n)} \left(\neg mt_{j,f_u(r_n)-1} \rightarrow \neg mt_{j,f_u(r_n)}\right) \right) \end{aligned} \tag{4}$$

Activatable Time (M')
begin
  $d := 0$;
  while ($M' \neq \emptyset$)
    mark $\forall m_i \in M'$ as visited;
    $M'' = \emptyset$;
    $d := d + 1$;
    loop for $m_i \in M'$
      loop for $r_j \in s^{-1}(m_i)$
        if $r_j \notin \{r_k \mid (r_k, n) \in f_e, n \leq d\}$ and $\forall m_k \in s(r_j)$ is visited **then**
          $f_e := f_e \cup \{(r_j, d)\}$;
          loop for $m_k \in p(r_j)$
            if m_k is not visited **then**
              $M'' := M'' \cup \{m_k\}$;
    $M' := M''$;
return (f_e, d);
end

Figure 3: Procedure for f_e

Unique Time (f_e)
begin
  $u := 0$;
  loop for $d \in \{1, \ldots, d_{max}\}$
    R_{sorted} := **sort** $\{r_i \mid (r_i, d) \in f_e\}$;
    loop for $r_j \in R_{sorted}$
      $u := u + 1$;
      $f_u := f_u \cup \{(r_j, u)\}$;
return f_u;
end

Figure 4: Procedure for f_u

According to our translation, the number of the elements of $f_u(r_n)$ is the number of reactions $n(R')$. Thus, only $n(R')$ formulas of D_{r_n} are generated. Although the size of translated formulas is enough tractable, we sometimes cannot find objective solutions since the translation is incomplete. To generate more solutions, we here extend D_{r_n} to $D^k_{r_n}$. Let o be an integer such that $o = n(R') * (k-1) + f_u(r_n)$. We then have the following formula $D^k_{r(n)}$:

$$D^k_{r(n)} = \left(rt_{n,o} \rightarrow \bigwedge_{m_i \in s(r_n)} mt_{i,o-1} \wedge \bigwedge_{m_j \in p(r_n)} mt_{j,o} \right) \wedge \left(\neg rt_{n,o} \rightarrow \bigwedge_{m_j \in p(r_n)} (\neg mt_{j,o-1} \rightarrow \neg mt_{j,o}) \right) \quad (5)$$

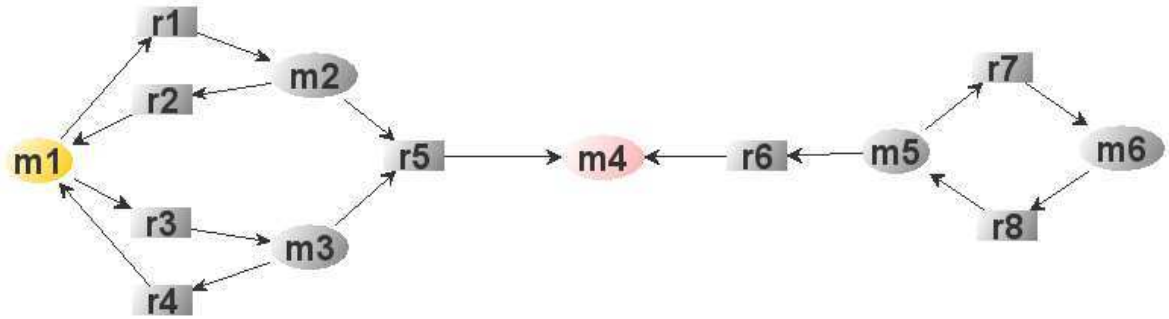


Figure 5: A Pathway Including Reversible Reactions

Obviously, D_{r_n} corresponds to $D^k_{r_n}$ when $k = 1$. Let z be an integer representing *step*. We have the following formula:

$$\bigwedge_{k=1}^{z} D^k_{r_n}. \tag{6}$$

In practice, $z = 3$ is enough to obtain the objective sub-pathways for the benchmark we used this time. We also need to have an initial condition and a target condition:

$$C(0) = \bigwedge_{m_i \in M_s \cup M_i} mt_{i,0} \wedge \bigwedge_{m_j \in M \setminus (M_s \cup M_i)} \neg mt_{j,0} \tag{7}$$

$$C(n(R') * z) = \bigwedge_{m_i \in M_t} mt_{i,n(R')*z} \tag{8}$$

Finally, we have the translated formula as follows:

$$\Psi = \bigwedge_{k=1}^{z} D^k_{r_n} \wedge C(0) \wedge C(n(R') * z) \wedge \psi_s \tag{9}$$

We use a set of propositional variables $V_p = \{mt_{i,n(R')*z} | m_i \in M\} \cup \{rt_{j,n(R')*z} | r_j \in R'\}$ to be minimized. Although we can restrict a number of solutions using a step z, sometimes there is a case that we want to reduce more number of solutions. In this case, we can choose more essential solutions by setting $V_p = \{mt_{i,n(R')*z} | m_i \in M\}$. We then compute minimal models of the formula Ψ with respect to V_p by the minimal model generation procedure shown in Figure 1.

4.2 Treate Reversible Reactions

Treatment of reversible reactions frequently becomes a problem. Ray *et al.* reported the difficulty of that and answer set semantics is suitable to resolve the problem [13]. Some other approaches took pre-processing or post-processing which breaks reversible reactions in a pathway [1, 11, 16]. Unlike those approaches, our method resolve the problem by considering the notion of activatable and finding minimal models of translated formulas. We show an toy example including reversible reactions in Figure 5. Metabolites $m1$ and $m4$ are a source metabolite and a target metabolite, respectively. We represent a model in a set of reactions to simplify explanations. A set of reactions $\{r_6, r_7, r_8\}$ cannot be a model of translated formula due to the formula (3). The formula (3) traces the origin of the producibility of

the metabolite as well as state maintenance, that is, if a metabolite is producible at a time $t = e$ then the formula (3) guarantees either the metabolite is producible at a time $t < e$ or the reaction is activatable at a time $t = e$. Therefore reversible reactions without feeding from $M_s \cup M_i$ are not activatable. Practically, such reactions are omitted in the first place by the procedure shown in Figure 3. A set of reactions $\{r_1, r_2, r_3, r_4, r_5\}$ can be a model including reversible reactions. However it cannot be a minimal model because there is a model $\{r_1, r_3, r_5\}$. Finally, we found only one minimal model $\{r_1, r_3, r_5\}$ for the example.

4.3 Other Biological Applications

Simulating Effects of Deletion of Enzymes. The method allow us to simulate the difference between pathways of wild-type organisms and pathways of mutants or gene knockout organisms. For instance, we can obtain the effect of a gene knock out by removing the reaction related to the gene we want to omit. This is achieved by adding the following formula.

$$\neg rt_{i,n(R')*z} \tag{10}$$

Simulating Effects of Inhibition. In metabolic pathways, each reaction is catalyzed by enzymes. Inhibition relations in some enzymes have been studied through biological experiments. Our method is capable to treat this relation by adding the following formula:

$$\neg rt_{i,n(R')*z} \vee \neg rt_{j,n(R')*z} \tag{11}$$

where reactions r_i and r_j are catalyzed by inhibited enzymes, respectively. This inhibition relations refine output sub-pathways of the method.

Forbidden Metabolites. A further potential application is in drug design, which restricts bi-products by the effect of compounds included in the drug. In this case, we can test by adding drug compounds as sources and unexpected bi-products as forbidden metabolites. This is achieved by adding the following formulas.

$$\bigwedge_{m_i \in M_f} \neg mt_{i,n(R')*z} \tag{12}$$

where M_f is a set of metabolites which are forbidden to present. Those constraints are useful to refine outputs when we know such forbidden metabolites in advance.

5 Experiments and Results

To evaluate our method, we use two reaction databases of *E. coli* K-12. One is the the reaction database from supplemental data of the literature [1]. Another one is from a well-known biological database *EcoCyc* [5] which gathers results of biological experiments and existence knowledge of *E. coli*. We downloaded the latest version 13.6 of the reaction database released on the November 2009. In the following experiments, we use experimentally elucidated sub-pathways as right solutions, which are respectively obtained from the literature [1] and the database *EcoCyc*. We modified the Main class of the SAT solver *Minisat*[1] [6] and used it as a minimal model generator shown in Section 2. Each

[1] `http://minisat.se/`

Table 1: Results for Pathways from [1]

Pathway#	Proposal			Beasley [1]		Planes [11]
	#Steps	#Sols.	cors.	cors. (a)	cors. (b)	cors.
1	3	1	yes	yes	no	no
2	1	1	yes	yes	no	yes
3	2	38	yes	yes	yes	no
4	1	1	yes	yes	no	no
5	3	4	yes	no	no	yes
6	2	7	yes	yes	no	yes
7	1	1	yes	yes	no	yes
8	3	28	yes	no	yes	no
9	1	3	yes	yes	no	yes
10	1	1	yes	yes	no	yes
Total # of yes in cors.			10	8	2	6

experiment has been done using a PC (Intel Centrino 1.84GHz CPU and 1GB RAM) running Ubuntu Linux 9.04 within 10 seconds. We have developed a graphical user interface integrating the proposed method, which aims for smooth evaluation. Figures 6 and 7 are screen shots of our experimental results from the interface.

5.1 Comparison with Previous Methods

There are two previous method. One is a method using optimization modeling for pathway analyses [1]. The input of this method is a reaction database with stoichiometry. Another one is a constraint based method for path finding [11]. The input of this method is a reaction database without stoichiometry as same as the proposed method. Due to the differences of each input, problem formalization and the number of solutions, it is difficult to make a direct comparison. We thus give an approximate comparison for 10 pathways which are used for both two methods. We use same source, initial, and target metabolites according to the literature [1]. As right solutions, the method by [11] used liner paths which are chosen from the experimentally elucidated sub-pathways from the supplemental data of the literature [1]. Similarly, we used those sub-pathways omitted bypass reactions as right solutions.

The results are shown in the table 1. First column shows the following pathways: #1 gluconeogenesis, #2 glycogen, #3 glycolysis, #4 proline bio-synthesis, #5 ketogluconate metabolism, #6 pentose phosphate, #7 salvage pathway deoxythymidine phosphate, #8 Kreb's cycle, #9 NAD biosynthesis, #10 arginine biosynthesis. Second column shows the number of steps where the most accurate sub-pathway was found. Third column shows the number of solutions found in the steps shown in the second column. Columns 4-7 show whether each method could find the structure exactly corresponding to the experimentally elucidated sub-pathways. In columns 5 and 6, (a) represents the objective of minimizing the total number of reactions and (b) represents the objective of maximizing the production of ATP in the literature [1]. As a result, we found every sub-pathway corresponding to the experimentally elucidated sub-pathways with the step $z \leq 3$. Moreover, the number of solutions are less than 10 except the pathway #3 and #8. Even for these two pathways, the number of solutions is less than 40, which is still tractable.

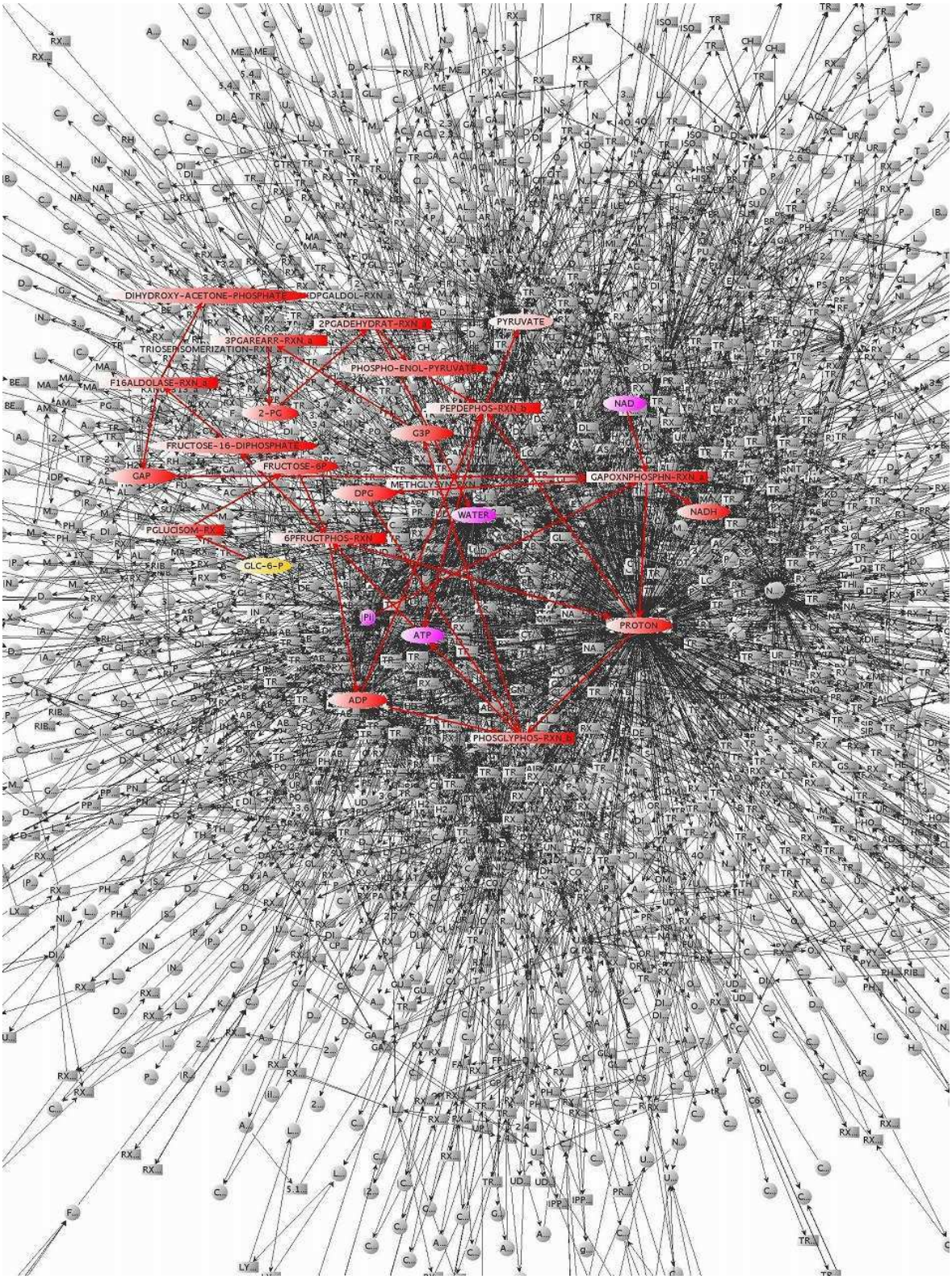


Figure 6: A Glycolysis Sub-pathway on a Whole *E. coli* Pathway

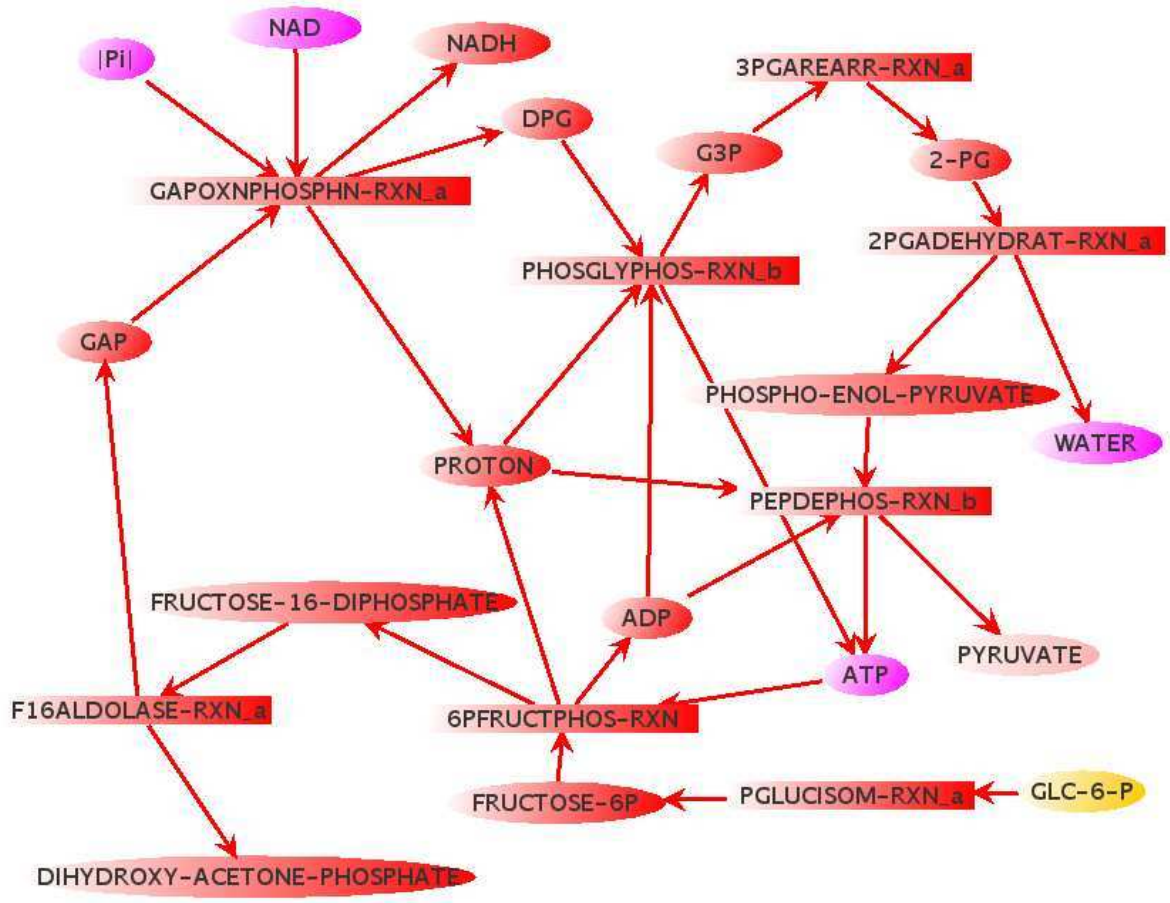


Figure 7: A Glycolysys Sub-pathway of the *E. coli* Pathway

5.2 Accuracy Evaluation on the *E. coli* Metabolic Pathway

We also apply our method to a whole metabolic pathway of *E. coli* (see Figure 6). In this experiment, we choose initial metabolites by calculating percentage of the presence of each metabolites as same as the literature [1]. For instance, in generally, a cell contains 60 percents water. In order to decide such initial metabolites we define the percentage of the presence of a metabolite $pr_m = (n_m \div n(R)) \times 100$, where n_m represents the number of reactions in which a metabolite m appears. We compute this percentage of the presence for every metabolite and if it has the presence more than 1.5 percent we use it as an initial metabolite. Although 1.5 percent is apparently small value, this is relatively high presence in the pathway because there are only 16 of 1073 metabolites which have the percentage of presence more than 1.5 percent. We particularly choose metabolites which are the first 6 of 1073 metabolites regarding its percentage of presence: `WATER`, `PROTON`, `ATP`, `ADP`, `|pi|` and `NAD`. We apply the method to find a glycolysis sub-pathway in a whole *E. coli* pathway. As a result, we found 5 sets of reactions (see Table 2). One of them called `M5` includes 8 reactions (see Figure 7) corresponding to the conventional glycolysis sub-pathway described in EcoCyc. We evaluate the obtained sub-pathway `M5` with the following evaluation value. True positive (TP) is a number of reactions found in the experimentally elucidated solution which are also part of a computational result. False positive (FP) is a number of reactions found in a computational result but not in the experimentally elucidated solution. False negative (FN) is a number of reactions found in the experimentally elucidated solution but not in a computational result. Here we

Table 2: Found Minimal Reaction Sets

Reaction Name (by [5])	M1	M2	M3	M4	M5	EcoCyc
2PGADEHYDRAT-RXN				x	x	x
3PGAREARR-RXN				x	x	x
6PFRUCTPHOS-RXN		x			x	x
6PGLUCONOLACT-RXN			x			
DLACTDEHYDROGNAD-RXN	x	x				
F16ALDOLASE-RXN		x			x	x
F16BDEPHOS-RXN						x
GAPOXNPHOSPHN-RXN				x	x	x
GLU6PDEHYDROG-RXN			x			
GLYOXIII-RXN	x	x				
KDPGALDOL-RXN			x			
METHGLYSYN-RXN	x	x				
NAD-KIN-RXN			x			
PEPDEPHOS-RXN				x	x	x
PEPSYNTH-RXN						x
PGLUCISOM-RXN	x	x		x	x	x
PGLUCONDEHYDRAT-RXN			x			
PHOSGLYPHOS-RXN				x	x	x
RXN0-313	x			x		
TRIOSEPISOMERIZATION-RXN	x					x

define sensitivity $Sn = TP/(TP+FN)$, positive predictive value $PPV = TP/(TP+FP)$ and accuracy $Ac = (Sn+PPV)/2$. As a result, we obtain $Sn = 0.727$, $PPV = 1$, $Ac = 0.864$ for the glycolysis sub-pathway. The value $PPV = 1$ means all reactions included in an obtained sub-pathway are also included in the experimentally elucidated sub-pathway. However, $Sn = 0.727$ means that some reactions included in the experimentally elucidated sub-pathway are not included in the obtained sub-pathway. This is because the experimentally elucidated sub-pathway from EcoCyc contains bypass reactions. While the sub-pathway of EcoCyc contains a bypass reaction `PEPSYNTH-RXN` which may be needed from a stoichiometry viewpoint. In the case of the glycolysis sub-pathway, `PEPSYNTH-RXN` is such a bypass reaction. To support such bypass reactions is a future work.

6 Related Work

As far as the authors are aware, the exactly same problem of the sub-pathway finding problem has not been yet formalized. Kuffer *et al.* reported an approach via translation to petri net [9]. Although their approach considered producible and activatable, did not consider subset minimality of the solution. Schuster *et al.* proposed a concept of elementary flux modes and found minimal flux distribution [15]. Their concept of *elementary flux mode* is closed to our problem, however, they used stoichiometry information to solve their problem while our problem only consider the topology of pathways. Beasley and Planes [1] used an optimization technique to stoichiometry analyses of pathways. Although their outputs are sometimes correspond to our outputs, their optimization does not guarantee the subset minimality.

Tiwari *et al.* proposed a method which uses weighted Max-SAT solver [18]. They translated biological laws into soft constraint represented in a weighted Max-SAT problem. The method thus can order solutions according to its total weights. However, their ordering of solutions is sometimes not ac-

ceptable from a biological viewpoint since it does not permit the activation of two reactions which uses same metabolite simultaneously. Ray *et al.* reported the logical approach for analyzing pathways using answer set programming (ASP) and reported how it suits for pathway analyzing. Similarly, Schaub and Thiele [14] apply ASP technique to analyze pathways. These approaches are also interesting in terms of translating the relations of reactions into logical form. As far as the authors know, there is a few methods have been reported for analyses of a whole organism pathway. We believe that our method provides a new linking method between simple graph-based approaches and those logical methods, which enables us to analyze complex networks like cells, organisms and life.

7 Conclusion

In this paper, we formalized the sub-pathway finding problem which identifies necessary reactions to produce target metabolites and presented a translation into a propositional formula. Our method uses a SAT solver as a model generator and it has the following features. First, our method can treat reversible reactions without pre-processing and post-processing. Second, it is capable to treat a whole *E. coli* pathway. Third, it can restrict the number of solutions to be tractable. These are important features for the realistic size pathways such as a whole cell or more extended pathways which includes metabolic, signalling, and gene regulatory networks. There are several important future topics. The proposed method found each conventional sub-pathways of the 11 pathways on *E. coli*. For more general evaluation, statistical analyses with more number of pathways are needed. We also need to consider the quality of solutions as well as ranking. Translating more biological knowledge is important to find sub-pathways of more extended pathways.

Acknowledgement This research is supported in part by the 2008-2011 JSPS Grant-in-Aid for Scientific Research (A) (No.20240016) and by the JSPS Research Fellowships for Young Scientists. We would like to thank Gauvain Bourgne and colleagues for their helpful comments. We also thank Oliver Ray for useful discussions.

References

[1] John E. Beasley and Francisco J. Planes. Recovering metabolic pathways via optimization. *Bioinformatics*, 23(1):92–98, 2007.

[2] Didier Croes, Fabian Couche, Shoshana J. Wodak, and Jacques van Helden. Metabolic pathfinding: inferring relevant pathways in biochemical networks. *Nucleic Acids Research*, 33(Web-Server-Issue):326–330, 2005.

[3] Didier Croes, Fabian Couche, Shoshana J. Wodak, and Jacques van Helden. inferring meaningful pathways in weighted metabolic networks. *Journal of Molecular Biology*, 356(1):222–236, 2006.

[4] Luis F. de Figueiredo, Stefan Schuster, Christoph Kaleta, and David A. Fell. Can sugars be produced from fatty acids? a test case for pathway analysis tools. *Bioinformatics*, 24(22):2615–2621, 2008.

[5] EcoCyc. `http://biocyc.org/download.shtml`.

[6] Niklas Eén and Niklas Sörensson. An extensible SAT-solver. In *Proceedings of SAT*, pages 502–518, 2003.

[7] Hidde De Jong. Modeling and simulation of genetic regulatory systems: A literature review. *Journal of Computational Biology*, 9:67–103, 2002.

[8] Miyuki Koshimura, Hidetomo Nabeshima, Hiroshi Fujita, and Ryuzo Hasegawa. Minimal model generation with respect to an atom set. In *Proceedings of FTP'09*, pages 49–59, 2009.

[9] Robert Küffner, Ralf Zimmer, and Thomas Lengauer. Pathway analysis in metabolic databases via differetial metabolic display (dmd). In *German Conference on Bioinformatics*, pages 141–147, 1999.

[10] I. Niemelä. A tableau calculus for minimal model reasoning. In *Proceedings of the TABLEAU '96*, pages 278–294, 1996.

[11] Francisco J. Planes and John E. Beasley. Path finding approaches and metabolic pathways. *Discrete Applied Mathematics*, 157(10):2244–2256, 2009.

[12] Syed Asad Rahman, P. Advani, R. Schunk, Rainer Schrader, and Dietmar Schomburg. Metabolic pathway analysis web service (pathway hunter tool at cubic). *Bioinformatics*, 21(7):1189–1193, 2005.

[13] Oliver Ray, Ken E. Whelan, and Ross D. King. Logic-based steady-state analysis and revision of metabolic networks with inhibition. In *CISIS*, pages 661–666, 2010.

[14] Torsten Schaub and Sven Thiele. Metabolic network expansion with answer set programming. In *ICLP '09: Proceedings of the 25th International Conference on Logic Programming*, pages 312–326, Berlin, Heidelberg, 2009. Springer-Verlag.

[15] Stefan Schuster, David A. Fell, and Thomas Dandekar. A general definition of metabolic pathways useful for systematic organization and analysis of complex metabolic networks. *Nature Biotechnology*, 18:326–332, 2000.

[16] Takeyuki Tamura, Kazuhiro Takemoto, and Tatsuya Akutsu. Measuring structural robustness of metabolic networks under a boolean model using integer programming and feedback vertex sets. In *CISIS*, pages 819–824, 2009.

[17] Marco Terzer, Nathaniel D. Maynard, Markus W. Covert, and Jörg Stelling. Genome-scale metabolit networks. *Systems Biology and Medicine*, 1(3):285 – 297, 2009.

[18] Ashish Tiwari, Carolyn L. Talcott, Merrill Knapp, Patrick Lincoln, and Keith Laderoute. Analyzing pathways using sat-based approaches. In *AB*, pages 155–169, 2007.

Perspectives on Constraints, Process Algebras, and Hybrid Systems

Luca Bortolussi
Dept. of Mathematics and Informatics, University of Trieste, Italy.
luca@dmi.units.it
Alberto Policriti
Dept. of Mathematics and Informatics, University of Udine, Italy.
Istituto di Genomica Applicata, Udine, Italy.
policriti@dimi.uniud.it

Abstract

Building on a technique for associating Hybrid Systems (HS) to stochastic programs written in a stochastic extension of Concurrent Constraint Programming (sCCP), we will discuss several aspects of performing such association. In particular, as we proved an sCCP program can be mapped in a HS varying in a lattice at a level depending on the amount of actions to be simulated continuously, we will discuss what are the problems involved in a semi-automatic choice of such level. Decidability, semantic, and efficiency issues will be taken into account, with special emphasis on their links with biological applications. We will also discuss about the role of constraints and of the constraint store in this construction.

1 Introduction

Systems biology emerged in recent years as the discipline promising to deepen the understanding of living beings, studying them from a systemic perspective. Interactions among constituents are considered in their concerted activity, and biological behavior is seen as an emergent property of these intricate patterns of cooperation and repression [13].

However, studying biological systems from this perspective is rather difficult for many reasons: the number of actions and interactions into play is huge, most of them are still unknown or poorly understood, the complexity of mathematical descriptions grows combinatorially, and efficiency and precision issues of models are critical.

Stochastic process algebras (SPA) [17], despite coming from the different context of performance analysis of distributed computing systems, proved to be a promising tool, inasmuch they offer a simple, compositional, description language that is automatically mapped into the complex mathematical formalism of continuous time stochastic processes, for simulation and analysis. Indeed, SPA have at their disposal automatic reasoning tools both at the syntactic and semantic levels, making them a powerful framework.

Commonly used SPA in systems biology are stochastic π-calculus [8] and bioPEPA [9]. However, their simple primitives may make difficult the description of complex interactions, like those typical of combinatorial biochemical networks [14]. Furthermore, they lack any ready-to-use computational or reasoning power, which is a limitation when facing the issue of modeling systems at greater level of detail.

Concurrent Constraint Programming [18], in its stochastic variant, **sCCP** [3], can be a way to circumvent these problems. In fact, it combines the simplicity of process algebras for describing agents with the reasoning and computing capabilities of constraints.

In our attempt to use **sCCP** as a modeling framework for biological systems, we found two main advantages:

1. the "computational twist" of constraints allow a compact description of complex operations, as those needed to describe combinatorial biochemical networks [2] or spatiality [6], without the need of introducing further primitives in the language.

2. The separation between agents and the constraint store (cf. Section 2) forces a modeling discipline that requires to separate the description of the *control logic* of the system (modeled by agents) from the description of the *configurations* of the system (naturally modeled in the constraint store).

This second aspect proved its utility as it greatly simplified the definition of an additional semantics for **sCCP** in terms of (a family of) Hybrid Automata [5], thus enhancing the mathematical instruments at disposal for analysis

However, these two features of **sCCP**, namely the "computational twist" and the flexible semantics based on hybrid automata, are somehow in conflict. In fact, the first one exploits the reasoning power of the constraint store, while the second one works when constraints are simplified to very basic interactions. Reconciling these aspects is an open problem.

In addition to the above issues, our reflection on the perspectives of usage of **sCCP** for Systems Biology must take into account realistic and effective semantics for **sCCP**. If, on the one hand, the control mechanisms are naturally interpreted in fully discrete terms, on the other hand, the stochastic as well as some of the substance-level modelling (constraint) variables, are more naturally kept continuous. This was the main reason that lead us to the above mentioned Hybrid Automata semantics, which can be seen as a way to associate **sCCP** programs to Hybrid Automata organized in a *lattice* at varying levels of discreteness.

Some of the features of such (variable) association were expected: the higher the level of discreteness to be maintained, the more adherent the program behaviour to the automaton time-evolution to be observed. Some other feature we found surprising: stochasticity can sometimes be dropped in favor of discreteness alone, with a very positive drawback in terms of simulation costs.

We conclude our discussion here putting forward a pair of open problems naturally arising in the above outlined framework:

1. Is there an *optimal* level of discreteness to be maintained when associating a hybrid automaton with a(n **sCCP**) program?

2. Can we automatically or semi-automatically address the above question by some sort of analysis of the (**sCCP**) program?

In this paper, we will not provide answers to such questions, but rather we will discuss these open problems in more detail, suggesting possible directions of attack. Before doing this, we will briefly survey the previous work on **sCCP** and on the hybrid semantics in a non-formal manner, illustrating the relevant notions by means of an example.

2 Stochastic Concurrent Constraint Programming

sCCP [3] has two basic ingredients: *agents* and *constraints*. Agents are the main actors, interacting by asynchronously exchanging information in form of *constraints*, through the *constraint store*. **sCCP** has been mainly applied as a modeling language for biological systems [3], using the constraint store to describe the state of the system, e.g. numerosity of molecular species. As these quantities evolve in time (and one is precisely interested in such a dynamics), we considered special variables, called *stream variables*, which can change value during computation (in contrast with standard logical variables, that

can be instantiated just once). At least for modelling simple biological scenarios, one needs very simple constraints, basically comparing and assigning new values to stream variables.

As an example, consider a simple genetic network, with one single gene producing a protein X at a basal rate k_p, acting as its own repressor (by binding in the promoter region of the gene). When the gene is repressed, it does not produce protein X and, after a delay, it goes back in the normal state (i.e., the repressor unbinds from the gene). Each copy of protein X is also constantly subject to degradation at rate k_d.

In order to model such a system in **sCCP**, we need one stream variable keeping track of the amount of protein X in the system. All interactions, instead, are described by **sCCP** agents. In particular, we need a simple recursively looping agent to model degradation and an agent modelling the gene. This latter agent is slightly more complex, as it describes the control mechanisms that the gene is subject to. The **sCCP** program is `gene_on` $\|$ `degrade`, where (* stands for true):

$$\begin{array}{lcl} \texttt{gene\_on} & \stackrel{\text{def}}{=} & [* \to X' = X + 1]_{k_p} \texttt{.gene\_on + } [X > 0 \to *]_{k_b X} \texttt{.gene\_off} \\ \texttt{gene\_off} & \stackrel{\text{def}}{=} & [* \to *]_{k_u} \texttt{.gene\_on} \\ \texttt{degrade} & \stackrel{\text{def}}{=} & [X > 0 \to X' = X - 1]_{k_d X} \texttt{.degrade} \end{array}$$

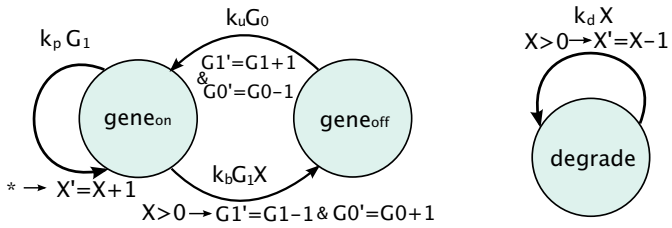
The basic actions executable by the agents above are *guarded updates* of the form $[G \to R]_\lambda$, where G is a *guard* that must be satisfied for the action to be performed and R is the *update* policy—basically a conjunction of atoms of the form $X' = X + k$. Furthermore, each action has a *stochastic duration*, given by an exponentially distributed random variable with rate depending on the state of the system through a positive real-valued function λ. Additionally, the language has standard constructs of SPA: stochastic choice $+$, parallel composition $\|$, and recursion.
The semantics of **sCCP** is given by a Continuous Time Markov Chain [16] (CTMC). Definitions and further details can be found in [3].

Remark 2.1. The constraints that can be used to update the constraint store are rather limited, as they simply add a constant term to some stream variables. This restriction, however, allows to interpret **sCCP**-actions as continuous fluxes, a required condition to define the hybrid semantics, see also Section 4. To model more involved biological systems, more complex update constraints and more complex constraint stores can/should be considered. For instance, in [2], we used a class of constraints operating on graphs and stream variables to tame the combinatorial complexity of modelling the formation of protein complexes. We will return on the issue of interfacing these two needs in Section 5.

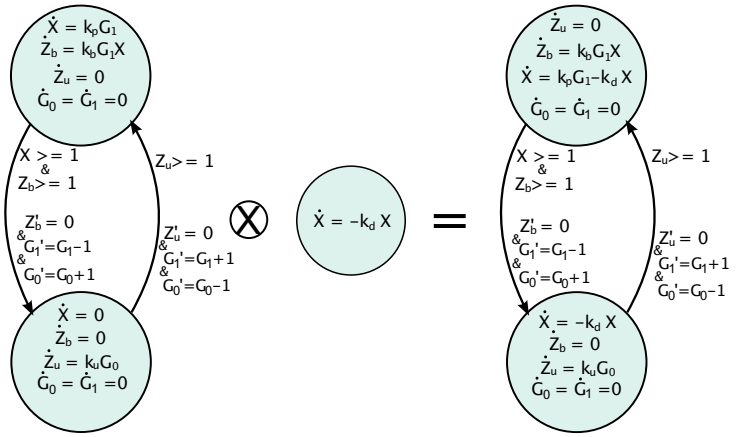
Looking again at the example, we can see that the parallel operator is used only to compose the single agents, but not within agents. When such condition is in force, we can represent **sCCP** agents as automata, synchronizing on store variables, called *Reduced Transition Systems* (RTS) [4]. The RTS of the agents of the example are shown in Figure 1(a). As can be seen, recursion is basically dealt with by introducing loops in the graphs, whose edges are labelled by rate/guard/update of the corresponding **sCCP** action. In Figure 1(a), there are also two additional variables: G_1 and G_0. They keep track of which state of the agent is the active one, with $G_1 = 1$ and $G_0 = 0$ corresponding to `gene_on` and $G_1 = 0$ and $G_0 = 1$ to `gene_off`. The updates of edges deal with such variables mimicking the program structure, while explicit dependence on G_0 and G_1 is introduced in rates (i.e. production rate becomes $k_p G_1$, because production is possible only in state `gene_on`). This is a technical trick useful to introduce the hybrid semantics.

3 Hybrid Automata

The hybrid semantics of **sCCP** will be defined in terms of Hybrid Automata (HA), see [11] for more details.



(a) Reduced Transition System



(b) Hybrid Automaton

Figure 1: (bf1(a)) Reduced Transition System for the agents of Section 2. (bf1(b)) Hybrid Automata obtained from gene example. Variables Z_b and Z_u are associated with the edge from gene$_{on}$ to gene$_{off}$ and from gene$_{off}$ to gene$_{on}$ of the RTS. See the text for a more detailed discussion on these edge variables.

The basic idea of HA is that they have a mixed discrete/continuous evolution. The discrete part of the system is described as a labelled graph, while the continuous part is modelled by an array of real-valued variables $\mathbf{X}$. In each vertex q of such a graph, called *mode*, variables are subject to a continuous evolution, usually defined by a set of ordinary differential equations (ODE) $\dot{\mathbf{X}} = F_q(\mathbf{X})$. The continuous evolution within a mode can be interrupted by the happening of a discrete event, corresponding to an edge of the graph. This event happens as soon as specific conditions on variables (described by a guard predicate) becomes true. Its execution changes the mode of the automaton (hence, also the ODE may change) and modifies discontinuously the value of variables $\mathbf{X}$, according to an edge-dependent reset policy.
Usually, compositionality of HA is achieved by a suitable definition of a HA-product, cf. [11, 7].

4 Hybrid Semantics of sCCP

In this section we informally explain the definition of the hybrid semantics for **sCCP** [5, 7]. The construction is compositional: first, single **sCCP** agents are converted into HA, then these HA are combined by taking their product.

The mapping starts from the RTS of each **sCCP** agent. The first step consists in partitioning the edges of such a RTS into two sets: those that will contribute to continuous dynamics and those that will define the discrete skeleton.

Consider again the RTS of the gene agent of Figure 1(a). It has three edges: we will treat as continuous only the looping edge on `gene_on`, corresponding to the production of X, while the ones corresponding to binding and unbinding will be dealt discretely. This is nothing but one possible choice. Think, for instance, of the case in which *all* edges are treated as continuous. Actually, all possible partitions are admissible, and the final choice is left to the modeler.
Once the edges are partitioned, we can construct the graph of the hybrid automaton. This is essentially derived from the RTS, collapsing nodes connected by continuous transitions and removing edges to be treated as continuous.
The continuous dynamics within each mode is defined according to continuous transitions connecting collapsed RTS-states. Consider the continuous transition producing protein X. It modifies only variable X, increasing it by 1 unit, with rate $k_P G_1$. The associated ODE is $\dot{X} = (+1) \cdot k_P G_1$, an it can be seen as obtained by multiplying the net increase of X by the rate. In case more than one transition is acting on a variable, their effect will be summed up.[1]
The definition of the discrete dynamics, instead, is slightly more complicated. In fact, we need to render the fact that **sCCP** actions *take time to be executed*. This is somehow un-natural for HA, in which discrete transitions are instantaneous. The idea is to introduce extra (continuous) variables to faithfully govern firing of discrete transitions. Such firing will happen when a threshold value set at the expected time of the stochastic transition is reached. Consider the RTS-edge corresponding to the repression of the gene in our running example. As discussed in [7], we can introduce a new continuous variable—Z_b in this case—and let it evolve according to $\dot{Z}_b = k_b G_1 X$, i.e. according to the rate of the stochastic transition. When Z_b reaches 1, we fire the transition and reset Z_b to zero. Z_b can be seen as a clock evolving at a non-constant speed.
In addition, each edge in the HA will be subject to the same guard and update policies of the corresponding **sCCP** action.
Once an HA has been build for each **sCCP** agent, these are composed together by a special product construction, which adds the right end sided of differential equations, cf. [7] for further details. The HA obtained for our example is shown in Figure 1(b).

The Lattice of HA. The previous construction is parametric with respect to the partitioning of **sCCP** actions into discrete and continuous. We can arrange the different HA so obtained in a lattice, where at the top element all **sCCP** actions are treated as continuous, while at the bottom element they are all kept discrete. Essentially, the fully continuous HA corresponds to the set of ODE associated to an **sCCP** program by fluid-flow approximation [4], while the fully discrete HA is a timed automata with skewed clocks (the so called Multi-Rate Timed Automata, [12]).

In this section, we assumed that the HA obtained has a (non-)deterministic evolution. We can also maintain the discrete dynamics stochastic, by simply replacing the threshold 1 in the guards of variables associated to edges by a randomly chosen threshold (exponentially distributed with rate 1) [7].

5 Perspectives

In the introduction, we argued that **sCCP** has two characterizing features that are somehow in conflict. On the one side, in order to model more complex systems, we would like to increase the complexity of constraint-based operations acting on the constraint store. On the other side, however, we want a simple form for such interactions in order to use hybrid-automata based semantics.

[1] This is the motivation for requiring constant updates of variables. In fact, it is not clear how to describe in terms of continuous fluxes other kind of updates, think for instance at $X' = k$.

How can we reconcile these two aspects? Considering the use of constraints of [2, 6], we can observe that, in all cases, the basic entity involved in modeling are stream variables. Constraints build a structure upon them to define complex manipulations and bookkeeping, in order to execute a sequence of simple operations as a single step activity.

One possibility is, therefore, to "make explicit" the constraints used, finding a low level description of the constraint store based only on stream variables, and precisely define the effect of each constraint in this new store. This construction can be hampered by combinatorial explosion of store size and even by the emergence of infinity in order to deal correctly with recursion. However, this direction is worth investigating, for it would reconcile these two conflicting aspects of **sCCP**.

Moving forward to the two open problems stated at the end of our introduction, let us observe that at the top of the lattice mentioned at the end of the previous section we have a single-mode automaton whose evolution is entirely described by a set of ODEs. Such an automaton can be seen as a full "mathematical" reduction of our initial **sCCP** program: being able to solve the ODEs we would have a complete solution of the system simulated by the **sCCP** program. The discrete dimension plays no role at the top of the lattice.

An attempt to push as down as possible this property, motivates us in the following definition:

Definition 5.1. An hybrid automaton $\mathscr{H}$ in the lattice associated to an **sCCP** program A is said to be an *optimal approximation* of A if and only if its bisimulation quotient[2] [15] is finite and every $\mathscr{H}'$ below $\mathscr{H}$ in the lattice does not enjoy this property.

On the ground of the previous discussion we have that given an **sCCP** program A there always exists a (not necessarily unique) optimal approximation of A.

Notice that it is not clear the role of stochasticity in Definition (5.1). In fact, we need to explain in some more detail the relationships intervening among continuity, discreteness and stochasticity in our construction. The key point is that, CTMC *are* decidable, in the sense that reachability is computable (one can compute the probability of reaching any subset of states with arbitrary precision), and model checking of CTL [10] formulae (or better, its stochastic version CSL [1]) is decidable. However, when we simplify a CTMC, replacing it by a Multi-Rate Automaton (MRA)[3], decidability is lost. This phenomenon is a consequence of the fact that in CTMCs time enters the picture orthogonally with respect to evolution: The choice of next state and the elapsed time before reaching it are probabilistically independent. In MRA, instead, time drives the evolution and the infinite precision involved in its density may result in the high expressiveness leading to undecidability.

The above discussion suggests that we can can tackle the open problems we proposed also focusing on the interplay among continuity, discreteness, and stochasticity. Alternatively, we can restrict our analysis on the removal of stochasticity, perhaps studying the effect of trading non-determinism and probability in our models. Again, a precise assessment of the level of decidability becomes an important benchmark for the approach.

As a final consideration we briefly comment on the following issue: is this circle of ideas/problems peculiar of Systems Biology? Biological systems seem naturally described by Hybrid Automata at a certain level of abstraction. At the finest level they can be modeled by CTMC, but these models may be

[2]The bisimulation quotient of an hybrid automaton $\mathscr{H}$ can be seen statically or dynamically. Statically, is the coarsest partition refinement of the infinite state system (whose states are points in the n-dimensional space of flows), stable with respect to $\mathscr{H}$'s continuous and discrete transitions. Dynamically, is the fix point of the partitioning procedure splitting states with respect to the predecessor relation of arcs in $\mathscr{H}$.

[3]MRA correspond to CTMC in the lattive of non-stochastic HA, as they are the HA associated to the fully discrete case.

not manageable in practice because of their complexity. However, the efficiency issue is not the most peculiar one for Systems Biology applications. In a certain sense, the features of biological systems hinting more directly at the necessity of a study of the above mentioned interplay, are their inherent uncertainty (consequence of realistic quantitative environmental interaction) and their complexity (consequence of our lack of knowledge of the internal biological control mechanisms). In this perspective, Systems Biology can be seen as a most interesting and promising arena in which testing model building techniques to mix different levels of discrete, continuous, and stochastic components.

References

[1] A. Aziz, V. Singhal, F. Balarin, R. Brayton, and A. Sangiovanni-Vincentelli. Verifying continuous time markov chains. In *Proceedings of CAV96*, 1996.

[2] L. Bortolussi, S. Fonda, and A. Policriti. Constraint-based simulation of biological systems described by molecular interaction maps. In *Proceedings of IEEE conference on Bioinformatics and Biomedicine, BIBM 2007*, 2007.

[3] L. Bortolussi and A. Policriti. Modeling biological systems in concurrent constraint programming. *Constraints*, 13(1), 2008.

[4] L. Bortolussi and A. Policriti. Dynamical systems and stochastic programming — from ordinary differential equations and back. *Transactions of Computational Systems Biology*, 2009.

[5] L. Bortolussi and A. Policriti. Stochastic programs and hybrid automata for (biological) modeling. In *Proceedings of CiE 2009*, 2009.

[6] L. Bortolussi and A. Policriti. Tales of spatiality in stochastic concurrent constraint programming. In *Proceedings of BioLogic09*, 2009.

[7] L. Bortolussi and A. Policriti. Hybrid dynamics of stochastic programs. *Theoretical Computer Science*, 2010.

[8] L. Cardelli. Abstract machines of systems biology. *Transactions on Computational Systems Biology*, III, LNBI 3737:145–168, 2005.

[9] F. Ciocchetta and J. Hillston. Bio-PEPA: an extension of the process algebra PEPA for biochemical networks. In *Proceeding of FBTC 2007*, 2007. Workshop of CONCUR 2007.

[10] E. Clarke, A. Peled, and A. Grunberg. *Model Checking*. MIT press, 1999.

[11] T. A. Henzinger. The theory of hybrid automata. In *LICS '96: Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science*, 1996.

[12] T.A. Henzinger, P.W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? In *Proceedings of 27th annual ACM symposium on Theory of Computing*, 1995.

[13] H. Kitano. Computational systems biology. *Nature*, 420:206–210, 2002.

[14] K. W. Kohn, M. I. Aladjem, J. N. Weinstein, and Y. Pommier. Molecular interaction maps of bioregulatory networks: A general rubric for systems biology. *Molecular Biology of the Cell*, 17(1):1–13, 2006.

[15] G. Lafferriere, G.J. Pappas, and S. Sastry. O-minimal hybrid systems. In *Proceedings of Mathematics of Control, Signals, and Systems (MCSS)*, 2000.

[16] J. R. Norris. *Markov Chains*. Cambridge University Press, 1997.

[17] A. Regev and E. Shapiro. Cellular abstractions: Cells as computation. *Nature*, 419, 2002.

[18] V. Saraswat and M. Rinard. Concurrent constraint programming. In *Proceedings of 18th Symposium on Principles Of Programming Languages (POPL)*, 1990.