Proceedings of WCB07

Workshop on Constraint Based Methods for Bioinformatics

Rolf Backofen, Alessandro Dal Palù, and Sebastian Will

September 13, 2007, Porto (Portugal)

Program Comittee

Rolf Backofen (co-chair) Freiburg Univ., Germany
Pedro Barahona Univ. Nova de Lisbona, Portugal
Alexander Bockmayr Freie Universitt Berlin, Germany
Mats Carlsson SICS, Uppsala Sweden
Alessandro Dal Pal (co-chair) Parma Univ., Italy
David Gilbert University of Glasgow
Simon De Givry INRA, Toulouse, France
Agostino Dovier Udine Univ., Italy
Francois Fages INRIA Rocquencourt, France
Enrico Pontelli NMSU (USA)
Sebastian Will (co-chair) Freiburg Univ., Germany

Preface

Bioinformatics is a challenging and fast growing area of research, which is of utmost importance for our understanding of life. Major contributions to this discipline can have thousands of positive effects in medicine, agriculture, or industry. To pick out only a few examples, Bioinformatics tackles problems related to:

- Recognition, analysis, and organization of DNA sequences
- Biological systems simulations (for metabolic or regulatory networks)
- Prediction of the spatial conformation of a biological polymer, given its sequence of monomers (in particular for proteins and RNA)

All these problems can be naturally formalized using constraints over finite domains or intervals of reals. Biology is a source of extremely interesting and challenging problems that can be encoded exploiting the application of recent and more general techniques of constraint programming. In this framework, some problems that have been successfully tackled are:

- The fundamental bioinformatics problem of sequence alignment can be solved by recent inference based constraint methods
- Biological systems simulations can be easily designed using concurrent constraint programming, and
- The constrained-based prediction of protein conformations promoted the development of new search strategies, new constraint solvers, and general symmetry breaking.

The main aim of this workshop is twofold. On the one hand, to share recent results in this area (new constraint solvers, new prediction and simulation programs). On the other hand, to present new challenging problems formalized and/or solved with constraint based methods.

Rolf Backofen
Alessandro Dal Palù
Sebastian Will

Contents

Constraint-based simulation of biological systems described by Molecular Interaction Maps

Luca Bortolussi[1], Simone Fonda[4], and Alberto Policriti[2,3]

[1] Dept. of Mathematics and Computer Science, University of Trieste, Italy.
luca@dmi.units.it
[2] Dept. of Mathematics and Computer Science, University of Udine, Italy
alberto.policriti@dimi.uniud.it
[3] Istituto di Genomica Applicata, Udine, Italy
[4] Dept. of Computer Science, University of Pisa, Italy.
simone.fonda@gmail.com

Abstract. We present a method to simulate biochemical networks described by the graphical notation of Molecular Interaction Maps within stochastic Concurrent Constraint Programming. Such maps are compact, as they represent implicitly a wide set of reactions, and therefore not easy to simulate with standard tools. The encoding we propose is capable to stochastically simulate these maps implicitly, without generating the full list of reactions.

1 Introduction

The aim of this work is the simulation of biological regulatory networks described by the graphical notation of Molecular Interaction Maps (MIM) [11]. In order to achieve this goal, we define an encoding of such maps in stochastic Concurrent Constraint Programming (sCCP) [1], a stochastic, concurrent, and constraint-based programming language.

In scientific literature, many mathematical tools have been proposed to describe, model, and simulate the behaviors of biological systems, all sharing the common goal of organizing and analyzing the available knowledge and understanding of such systems [9]. As happens with computer programming languages, one formalism could be better suited than another for a specific purpose, depending on the features it provides to its users. When choosing a formalism many parameters must be taken into account: continuous or discrete representation of the entities, expressivity, availability of mathematical analysis techniques, possibility of a computer aided simulation and visualization, and so on. Other important properties of a modeling language concern the process of writing and maintaining a model: compositional languages are usually preferable for systems composed of many interacting parts, like biological ones [12]. Formalisms can also differ in the size of produced models; generally, the more compact the better. Moreover, the same system can be modeled at different levels of detail; for example, a cell can be described as a functional black box or specifying its internal behaviors and mechanisms in detail.

As the focus of modeling in biology is to understand dynamical properties of biological systems, most of the modeling languages have a semantic based on ordinary differential equations [15] or stochastic processes [16]. Stochastic processes, usually continuous-time Markov Chains [16], have been used in biochemistry and biology since a long time, especially after the publishing of a simple and efficient simulation algorithm by Gillespie [8]. These models are generally more realistic than the ones based on Ordinary Differential Equations (ODEs), as they represent molecules as discrete quantities (compared to the continuous approximation of ODEs) and, moreover, they have a noisy evolution (compared to the determinism of ODEs), an important issue in biology [16].

Generally, both stochastic and differential models can be obtained in a canonical way when the list of reactions of the system is fully specified [16] (i.e., we need to specify all the possible ways of reacting of all the possible reactants). A big problem with this approach is the *cost* of the notation: in some cases, the effort required for a complete specification can be overwhelming; this is true especially for bio-regulatory networks, due to the central role played by multi-molecular complexes and protein modifications [10, 11, 6]. In fact, even a small set of proteins can potentially generate a big number of different complexes, of which only a few different kinds may be present at a certain time. Manually listing all these complexes can be a very hard task.

In order to tackle this problem, we need a formalism that can work at an higher level of abstraction, representing entities and behaviors in an implicit way. One possibility is offered by the Molecular Interaction Maps [11]: they are a compact graphical notation proposed by Kohn, with the goal to be clear, easy to use, compact, unambiguous, and (hopefully) widespread. In the author's expectations, the equivalent of electronic circuit diagrams for biologists.

The contribution of this work is the definition of an encoding of MIM in sCCP, a stochastic extension of Concurrent Constraint Programming [1], already used to model biological systems at different degrees of complexity [3, 2]. The crucial ingredient of such an encoding will be the *implicit* representation of complexes and reactions. Essentially, molecular complexes will be represented by graphs, modified locally by reactions. This work is related to κ-calculus [5], a ruled based language describing complexation implicitly, and to β-binders [4], an extension of π-calculus where π-processes are encapsulated in boxes that can be complexed together. Before presenting the encoding, however, we need to discuss in more detail Molecular Interaction Maps (Section 2). Then, after briefly recalling sCCP (Section 3), we present the main ideas of the implicit simulation of MIMs in sCCP (Section 4). Finally, conclusions and future works can be found in Section 5.

2 Molecular Interaction Maps

We briefly introduce now the MIM notation; the interested reader is referred to [11] for a detailed presentation. A MIM is essentially a graph, where nodes correspond to different (basic) molecular species, linked by different kinds of

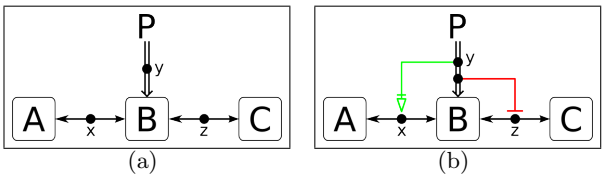


Fig. 1. Two very simple MIMs

connecting lines, see Figure 1 for an example. The notation follows few general principles: to keep the diagram compact an *elementary molecular species*, like A, B or C in Figure 1(a), generally occurs in *only one place* on a map. Different interactions between molecular species, instead, are distinguished by different lines and arrowheads; for instance, the double line in Figure 1(a) represents a covalent modification of B (in this case a *phosphorylation*, i.e. the attachment of a phosphate group in a specific place in the protein), while the single double-barbed arrows denote complexation operations. *Complex molecular species*, or simply complexes, are created as a consequence of interactions and are indicated by small circles on the corresponding interaction line; e.g. in Figure 1(a) x represents the complex $A : B$, the result of a complexation between A and B, while y represents the phosphorylated B molecule (denoted hereafter with pB).[5]

In general, there are two types of interaction lines: *reactions* and *contingencies*. The former operate on molecular species, the latter on reactions or other contingencies. The line with a T-shaped end of Figure 1(b) is an *inhibition* line; it states that phosphorylated B (indicated by y) cannot bind to C, (in fact the line terminates on the barbed arrow connecting B to C). Another contingency symbol of Figure 1(b) is the arrow with a bar preceding its empty arrowhead, terminating in x. This line represents *requirement*: B must be phosphorylated in order to bind to A. Note that multiple nodes on an interaction line represent exactly the same molecular species.

In order to explain the differences between Figure 1(a) and Figure 1(b), we need to introduce the *interpretations* of MIMs. The MIM notation, in fact, can be equipped with two different interpretations: *explicit* and *combinatorial*.
In the explicit interpretation, an interaction line applies only to the molecular species directly connected to it. Looking again at Figure 1(a), we can say that B can bind to A (forming the complex $A : B$) or to C (forming the complex $B : C$) but there is no way the complex $A : B : C$ will be formed. Moreover if B is phosphorylated, it cannot bind neither to A nor to C.
In the combinatorial interpretation, an interaction line represents a functional connection between domains or sites that (unless otherwise indicated) is inde-

[5] MIMs have also special symbols for gene transcription and regulation and for compartments and transportation; hence they can, in principle, be used to represent large scale networks integrating, for instance, biochemical, transportation and genetic networks.

pendent of the modifications or bindings of the directly interacting species. In this way the map of Figure 1(a) states that A can bind to B, independently of the state of B. For instance, B could be phosphorylated and thus forming the $A : pB$ complex, or it could be bound to C, resulting in the $A : B : C$ complex. Moreover, the combinatorial interpretation represents implicitly a large number of molecules: a single complexation arrow usually handles, on both sides, a big set of reactants. Consider, for example, the arrow connecting A to B in Figure 1(a); in the combinatorial interpretation it represents four reactions, namely the complexation of A with B, pB, $B : C$ and $pB : C$.[6]

In principle, it is always possible to create an explicit MIM with the same behaviors of a combinatorial MIM, introducing more reaction arrows and contingencies.[7] Explicit MIMs can be easily translated into a set of ODEs or into an explicit stochastic model for computer simulation, see [10]. Unfortunately, expliciting a combinatorial MIM is a non-trivial job, due to the combinatorial explosion of reaction arrows and contingencies needed in the map (expliciting map of Figure 1(a) requires nine more arrows). In addition, an explicit ODE-based (or stochastic) simulation, like the one described in [10], requires to consider all possible molecular complexes that can be generated in the system as reactants or products of some reaction. However, at a given time, usually only a small subset of these complexes is present, hence the effort needed to generate explicitly this large number of complexes is largely unmotivated.

Our goal is precisely to define a simulation operating directly at the level of the combinatorial interpretation of MIMs, hence this is the interpretation we will consider in the rest of the paper. The advantages of this choice are clear: during each stage of a simulation, we need to represent only the complexes present in the system at that time. Moreover, models can be defined using the compact notation of combinatorial MIMs, hence the resulting map is usually smaller, thereby easier to build and understand.

The biggest obstacle towards an implicit simulation of MIMs is the fact that its interpretations are ambiguous. We tackled this problem defining a set of graph rewriting rules that disambiguate the maps, in what we deem a biologically plausible way. We omit here further details on this preprocessing step.

3 Stochastic Concurrent Constraint Programming

Concurrent Constraint Programming (CCP [13]) is a process algebra having two distinct entities: agents and constraints. Constraints are interpreted first-

[6] The combinatorial interpretation considers these reactions as happening uniformly, i.e. with the same rate whatever the context. This may not be true biologically: for instance, A may complexate faster with pB than with B. This behavior needs to be specified explicitly using specific contingency arrows, like stimulation arrows. A different approach worth considering may be that of assigning basic rates as a function of the presence/absence of certain bounds in the complex.

[7] Contingencies are used in the combinatorial interpretation to restrict behaviors. For instance, in the map of Figure 1(b), the explicit and the combinatorial interpretations coincide.

order logical formulae, stating relationships among variables (e.g. $X = 10$ or $X + Y < 7$). Agents in CCP, instead, have the capability of adding constraints (`tell`) into a "container" (the *constraint store*) and checking if certain relations are entailed by the current configuration of the constraint store (`ask`). The communication mechanism among agents is therefore asynchronous, as information is exchanged through global variables. In addition to `ask` and `tell`, the language has all the basic constructs of process algebras: non-deterministic choice, parallel composition, procedure call, plus the declaration of local variables. Moreover, constraints of the store can be defined using the computational machinery of Prolog [14, 13].

The stochastic version of CCP (sCCP [1, 3]) is obtained by adding a stochastic duration to all instructions interacting with the constraint store $\mathcal{C}$, i.e. `ask`, `tell`. Each instruction has an associated random variable, exponentially distributed with rate given by a function associating a real number to each configuration of the constraint store: $\lambda : \mathcal{C} \to \mathbb{R}^+$.

The underlying semantic model of the language (defined via structural operational semantic, cf. [1]) is a Continuous Time Markov Chain [16] (CTMC), i.e. a stochastic process whose temporal evolution is a sequence of discrete jumps among states in continuous time. States of the CTMC correspond to configurations of the sCCP-system, consisting in the current set of processes and in the current configuration of the constraint store. The next state is determined by a stochastic race among all active instructions such that the fastest one is executed. More details on sCCP and on its operational semantics can be found in [2].

Time-varying quantities, an important ingredient to deal with biological systems [3], are modeled as *stream variables*, i.e. growing lists with an unbounded tail.

In [2, 3] we argued that sCCP can be conveniently used for modeling a wide range of biological systems, like biochemical reactions, genetic regulatory networks, the formation of protein complexes, and the process of folding of a protein. In fact, while maintaining the compositionality of process algebras, the presence of a customizable constraint store and of variable rates gives a great flexibility to the modeler.

4 Encoding MIMs in sCCP

The starting point for the direct simulation of MIM in sCCP is the definition of a suitable representation of molecules and complexes. Actually, with respect to other process algebras like π-calculus, sCCP offers a crucial ingredient in this direction, namely the presence of the constraint store. In fact, the store is customizable and every kind of information can be represented by the use of suitable constraints, i.e. logical predicates. Manipulating and reasoning on such information can be performed in a logic programming style [13]. These ingredients make the constraint store an extremely flexible tool that can be naturally used to represent the data structures needed to operate on MIMs.

`molecular_type`(*molecular_type_id*,*port_list*,*contingency_list*)
`node`(*molecular_type_id*, *mol_id*)
`edge`([mol_id_1, $port\_type\_id_1$],[mol_id_2, $port\_type\_id_2$])
`complex_type`(*complex_type_id*, *node_list*,*edge_list*, *contingency_list*)
`complex_number`(*complex_type_id*, X)
`port_number`(*port_type_id*, X)

Table 1. Predicates describing MIM structures and counting their occurrences.

The idea to simulate MIMs is simple: we operate directly on the graphical representation. Of course, a MIM contains all possible interactions of the system, hence the graphical representation used in the simulation must be specialized to single molecules and complexes. Complexes, in particular, can be seen as *graphs*, with *nodes representing the basic molecules* (i.e. the proteins) of the complex, and with *edges representing the chemical bonds* tying them together. Such graphs will be referred in the following as *complex-graphs*.

Recalling Figure 1 and the combinatorial interpretation of Section 2, we can easily convince ourselves that molecular species are defined by the collection of their interaction sites. In our encoding, these sites are represented by *ports* (or better, *port-type*), i.e. terminating points of one single arrow in the MIM. Port-types are characterized by an unique identifier, called *port_type_id*, and by a boolean variable, INH_p, storing the state of the port. In fact, each port can be active ($INH_p = false$), meaning that it can take part to the corresponding reaction, or inhibited ($INH_p = true$) by biological mechanisms specified in the map.
Molecular-types correspond to nodes of the MIM or to points in the middle of an arrow (i.e., terminating points of reaction arrows). They consist of an unique identifier, *molecular_type_id*, of a list of port-types, implicitly determining all the possible reactions the molecule can be involved into, and of a list of contingencies starting from it (we present the treatment of contingencies at the end of the section). For instance, in Figure 1(a), A and x nodes define two distinct molecular-types.

Each graph of a complex can contain, in principle, several instances of the same molecular-type, just think at the case in which two copies of the same molecule are bound together (the so-called homodimers). These different copies are distinguished by an unambiguous naming system inside each complex: each node of a complex-graph is numbered by an integer, local to that complex, called *mol_id*. Moreover, each different complex graph that can be constructed according to the prescription of the MIM, identifies a *complex-type*; complex-types are also given an unique id, *complex_type_id*, assigned at run-time whenever a new complex-type is created.

Complex-graphs and molecular-types can be easily represented in sCCP. In fact, we just need to store all the characterizing information in suitable logical predicates, listed at the beginning of Table 1.
Another important class of predicates, crucial for the run-time engine, are those counting the number of objects of a certain type. Specifically, at run-time we need to count how many complexes we have for each different complex-type (predicate `complex_number`), and how many active ports we have, for any port-type (predicate `port_number`). The variable X used in these predicates is a stream variable, defined in Section 3.

The reason for updating the number of ports or complexes at run-time, lies in the definition of the stochastic model for the simulation of MIMs. We adopt a classical approach, defining the speed of a reaction according to the *principle of mass action* [8]: the speed of each reaction is proportional to the quantity of each reactant. In our encoding, each reaction involves one or two port-types, hence its speed will be proportional to the number of active instances of such port-types.

The sCCP program associated to a MIM can be seen as a loop composed of 4 basic steps:

1. choose the next reaction to execute;
2. choose the reactants;
3. create the products;
4. apply contingency rules to products.

It gives rise to a stochastic model (a Continuous Time Markov Chain) that can be simulated by a classical Monte Carlo algorithm like Gillespie's direct method [8] (see [2] for details on a meta-interpreter for sCCP).
We give now some details on the sCCP program. The choice of the next reaction can be seen as a stochastic race among all the enabled reactions. In sCCP, this effect is obtained associating an agent to each reaction arrow of the molecular interaction map, called *reaction agent*. This agent tries to execute at a rate defined according to the principle of mass action. If the agent wins the competition, it needs to identify the actual reactants. In fact, a reaction involves one or two complexes with available ports of the required port-type. However, as different complex-types can have such ports available, we must identify the ones really involved in the reaction. Essentially, we need to pick, with uniform probability, one complex among all those having an active port of the required type. This operation is performed by *port managers*: there is one agent for each port-type, keeping an updated list of all complex types containing active ports of its type and choosing one of them upon request from a reaction agent.

When reactants have been chosen by port agents, the reaction agent generates the products of the reaction. For instance, in case of a complexation reaction, the agent has to merge two complexes into one, adding nodes and edges to the complex description and marking as bound the ports involved in the reaction (i.e. removing them from lists of the corresponding port agents).

If, after these operations, a new (i.e. not present in the system) complex-type is obtained, then all the necessary predicates are added to the constraint store. A bookkeeping of the predicates counting complexes and ports is then performed.

The last point of the simulation algorithm consists in the application of contingencies. These are the inhibition and requirement arrows, briefly introduced in Section 2. To grasp the rationale behind their implementation, consider again Figure 1(b). The inhibition arrow from y (the head of the contingency rule) to A-B complexation line (the tail) was used to forbid complexation between phosphorylated B and A. This essentially means that, if B is phosphorylated (i.e., the corresponding edge e is in the complex description), then B cannot bind to A, and so the port p_{B-A} connecting B to A, must become inactive.
This example suggests that contingencies are nothing but logical implications of the form:

IF (a set of edges E is in the complex)
THEN (some ports must become active/inactive)

Rules of this kind are stored in each molecular-type descriptor, so that each complex-type has associated a set of contingency rules potentially applicable.
When a new complex type is created in a reaction, then the reaction agent checks what rules among those listed in the complex can be applied, and it modifies accordingly the value of INH_p of ports and their associated global counters, like the predicate `port_number` and the lists used by port agents.

5 Conclusions

In this paper we presented an implementation in stochastic concurrent constraint programming of an algorithm to simulate biological regulatory networks defined by means of Molecular Interaction Maps. This notation is implicit, as each edge in such a diagram represents a set of reactions potentially very large. Our sCCP-simulation, instead of generating the full list of reactions, is able to simulate the map implicitly, generating only those complexes that are actually present in the system at run-time. This is achieved using a graph-based representation of complexes, so that new complexes are dynamically constructed merging and splitting other complex-graphs. A prototype implementation has been written in SICStus prolog [7] and used to perform some preliminary tests.

The choice of sCCP as a language to describe (implicitly) such maps is motivated by different reasons. First of all, the details of the stochastic evolution are dealt automatically by its (stochastic) semantics. In addition, the power of constraints allows to represent and reason directly on the graph-based representation of complexes, separating this description from the definition of agents performing the simulation. Another important motivation is that the model built in such way is compositional w.r.t. the addition of new edges (and nodes) in a MIM. Finally, the size of an sCCP program associated to a MIM scales linearly

with the description of the map (i.e. with the number of symbols needed in the description), thus taming the combinatorial explosion of possible reactions.

Currently, we are planning to realize a more efficient implementation, interfacing it with graphical tools to design MIMs, in order to analyze big bioregulatory networks.

Acknowledgements

This work has been supported by project FIRB03-RBNE03B8KK.

References

1. L. Bortolussi. Stochastic concurrent constraint programming. In *Proceedings of 4th International Workshop on Quantitative Aspects of Programming Languages, QAPL 2006, ENTCS*, volume 164, pages 65–80, 2006.
2. L. Bortolussi. *Constraint-based approaches to stochastic dynamics of biological systems*. PhD thesis, PhD in Computer Science, University of Udine, 2007. Available at http://www.dmi.units.it/ bortolu/files/reps/Bortolussi-PhDThesis.pdf.
3. L. Bortolussi and A. Policriti. Modeling biological systems in concurrent constraint programming. In *Proceedings of Second International Workshop on Constraint-based Methods in Bioinformatics, WCB 2006*, 2006.
4. F. Ciocchetta, C. Priami, and P. Quaglia. Modeling kohn interaction maps with beta-binders: an example. *Transactions on computational systems biology*, LNBI 3737:33–48, 2005.
5. V. Danos and C. Laneve. Formal molecular biology. *Theor. Comput. Sci.*, 325(1):69–110, 2004.
6. J.R. Faeder, M.L. Blinov, B. Goldstein, and W.S. Hlavacek. Rule-based modeling of biochemical networks. *Complexity*, 10:22–41, 2005.
7. Swedish Institute for Computer Science. Sicstus prolog home page.
8. D.T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. of Physical Chemistry*, 81(25), 1977.
9. H. Kitano. Computational systems biology. *Nature*, 420:206–210, 2002.
10. K. W. Kohn. Functional capabilities of molecular network components controlling the mammalian g1/s cell cycle phase transition. *Oncogene*, 16:1065–1075, 1998.
11. 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.
12. C. Priami and P. Quaglia. Modelling the dynamics of biosystems. *Briefings in Bioinformatics*, 5(3):259–269, 2004.
13. V. A. Saraswat. *Concurrent Constraint Programming*. MIT press, 1993.
14. L. Shapiro and E. Y. Sterling. *The Art of PROLOG: Advanced Programming Techniques*. The MIT Press, 1994.
15. S. H. Strogatz. *Non-Linear Dynamics and Chaos, with Applications to Physics, Biology, Chemistry and Engeneering*. Perseus books, 1994.
16. D. J. Wilkinson. *Stochastic Modelling for Systems Biology*. Chapman & Hall, 2006.

The Density Constraint

Alessandro Dal Palù[1], Agostino Dovier[2], and Enrico Pontelli[3]

[1] Dip. di Matematica, Univ. di Parma, alessandro.dalpalu@unipr.it
[2] Dip. di Informatica, Univ. di Udine, dovier@dimi.uniud.it
[3] Dept. Computer Science, New Mexico State University, epontell@cs.nmsu.edu

Abstract. The paper expands on the growing body of literature which makes use of constraint programming techniques to solve bioinformatics problems, especially in the context of discrete models. The focus is on the use of constraint programming methodologies to solve the problem of prediction of real-size protein structures. In particular, the paper builds on the good results obtained using simplified energy models (e.g., *HP* [1]) and bioinformatics-specific global constraints [3]. The paper shows how information obtained through *electron cryomicroscopy* can help this research. The density maps obtained from cryomicroscopy are formalized as *global constraints* and used to further prune the space of possible configurations of amino acids in the 3D space. The paper investigates the complexity of these new global constraints, and presents a preliminary implementation of the associated propagation procedures.

1 Introduction

Electron cryomicroscopy is an experimental technique that is believed to have the potential to allow structure determination for large and membrane proteins [11, 7, 2]. Using cutting edge techniques in this field, the 3D structure of large complexes, such as the Herpes virus, have been successfully generated at a 8.5Å resolution [11]. Although it is not possible to determine the backbone chain of the protein at a resolution lower than 6Å–current methods to determine the protein structure require a much higher resolution, such as 3Å or 4Å [10, 5]—this resolution allows the identification of various secondary structure elements, such as α-helices and β-sheets [11].

In general, an electron microscope produces 2D scans of a sample, which correspond to a projection of a 3D object electron density on one of the viewing planes. This technique can be employed to study the structure of proteins in a potentially faster—since no protein crystal growing is required—and cheaper way—since the microscope and the technique used are less expensive than a NMR apparatus. The basic idea is to retrieve several 2D scans of a high concentration protein solution, in which many copies of the same molecule are freely oriented in the fluid. Each scanned protein reveals a projection of its electron density on a randomly oriented plane. The combination of the information carried by individual projections allows us to obtain a 3D model of density.

The main drawbacks of this method are represented by the resolution, which in general is lower than NMR, and the noise gathered by the microscope, which is usually reduced by working at low temperatures. Thus, it is a significant challenge to exploit this low resolution data in order to predict high resolution models (i.e., all atoms) of

the protein. Nevertheless, even at low resolution, many of the features of the protein are recognizable, and information about the general shape is available.

In this paper we introduce a constraint framework that models the density information as a global constraint to be integrated in constraint-based protein structure determination tools (e.g., like COLA [4]). We analyze the computational complexity of (some variants of) the proposed global constraint, and we describe some preliminary implementation results.

2 Density maps

A density map D is a data set obtained through electron cryomicroscopy analysis. It represents the electron density of molecule in a given portion of space and usually it has a resolution ranging from 6Å to 12Å. This means that two density contributions can be distinguished only if they are at distance greater than 6Å (resp. 12Å). The density map is sampled at uniform rate R in the 3D space. This generates a partition of the space into cubes with edges of length R. Each cube contains a certain amount of density that is associated to a real number, which represents the sample measurement.

In principle it is sufficient to sample the density map at the Nyquist frequency [8] to avoid information loss—i.e., the sampling rate should be less than or equal to half of the resolution—from 3Å to 6Å in our case. However, it is preferable to intensify the sampling rate (i.e., reduce R). A common choice is to sample at 1Å. Because of this, the density map results to be smooth, due to the over-sampling process. This is analogous to a low resolution photo printed and over-sampled and/or scanned by a higher resolution scanner.

The density information is a 3D function which is generated by the presence of molecule's components (e.g., atoms, nucleotides, amino acids). Each component provides a typical contribution function, which combines additively to form the molecule density map. We will consider the density map overlapped on a discrete version of the 3D space. Formally, we will use the notation $D(x, y, z) \geq 0$ to refer to the value of the density function sampled at location $(x, y, z) \in \mathbb{N}^3$.

In Figure 1, on the right, we show a simple density map of a protein with 4 identical amino acids, arranged on the xy plane. We plot only the maximal z layer. The map is simulated with a resolution of 10Å and sampled at 1Å. The darker colors indicate higher densities. The white circles highlight the position of the center of the individual amino acids. In the same figure, on the left, we show the contribution of a single amino acid.

When using low resolution density maps, each component of a molecule can be approximated by a specific contribution function $\mathcal{F} : \mathbb{N}^3 \times \mathbb{N}^3 \rightarrow \mathbb{R}^+ \cup \{0\}$. For example, a reasonable choice could be to use a *Gaussian* contribution:

$$\mathcal{F}(\boldsymbol{x}, \boldsymbol{p}) = \mathcal{G}_{a,\sigma}(\boldsymbol{x}, \boldsymbol{p}) = ae^{-\frac{|\boldsymbol{x}-\boldsymbol{p}|^2}{2\sigma^2}}$$

where $\boldsymbol{p} \in \mathbb{N}^3, a \in \mathbb{R}^+, \sigma \in \mathbb{R}^+$ are respectively the reference point for the center of the object, the intensity of the map, and the decay control parameter. The parameters a and

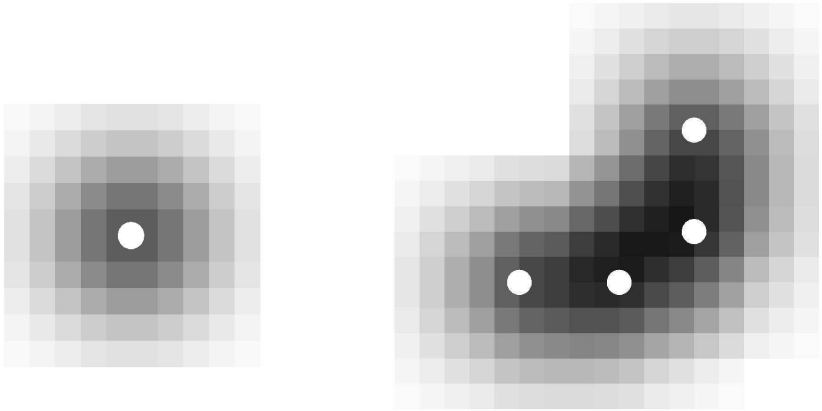


Fig. 1. Density map of an amino acid (left) and of a 4-amino acid protein (right)

σ can be estimated according to the type of the component, by first generating density maps for the single components and then performing a least square approximation.

Let us observe that the Gaussian is only one of the possible forms for the contribution functions. Our approach intended to be parametric w.r.t. the function used.

Given a molecule chemical description—i.e., the set of components and their chemical bonds—and an observed density map D, it is possible to decompose the molecule into n components (e.g., a protein can be decomposed into the set of composing amino acids, or, with higher precision, into the set of atoms composing it). Each component $i \in 1 \dots n$ can be placed in the space in the position $\boldsymbol{p}_i$ and provides a specific contribution function $\mathcal{F}_i(\boldsymbol{x}, \boldsymbol{p}_i)$ which can be pre-computed.

The ultimate goal is to find the possible placements of the components $[\boldsymbol{p}_1, \dots, \boldsymbol{p}_n]$ so that their combination produces a density map close to D.

More formally, for each $\boldsymbol{x} \in \mathbb{N}^3$

$$\sum_{i=1}^{n} \mathcal{F}_i(\boldsymbol{x}, \boldsymbol{p}_i) \approx D(\boldsymbol{x})$$

where $\approx$ means that we could introduce some tolerance in the equality, due to the errors contained in the experimental data and introduced by approximations of $\mathcal{F}$ functions.

In Figure 1, for example, given the density map and the approximation of the single amino acid, we would retrieve the four placements (white circles) of the amino acids in the original protein. Darker squares mean higher contributions (white circles only specify the placement of components).

In order to produce the density map, the chemical formula of the molecule has to be known. From the chemical formula, it is possible to derive exactly the electron charge of the molecule, and thus the expected amount of density in the map. This information provides the reference to the experimental data and allows us to relate correctly the single contributions to the density map.

3 Average and punctual density constraint

In this section we formalize the constraint induced by the knowledge of a density map associated to a molecule.

We propose two global constraints. Both of them relies on the idea of discretization of the space into regions (e.g., cubes with edges of length $d \in \mathbb{N}$). In the first one, called *average density constraint*, *regions* of space are "big" i.e., capable of containing more than one component. In this case, we consider the average density in the region. The second constraint, called *punctual density constraint*, considers instead "small" (pointwise) regions. The center of a component is placed exactly on a discrete point in the space, and it also affects the neighboring points.

We assume that each variable X represents the position of a component in the space. The domain of X is thus a set of points in $\mathbb{N}^3$ and possibly organized according to a crystal lattice disposition [4].

Definition 1 (Average Density Constraint). *Let us consider the following components:*

- *k disjoint regions $i = 1, \ldots, k$, each described by:*
  - *the set of points $R_i \subseteq \mathbb{N}^3$ of the region, and*
  - *a (density) value $s_i \in \mathbb{R}^+ \cup \{0\}$*

  Let us denote $\boldsymbol{R} = (R_1, \ldots, R_k)$ and $\boldsymbol{s} = (s_1, \ldots, s_k)$.
- *n molecules $1, \ldots, n$, each of them characterized by the corresponding contribution function $\mathcal{F}_i$, where $\mathcal{F} = (\mathcal{F}_1, \ldots, \mathcal{F}_n)$,*
- *n variables $\boldsymbol{X} = (X_1, \ldots, X_n)$ with domains $O_1, \ldots, O_n$.*

The global constraint

$$\mathsf{a\_density}(\boldsymbol{X}, \mathcal{F}, \boldsymbol{R}, \boldsymbol{s})$$

is satisfied by all the n-tuples $\langle p_1, \ldots, p_n \rangle \in O_1 \times \cdots \times O_n$ such that for all $i = 1, \ldots, k$ it holds that:

$$\sum_{j=1}^{n} \sum_{\boldsymbol{x} \in R_i} \mathcal{F}_i(\boldsymbol{x}, p_j) \leq s_i \tag{1}$$

The constraint states that each region i provides an upper bound s_i to the sum of the density contributions provided by all the molecules in that area. The values of s_i for the various regions can be obtained by the sum of the density values $D(x, y, z)$ for all the points (x, y, z) in the region R_i. Since region sizes are known, we have decided to avoid to divide the sum of the contributions by the volume, thus keeping the average value implicit.

Definition 2 (Punctual Density Constraint). *Let us consider a* discrete *density map $D(x, y, z) \in \mathbb{R}^+ \cup \{0\}$, where $x, y, z \in \mathbb{N}$. Given n molecules $1, \ldots, n$, each of them characterized by the corresponding contribution function $\mathcal{F}_i$, where $\mathcal{F} = (\mathcal{F}_1, \ldots, \mathcal{F}_n)$, n variables $\boldsymbol{X} = (X_1, \ldots, X_n)$ with domains $O_1, \ldots, O_n$. Then, the global constraint*

$$\mathsf{p\_density}(\boldsymbol{X}, \mathcal{F}, D)$$

is satisfied by the set of n-tuples $\langle p_1, \ldots, p_n \rangle \in O_1 \times \cdots \times O_n$ such that for all $\boldsymbol{x} \in \mathbb{N}^3$ it holds that:

$$\sum_{i=1}^{n} \mathcal{F}_i(\boldsymbol{x}, p_i) \leq D(\boldsymbol{x}) \tag{2}$$

Remark 1. In this paper we have chosen to require a $\leq$ condition in (1) and in (2) instead of a (possibly weak) form of equality $\approx$. The idea is that, since the whole map and the global amount of molecule contributions are known in advance, when all the $\leq$ constraints are satisfied, the equivalence is a consequence.

4 Complexity Results

We will study the complexity of the consistency problem for slightly simplified versions of the density constraints. For the sake of simplicity, our analysis is performed in the 2D space.

Definition 3 (Density 1). *Let us consider the* `Density 1` *problem defined as follows:*
Input:

- *k disjoint regions $1, \ldots, k$, each of one characterized by:*
  - *the set of points R_i of the region i,*
  - *a (density) value $s_i \in \mathbb{R}^+ \cup \{0\}$*
- *n molecules providing density contributions $a_1, \ldots, a_n \in \mathbb{N}$.*

Question: *Establish whether there is an assignment $\sigma : \{X_1, \ldots, X_n\} \longrightarrow \mathbb{N}^2$ such that for all $i = 1, \ldots, n$:*

$$\sigma(X_i) \in \bigcup_{j=1}^{k} R_j \tag{3}$$

$$\sum_{j=1 \ldots n, \sigma(X_j) \in R_i} a_j \leq s_i \tag{4}$$

Let us define $\mathsf{box}[(x_1, y_1), (x_2, y_2)] = \big\{(x, y) \in \mathbb{N}^2 \,:\, x_1 \leq x \leq x_2, y_1 \leq y \leq y_2\big\}$.

Proposition 1. *The* `Density 1` *problem is NP complete.*

Proof. It is clearly in NP. To prove its NP completeness, let us reduce the bin packing problem to the `Density 1` problem.[1]

Consider an instance $a_1, \ldots, a_n, C, B$ of bin packing, where the a_i's are the tokens, C is the bin capacity, and B the number of bins. Let $d \geq 1$ be an arbitrary integer number. We can construct an instance of `Density 1` as follows:

- $k = B$ and $s_1 = \cdots = s_k = C$

[1] Observe that the bin packing problem is (strongly) NP-complete (see, e.g., [9, pp. 203–205]).

– $R_1 = \mathsf{box}[(0,0),(d-1,d-1)], R_2 = \mathsf{box}[(d,0),(2d-1,d-1)], \ldots, R_k = \mathsf{box}[((k-1)d,0),(kd-1,d-1)]$
– The n molecules provide density contributions $a_1, \ldots, a_n$
– The contribution functions are as follows:

$$\mathcal{F}_i(\boldsymbol{x}, \boldsymbol{p}) = \begin{cases} a_i, \text{if } \boldsymbol{x} = \boldsymbol{p} \\ 0 \text{ otherwise} \end{cases}$$

It is easy to see that this instance of bin packing has a solution iff the corresponding instance of `Density 1` has. □

This result proves the NP-completeness both of the consistency problem for the a_density constraint and for the p_density constraint (in the reduction the size of R_i does not matter, so we can set $d = 1$).

Remark 2. Observe that the previous proof has some shortcomings. First, the bin packing reduction assumes arbitrary density contributions for the molecules. However, in a real problem at hand, density contributions come from a specific finite set of values, depending on the nature of the individual molecules. Second, in the case of p_density, we allow more than one molecule in a single point, which is not a realistic assumption in the density problem. Third, the density distribution function used in the proofs is very simple. We are working on to a more general NP-completeness proof.

The following version of the density problem, in which the sets of molecule values are known in advance (let α be its cardinality), and a maximum number of molecules per region is fixed (bounded by a parameter β), a polynomial consistency check.

Definition 4 (Density 2). *Let $\alpha \in \mathbb{N}$ and $\beta \in \mathbb{N}$ be fixed. Then the* `Density 2` *problem is defined as follows.*
Input:

– *k disjoint regions $1, \ldots, k$, each described by:*
  • *R_i be the set of points of the region i*
  • *a (density) value $s_i \in \mathbb{R}^+ \cup \{0\}$*
– *n molecules with density contributions $a_1, \ldots, a_n$ chosen from a set of α different elements and such that $a_i\beta \geq \max\{s_1, \ldots, s_k\}$*
– *n variables $X_1, \ldots, X_n$*

Question: *Establish whether there is a one-to-one assignment $\sigma : \{X_1, \ldots, X_n\} \longrightarrow \mathbb{N}^2$ such that for all $i = 1, \ldots, n$ formulas (3) and (4) hold.*

Proposition 2. *The* `Density 2` *is in P.*

Proof. By hypothesis, each a_i is chosen in a set of α elements, say $\{m_1, \ldots, m_\alpha\}$. The set of molecules in a region R can therefore be identified as a tuple $\langle t_1, \ldots, t_\alpha \rangle$, where t_i represents the number of occurrences of molecules of value m_i in that region.

Let $S = \max\{s_1, \ldots, s_k\}$. By hypothesis, for all $j = 1, \ldots, n$, $a_j\beta \geq S$: thus for all $i = 1, \ldots, \alpha$, $m_i\beta \geq S$. We know that for every region R and every solution σ it holds that

$$\sum_{i=1}^{\alpha} t_i m_i = \sum_{j=1\ldots n, \sigma(X_j) \in R} a_j \leq S$$

Let $m = \min\{m_1, \ldots, m_\alpha\}$. Assume, by contradiction, that $\sum_{i=1}^{\alpha} t_i > \beta$. Then,

$$\sum_{i=1}^{\alpha} t_i m_i \geq \sum_{i=1}^{\alpha} t_i m > \beta m \geq S$$

which is an absurdum. Therefore, $\sum_{i=1}^{\alpha} t_i \leq \beta$ (and, in particular, $t_i \leq \beta$).

Now, let us find an upper bound for the number of these tuples. Let us distribute in a line α white balls (from left to right, one for each element m_i) and β black balls. The number of black balls immediately on the right to the i-th white ball denotes the number of occurrences of elements m_i in the region. For instance, if $\alpha = 4$ and $\beta = 3$:

- $\bullet\bullet\bullet\circ\circ\circ\circ$ stands for $\langle 0, 0, 0, 0\rangle$
- $\circ\bullet\bullet\circ\bullet\circ\circ$ stands for $\langle 2, 1, 0, 0\rangle$
- $\bullet\circ\bullet\circ\bullet\circ\circ$ stands for $\langle 1, 1, 0, 0\rangle$
- $\bullet\bullet\circ\circ\circ\circ\bullet$ stands for $\langle 0, 0, 0, 1\rangle$

It is easy to see that there is a disposition for each possible tuple. Thus, the number of possible tuples is $q = \frac{(\alpha+\beta)!}{\alpha!\beta!}$ and any possible solution can be given (and tested) by enumerating, for each value of q, how many regions are in that way. An upper bound is therefore $k^q = O(n^q)$. □

Note, however, that in practice the exponent q is rather large. For instance, assuming $\alpha = 20$ (one characteristic value for each amino acid) and $\beta = 5$ (at most 5 amino acids can be in a region), we obtain $q = 637,560,000$.

5 Preliminary results and Conclusions

We can include density constraints in the context of a CSP framework used to determine the placement of amino acids in $\mathbb{N}^3$. Given a protein sequence $S \in \{1, \ldots, 20\}^n$, its density map D and the j-th amino acids density map parameters a_j and a Gaussian function $\mathcal{G}_j$, $j \in \{1, \ldots, 20\}$,[2] we define the variables X_i with domains $D_i \subseteq \mathbb{N}^3$, with $i \in \{1, \ldots, n\}$. The constraints added to the problem are:

- $\mathsf{p\_density}(\boldsymbol{X}, \boldsymbol{a}, \boldsymbol{\mathcal{G}}, D)$
- $\mathsf{contiguous}(\boldsymbol{X})$ which imposes that each pair of consecutive amino acids are placed at Euclidean distance equal to 3.8Å.

The CSP defined above cannot provide any solutions for two reasons. The first one is that, given the discrete domains of amino acids positions, it is not possible to find a pair of points at exact distance equal to 3.8Å. The distance constraint contiguous has to be in fact relaxed, in order to allow admissible solutions. In the implementation we used the range 3.3Å–5.2Å as acceptable distances. The second reason is more subtle: given the approximations made to model the $\mathcal{F}$ functions as Gaussian distributions and the discretized locations of contributions, it is impossible to recreate exactly the original density map D. This translates to the fact that the best solution (in which the

[2] I.e., we have individual parameters a and σ for each one of the 20 distinct amino acids.

components are mapped as close as possible to the original positions) would provide fluctuations w.r.t. the original density map. The slight local excess of density would immediately falsify the p_density constraint. It is convenient, thus, to increase the original density map by adding a density threshold, obtained, e.g., as a fraction of a single amino acid maximal punctual contribution.

The combination of the two constraints allows a filtering that can be performed as follows. A domain point $\boldsymbol{p}$ for amino acid S_i has support only if the presence of S_i does not violate D. Moreover there must be a compatible assignment of S_{i-1} and S_{i+1} that respects both the contiguous and the p_density constraints.

It is possible to further enhance the filtering, considering that for each supported assignment of S_i, if there is a point $\boldsymbol{t}$ such that $D(\boldsymbol{t})$ is greater than the sum of contributions of S_{i-1}, S_i and S_{i+1} in $\boldsymbol{t}$, then it should be possible to add another generic amino acid that contributes to $\boldsymbol{t}$, in order to reduce the gap. This boils down to finding a support for this amino acid.

To show the impact of the density constraint, we focus on simple toy examples in the 2D plane. Here we generate two types of density maps, assuming that each amino acid contributes with a Gaussian contribution function. In the first example, we arrange a chain of 30 amino acids with a spiral shape, respecting the distance of 3.8Å between consecutive elements. This example shows that non-trivial arrangements can be reconstructed, despite the fact that the domain of admissible positions is discrete. In Figure 2, we show the input density map (on the left) and the computed density map (in the center) generated by an admissible solution (on the right). The darker pixels represent the denser regions. The side of a pixel is 1Å and the simulated resolution is 10Å.

As a second example, we show how a uniform density map (obtained by a square tiling arrangement of 16 amino acids on the plane) can be reconstructed. The density maps and resulting configuration are shown in Figure 3.

The prototype, implemented on an AMD Opteron 2.2GHz Linux machine, has been validated on a number of benchmarks. We report the first admissible arrangement found. The results are summarized in Table 1. The Filtering column shows the time required for the initial pruning of domains before the search is started. In the spiral example, the filtering allows to reduce significantly the size of initial domains for central variables, while for side amino acids the domain covers the whole spiral. The Domain size column reports the size of the domains of the variable involved. Observe that after the filtering only 16 points are allowed to central variables. The first and last variables of the spiral, instead, have 387 points allowed. The presence of small domains allows to exploit a first fail strategy that is able to find a solution after few backtracks.

Fig. 2. Spiral with 30 amino acids: density (left), predicted density (center) and placement (right)

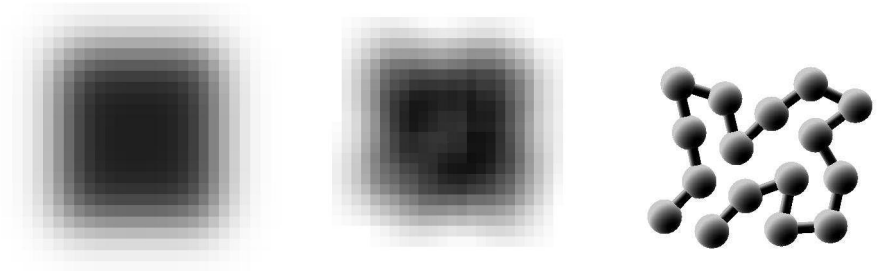


Fig. 3. Square with 16 amino acids: density (left), predicted density (center) and placement (right)

	Amino acids	Filtering	Domain size	Search	Nodes
Spiral	30	119.1 s	16–387	0.13 s	103
Square	16	23.8 s	214	95.6 s	124,663

Table 1. Summary of the experimental results

Observe how, in the square example, there are many equivalent solutions, roughly bounded by the number of possible self avoiding walks inside the square. The freedom in placing the amino acids allows the search to exponentially grow, both in finding the first admissible solution and in enumerating all of them.

We would like to stress that the space searched depends exponentially on the number of neighbors, once an amino acid is fixed. In these examples there are 52 neighbors that respect the contiguous constraint (recall that we use a range of allowed distances). The presence of the p_density constraint allows us to deal with a 52^n search space working with 1Å accuracy. This is extremely desirable in order to obtain sound results. In systems like COLA [4], the accuracy was set at 3.8Å with only 12 neighbors per point (due to the Face Centered Cubic lattice used).

In general, these examples suggest the possible application to real proteins. Often real density maps contain uniform density areas (large cylinders and planes) that are generated by the possible presence of secondary structure patterns, i.e., α-helices and β-sheets. From the complexity point of view, these areas allow a wide number of dispositions of subsequences, like in the case of the square example. These are sources of exponentially many solutions, while other regions of proteins (coils) are more linearly shaped, like in the spiral example. The addition of further secondary structure constraints could limit the explosion of equivalent solutions and retain only the biologically feasible ones.

An idea to improve the performances could be to use the density constraint dissociated from the contiguous constraint in the presence of secondary structure patterns. In those cases, it is more important to obtain a placement of amino acids rather than exploring all possible self-avoiding walks. Once a placement is found, then the con-

tiguous constraint will guide the correct linking between amino acids, following the secondary structure properties. Dividing the phases allows a fast retrieval of a placement compared to an expensive enumeration of self-avoiding walks until an admissible solution is found.

Acknowledgments

This work is partially supported by FIRB Project RBNE03B8KK, by PRIN Project 2005015491, and by NSF grants CNS-0220590, CNS-0454066, and HRD-0420407.

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. B. Böttcher, S.A. Wynne, and R.A. Crowther. Determination of the Fold of the Core Protein of Hepatitis B Virus by Electron Cryomicroscopy. *Nature*, 386:8891, 1997.
3. A. Dal Palù, A. Dovier, and E. Pontelli. Global constraints for Discrete Lattices. Proc. of WCB06, pp. 55–68, Nantes (FR), Sept. 2006.
4. A. Dal Palù, A. Dovier, and E. Pontelli. A constraint solver for discrete lattices, its parallelization, and application to protein structure prediction. Software-Practice and Experience, 2007. DOI: `10.1002/spe.810`
5. J. Greer. Automated Interpretation of Electron Density Maps of Proteins: Derivation of Atomic Coordinates for the Main Chain. *J. MOl. Biol.*, 100:427–458, 1974.
6. D. Lichtenstein. Planar Formulae and Their Uses. SIAM J. Comput. 11(2):329–343 (1982).
7. E.J. Mancini, M. Clarke, B.E. Gowen, T. Rutten, and S.D. Fuller. Cryo-electron Microscopy Reveals the Functional Organization of an Enveloped Virus. *Mol. Cell*, 5:255266, 2000.
8. A.V. Oppenheim, R.W. Schafer, J.R. Buck. *Discrete-Time Signal Processing, 2nd ed.* Englewood Cliffs, NJ: Prentice-Hall, 1999.
9. C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.
10. C.E. Wang. ConfMatch: Automating Electron-density Map Interpretation by Matching Conformations. *Acta Crystallogr. D. Biol Crystallogr*, 56:1591–1611, 2000.
11. Z.H. Zhou, M. Dougherty, J. Jakana, J. He, F. Rixon, and W. Chiu. Seeing the Herpesvirus Capsit at 8.5Å. *Science*, 288:877880, 2000.

Module identification using biological constraints

E. De Maria[1], M. Zantoni[1], A. Dovier[1], and A. Policriti[1,2]

[1] Dipartimento di Matematica e Informatica, Università di Udine
[2] Istituto di Genomica Applicata, Udine.

Abstract. One of the main issues of computational biology consists in the identification of short strings (motifs) that occur approximately in a set of longer strings. A challenging problem is the one of extracting sets of motifs (modules) that satisfy some constraints on the distance between composing motifs and appear in a significant portion of the string input data set. In this work, we encoded such a problem in CLP(FD) and we tested our program on different data sets, obtaining significant results.

Introduction

After the first major effort in sequencing, the Human Genome Project, many other genomes have been sequenced and the next big challenge is to build reliable maps the functional elements. The essence of this challenge is finding specific patterns in a vast amount of information, and it is therefore natural for the fields of biology and information technology to cooperate in creating solutions to meet it. The discovery/identification of short strings occurring approximately in a set of longer strings/sequences is a classic in today's computational biology. In our particular case we refer to these short strings as *motifs*. The field of motif discovery has seen, in recent years, a shift of attention from single motifs to sets of motifs that cooperate in regulating the expression of a gene. This has resulted in a new and more challenging objective. In biological terms, instead of finding single *Transcription Factor Binding Sites* (TFBS, genomic motifs responsible for the binding of Transcription Factors to Promoters and other regulative elements) we now want to tackle the problem of finding (constrained) collections of TFBSs modeled as composite motifs also known as *modules*.

In this paper we focus our attention on the problem of finding one or more modules common to (a significant portion of) a given set of strings, imposing and managing distance constraints on the collection of simple motifs generated inspecting the input.

1 Biological Background

An important concept behind this work is gene regulation. Regulation is the amount or timing of introduction of a functional product of a gene. The product can be anything from mRNA to a protein or even a modified protein. In this

text the term is used as a descriptor of the affinity to transcribe a given gene. An up-regulated gene has higher likelihood of being transcribed and is transcribed more often.

A Transcription Factor is a description of proteins that attach themselves to binding sites in a gene's promoter region and, thereby, regulate the transcription of the gene. For this reason the discovery of such functional elements in the non-coding regions of DNA can be very valuable. The regulation of a gene is much more complex than this description might suggest. Orientation and folding of the DNA are significant, among many other factors. Still the search and determination of transcription factor binding sites are two of the most important functional elements in any genome, for understanding regulation. TFBSs are usually short, around 5-15 base pairs (bp), and are frequently degenerate sequence motifs. As a result of this, potential binding sites can occur frequently by chance in large genomes of higher eukaryotes. This makes the functional elements even harder to find.

Predicting promoter elements or extracting their consensus sequence are important steps towards the global comprehension of the mechanisms undergoing genes co-regulation. In order to achieve this goal, it is necessary to take into consideration aspects like the correct combination and the precise spatial organization of the regulatory sites, as demonstrated in recent papers [6, 11]. The order and relative distances between the binding sites, thus, can no longer be considered negligible constraints, and whatever the method used to extract a consensus sequence is, the prediction of precise promoters structure cannot be considered completed unless more biological knowledge is used during the prediction [1, 5, 8, 9]. Transcription Factor (TF) molecules interact with each other and bind to DNA to establish a gene transcription signal. The number of different TFs in a module, the number of TFBS, the spatial constraints, the order of TFBS and relative strands of TFBS differ for different regulatory pathways.

It is possible to extend the concept of motifs searching from the single binding site approach to a broader, module-based approach, to identify groups of at most 3-4 different TFBS that are conserved in different co-regulated genes and that maintain a constant overall distance with respect to one another. More complex modules composed by several motifs are difficult to localize, as they might be distributed in a large portions of the DNA.

It is important to underline that the module search/determination is a follow-up of the motif search activity. As a matter of fact, the goodness of the obtained results is tightly linked to the accuracy of the motifs discovery model used.

In literature, the issue of motif discovery has been well studied [10, 2].

2 Finding a common substring

Finding a consensus sequence representing the best approximation of all the similar results that have been obtained by a search, is crucial in order to recover useful information from the examined (biological) sequences. The problem can be formulated as follows: find a substring or a "similar" subsequence that is

common to many of the strings in the set. We use the Hamming distance d_H to define the concept of "similarity" among substrings, even though our results can be easily adapted to a different notion of distance.

The problems we are interested in include the *Closest String Problem* (CStrP) and the *Closest Substring Problem* (CSStrP), with or without a threshold.

Initially, guided by the needs of genomic research, statistical approaches were used to give solutions for the CStrP. The problem had been previously studied because of its connection with the area of coding theory, where it was proved to be NP-hard [4]. The CSStrP models the more general situation where the strings that must be compared do not have the same length, and one wants to find just parts of the string that are similar. The CSStrP, being a generalization of the CStrP, can be easily shown to be NP-hard. In terms of parameterized complexity, the main results for the CSStrP is that it cannot be solved in polynomial time, even when the distance parameter is fixed [3]. This is expressed, in terms of parameterized complexity theory, by showing that the CSStrP is in the class W[1]-hard.

As said before, it is important to underline that this task produces the input for the module identification problem. As a matter of fact, the goodness of the obtained results in the following is linked to the accuracy of the motifs discovery model used.

The notion of module has been well formalized in [7] with the definition of *structured motif* and the notion of module introduced below is a natural adaptation to the case of discovery (as opposed to search).

The approach described in this paper is based on results obtained using the algorithm ScanPro [13, 14, 15], which in turn was presented using a constraint programming approach [12].

3 Finding sets of motifs

Let $\mathcal{F} = \{s_1, \ldots, s_\alpha\}$ be e a set of α strings, each one of length β or less, over an alphabet Σ (i.e. nucleotides, aminoacids,...). Let $K = \{1 \ldots \alpha\}$ and $H = \{1 \ldots \beta\}$. Let $\mathcal{MT} \subseteq \Sigma^\star$ be a set of finite strings (*motifs*) such that each $\mu \in \mathcal{MT}$ is a substring that occurs in one or more strings of $\mathcal{F}$. We define the *motif alphabet* Γ to be a set isomorphic to $\mathcal{MT}$.

Definition 1. *Let* $\mathfrak{M} : \Gamma \to \mathscr{P}(K \times H)$*, the* instance function*, be such that given a motif* $\mu \in \Gamma$*,* $\mathfrak{M}(\mu) = \{(k_1, h_1), \ldots, (k_t, h_t)\}$ *is a set of pairs of integer numbers representing the indices of elements of* $\mathcal{F}$ *where* μ *occurs (even with repetitions) and the relative positions.*

We denote by $(\mathfrak{M}(\mu))_1 = \{k_1, \ldots, k_t\}$ and $(\mathfrak{M}(\mu))_2 = \{h_1, \ldots, h_t\}$ the *projections* of $\mathfrak{M}(\mu)$, and by μ^{k_i,h_i} the occurrence of the motif μ in string k_i starting at position h_i (that is, $s_{k_i}[h_i \ldots |\mu| - 1] = \mu$).

A *module* is defined as an ordered sequence of characters of Γ that orderly occurs in a given input sequence and a pair of integers representing the minimum and maximum distance between two adjacent motifs in the considered structured motif. Let $\mathbb{N}^\perp = \mathbb{N} \cup \{\perp\}$.

Definition 2 (Module). *A* module ϕ *is a triple*

$$\left(\langle \mu_j \rangle_{j \in \langle 1,\ldots,J \rangle}, d_{min}, d_{max}\right) \in (\Gamma^\star \times \mathbb{N}^\perp \times \mathbb{N}^\perp),$$

such that for some $k \in ((\mathfrak{M}(\mu_j))_1 \cap (\mathfrak{M}(\mu_{j+1}))_1)$ *and all* $j < J$*, there exist* $(k,h) \in \mathfrak{M}(\mu_j)$ *and* $(k,h') \in \mathfrak{M}(\mu_{j+1})$*, for which*

$$d_{min} \leq |h' - h| \leq d_{max}.$$

Formalization of the problem

Our goal is to find all modules that occur in at least q different sequences of $\mathcal{F}$ and satisfy constraint on composing motifs and their relative distances. Each module occurrence $(\langle \mu_1 \mu_2 \ldots \mu_J \rangle, d_{min}, d_{max})$ is a subsequence built with the strings $\mu_1, \mu_2, \ldots, \mu_J$ (in this order) and satisfying $d \leq d_{min}$ and $D \geq d_{max}$, where d (resp. D) is the minimum (resp. maximum) distance between (starting points of) μ_j and μ_{j+1}.

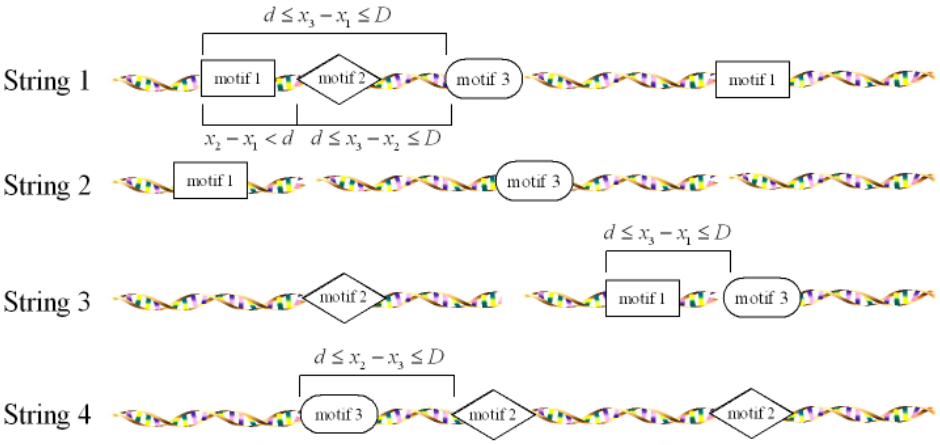


Fig. 1. For $q = 1$ (no constraints on the number of strings) there are 3 modules of length 2:[motif1, motif3], [motif2,motif3] and [motif3,motif2]. For $q = 2$ there is 1 module of length 2: [motif1, motif3]. For $q = 3$ there are no modules. [motif1, motif2, motif3] is not a module of length 3 because in String1 the distance between motif1 and motif2 is lower than d.

The complexity of the problem is dominated by $|\Gamma|^J$. If J is a constant, then the complexity of the decision test and of finding all the solutions is polynomial. Otherwise, if J is part of the input, $|\Gamma|^J$ grows exponentially on the inpout. In this case, finding all the solutions is inherently exponential. We have to figure out whether the associated decisional problem is NP complete or not.

4 Encoding in CLP(FD)

Given the alphabet Γ of motifs, a set $\mathcal{F} = \{s_1, \ldots s_k\}$ of strings, and a list holding the occurrences of each motif in each string, our aim is to find out all the possible modules of length t with occurrences above a given threshold q, i.e., modules with at least t Γ-characters that appear in a number of strings greater than or equal to q. Moreover, we require modules to satisfy minumum/maximum constraints on relative distances between composing motifs. Whenever we find a module, we want to count its occurrences in each one of the strings where it is present and to keep track of the positions where it occurs. We decided to encode this problem using Constraint Logic Programming over Finite Domains (in particular, SICStus 4 with clpfd).

The main predicate is `mod_search(+M, +Q, +Lb, +Lb)`, where `M` is the number of motifs present in the string set $\mathcal{F}$, `Q` is the lower bound on the number of strings where each module must appear, and `Lb` (resp. `Ub`) is the lower (resp. upper) bound on the distance between subsequent motifs in a module. `mod_search` retrieves from a file a list $L = \{l_1 \ldots l_k\}$ of lists containing the occurrences of each motif in each string. For each $1 \leq i \leq k$, list $l_i = \{\ell_{i1} \ldots \ell_{iM}\}$ is a list of lists containing the occurrences of each motif in string s_i. For each $1 \leq j \leq M$, list ℓ_{ij} contains the occurrences of the j-th motif in string s_i. As an example, list $L =$ [$[[1, 38], [24], [55, 70]]$, $[[12], [], [1, 25, 47]]$] gives the information that there are 2 strings and 3 motifs.
In the first string

- the first motif occurs in positions 1 and 38;
- the second motif occurs in position 24;
- the third motif occurs in positions 55 and 70.

In the second string

- the first motif occurs in position 12;
- the second motif does not occur;
- the third motif occurs in positions 1, 25, and 47.

Predicate `mod_search` definition mainly exploits two predicates, `callconstrain` and `allsolutions`. The first one looks for the presence of each structured motif of length 2 in the different strings of $\mathcal{F}$; the second one distinguishes between the structured motifs that appear in at least `Q` strings (modules) and the other ones. For each module (structured motif of the former type), it counts its occurrences in every string where it appears and it keeps trace of the positions where it occurs.

As far as predicate `callconstrain` is concerned, to perform the search of a single structured motif in a single string it takes advantage of predicate `onestring (+Lb, +Ub, +S, +[A,B], -V_AB , -M_AB, -N_AB)`, where `Lb` and `Ub` are the input lower and upper bounds, `S` is a list of lists, which contains the occurrences of each motif in a given string, `[A,B]` is a structured motif of length 2 made by motif `A` and motif `B`, `V_AB` is a boolean variable and `M_AB`, `N_AB` are

the positions where the structured motif occurs. Variable V_AB is set to 1 if in the given string there exists a pair of motif occurrence positions M_AB and N_AB such that Lb $\leq$ N_AB - M_AB $\leq$ Ub; otherwise V_AB is set to 0 and M_AB and N_AB are set to -1.

The definition of predicate onestring/7 is the following:

```
onestring(Lb,Ub,S,[A,B],V_AB , M_AB,N_AB):-
            nth1(A,S,L1), % L1 is the occurrences list of A in S
            nth1(B,S,L2), % L2 is the occurrences list of B in S
            list_to_fdset(L1,D1),
            list_to_fdset(L2,D2),
            M_AB in_set D1, % M_AB is the position of A in S
            N_AB in_set D2, % N_AB is the position of B in S
            N_AB-M_AB #>= Lb, N_AB-M_AB #=< Ub, !, V_AB #= 1.          (*)

onestring(_,_,_,[_,_],0 , -1,-1).
```

Since in the library clpfd of SICStus Prolog the domains of variables are internally represented as FD set terms, we use operation list_to_fdset to turn the list of occurrence positions of each motif in each string into a FD set. Then we take advantage of operation in_set to state the belonging of the occurrence position of each motif of the structured motif in each string in the proper FD set.

As far as predicate allsolutions is concerned, its inputs are the threshold Q and, for each structured motif, a list of boolean variables Bool and a list of M_AB and N_AB values (the length of Bool equals $|\mathcal{F}| = k$ because we associate a boolean variable to each string). Given a structured motif, if it appears in at least Q strings, that is, sum(Bools,#>=,Q), then we count its occurrences in each string where it is present; otherwise we avoid it. To count such occurrences and to keep trace of them, we use a standard mechanism based on assert and retract.
The source code is available at http://www.dimi.uniud.it/demaria/modules.html.

5 Results

We carried out several tests on an AMD Opteron 2.2 GHz Linux machine. First of all, we tested our program on a data set consisting in 26 strings of length 500 containing 23 motifs of length 6. Following biologists suggestions, we constrained the distance between subsequent motifs in a module to belong to the interval [10,90] and we looked for modules of length 2 appearing in at least q strings, with $q \geq 1$. The results of the test are presented in Figure 2. As the threshold q increases, the number of modules satisfying the constraint decreases. The minimum q such that there are not modules which satisfy the constraint is 15. As far as execution time is concerned, it decreases when q increases, that is, the number

of modules decreases. In fact, the less modules are, the less module occurrences to count are and the less `assert` and `retract` operations are needed.

Quorum	Modules	Time(ms)
1	385	410
2	275	350
3	198	300
4	143	270
5	117	240
6	94	250
7	60	190
8	39	190
9	24	190
10	12	180
11	8	150
12	3	170
13	1	180
14	1	170
15	0	150

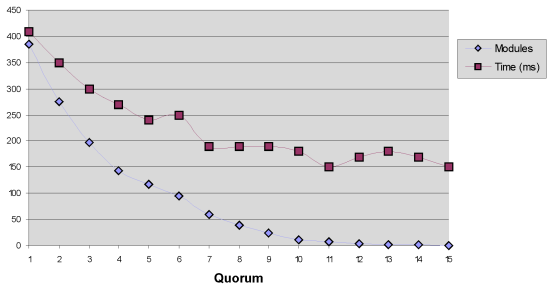


Fig. 2. Modules search on 26 strings of length 500. Decreasing of module number and execution time according to the growth of the quorum.

The results of such tests were analyzed by biologists who extrapolated relevant conclusions. In fact one of the input motifs, namely `GCAGNG`, was classified by biologists as an unknown one. Surprisingly our test showed that such a motif, a part from being abundant, is the first component of a module (`[GCAGNG,GCTGNG]`) that appears very often (in 11 strings). Such a result has a biological relevance because it means that the unknown motif appears very often in combination with other known motifs.

Another experiment consisted in testing our program on 7 data sets, each one made by 20 strings of length 500, 1000, 1500, 2000, 2500, 3000, and 3500 respectively and containing 15, 21, 27, 33, 39, 42, and 50 motifs respectively. In order to compare execution times, we launched our program on each data set with $q = 12$ and, as in the previous experiment, we constrained the distance between subsequent motifs in a module to belong to the interval [10,90]. The results are present in Figure 3. Time increases quickly because at each step we increase not only the strings length (and consequently the number of occurrences of different motifs in each string) but also the number of motifs.

At last, we tested our program on 10 data sets consisting respectively of 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100 strings of length 500. Each data set contains 15 motifs. We launched our program with q equal to the number of strings divided by 2. The results are present in Figure 4.

String length	Motif number	Time(ms)
500	15	110
1000	21	650
1500	27	1960
2000	33	5100
2500	39	15800
3000	42	29170
3500	50	51450

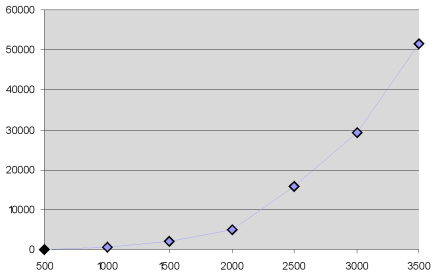


Fig. 3. Modules search on data sets of 20 strings with increasing length and motif number. Growth of execution time according to string length and motif number.

String number	Time(ms)
10	110
20	130
30	210
40	300
50	360
60	410
70	500
80	550
90	620
100	710

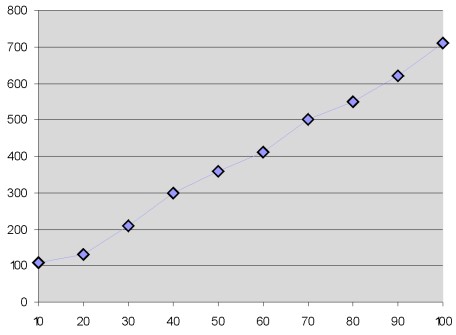


Fig. 4. Modules search on data sets with increasing number of strings. Growth of execution time according to the cardinality of the data set (string number).

6 Conclusions and Future work

Our program for extracting modules can be easily extended in several directions. First of all, it is immediate to generalize it to the research of modules of lengths greater than 2. Modules of length 3 could be thought as a combination of a motif with a module of length 2. Using such an approach, modules of length greater than 2 can be searched recursively. If we were asked to look for a restricted number of modules, we could improve the research by adopting a strategy that consists in considering at first motifs which occur in a greater number of strings, maximizing in such a way the chances to find modules which respect the quorum on the number of strings. We could start considering neighborhoods of radius greater than `Ub` of the occurrence positions of the most frequent motif, then look for occurrences of frequent motifs in these neighborhoods and so on.

Another possible extension concerns the constraints to impose when looking for a module in a set of strings. As an example, it would be reasonable to consider only modules that are sufficiently distant from the the beginning or the end of the string. As another example, biologists are interested in modules whose set of beginning positions in the different strings where they appear is limited by a lower and an upper bound. Such constraints can be very easily added to our program. After substituting the last line of predicate `onestring` (*) by the reified constraint

```
V_AB #<=>(N_AB-M_AB #>= Lb #/\ N_AB-M_AB #=< Ub),
```

one can add in predicate `allsolutions` explained in Section 4 constraints on the 2 lists containing respectively `V_AB` variables (one boolean variable is associated to each string) and `M_AB` and `N_AB` values.

Acknowledgments This work is partially supported by MIUR projects PRIN05-015491 and FIRB03-RBNE03B8KK.

References

[1] G.M. Church, A.M. Michelson, M.S. Halfon, and Y. Grad. Computation-based discovery of related transcriptional regulatory modules and motifs using an experimentally validated combinatorial model. *Genome Research*, 12:1019–1028, 2002.

[2] I. Eidhammer, I. Jonassen, S. H. Grindhaug, D. Gilbert, and M. Ratnayake. A constraint based structure description language for biosequences. *Constraints*, 6(2-3):173–200, 2001.

[3] M.R. Fellows, J. Gramm, and R. Niedermeier. On the parameterized intractability of closest substring and related problems. *Lecture Notes in Computer Science*, pages 262–273, 2002.

[4] M. Frances and A. Litman. On covering problems of codes. *Theor. Comput. Syst.*, 30:113–119, 1997.

[5] G. Kreiman. Identification of sparesely distributed clusters of cis-regulatory elements in sets of co-expressed genes. *Nucleic Acid Research*, 32, 2004.

[6] V.J. Makeev, A.P. Lifanov, A.G. Nazina, and D.A. Papatsenko. Distance preferences in the arrangement of binding motifs and hierarchical levels in organization of transcription regulatory information. *Nucleic Acids Research*, 31(20):6016–6026., October 2003.

[7] M. Morgante, A. Policriti, N. Vitacolonna, and A. Zuccolo. Structured motifs search. *Journal of Computational Biology*, 12(8), October 2005.

[8] B.D. Pfeiffer, P. Tomancak, S. Celniker, M. Levine, G.M. Rubin, M.B. Eisen, B.P. Berman, and Y. Nibu. Exploiting transcription factor binding site clustering to indentify cis-regulatory modules involved in pattern formation in the drosophila genome. *PNAS*, 99:757–762, 2001.

[9] R. Sharan, R. Shamir, Y. Shiloh, R. Elkon, and C. Linhart. Genomewide in silico identification of transcriptional regulators controlling the cell cycle in human cells. *Genome Research*, 13:773–780, 2003.

[10] R. Staden. Searching for patterns in protein and nucleic acid sequencies. *Methods in Enzymology*, 183:193–211, 1990.

[11] G. Terai and T. Takagi. Predicting rules on organization of cis-regulatory elements, taking the order of elements into account. *Bioinformatics*, 20(7):1119–1128, May 2004.

[12] M. Zantoni, E. Dalla, A. Policriti, and C. Schneider. Biological constraints for the consensus subsequence problem. In *Workshop on Constraint Based Methods for Bioinformatics (WCB05)*, 2005.

[13] M. Zantoni, E. Dalla, A. Policriti, and C. Schneider. Finding regulatory elements fixing error layouts. In *International Symposium on Computational Biology & Bioinformatics (ISBB)*, 2006.

[14] M. Zantoni, E. Dalla, A. Policriti, and C. Schneider. Motif discovery fixing mismatch positions. In *Communications to SIMAI Conferences, ISSN 1827-9015*, May 2006.

[15] M. Zantoni, E. Dalla, A. Policriti, and C. Schneider. TFBS discovery fixing layouts. Extended Abstract - RECOMB 2006, April 2006.

Analyzing Biological Data Time Series in Constraint-LTL

François Fages, Aurélien Rizk

Firstname.Lastname@inria.fr
Projet Contraintes, INRIA Rocquencourt,
BP105, 78153 Le Chesnay Cedex, France.
http://contraintes.inria.fr

Abstract. Temporal logics and model-checking techniques have proved successful to respectively express biological properties of complex biochemical systems, and automatically verify their satisfaction in both qualitative and quantitative models. In this paper, we propose a finite time horizon model-checking algorithm for the existential fragment of LTL with numerical constraints over the reals, with the ability to compute the range of values of the real variables occurring in a formula that makes it true in a model. We illustrate this approach for the analysis of biological data time series, provide a set of biologically relevant patterns of formulas, and evaluate them on a model of the cell cycle control.

1 Introduction

Temporal logics and model-checking techniques [1] have proved useful to respectively express biological properties of complex biochemical systems and automatically verify their satisfaction in both qualitative and quantitative models, i.e. in boolean [2–4], discrete [5, 6], stochastic [7, 8] and continuous models [9, 10, 3]. This approach relies on a logical paradigm for systems biology that consists in making the following identifications [11]:

biological model = transition system
biological property = temporal logic formulae
biological validation = model-checking

Having a formal language not only for describing models, i.e. transition systems by either process calculi [12–16], rules [2, 17, 18], Petri nets etc..., but also for formalizing the biological properties of the system known from biological experiments under various conditions, opens a whole avenue of research for designing automated reasoning tools inspired from circuit and program verification to help the modeler [19]. However, the formalization of the biological properties as a specification in temporal logic remains a delicate task and a bottleneck of the method.

In this paper, we investigate the use of this logical paradigm to analyze numerical data, and infer temporal logic specifications from experimental data time

series. There has been work on the inference of correlations and positive and negative influences between entities from numerical data time series, especially for gene expression temporal data [20, 21]. However, to our knowledge, the inference of temporal logic formulae with real valued variables from numerical data time series is new.

In this paper, we generalize the finite time horizon model-checking algorithm described in [9] to the existential fragment of LTL with numerical constraints over the reals. This first-order setting provides the ability to compute those instantiations of a formula that are true in a model, by giving the range of values of the real valued variables occurring in the formula for which it is true.

We illustrate the relevance of this approach to the analysis of biological data time series, by providing a set of biologically relevant patterns of formulas and by evaluating them on models of cell cycle control in Sec. 3. We then conclude on the results achieved so far, their generality, and their use in on-going work.

2 Formula Instantiation in Constraint-LTL over Reals

The *Linear Time Logic* LTL is a temporal logic [1] that extends propositional or first-order logic with modal operators for qualifying when a formula is true in a tree of timed states, called a Kripke structure.

We consider LTL formulas with real valued variables and inequality constraints. More precisely, the language of constraint-LTL formulae considered in this paper is defined by the following grammar :

$$
\begin{aligned}
Constraint-ltl = \;& Atom \mid F(Constraint-ltl) \\
& \mid G(Constraint-ltl) \mid X(Constraint-ltl) \\
& \mid (Constraint-ltl)U(Constraint-ltl) \\
& \mid (Constraint-ltl)\ and\ (Constraint-ltl) \\
& \mid (Constraint-ltl)\ or\ (Constraint-ltl) \\
& \mid (Constraint-ltl) \Rightarrow (Constraint-ltl) \\
& \mid not\ (Constraint-ltl) \\
Atom = \;& Value\ Op\ Variable \mid Value\ Op\ Value \\
Op = \;& < \mid > \mid \leq \mid \geq \\
Value = \;& float \mid [molecule] \mid d[molecule]/dt \mid d^2[molecule]/dt^2 \\
& \mid Value + Value \mid Value - Value \mid -Value \mid Value \times Value \\
& \mid Value/Value \mid Value^{Value}
\end{aligned}
$$

By an obvious transformation, negations and implications can be eliminated, by propagating the negations down to the atomic constraints in the formula.

From now on, we will assume that all constraint-LTL formulae are in negation free normal form.

Traces considered are linear Kripke structures in which constraint-LTL formulae can be interpreted. Since constraints refer not only to concentrations, but also to their derivatives, traces are of the form

$$(< t_0, x_0, dx_0/dt, d^2x_0/dt^2 >, < t_1, x_1, dx_1/dt, d^2x_1/dt^2 >, ...)$$

where at each time point, t_i, the trace associates the concentration values x_i of the variables, and the values of their first and second derivatives dx_i/dt and d^2x_i/dt^2. For instance `F([A]>10)` expresses that the concentration of `A` eventually gets above the threshold value 10. `G([A]+[B]<[C])` expresses that the concentration of C is always greater than the sum of the concentrations of A and B.

Given a trace T representing a linear Kripke structure, and a constraint-LTL formula ϕ with n variables, the *formula instantiation problem*, $\exists \boldsymbol{v} \in \mathbb{R}^n \ (\phi(\boldsymbol{v}))$, is the problem of determining the valuation $\boldsymbol{v}$ of the variables for which the formula ϕ is true in T. In other words, we look for the domain of validity of the variables $\mathcal{D}_\phi \subset \mathbb{R}^n$ such that $T \models_{LTL} \forall \boldsymbol{v} \in \mathcal{D}_\phi \ (\phi(\boldsymbol{v}))$.

The domain of validity $\mathcal{D}_\phi$ of ϕ can be computed using an algorithm that generalizes trace-based model-checking [9]:

Algorithm 1 (trace-based constraint-LTL formula instantiation) *Given a finite trace and a temporal property ϕ with variables,*

1. *label each trace point by the atomic sub-formulae of ϕ and their domain of validity as follows :*
   - *for an atomic formula ψ without variables, label a time point t_i by $(\psi, \mathcal{D}_\psi(t_i) = \mathbb{R}^n)$ if ψ is true at time t_i, and $(\psi, \mathcal{D}_\psi(t_i) = \varnothing)$ otherwise;*
   - *for an atomic formula $[A] \geq p$ (that is, of the form value $\geq$ variable) label a time point t_i by $([A] \geq p, \mathcal{D}_{[A]\geq p}(t_i))$ where $\mathcal{D}_{[A]\geq p}(t_i)$ is the half-space of $\mathbb{R}^n$ defined by $p \leq [A](t_i)$;*
   - *proceed similarly for other atomic formulae containing variables;*
2. *starting from the end of the trace, label each time point t_i by the sub-formula $F\psi$ and its domain of validity $\mathcal{D}_{F\psi}(t_i) = \mathcal{D}_{F\psi}(t_{i+1}) \cup \mathcal{D}_\psi(t_i)$;*
3. *starting from the end of the trace, label each time point t_i by the sub-formula $G\psi$ and its domain of validity $\mathcal{D}_{G\psi}(t_i) = \mathcal{D}_{G\psi}(t_{i+1}) \cap \mathcal{D}_\psi(t_i)$;*
4. *starting from the end of the trace, label each time point t_i by the sub-formula $\psi_1 U \psi_2$ and its domain of validity $\mathcal{D}_{\psi_1 U \psi_2}(t_i) = \mathcal{D}_{\psi_2}(t_i) \cup (\mathcal{D}_{\psi_1 U \psi_2}(t_{i+1}) \cap \mathcal{D}_{\psi_1}(t_i))$;*
5. *label each time point t_i by the sub-formula $X\psi$ and its domain of validity $\mathcal{D}_{X\psi}(t_i) = \mathcal{D}_\psi(t_{i+1})$;*
6. *label each time point t_i by the sub-formula ψ_1 or ψ_2 and its domain of validity $\mathcal{D}_{\psi_1 \text{ or } \psi_2}(t_i) = \mathcal{D}_{\psi_1}(t_i) \cup \mathcal{D}_{\psi_2}(t_i)$;*
7. *label each time point t_i by the sub-formula ψ_1 and ψ_2 and its domain of validity $\mathcal{D}_{\psi_1 \text{ and } \psi_2}(t_i) = \mathcal{D}_{\psi_1}(t_i) \cap \mathcal{D}_{\psi_2}(t_i)$;*
8. *return the domain $\mathcal{D}_\phi(t_i)$ for all time points t_i where it is not empty.*

This algorithm enjoys a strong completeness theorem for the chosen fragment of constraints over the reals.

Theorem 1. *The instantiation algorithm is correct and complete: a valuation* $\boldsymbol{v}$ *makes* ϕ *true at time* t_i, $T, t_i \models_{LTL} (\phi(\boldsymbol{v}))$, *if and only if* $\boldsymbol{v}$ *is in the computed domain of* ϕ *at* t_i, $\boldsymbol{v} \in \mathcal{D}_\phi(t_i)$.

Now, let us define a polytope of $\mathbb{R}^n$ as a finite intersection of half-spaces and the size $S(\mathcal{D})$ of a domain as the minimum number of polytopes the domain is made of.

Theorem 2. *In the worst case, the size of the validity domain of a LTL formula of size* k *on a trace of length* n *is* n^{k^2}.

3 Application to the Inference of Temporal Properties from Biological Time Series

Temporal logic is sufficiently expressive to formalize a wide range of biological properties known from experiments under various conditions. The formula instantiation algorithm in constraint-LTL makes it possible to analyze concentration traces and obtain semi-quantitative information such as:

Reachability : `F([A]>=p)`, what threshold p species A attain in the trace ?

Stability : `G([A]=<p1 & [A]>=p2)`, what is the range of values taken by $[A]$? This range can be looked for in some context given by a condition like in `G(Time>10 -> ([A]<p1 & [A]>p2))`.

Oscillation : `F((d([A])/dt>0 & [A]>v1) & (F((d([A])/dt<0 & [A]<v2))))`, what amplitude $(v1 - v2)$ is attained in at least one oscillation ? An oscillation is defined as the change of sign of the derivative. This formula can be extended for more oscillations and is abbreviated by `oscil(M,K,p)`. It states that M must have amplitude `p` in at least K oscillations. By applying the algorithm for each value of K, beginning with 1, we can find the number of oscillations in the trace and minimal amplitude `p` attained by K oscillations for any K.

Influence : `G(d[A]/dt>p1 -> d2[B]/dt2>=0)`, above which threshold does the derivative of A have an influence on B ? The influence is positive if a high value of `d[A]/dt` entails a positive second derivative of `[B]`. It is worth noticing that, as multiple species might influence B, this formula only indicates a correlation between the value of the derivative of A and the second derivative of B and gives no proof of direct influence.

3.1 Cell Cycle Simulation Data

For the purpose of evaluation of the method, we use here simulation data obtained from a model of the cell cycle in budding yeast [22]. Concentration traces are obtained by simulating the cell cycle control model in Biocham [18]. Then,

we try to recover relevant properties of the model by automatically analyzing the traces.

Notations `~{p1}` and `~{p1,p2}` denote phosphorylated forms of a molecule, that is the addition of a phosphate group to a molecule. Figure 1 displays a simulation trace obtained in this model for four species of this model.

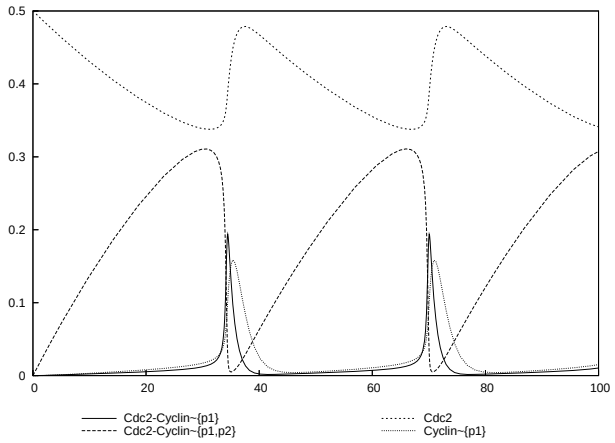


Fig. 1. Budding yeast cell cycle simulation trace over 100 time units made of 94 time points.

A reachability query provides the maximum concentration attained by an entity. The result returned is a list of domains represented by lists of constraints on the variables.

```
biocham: trace_analyze(F([Cdc2-Cyclin~{p1}]>=v)).
[[v=<0.194]]
```

Here a single domain is returned with a single constraint on v. The most relevant value in this domain is its boundary, `0.194`, which is here the maximum concentration of `Cdc2-Cyclin~{p1}` in the trace. Table 1 gives the maximum reachable values for the four species displayed in Figure 1.

For stability, let us find the range of values taken by `[Cdc2]` in the last third part of the trace:

```
biocham: trace_analyze(G(Time>66 -> ([Cdc2]=<v1 & [Cdc2]>=v2))).
[[v1>=0.479, v2=<0.338]]
```

The domain is defined by the conjunction of the two constraints `v1 >= 0.479` and `v2 =< 0.338`. These values are the maximum and minimum values attained

Species	Reachability	Stability	Amplitude of at least n oscillations	
			$n = 1$	$n = 2$
`Cdc2`	0.500	(0.338,0.479)	0.141	0.138
`Cdc2-Cyclin~{p1,p2}`	0.311	(0.005,0.310)	0.306	0.306
`Cdc2-Cyclin~{p1}`	0.194	(0.002,0.194)	0.192	0.192
`Cyclin~{p1}`	0.159	(0.004,0.158)	0.155	0.154

Table 1. Results for reachability (maximum value), stability (bottom and top values in the last third part of the trace) and amplitude of at least n oscillations.

by `[Cdc2]` in the last third part of the trace. The results for the other species are given in Table 1.

An oscillation query may compute several interval domains:

```
biocham: trace_analyze(oscil(Cdc2,1)).
[[v2>=0.338, v1=<0.479], [v2>=0.341, v1=<0.479]]
```

The result is the union of two boxes. In such domains, the most relevant point is not obvious. Here we look for the maximum amplitude $v1 - v2$. The maximum is obtained in the domain with $v1 - v2 = 0.479 - 0.338 = 0.141$. This result states that at least one oscillation of `Cdc2` has an amplitude greater or equal to 0.141. The number of oscillations is then incremented until obtaining an empty validity domain. It is obtained for `Cdc2` with the query `oscil(Cdc2,3)`, stating that there are only two oscillations of `Cdc2` in the trace.

The results for the other species are given in Table 1. Obtaining the amplitude of the oscillations is useful to distinguish between mixed amplitudes oscillations in the trace. For instance, in noisy data the amplitude can be used to count the number of oscillations regardless of small noise induced oscillations.

The influence of a molecule A on a molecule B is looked for with formula `G(d[A]/dt>p1 -> d2[B]/dt2>0)`. The idea behind this formula is that if a species B appears only in a reaction rule of the form $A \rightarrow B$ with a mass action law kinetic, the following constraint-LTL formulae are true : $G(d[A]/dt > 0 \Rightarrow d^2[B]/dt^2 > 0)$ and $G(d[A]/dt < 0 \Rightarrow d^2[B]/dt^2 < 0)$. Table 2 gives influence scores based on the results of these formulas for species `Cdc2` and `Cdc2-Cyclin~{p1,p2}`.

3.2 Experimental Data

Experimental data for measuring the evolution over time of gene expression levels or of protein concentrations, typically involve between 6 and 50 time points taken at regular intervals. Furthermore, experimental data are noisy, and it is not one trace but several ones that have to be analyzed in order to extract their significant features. The strategy here is thus to analyze the traces separately and retain the intersection set of their properties, or the most frequent ones only.

In order to evaluate the instantiation algorithm on similar experimental-like concentration traces, we extracted eleven equally spaced time points from the cell cycle simulation trace. The obtained trace is displayed in Figure 2.

Species	Cdc2	Cdc2-Cyclin~{p1,p2}
Cdc2	0.00	0.11
Cdc2~{p1}	0.01	0.12
Cyclin	0.00	**0.34**
Cdc2-Cyclin~{p1,p2}	0.00	0.02
Cdc2-Cyclin~{p1}	**0.90**	0.00
Cyclin~{p1}	0.50	0.09

Table 2. Positive influence scores of all species on Cdc2 and Cdc2-Cyclin~{p1,p2}. Molecules appearing in rows (resp. columns) act as molecule A (resp. B) in formulae $G(d[A]/dt > v1 \Rightarrow d^2[B]/dt^2 > 0)$ and $G(d[A]/dt < v2 \Rightarrow d^2[B]/dt^2 < 0)$ used to compute these scores.

Species	Reachability	Stability	Amplitude of at least n oscillations	
			$n = 1$	$n = 2$
Cdc2	0.500	(0.341,0.441)	0.125	0.089
Cdc2-Cyclin~{p1,p2}	0.311	(0.031,0.308)	0.279	0.222
Cdc2-Cyclin~{p1}	0.194	(0.002,0.194)	0.192	0.012
Cyclin~{p1}	0.100	(0.005,0.100)	0.095	0.018

Table 3. Results for reachability, stability and oscillation queries in experimental-like data.

We applied on this trace the queries used on the original simulated one, results are given in Tables 3 and 4. Oscillations properties are still obtained but with smaller amplitudes, because the peaks are missed in the sampling. For instance, Cdc2-Cyclin~{p1} has one oscillation of size 0.192 but two oscillations of size only greater than 0.012. This is a limit inherent to a low number of time points as the first peak of Cdc2-Cyclin~{p1} almost disappeared in this trace. Having a small number of time points also tends to give high self positive influence scores but considering only highest scores except self influence still correctly determines the influence between species.

4 Conclusion

Considering the bottleneck of specifying in temporal logic with numerical constraints the biological properties of a system known from experiments, we have proposed an algorithm for inferring constraint-LTL formulae from numerical data time series. To this end, the finite time horizon model-checking algorithm described in [9] has been generalized to an instantiation algorithm in the existential fragment of LTL with numerical constraints over the reals. A strong completeness theorem stating that the ranges of real valued variables computed for a formula describe exactly the solution space, has been shown, together with a complexity bound in n^{k^2} on the representation of the domain, where n is the number of time points and k the size of the formula.

For the purpose of evaluating the method, we worked with time series generated from models by simulation, and considered one experimental-like time

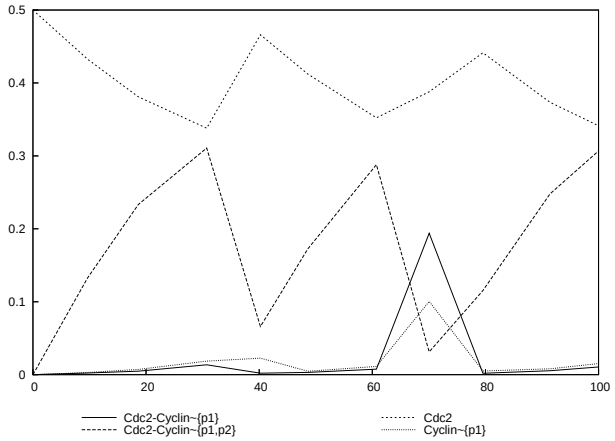


Fig. 2. Curve of concentrations every 10 units of time extracted from the cell cycle simulation trace.

Species	`Cdc2`	`Cdc2-Cyclin~{p1,p2}`
`Cdc2`	**0.59**	0.00
`Cdc2~{p1}`	**0.59**	0.00
`Cyclin`	0.00	**0.73**
`Cdc2-Cyclin~{p1,p2}`	0.00	0.59
`Cdc2-Cyclin~{p1}`	0.49	0.00
`Cyclin~{p1}`	0.48	0.00

Table 4. Positive influence scores of all species on `Cdc2 and Cdc2-Cyclin~{p1,p2}`.

series extracted from the simulation trace in few time points taken at regular intervals of time. In the near future, we plan to apply the method to the analysis of experimental temporal data of FSH signaling proteins for designing a model of FSH signal transduction together with its temporal specification, and proceed similarly with cell cycle and circadian cycle data for cancer chronotherapies in the framework of the EU project Tempo[1].

References

1. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. MIT Press (1999)
2. Eker, S., Knapp, M., Laderoute, K., Lincoln, P., Meseguer, J., Sönmez, M.K.: Pathway logic: Symbolic analysis of biological signaling. In: Proceedings of the seventh Pacific Symposium on Biocomputing. (2002) 400–412

[1] `http://www.chrono-tempo.org/`

3. Chabrier, N., Fages, F.: Symbolic model cheking of biochemical networks. In Priami, C., ed.: CMSB'03: Proceedings of the first workshop on Computational Methods in Systems Biology. Volume 2602 of Lecture Notes in Computer Science., Rovereto, Italy, Springer-Verlag (2003) 149–162
4. Chabrier-Rivier, N., Chiaverini, M., Danos, V., Fages, F., Schächter, V.: Modeling and querying biochemical interaction networks. Theoretical Computer Science **325**(1) (2004) 25–44
5. Bernot, G., Comet, J.P., Richard, A., Guespin, J.: A fruitful application of formal methods to biological regulatory networks: Extending thomas' asynchronous logical approach with temporal logic. Journal of Theoretical Biology **229**(3) (2004) 339–347
6. Batt, G., Bergamini, D., de Jong, H., Garavel, H., Mateescu, R.: Model checking genetic regulatory networks using gna and cadp. In: Proceedings of the 11th International SPIN Workshop on Model Checking of Software SPIN'2004, Barcelona, Spain (2004)
7. Calder, M., Vyshemirsky, V., Gilbert, D., Orton, R.: Analysis of signalling pathways using the prism model checker. In Plotkin, G., ed.: CMSB'05: Proceedings of the third international conference on Computational Methods in Systems Biology. (2005)
8. Heath, J., Kwiatkowska, M., Norman, G., Parker, D., Tymchyshyn, O.: Probabilistic model checking of complex biological pathways. In: Proc. Computational Methods in Systems Biology (CMSB'06). Volume 4210 of Lecture Notes in Computer Science., Springer-Verlag (2006) 32–47
9. Calzone, L., Chabrier-Rivier, N., Fages, F., Soliman, S.: Machine learning biochemical networks from temporal logic properties. In Plotkin, G., ed.: Transactions on Computational Systems Biology VI. Volume 4220 of Lecture Notes in BioInformatics. Springer-Verlag (2006) 68–94 CMSB'05 Special Issue.
10. Antoniotti, M., Policriti, A., Ugel, N., Mishra, B.: Model building and model checking for biochemical processes. Cell Biochemistry and Biophysics **38** (2003) 271–286
11. Fages, F.: Temporal logic constraints in the biochemical abstract machine biocham (invited talk). In Springer-Verlag, ed.: Proceedings of Logic Based Program Synthesis and Transformation, LOPSTR'05. Number 3901 in Lecture Notes in Computer Science, London, UK (2005)
12. Regev, A., Silverman, W., Shapiro, E.Y.: Representation and simulation of biochemical processes using the pi-calculus process algebra. In: Proceedings of the sixth Pacific Symposium of Biocomputing. (2001) 459–470
13. Cardelli, L.: Brane calculi - interactions of biological membranes. In Danos, V., Schächter, V., eds.: CMSB'04: Proceedings of the second international workshop on Computational Methods in Systems Biology. Volume 3082 of Lecture Notes in BioInformatics., Springer-Verlag (2004) 257–280
14. Regev, A., Panina, E.M., Silverman, W., Cardelli, L., Shapiro, E.: Bioambients: An abstraction for biological compartments. Theoretical Computer Science **325**(1) (2004) 141–167
15. Danos, V., Laneve, C.: Formal molecular biology. Theoretical Computer Science **325**(1) (2004) 69–110
16. Phillips, A., Cardelli, L.: A correct abstract machine for the stochastic pi-calculus. Transactions on Computational Systems Biology (to appear) Special issue of BioConcur 2004.

17. Fages, F., Soliman, S., Chabrier-Rivier, N.: Modelling and querying interaction networks in the biochemical abstract machine BIOCHAM. Journal of Biological Physics and Chemistry **4**(2) (2004) 64–73
18. Calzone, L., Fages, F., Soliman, S.: BIOCHAM: An environment for modeling biological systems and formalizing experimental knowledge. BioInformatics **22**(14) (2006) 1805–1807
19. Fages, F.: From syntax to semantics in systems biology - towards automated reasoning tools. Transactions on Computational Systems Biology IV **3939** (2006) 68–70
20. Xu, R., Hu, X., II, D.C.W.: Inference of genetic regulatory networks from time series gene expression data. In: JCNN. Volume 2. (2004) 1215–1220
21. Nachman, I., Regev, A., Friedman, N.: Inferring quantitative models of regulatory networks from expression data. In: ISMB/ECCB (Supplement of Bioinformatics). (2004) 248–256
22. Chen, K.C., Csikász-Nagy, A., Györffy, B., Val, J., Novàk, B., Tyson, J.J.: Kinetic analysis of a molecular model of the budding yeast cell cycle. Molecular Biology of the Cell **11** (2000) 396–391

Elucidating transient protein interactions with multiple dockings

Ludwig Krippahl and Pedro Barahona

Dep. de Informática, Universidade Nova de Lisboa, 2825 Monte de Caparica, Portugal
ludi@di.fct.unl.pt, pb@di.fct.unl.pt

Abstract. Intelligent drug design requires a good understanding of protein interaction mechanisms. The rapid advance of high-throughput methods of protein structure determination gives us an exponential growth in known protein structures, but the problem of determining the structure of protein complexes is still a hard one to solve, especially for fast transient interactions, which are very common in catalysis and important targets for drugs. In this paper we present a modeling approach for trying to predict protein-protein complexes [1,2] by using constraint propagation techniques to incorporate information about the mechanism [3], and also to speed up calculations and make it more practical to compare a large number of docking simulations with similar partners in order to overcome the difficulty of modeling very weak interactions. As an example, we tested this approach on an electron transfer interaction that is part of photosynthesis in plants (FNR-ferredoxin).

1. Introduction

The improvement of experimental methods of protein structure determination provides an exponentially growing set of protein structures, freely available on the Protein Data Bank. Nevertheless, experimental determination of transient protein complexes is still extremely difficult. At the time of writing (June 2007) the Protein Data Bank contains over 44 thousand structures, but only five hundred transient complex structures according the latest version of PROTCOM database [4], with the disparity being higher for weaker interactions. Given that catalytic action requires a high turnover, and thus a weak interaction, this means that the more important the system the harder it is to determine the complex structure by experimental methods. This justifies the importance of docking algorithms to model protein interactions from the known structures of the partners, for understanding such interactions is crucial at many levels, from fundamental research in biochemistry through medicine and drug design.

Two decades since the pioneering work of Katzir and others [5] have seen significant developments in algorithms to generate models and scoring functions to select the most likely candidates. Examples from the CAPRI (Critical Assessment of Protein Interactions) experiment [6] illustrate the diversity of protein interaction modelling packages currently available. A common trend in these approaches is to try to model interactions using only knowledge derived from the structure and

physicochemical properties of the proteins involved. Some algorithms have been developed [1, 7] or adapted [8] to use data on the interaction mechanisms, but this approach is still the exception rather than the norm. Our algorithm, BiGGER (Bimolecular complex Generation with Global Evaluation and Ranking), is one of these exceptions, and the Chemera modelling package has been developed from the start to help the researcher bring into the modelling process as much data as available. Previous results show that BiGGER can be a powerful modelling tool when used in this manner [e.g. 2, 9, 10, 11, 12, 13]. In the following sections we will highlight some aspects of these results to provide context for our current research.

The main motivation for the multiple partners and multiple dockings approach is that the same reasons that make transient complexes difficult to determine make them difficult to model. For such weak and fast interactions surface contact area can be small, and electrostatics and solvation effects may play a more subtle role in guiding the partners instead of attaching them strongly together. Furthermore, the real "complex" may actually be a population of configurations the partners move through in an interaction rather than a more lasting structure.

This paper gives an overview of different levels at which docking simulations can help elucidate interaction mechanisms. The goal at the more detailed level is to predict the structure of a protein complex, but for transient interactions this is often not feasible, or even meaningless, as these interactions are fluid and dynamic rather than static assemblies. So we move on to the next level of providing ensembles of models that capture the likely dynamics of the interaction. Experimental data provides constraints at this stage to narrow down the possibilities, and to provide a different kind of analysis. Section 3.2 shows how constrained docking was used to find regions of fibrinogen susceptible to cleavage by a protease, even though in this case specific models of the complex would be neither reliable nor useful. Section 3.3 presents our current work in taking this approach even further. Models of transient complexes are not sufficiently reliable to detect minor modifications, such as the interference of a small molecule like most drugs. However, we show that it is possible to detect the effects of even small changes by standing back from the detail and assessing the pattern of docking models generated by BiGGER. Combining information from simulations with different partners and considering thousands of models together seems to be a powerful way of predicting some aspects of transient interactions.

The ability to predict the effect of small changes in protein structure or chemistry can be used in protein engineering (*e.g.* in guiding site directed mutagenesis experiments), diagnosis of genetic diseases due to particular combinations of alleles, predicting side effects of drugs, and potentially aiding future techniques such as gene therapy.

The paper is organized as follows. Section 2 explains the protein docking problem, which is to model the interaction between two proteins by determining how the two docking partners fit together, and the BiGGER algorithm, and gives an overview of our method. Section 3 illustrates the different levels of information provided by docking simulations with examples from previous and current work. Section 4 concludes this paper and discusses plans for future work.

BiGGER and Chemera are available at http://www.cqfb.fct.unl.pt/bioin/chemera/

2. The Docking Method

The foundation of the method described here is the BiGGER docking algorithm [1,2]. Like many other docking algorithms, BiGGER represents the protein structures using a regular cubic lattice of cells. Thus grid is a very straightforward representation where each cell can be either an empty cell, a surface cell, or a core cell. The surface cells define the surface of the protein, with the overlap of surface cells indicating a surface contact.

This grid representation has been optimized to model macromolecular interactions by fine-tuning the placement of the surface region relative to the structure to be modeled. Surface cells lie outside the Van der Waals surfaces of the atoms. When these surface cells overlap with those of another grid, the placement of the grids corresponds to a small separation of the Van der Waals surfaces on the protein complex, which is more realistic than actual Van der Waals contact.

The core cells are used to rule out overlaps, since overlapping core cells indicate that parts of the two structures are occupying the same space. Overlap between core and surface cells can be ignored because in our model the surface region corresponds to a layer external to the structure, as explained above. This makes it possible to model some flexibility, especially of exposed side chains, by removing their respective core region cells. This still favors surface contacts with the structure as specified by the PDB file, but allows the overlap or the more flexible side chains, which simulates their ability to rearrange during complex formation. This soft docking method is described in more detail in [1].

2.1 The Geometric Search

Given the definitions of allowed configurations and of how to measure surface contact, one can search all configurations by moving one grid relative to the other and examining the overlapping cells of the two grids. This translation search must be repeated for each orientation of one partner relative to the other, in order to search the rotation space as well. Typically, the rotational space is sampled in steps of 15º around each of the three orthogonal axes of rotation, for a total of approximately six thousand orientations.

The translation space is searched in small steps in all directions, each step corresponding to the size of one grid cell (1Å cube). BiGGER reduces the large number of possibilities by pruning the regions where the search can be determined to produce no useful models, using constraint propagation techniques. For details, please see [3], but most of the efficiency comes from encoding the grids as lists of intervals specifying the segments of similar cells along the X coordinate instead of encoding each cell individually. These lists are arranged in a two-dimensional array on the Y-Z plane.

This encoding reduces the memory requirements for storing the grids by more than two orders of magnitude relative to the classical Fast Fourier Transform approach [5] and also improves search efficiency.

2.2. Scoring

The geometric search is the first step in generating adequate models. At this stage BiGGER retains the configurations with the highest surface, typically 1000 or 5000. After the full search through all relative displacements and orientations, BiGGER scores these candidate models according to surface contact (number of overlapping surface cells), electrostatic interactions (based on a simple Coulombic model), estimated effect of desolvating the contact regions (solvent exclusion) and a statistical evaluation of side-chain contacts at the interface.

These measures are aggregated with a neural network trained to distinguish correct from incorrect models [1], and ranked according to the aggregated value.

2.3 Using Experimental Data

In some cases there is information about distances or contacts between parts of the proteins. This may be the distance between the redox centers to allow electron transfer, distance to active sites in general, NMR titration data indicating perturbations at some residues, site directed mutagenesis experiments, and so forth.

The main problem is that experimental data is always noisy, so it is not feasible to simply impose a constraint forcing all contacts and distances suggested by the data. The most common situation is to have a set of likely distance constraints of which not all necessarily hold. Typically, we have to impose a constraint of the form:

At least K atoms of set A must be within R of at least one atom of set B (**1**)

where set A is on one protein and set B on the other, and R a distance value. This leads to a combinatorial problem with a large number of disjunctions that can be solved by simply evaluating each candidate structure generated. Any docking method is suited to incorporate experimental data in this generate-and-test fashion, and this was our first approach [10, 11, 12,]. However, this approach is very inefficient, and BiGGER is especially suited to incorporate these constraints by pruning the search space and decrease computation time. This is currently implemented, and was already used in published work [3, 13]. It is especially important when doing a large number of comparative docking runs, in the multiple docking method (below).

2.4 Multiple docking

Previous work showed us the importance of comparing different docking simulations. For example, in [12], we compared the models generated for the interaction of ferredoxin with ferredoxin NADP reductases from different organisms. The purpose was to validate the method with known complex structures to support the prediction of a novel complex, and this involved comparing only a few simulations. Even so it revealed the difficulties in processing large data sets of hundreds or thousands of relevant models.

Multiple docking is an extension of this to a large number of simulations, combining different partners in order to extract higher level of information such as the

likely relative affinity, dominating factor for the interaction, most likely interface regions, and so forth. While the basic idea is simply to do a large number of docking runs, it raises some performance issues that can be addressed by the efficient incorporation of additional data, and especially it raises the problem of processing the results. For example, current work in comparing different ferredoxin and ferredoxin reductase complexes involves six reductases, eight ferredoxins, 48 different docking simulations, and tens of thousands of models.

2.5 Interpreting results

The complexity of multiple docking results motivated our recent focus in better ways to interpret the data at a high level. Part of this task is a simple matter of doing statistics on the sets of models generated, such as different percentiles for each interaction score, maximum and minimum values, and so forth. But one good indicator of a reliable docking simulation is the clustering of the models. Figure 1 illustrates showing two sets of docking models. Each set is represented by the structure of one partner surrounded by spheres indicating the geometric center of the other partner in each model generated.

Although we can detect these patterns by visual inspection, it is not practical to do so for dozens of docking simulations, and especially hard to compare the results without quantifying this distribution. So we developed a simple way to quantify the scattering of models. For each pair of models we measure the distance between the geometric centers. Thus if we have proteins A and B, for models 1 and 2 we measure the distance between the center of A in model 1 and the center of A in model 2, considering B fixed, and then do the same for B. With these distance values we build a histogram counting the number of occurrences of each distance range (0Å to 1Å, 1Å to 2Å and so forth). Finally we divide the number in each bin by the number expected if the models were uniformly distributed on the surface of a sphere with a radius identical to the average distance between the centers of the two partners,

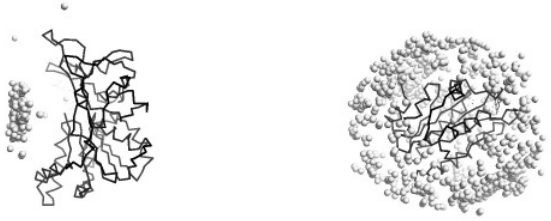


Fig. 1. The left panel shows a strongly clustered set of docking models (spheres to the left of the structure). The right panel shows a set of models with no evident clusters.

A peak in the distribution at the lower distance ranges indicates clustering, while no clustering results in a flat line. This makes it easy to compare many docking runs with different partners at the same scale and in a quantifiable manner (the distance corresponding to the peak and the peak height).

3 Previous Results and Current Work

The first part of this section covers our previously published work in this area. This gives context to the more recent results and current work, and helps explain method as a whole. We refer the reader to the cited publications for more details on each case.

3.1 Modeling Transient Complexes

Our initial experience with electron transfer complexes was invaluable by revealing in first hand the difficulties of modeling transient complexes. To compare the interactions of a bacterial Cytochrome c Peroxidase with cytochrome c and non-physiological partners from different organisms (9) it was necessary to consider experimental data (especially titration data) and the interaction mechanism, which requires proximity between the redox centers. This was the best way to properly judge the interaction scores and evaluate the candidate models to select the most likely structures. This effort also revealed the problem of analyzing the results. One docking simulation generates thousands of candidate structures, and it proved to be a laborious task to compare and analyze even a few simulations.

3.2 Constrained Docking

In order to improve results, we included experimental data in the docking simulation itself. The first approach was to score the candidate models according to distances or contacts between defined regions. The results were promising in modeling electron transfer complexes (10, 11, 12, 13) and protein digestion [14], and eventually we included the constraints during the search stage instead of at the scoring stage only, in order to prune the search space and reduce computation costs. In [3] we report the comparison of constrained and unconstrained docking in realistic conditions, for five CAPRI [6] targets, where computation time was approximately one fifth that of the unconstrained docking.

3.3 Multiple docking

We are currently applying the multiple docking method to the interaction of ferredoxins with different ferredoxin NADP reductases (FNRs), an important step in photosynthesis. One interesting preliminary result is the difference between the docking patterns for the interaction of several ferredoxins with the FNR from spinach (*Spinacia oleracea*) when there is an ADP molecule bound to the enzyme. The ADP molecule is only about 1% the size of FNR, and the conformational changes on the enzyme are minimal (the root mean square deviation between the two structures is only 0.26Å).

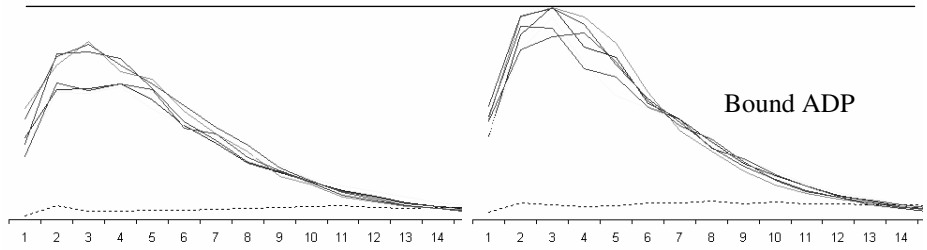


Fig. 2. Normalized histograms for docking 6 ferredoxins (gray solid lines) with spinach FNR. On the right side for the FNR without ADP, on the left side FNR with bound ADP. The dashed line lower in the graphs shows the control docking with an unrelated protein (phthalate reductase from *Pseudomonas*).

However, as Figure 2 shows, the effect is quite evident when we quantify the clustering in a normalized histogram, as described in section 2.5. With only visual inspection of the docking results this would not be noticeable, and any differences in the highest ranking models would be attributable to the variations expected from modeling weak interactions.

4. Conclusion

Transient protein complexes are difficult to model, and the information that can be obtained from individual models is unreliable and often unrepresentative of the interaction. Dealing with ensembles of models allows us to analyze the results at different levels, and trade structural detail for sensitivity to small differences such as mutations or bound cofactors and drugs. The role of AI in this method is not, as is often the case, to apply one technique to solve part of the problem, but to bring together different techniques to attack the problem from different directions. Clustering, neural networks, and constraint programming are part of the docking algorithm and data processing. Our current work includes reasoning and reactivity to help keep interactions analysis synchronized with the availability of new structures.

The ultimate goal of this approach is to have a computational method sensitive to small effects such as point mutations or small molecules on protein interactions, to help predict the effects that different drugs or drug combinations can have in different people.

References

1. Palma PN, Krippahl L, Wampler JE, Moura, JJG. 2000. BiGGER: A new (soft) docking algorithm for predicting protein interactions. Proteins: Structure, Function, and Genetics 39:372-84.
2. Krippahl L, Moura JJ, Palma PN. 2003. Modeling protein complexes with BiGGER. Proteins: Structure, Function, and Genetics. V. 52(1):19-23.
3. Krippahl, L, Barahona P. Applying Constraint Programming to Rigid Body Protein Docking (2005), Lecture Notes in Computer Science CP 2005: 373-387.

4. P.J.Kundrotas and E.Alexov (2007) Nucleic Acids Research, v.35, pp. D575-D579.
5. Katchalski-Katzir E, Shariv I, Eisenstein M, Friesem AA, Aflalo C, Vakser IA. 1992 Molecular surface recognition: determination of geometric fit between proteins and their ligands by correlation techniques. Proc Natl Acad Sci U S A. 1992 Mar 15;89(6):2195-9.
6. Janin J. 2002. Welcome to CAPRI: A Critical Assessment of PRedicted Interactions. Proteins: Structure, Function, and Genetics Volume 47, Issue 3, 2002. Pages: 257
7. Dominguez C, Boelens R, Bonvin AM. HADDOCK: a protein-protein docking approach based on biochemical or biophysical information. J Am Chem Soc. 2003 Feb 19;125(7):1731-7.
8. Gabb HA, Jackson RM, Sternberg MJ. Modelling protein docking using shape complementarity, electrostatics and biochemical information. J Mol Biol. 1997 Sep 12;272(1):106-20.
9. Pettigrew GW, Goodhew CF, Cooper A, Nutley M, Jumel K, Harding SE. 2003, The electron transfer complexes of cytochrome c peroxidase from Paracoccus denitrificans. Biochemistry. 2003 Feb 25;42(7):2046-55.
10. Morelli X, Dolla A., Czjzek M, Palma PN, Blasco, F, Krippahl L, Moura JJ, Guerlesquin F. 2000. Heteronuclear NMR and soft docking: an experimental approach for a structural model of the cytochrome c553-ferredoxin complex. Biochemistry 39:2530-2537.
11. Morelli X, Palma PN, Guerlesquin F, Rigby AC. 2001. A novel approach for assessing macromolecular complexes combining soft-docking calculations with NMR data. Protein Sci. 10:2131-2137.
12. Palma PN, Lagoutte B, Krippahl L, Moura JJ, Guerlesquin F.. Synechocystis ferredoxin/ferredoxin-NADP(+)-reductase/NADP+ complex: Structural model obtained by NMR-restrained docking. FEBS Lett. 2005 Aug 29;579(21):4585-90.
13. Krippahl L, Palma PN, Moura I., Moura JG. Modelling the Electron-Transfer Complex Between Aldehyde Oxidoreductase and Flavodoxin (2006). Eur. J. of Inor. Chem., Volume 2006, Issue 19 , Pages 3835 - 3840
14. Monaco S, Gioia M, Rodriguez J, Fasciglione GF, Di Pierro D, Lupidi G, Krippahl L, Marini S, Coletta M.Modulation of the proteolytic activity of matrix metalloproteinase-2 (Gelatinase A) on fibrinogen.Biochem J., Biochem J. 2007 Mar 15;402(3):503-13.

Constraint-Based Analysis of Gene Deletion in a Metabolic Network

Abdelhalim Larhlimi and Alexander Bockmayr

DFG-Research Center Matheon, FB Mathematik und Informatik, Freie Universität Berlin, Arnimallee, 3, 14195 Berlin, Germany
e-mail: larhlimi@mi.fu-berlin.de,bockmayr@mi.fu-berlin.de

Abstract. Constraint-based models of metabolic networks use governing constraints to restrict potential cellular behavior. The range of possible behaviors, which is mathematically described by the steady-state flux cone, can be altered by gene deletion. Several optimization-based approaches have been proposed to find, in the altered network, behaviors optimizing a particular network function. Here, we present a constraint-based approach that allows analyzing the changes in the overall capabilities of a metabolic network following a gene deletion. Furthermore, we establish a relationship between the altered flux cone and the reversibility type of the reaction associated with the deleted gene.

1 Introduction

Constraint-based models have become a fundamental tool to study genome-scale metabolic networks [10]. Such models use governing constraints to restrict potential cellular behavior. The range of all possible behaviors, which is mathematically described by the steady-state flux cone, can be altered by gene deletion. Several optimization-based approaches have been developed that allow computing, in the altered network, behaviors optimizing a particular network function [2]. Mathematically, this requires defining a hypothetical objective function. However, although the assumption of optimality for a wild-type biological system is justifiable, the same assumption may not be valid for studying an altered system [15]. Furthermore, these approaches consider only optimal states with respect to the predefined objective function. These particular states form a restricted subset of all possible behaviors of the altered system. Hence, these optimization-based approaches loose information about how the achievable behaviors of the network could change following a gene deletion.

In this paper, we analyze the changes in the overall capabilities of a metabolic network caused by gene deletion. In particular, we show how to obtain in a constraint-based approach a description of the altered steady-state flux cone. The analysis is based on a refined classification of reactions.

The organization of this paper is as follows. In Sect. 2, we recall some basic facts about metabolic network analysis and present the notions of minimal metabolic behavior and reversible metabolic space. This leads to a refined classification of reactions. In Sect. 3 we use this for a constraint-based analysis of gene deletion.

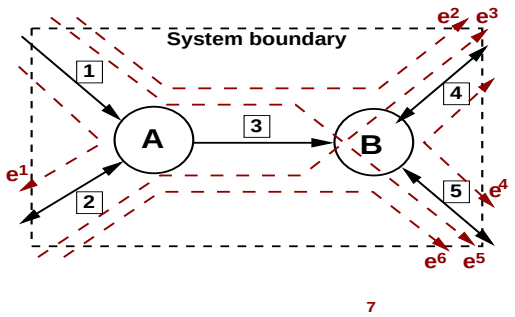


Fig. 1. Network ILLUSNET with the corresponding elementary modes.

2 Metabolic network analysis

In the context of metabolic network analysis, metabolic systems are assumed to operate at steady state such that for all internal metabolites the flux is balanced. In addition, the flux through each irreversible reaction must be non-negative. Fluxes through reversible reactions are not restricted with respect to their sign. The set of all possible flux distributions over the network at steady state defines the (steady-state) *flux cone* [10]

$$C = \{v \in \mathbb{R}^n \mid Sv = 0,\ v_i \geq 0,\ \text{for all } i \in Irr\}, \tag{1}$$

where S is the $m \times n$ stoichiometric matrix of the network, with m internal metabolites (rows) and n reactions (columns), and the vector $v \in \mathbb{R}^n$ gives a *flux distribution*. $Irr \subset \{1, \ldots, n\}$ denotes the set of *irreversible* reactions in the network, and $Rev = \{1, \ldots, n\} \setminus Irr$ the set of *reversible* reactions.

Example 1. For illustration we consider the hypothetical network ILLUSNET depicted in Fig. 1. It consists of two metabolites (A, B), and five reactions $\{1, \ldots, 5\}$. The flux cone is defined by $C = \{v \in \mathbb{R}^5 \mid Sv = 0, v_i \geq 0,\ \text{for all } i \in Irr\}$, with the set of irreversible reactions $Irr = \{1, 3\}$, and the stoichiometric matrix

$$S = \begin{pmatrix} 1 & -1 & -2 & 0 & 0 \\ 0 & 0 & 3 & -1 & -1 \end{pmatrix}.$$

The flux cone contains the full range of achievable behaviors of the metabolic network at steady state. Hence, it is of great interest to describe this cone in a mathematically and biologically meaningful way. There are two mathematical ways for describing the flux cone. The first is an *inner* description based on a set of generating vectors that span the cone. The second is an *outer* description based on sets of constraints, which gives a test for determining whether a given flux vector belongs to the cone [1].

The concept of *elementary mode (EM)* [13, 14] has been proposed to characterize the flux cone using an inner description. An EM corresponds to a flux

distribution $v \in C \setminus \{0\}$ involving a minimum set of reactions, i.e., the set $S^c(v) = \{i \in Rev \cup Irr \mid v_i \neq 0\}$ is minimal. Every possible flux distribution is then a non-negative combination of elementary modes. In contrast to this approach, we propose in [7, 9] an outer description of the flux cone, based on sets of non-negativity constraints. This approach defines a *metabolic behavior* as a set of *irreversible* reactions $D \subseteq Irr, D \neq \emptyset$, such that there exists a flux distribution $v \in C$ with $D = \{i \in Irr \mid v_i \neq 0\}$. A metabolic behavior D is *minimal (MMB)*, if there is no metabolic behavior $D' \subsetneq D$ strictly contained in D.

The set of flux distributions involving only reversible reactions defines the *reversible metabolic space (RMS)*,

$$RMS = \{v \in C \mid v_i = 0, \text{ for all } i \in Irr\}, \tag{2}$$

which corresponds to the lineality space of the flux cone C [12]. The dimension t of the reversible metabolic space, by definition, is equal to the dimension of the lineality space of C, which is a linear subspace of $\mathbb{R}^n$.

The minimal metabolic behaviors (MMBs) are closely related to the *minimal proper faces* of the flux cone C, i.e., the faces of dimension $t+1$ [12]. According to [7, 9], each minimal proper face is described by its *characteristic set* $D = \{j \in Irr \mid v_j > 0, \text{ for some } v \in G\}$. Indeed, G is given by

$$G = \{v \in C \mid \ v_i = 0, \text{ for all } i \in Irr \setminus D\}. \tag{3}$$

The characteristic set D is uniquely determined by G and all reactions from D are proportional to each other. The next theorem shows that the MMBs are in a 1-1 correspondence with (the characteristic sets of) the minimal proper faces of the flux cone C.

Theorem 1 ([9]). *Let $D \subseteq Irr$ be a set of irreversible reactions. Then, the following are equivalent:*

- *D is a minimal metabolic behavior.*
- *D is the characteristic set of a minimal proper face of the flux cone.*

If $G^1, \ldots, G^s$ are the minimal proper faces of the flux cone C, the corresponding MMBs $D^1, \ldots, D^s$ together with the RMS completely define C, see [7, 9] for additional details.

Example 2. In the metabolic network from Fig. 1, there are six elementary modes $\{e^1, \ldots, e^6\}$. The MMBs, the corresponding minimal proper faces and the RMS are the following:

$$\begin{aligned} &D^1 = \{1\}, \quad D^2 = \{3\}, \\ &G^1 = \{v \in C \mid v_1 \geq 0, \ v_3 = 0\}, \quad G^2 = \{v \in C \mid v_3 \geq 0, \ v_1 = 0\}, \\ &RMS = \{v \in C \mid v_1 = 0 \text{ and } v_3 = 0\} \end{aligned}$$

Fig. 2(a) shows a 3D illustration of the flux cone C. The RMS is a line generated by the flux distribution $v = (0, 0, 0, 1, -1)$, i.e., $RMS = \{\lambda(0, 0, 0, 1, -1) \mid \lambda \in \mathbb{R}\}$. The minimal proper faces G^1 and G^2 are two half-planes.

Note that minimal metabolic behaviors satisfy a simplicity condition similar to the one that holds for elementary modes. Furthermore, for each MMB D, there exists at least one EM e involving exactly the irreversible reactions from D, i.e., $D = \{i \in Irr \mid e_i \neq 0\}$. The number of MMBs is typically much smaller than the number of EMs. For instance, the central carbon metabolism of E. coli contains more than 500000 elementary modes [5], but only 3560 minimal metabolic behaviors. The RMS for this metabolism is reduced to the origin, i.e., $RMS = \{0\}$.

The MMBs and the RMS of a metabolic network can be determined from a set of generators of the flux cone C. Software packages for polyhedral computations, such as cdd [3], allow for computing these generators. The number of generators of C may be exponential in the size of the inequality description in Equation (1). For more about complexity issues, we refer to [4].

Based on the concepts of MMBs and the RMS, a refined classification of reactions has been proposed [7]. A reversible reaction $j \in Rev$ is called *pseudo-irreversible* if $v_j = 0$, for all $v \in RMS$. A reversible reaction that is not pseudo-irreversible is called *fully reversible.* Inside each minimal proper face, the irreversible and the pseudo-irreversible reactions take a unique direction. More precisely, we have the following properties.

Theorem 2 ([7]). *Let G be a minimal proper face of the flux cone C and let $j \in \{1, \ldots, n\}$ be a reaction.*

- *If $j \in Irr$ is irreversible, then $v_j > 0$, for all $v \in G \setminus RMS$, or $v_j = 0$, for all $v \in G$. Furthermore, $v_j = 0$, for all $v \in RMS$.*
- *If $j \in Rev$ is pseudo-irreversible, then the flux v_j through j has a unique sign in $G \setminus RMS$, i.e., either $v_j > 0$, for all $v \in G \setminus RMS$, or $v_j = 0$, for all $v \in G \setminus RMS$, or $v_j < 0$, for all $v \in G \setminus RMS$. For all $v \in RMS$, we have again $v_j = 0$.*
- *If $j \in Rev$ is fully reversible, there exists $v \in RMS$ such that $v_j \neq 0$. We can then find pathways $v^+, v^-, v^0 \in G \setminus RMS$ with $v_j^+ > 0$, $v_j^- < 0$ and $v_j^0 = 0$.*

Example 3. In the ILLUSNET network, reaction 2 is pseudo-irreversible, while reactions 4 and 5 are fully reversible. In the context of the minimal proper face G^1, the pseudo-irreversible reaction 2 operates only in the forward direction, i.e., $v_2 > 0$ for all $v \in G^1 \setminus RMS$, while it operates in the backward direction in the context of the face G_2.

In the following, *removing a reaction* means that the flux through this reaction is constrained to zero. Based on the refined reaction classification, we studied in [8] the consequences of removing a reaction in terms of the capabilities of the remaining reactions in the network. A zero flux through some reaction may imply a zero flux through many other reactions. It has been shown that the reversibility property is an important key to elucidate interactions between reactions. For instance, the removal of a (pseudo-)irreversible reaction has no effect on the fully reversible reactions, and all reactions in an enzyme subset [11] must have the same reversibility type (irreversible, pseudo-irreversible, or fully reversible).

In the following, we are interested in gene deletion. More precisely, we study how the flux cone is changed if a reaction associated with the deleted gene is removed. This reaction will be called the *target reaction*. Based on the mathematical results above, the following section establishes a relationship between the changes in the flux cone and the reversibility type of the target reaction.

3 Constraint-based analysis of gene deletion

Let $\tau \in \{1, \ldots, n\}$ be the target reaction associated with the deleted gene. To simulate gene deletion, we constrain the flux through reaction τ to zero. This leads to the altered flux cone

$$C' = \{v \in \mathbb{R}^n \mid Sv = 0,\ v_\tau = 0,\ v_i \geq 0,\ \text{for all } i \in Irr\} \tag{4}$$

which contains all possible steady-state flux distributions over the altered network.

The altered flux cone C' can be described using an existing description of the flux cone C. To do this, the elementary mode approach takes advantage of the *conservation property* of EMs: if the flux through a reaction is constrained to zero, the set of EMs of the altered network is the set of all EMs which do not involve this reaction [6, 14].

Example 4. In the ILLUSNET network, if we constrain the flux through reaction 4 to zero, the elementary modes of the altered flux cone C' are e^1, e^5 and e^6, which do not involve reaction 4.

To describe the metabolic network after a gene deletion, the EM approach does not explicitly take into account the reversibility type of the target reaction. However, this becomes possible by using minimal metabolic behaviors and the reversible metabolic space. Mathematically, the cone C' defined in Equation (4) is also given by $C' = C \cap \{v \in \mathbb{R}^n \mid v_\tau = 0\}$. Therefore, we may deduce an outer description of the altered cone C' from an outer description of the original cone C. This deduction depends on the reversibility type of the target reaction τ.

3.1 Removing an irreversible reaction

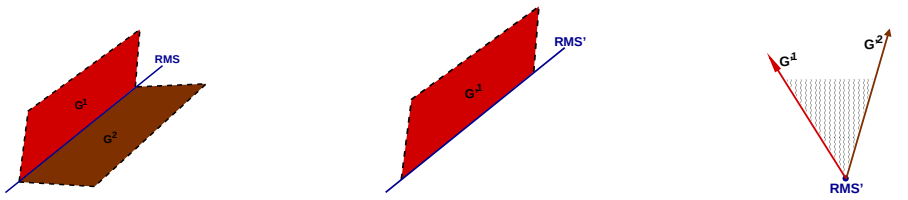
In analogy with EMs, there is the following *conservation property* for MMBs: if the flux through an irreversible reaction is constrained to zero, the set of MMBs of the new network is the set of all MMBs which do not involve this reaction. Hence, if reaction τ is irreversible, this conservation property guarantees that the MMBs of C' are exactly the MMBs D of C for which $j \notin D$. The reversible metabolic space does not change, i.e., $RMS' = RMS$.

3.2 Removing a pseudo-irreversible reaction

In this case, only MMBs in which reaction τ is not involved are kept, and new ones are generated. The generation of new MMBs relies on the *adjacency property* of the minimal proper faces of the cone C [3]. Indeed, consider the hyperplane $H = \{v \in \mathbb{R}^n \mid v_\tau = 0\}$. Let $H^+ = \{v \in \mathbb{R}^n \mid v_\tau > 0\}$ (resp. $H^- = \{v \in \mathbb{R}^n \mid v_\tau < 0\}$) be the positive (resp. negative) half-space supported by the hyperplane H. Then H partitions the set of minimal proper faces of C into three parts: the set J^+ of *positive* minimal proper faces G for which $G \setminus RMS \subseteq H^+$, the set J^- of *negative* minimal proper faces G for which $G \setminus RMS \subseteq H^-$, and the set J^0 of *zero* minimal proper faces contained in H. The new minimal proper faces of C' are obtained by combining a positive and a negative minimal proper face that are contained in a $(t+2)-$dimensional face of the flux cone C [3]. Again, t denotes the dimension of the RMS. Since only positive combinations are performed, all irreversible reactions defining a minimal proper face G^1 will define a new face G' if the latter is obtained by combining G^1 with another minimal proper face G^2. The MMB D' associated with G' is then the union of the MMBs D^1 and D^2 associated with G^1 and G^2, respectively. Therefore, the new MMBs of the cone C' can be computed by (a) identifying positive and negative MMBs associated with the positive and negative minimal proper faces of C, (b) computing all possible unions between positive and negative MMBs of C, and (c) keeping only those which are minimal.
Finally, the reversible metabolic space does not change, i.e., $RMS' = RMS$.

3.3 Removing a fully reversible reaction

The unique effect of removing a fully reversible reaction is the reduction of the dimension of the reversible metabolic space, i.e., $\dim(RMS') = \dim(RMS) - 1$. The MMBs of C and C' are the same.



(a) (b) (c)

Fig. 2. Here, 2(a) gives a 3D illustration of the flux cone C, 2(b) shows the altered flux cone C' after removing the irreversible reaction 3. Finally, 2(c) shows the altered cone C' after removing the fully reversible reaction 4.

Example 5. Fig. 2 shows the flux cone C and the altered cone C' depending on the reversibility type of the target reaction. Fig. 2(b) shows the cone C' after the removal of the irreversible reaction 3. In this case, the reversible metabolic space does not change and only the MMB D^1, which does not involve reaction 3, is still an MMB for the cone C'. Fig. 2(c) shows the cone C' after the removal of the fully reversible reaction 4. In this case, the flux cone becomes pointed, i.e., the reversible metabolic space is reduced to the origin $\{0\}$, and the MMBs of C and C' are the same.

The results above can be extended to predict the effect on the flux cone when constraining the reversibility of some reaction. If a reversible reaction ι is constrained to operate in the positive (resp. negative) direction only, the resulting flux cone will be $C'' = C \cap \{v \in \mathbb{R}^n \mid v_\iota \geq 0\}$ (resp. $C'' = C \cap \{v \in \mathbb{R}^n \mid v_\iota \leq 0\}$). Again, the description of C'' can be deduced from that of the flux cone C depending on the reversibility type of reaction ι. Indeed, if ι is pseudo-irreversible, the MMBs of C'' are the MMBs of the altered cone C', defined in Equation (4), together with the MMBs corresponding to the positive (resp. negative) minimal proper faces of C. The reversible metabolic space does not change, i.e., $RMS'' = RMS$. On the other hand, if ι is fully reversible, the MMBs of C'' are the MMBs of C, together with a new MMB $D = \{\iota\}$ and $\dim(RMS'') = \dim(RMS) - 1$.

4 Conclusion

In this paper, we have shown that the outcome of deleting a gene or constraining the reversibility of some reaction mainly depends on the reversibility type of the target reaction. Possible effects include not only changes in the steady-state flux cone, but also changes in the reversibility of reactions. Indeed, some reversible reactions become unable to operate in the forward and backward direction, while others become unable of carrying any flux under steady-state conditions. The importance of a target reaction can then be assessed by the amount of all these changes.

References

1. B. L. Clarke. Stability of complex reaction networks. In I. Prigogine and S.A. Rice, editors, *Advances in Chemical Physics*, volume 43, pages 1–216. John Wiley & Sons, 1980.
2. J. S. Edwards and B. O. Palsson. Metabolic flux balance analysis and the in silico analysis of escherichia coli k-12 gene deletions. *BMC bioinformatics*, 1:1, 2000.
3. K. Fukuda and A. Prodon. Double description method revisited. In *Combinatorics and Computer Science*, pages 91–111. Springer, LNCS 1120, 1995.
4. L. Khachiyan, E. Boros, K. Borys, K. Elbassioni, and V. Gurvich. Generating all vertices of a polyhedron is hard. In *SODA '06: 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 758–765, New York, 2006.

5. S. Klamt and J. Stelling. Combinatorial complexity of pathway analysis in metabolic networks. *Mol. Biol. Rep.*, 29(1-2):233–236, 2002.
6. S. Klamt and J. Stelling. Two approaches for metabolic pathway analysis? *Trends Biotechnol.*, 21:64–69, 2003.
7. A. Larhlimi and A. Bockmayr. Minimal metabolic behaviors and the reversible metabolic space. Preprint 299, DFG Research Center Matheon, December 2005. `http://page.mi.fu-berlin.de/~bockmayr/MMB.pdf`.
8. A. Larhlimi and A. Bockmayr. A new approach to flux coupling analysis of metabolic networks. In *Computational Life Sciences II, CompLife'06, Cambridge, UK*, pages 205–215. Springer, LNBI 4216, 2006.
9. A. Larhlimi and A. Bockmayr. A new constraint-based description of the steady-state flux cone of metabolic networks. Submitted, 2006.
10. J. A. Papin, J. Stelling, N. D. Price, S. Klamt, S. Schuster, and B. O. Palsson. Comparison of network-based pathway analysis methods. *Trends Biotechnol.*, 22(8):400–405, 2004.
11. T. Pfeiffer, I. Sánchez-Valdenebro, J. C. Nuno, F. Montero, and S. Schuster. Metatool: for studying metabolic networks. *Bioinformatics*, 15(3):251–257, 1999.
12. A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, 1986.
13. S. Schuster and C Hilgetag. On elementary flux modes in biochemical reaction systems at steady state. *J. Biol. Syst.*, 2(2):165–182, 1994.
14. S. Schuster, C. Hilgetag, J. H. Woods, and D. A. Fell. Reaction routes in biochemical reaction systems: algebraic properties, validated calculation procedure and example from nucleotide metabolism. *J. Math. Biol.*, 45:153–181, 2002.
15. D. Segrè, D. Vitkup, and G. M. Church. Analysis of optimality in natural and perturbed metabolic networks. *Proc. Natl. Acad. Sci. U.S.A.*, 99(23):15112–15117, 2002.

Mendelian error detection in complex pedigrees using weighted constraint satisfaction techniques

Marti Sanchez, Simon de Givry, Thomas Schiex
{msanchez,degivry,tschiex}@toulouse.inra.fr

INRA-UBIA Toulouse, France

Abstract. With the arrival of high throughput genotyping techniques, the detection of likely genotyping errors is becoming an increasingly important problem. In this paper we are interested in errors that violate Mendelian laws. The problem of deciding if Mendelian error exists in a pedigree is NP-complete [1]. Existing tools dedicated to this problem may offer different level of services: detect simple inconsistencies using local reasoning, prove inconsistency, detect the source of error, propose an optimal correction for the error. All assume that there is at most one error. In this paper we show that the problem of error detection, of determining the minimum number of error needed to explain the data (with a possible error detection) and error correction can all be modeled using soft constraint networks. Therefore, these problems provide attractive benchmarks for weighted constraint network (WCN) solvers. Because of their sheer size, these problems drove us into the development of a new WCN solver `toulbar2` [1] which solves very large pedigree problems with thousands of animals, including many loops and several errors. This paper is a summary of an extended version to appear [17].

Biological background and motivations

A pedigree is defined by a set of individuals, their parental relationship and their associated genetic information. For each individual we consider its *genotype*, that is the pair of *alleles* (genetic information at one position in the chromosome) inherited from the parents. An individual is called a founder if its parents are not among the individuals present in the pedigree. Genotypes are not always completely observable and the indirect observation of a genotype (its expression) is termed the *phenotype*. The genetic information may be corrupted because of experimental and human errors. We only consider in this paper typing errors, also called phenotype error. A typing/phenotype error means simply that the phenotype in the pedigree is incompatible with the true (unknown) genotype. Phenotype errors are also called Mendelian errors when they make a pedigree inconsistent with Mendelian law of inheritance which states that the pair of alleles of every individual is composed of one paternal and one maternal allele. The problem of checking pedigree consistency is actually shown to be NP-complete in [1].

The detection and correction of errors is crucial before the data can be further exploited. Because of its NP-completeness, most existing tools only offer a limited polynomial time checking procedure. The different tools we are aware of that try to tackle

[1] `toulbar2` and `mendelsoft` are accessible from: http://www.inra.fr/mia/T/MendelSoft/

this problem are either incomplete, restricted by strong assumptions (such as unique error), or incapable of dealing with large problems.

In this paper, we introduce soft constraint network models, the problem of determining the minimum number of errors needed to explain the data and the problem of proposing an optimal correction to an error. In Section 2, we describe the algorithms used to solve these problems. We report extensive results using the weighted constraint network solver `toulbar2` and other solvers in Section 3.

1 Modeling the problems

A Weighted Constraint Network (WCN) $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ [6] is defined by a set of n variables $\mathcal{X} = \{x_1, \ldots, x_n\}$, a set of matching domains $\mathcal{D} = \{D_1, \ldots, D_n\}$ with maximum size equal to d and a set of e constraints $\mathcal{C}$. Every variable $x_i \in \mathcal{X}$ takes its value in the associated domain D_i. A constraint $c_S \in \mathcal{C}$, is a cost function that assigns an integer cost (from 0 to a maximum k) to all the possible assignments of variables in S. The minimum and maximum costs will be also denoted by $\bot$ and $\top$. We redefine the sum to include an absorbing element k: $a \oplus b = min\{k, a+b\}$. The set S is called the scope of the constraint and $|S|$ its arity. For every variable assignment A, $c_S(A[S]) \in N$ represents the cost of the constraint for the given assignment where $A[S]$ is the projection of A on the constraint scope S. In this paper we consider arities of one, two and three: a unary weighted constraint C_i is a cost function $C_i(a \in D_i) \rightarrow [0..k]$. A binary constraint C_{ij} is a cost function $C_{ij}(a \in D_i, b \in D_j) \rightarrow [0..k]$ A ternary constraint C_{ijl} is a cost function $C_{ijl}(a \in D_i, b \in D_j, c \in D_l) \rightarrow [0..k]$. We assume the existence of a unary constraint C_i for every variable and a *zero*-arity constraint (i.e. a constant), noted $C_\varnothing$.

The aim is then to find an assignment A of all variables such that the sum of all tuple costs $\bigoplus_{c_S \in C} c_S(A[S])$ is minimum. This is called the Weighted Constraint Satisfaction Problem (WCSP), and is NP-hard. Several recent algorithms for tackling this problem, all based on the maintenance of local consistency properties have been recently proposed [8, 4, 7]. They are presented in Section 2.

Now consider a pedigree defined by a set I of individuals. For a given individual $i \in I$, we note $pa(i)$ the set of parents of i. Either $pa(i) \neq \varnothing$ (non founder) or $pa(i) = \varnothing$ (founder). At the locus considered, the set of possible alleles is denoted by $\{1, ..., m\}$. Therefore, each individual carries a genotype defined as an unordered pair of alleles (one allele from each parent, both alleles can be identical). The set of all possible genotypes is denoted by G and has cardinality $\frac{m(m+1)}{2}$. For a given genotype $g \in G$, the two corresponding alleles are denoted by g^l and g^r and the genotype is also denoted as $g^l|g^r$. By convention, $g^l \leq g^r$ in order to break symmetries between equivalent genotypes (e.g. 1|2 and 2|1). The experimental data is made of phenotypes. For each individual in the set of observed individuals $I' \subset I$, its observed phenotype restricts the set of possible genotypes to those which are compatible with the observed phenotype. This set is denoted by $G(i)$ (very often $G(i)$ is a singleton, observation is complete).

A corresponding weighted constraint network encoding this information uses one variable per individual *i.e.* $\mathcal{X} = I$. The domain of every variable $i \in \mathcal{X}$ is simply defined as the set of all possible genotypes G. If an individual i has an observed phenotype, a unary soft constraint that involves the variable i is added. To model the possibility

of typing errors, genotypes in G which are incompatible with the observed phenotype $G(i)$ should not be completely forbidden. Instead, a soft constraint forbids them with a cost of 1 (since using such a value represents one typing error). Finally, to encode Mendelian law, and for every non founder individual $i \in \mathcal{X}$, a single ternary constraint involving i and the two parents of i, $pa(i) = \{j,k\}$ is added. This constraint assigns cost 0 to triples (g_i, g_j, g_k) of genotypes that verify Mendelian inheritance *i.e.* such that one allele of g_i appears in g_j and the other appears in g_k. Equivalently: $(g_i^l \in g_j \wedge g_i^r \in g_k) \vee (g_i^l \in g_k \wedge g_i^r \in g_j)$. Ternary constraints assign the maximum cost k to forbidden combinations. For a pedigree with n individuals among which there are f founders, with m possible alleles, we obtain a final WCSP with n variables, a maximum domain size of $\frac{m(m+1)}{2}$, n unary constraints (the unobserved individuals have a trivially null unary constraint) and $n-f$ ternary constraints.

If we consider an assignment of all variables to indicate the real genotype of all individuals, the sum of all the costs induced by all unary constraints on this assignment precisely gives the number of errors made during typing. Finding an assignment with a minimum number of errors follows the traditional parsimony principle (or Occam's razor) and is consistent with a low probability of independent errors (quite reasonable here). This defines the Parsimony problem. One solution of the corresponding WCSP with a minimum cost therefore defines a possible diagnostic. The model shifts from satisfaction to optimization.

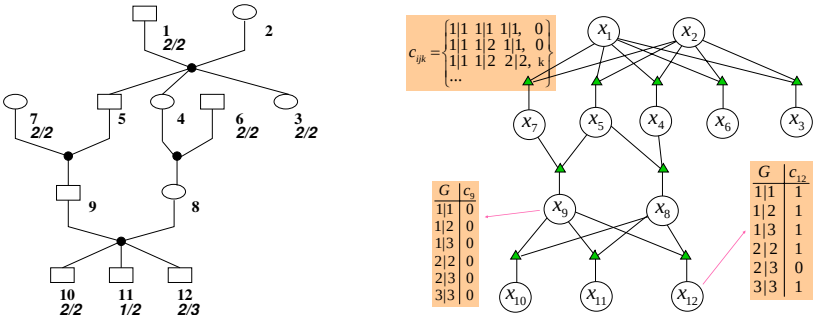


Fig. 1. Left: A pedigree taken from [14]. Right: its corresponding WCSP.

Example 1. An example is given in Fig. 1. There are $n = 12$ individuals and $m = 3$ alleles. A box corresponds to a male individual and an ellipse to a female. The arcs describe parental relations (individuals 1 and 2 have three children 3,4, and 5). Individuals 1,2,6, and 7 ($f = 4$) are founders. The possible genotypes are $G = \{1|1, 1|2, 1|3, 2|2, 2|3, 3|3\}$. 7 individuals (1,3,6,7,10,11, and 12) have an observed phenotype (a single genotype). The corresponding WCSP has 12 variables, 8 ternary constraints and 7 soft unary constraints. The minimum number of typing errors is one. An optimal solution is $\{(1,2|2), (2,1|2), (3,2|2), (4,1|2), (5,2|2), (6,2|2), (7,2|2), (8,1|2), (9,2|2), (10,2|2), (11,1|2), (12,2|2)\}$ such that the *erroneous* typing 2|3 of individual 12 has been changed to 2|2.

1.1 Error correction

When errors are detected, one would like to optimally correct them. The simple parsimony criterion is usually not sufficient to distinguish alternative values. More information needs to be taken into account. Being errors and Mendelian inheritance typically stochastic processes, a probabilistic model is attractive. A Bayesian network is a network of variables related by conditional probability tables (CPT) forming a directed acyclic graph. It allows to concisely describe a probability distribution on stochastic variables. To model errors, a usual approach is to distinguish the observation O and the truth T. A CPT $P(O|T)$ relates the two variables and models the probability of error.

Following this, we consider the following model for error correction: we first have a set of n variables T_i each representing the true (unknown) genotype of individual i. The domain is G. For every observed phenotype, an extra observed variable O_i is introduced. It is related to the corresponding true genotype by the CPT $P_i^e(O_i|T_i)$. In our case, we assume that there is a constant α probability of error: the probability of observing the true genotype is $1-\alpha$ and the remaining probability mass is equally distributed among remaining values.

For the individual i and its parents $pa(i)$, a CPT $P_i^m(T_i|pa(i))$ representing Mendelian inheritance connects T_i and its corresponding parent variables. Finally, prior probabilities $P^f(i)$ for each genotype must be given for every founder i. These probabilities are obtained by directly estimating the frequency of every allele in the genotyped population. The probability of a complete assignment $P(O,T)$ (all true and observed values) is then defined as the product of the three collections of probabilities (P^e, P^m and P^f). Note that equivalently, its log-probability is equal to the sum of the logarithms of all these probabilities.

The evidence given by the observed phenotypes $G(i)$ is taken into account by reducing the domains of the O_i variables to $G(i)$. One should then look for an assignment of the variables $T_i, i \in I'$ which has a maximum a posteriori probability (MAP). The MAP probability of such an assignment is defined as the sum of the probabilities of all complete assignments extending it and maximizing it defines an $\mathsf{NP}^{\mathsf{PP}}$-complete problem [15], for which there exists no exact methods that can tackle large problems. The PEDCHECK solver [13, 14] tries to solve this problem using the extra assumption of a unique already identified error. This is not applicable in large data-sets. Another very strong assumption (known as the Viterbi assumption) considers that the distribution is entirely concentrated in its maximum and reduces MAP to the so-called Maximum Probability Explanation problem (MPE) which simply aims at finding a complete assignment of maximum probability. Using logarithms this problem directly reduces to a WCSP problem where each CPT is transformed in an additive cost function. This allows to solve MPE using WCSP dedicated tools.

2 Soft constraint algorithms: extension to ternary constraints

In order to find an optimal solution and prove its optimality, a classical depth-first branch and bound algorithm is applied. An initial upper bound ($\top$) is given by the number of genotyping data plus one for the parsimony pedigree problem. For MPE, we

multiply for each individual the minimum probabilities different from zero of P^e, P^m and P^f (see Section 1.1) and take the opposite of its logarithm (to get additive positive costs). Each time a better solution is found, its cost becomes the new upper bound.

Inside branch and bound at each node of the search we enforce a local consistency. In this section, we present several local consistency properties, previously defined for binary constraints [7] and extended to the case of ternary constraints in order to deal with our pedigree problem. The technical difficulty resides in the simultaneous enforcement of two important soft local consistency properties (AC and DAC which are defined below) in polynomial time. We show how to enforce DAC in one direction of a ternary constraint without breaking AC in the two other directions. This allows to close an open question from [4] (Section 4) "whether a form of directional k-consistency can be established in polynomial time for k $>$ 2". The answer is yes for ternary constraints and we believe it can be generalized to any bounded constraint arity.

Two WCSPs defined over the same variables are said to be *equivalent* if they define the same cost distribution on complete assignments. Local consistency properties are widely used to transform problems into equivalent simpler ones. When enforcing a local consistency property at every node of the search, implicit costs can be deduced and so the search space can be hopefully reduced and variable values pruned earlier (a non trivial lower bound is given by $C_\varnothing$).

The simplest form of local consistency we used is node consistency (NC): $\forall x_i \in \mathcal{X}, (\exists a \in D_i / C_i(a) = \bot) \wedge (\forall a \in D_i / C_\varnothing \oplus C_i(a) < \top)$. Any WCSP can be easily transformed into an equivalent NC instance by projecting every unary constraint towards $C_\varnothing$ and subsequently pruning unfeasible values. We continue by extending the notion of soft (directional) arc consistency to ternary cost functions, for this, we extend the classic notion of support. Given a binary constraint C_{ij}, $b \in D_j$ is a *simple support* for $a \in D_i$ if $C_{ij}(a,b) = \bot$. Similarly, for directional arc consistency, $b \in D_j$ is a *full support* for $a \in D_i$ if $C_{ij}(a,b) \oplus C_j(b) = \bot$.

For a ternary cost function C_{ijk}, we say that the pair of values $(b \in D_j, c \in D_k)$ is a simple support for $a \in D_i$ if $C_{ijk}(a,b,c) = \bot$. Similarly, we say that the pair of values $(b \in D_j, c \in D_k)$ is a full support for $a \in D_i$ if $C_{ijk}(a,b,c) \oplus C_{ij}(a,b) \oplus C_{ik}(a,c) \oplus C_{jk}(b,c) \oplus C_j(b) \oplus C_k(c) = \bot$.

A WCSP is arc consistent (AC) if every variable is NC and every value of its domain has a simple support in every constraint. Given a static variable ordering, a WCSP is directional arc consistent (DAC) if every value of every variable x_i has a full support in every constraint C_{ij} such that $j > i$ and in every constraint C_{ijk} such that $j > i \wedge k > i$. A WCSP is full directional arc consistent (FDAC) if it is both AC and DAC [8].

The enforcing of simple and full supports for ternary constraints has to be carefully adapted from the previous existing algorithms for binary constraints. The idea is to extend unary and binary costs involved in the scope of ternary constraint C_{ijk} in such a way that a maximum projection is achievable on variable i without losing simple supports for variables j and k. The details of how this extension is done, proofs and implementation are given in the longer version [17].

The strongest form of local consistency we use is existential directional arc consistency (EDAC) [7]. A WCSP is existential arc consistent (EAC) if every variable x_i has at least one value $a \in D_i$ such that $C_i(a) = \bot$ and a has a full support in every

constraint. A WCSP is EDAC if it is both FDAC and EAC. EAC enforcement is done by finding at least one *fully supported* value per variable i.e. which is fully supported in all directions. If there is no such value for a given variable, then projecting all the constraints towards this variable will increase the lower bound, resulting in at least one *fully supported* value.

The complexity of ternary EDAC is time $O(ed^3 max\{nd, \top\})$ and space $O(ed^2)$, where n is the number of variables, d is the maximum domain size, e is the number of constraints and $\top$ is the maximum cost. The proof can be found in [17].

We maintain EDAC during search, producing a lower bound in $C_\varnothing$. The DAC variable ordering corresponds to the pedigree file order, which is usually a temporal order. If $C_\varnothing \geq \top$ then, the algorithm backtracks. We use dynamic variable and value ordering heuristics. We add a basic form of conflict back-jumping by always choosing the last variable in conflict (i.e. its assignment results in an empty domain or $C_\varnothing \geq \top$) [10]. The value ordering heuristic chooses first the fully supported value found by EAC. We use a binary branching scheme: the chosen variable is assigned to its fully supported value or this value is removed from its domain. Finally, we apply a limited form of variable elimination during the search as proposed in [9].

3 Experimental evaluation

In the experimental section we want to compare the accuracy of error detection for the different models introduced: MAP, MPE and Parsimony. The most complex MAP problem is a mixed optimization/integration problem that can be only solved by dedicated Bayes net solvers. Among them, we have chosen Samiam (version 2.2.1) because it is one of the most efficient and robust solver available according to the last BN solving competition. The MPE problem is a pure optimization problem which requires however to be able to deal with very large costs such as those produced by logarithms of probabilities (see Section 2). These problems can be addressed again by Samiam but also by `toulbar2` which has been extended to use very large integer costs. MPE can only be solved on small or mid-size instances. Finally, the simplest Parsimony problem can be directly tackled by `toulbar2`.We also used a version of toolbar called `toolbar/BTD` which integrates a specific tree-decomposition based branch and bound (version 2.2) [5] that should perform well on pedigree problems which have usually a tree-width much smaller than the number of variables. It also uses only binary EDAC and thus will show the interest of higher order consistencies. Parsimony problem can be solved on very large instances.

Because the pedigree analysis problem is not a new problem, one must also acknowledge the existence of different solvers for the real problem. However, none of these tools will be considered in the analysis because they either make very strong assumptions incompatible with the pedigree size considered (PedCheck [13] assumes that there is only one error), may be incomplete solvers (CheckFam [16] can prove inconsistency but produces only local corrections on nuclear families that may not always restore consistency while GenCheck [2] provide corrections that do not optimize parsimony of likelihood either) or have very limited efficiency compared to the solvers

considered here (GMCheck [18] tackles the MPE problem but is totally dominated by SamIam).

Two types of pedigree have been used to perform the evaluation: random pedigree and real pedigree. The random pedigree have been generated using a pedigree generator designed by geneticists at INRA. We then randomly erase the genotypes of some individuals with a given probability and introduce errors in some individuals with a given probability. The original correct genotypes are recorded in order to be able to evaluate the accuracy of error correction. We used a genotyping error probability $\alpha = 5\%$ (see Section 1.1). For random pedigree, all experiments have been performed on a 3 GHz Intel Xeon with 2 Gb of RAM.

Real instances are human genotyped pedigrees (genetic studies of eye, cancer and Parkinson diseases) as reported in [13, 14] and two groups (*berrichon* and *langlade*) are pedigree instances coming from sheep animals provided by the CTIG (*Centre de Traitement de l'Information Genetique*) in France. For real pedigree, all experiments have been performed on a 3 GHz Intel Xeon 64-bit with 16 Gb of RAM.

To compare the error prediction accuracy provided by the MAP, MPE and Parsimony, we had to limit ourselves to relatively small instances that could be solved to optimality by Samiam. The MPE problem has been solved using `toulbar2` and Samiam. Finally, Parsimony was solved using `toulbar2` only.

Two features were evaluated: the prediction of the individuals (denoted *ind*) containing an error in the pedigree and the prediction of the correct genotype (denoted by *geno*). The sensitivity of the prediction is the percentage of features that should be detected which are actually correctly predicted. Similarly, specificity is percentage of predicted features which are correct. Summarizing, MAP has 10% higher genotype specificity, meaning that is more robust in predicting the corrections of genotypes, as expected. However, it is too costly in terms of CPU time and can only deal with small instances. MPE gives very similar results while Parsimony is interesting for just restoring consistency. In our experiments `toulbar2` outperforms Samiam on the MPE problem. We further compared Parsimony and MPE on larger data sets using `toulbar2`. This is reported in Fig. 2. MPE has nearly a 10% better individual sensitivity and a 15% better genotype sensitivity and specificity on larger problems. We observe that the CPU-time needed to solve the instances is highly sensible to the treewidth for both MPE and Parsimony. For tree-widths above 50, `toulbar2` encountered some hard MPE instances it could not solve in the time limit [2].

Since our aim is to solve very large real size instances, we conclude the evaluation by comparing time efficiency of different solvers on the simplest Parsimony problem. Indeed, on the largest real instances defined by the sheep pedigree, MPE remained unsolvable in less than 10 hours. Despite its lower accuracy, Parsimony still provides the essential service of consistency restoration and this with minimum loss of data, a criterion that may look attractive in practice to biologists.

[2] Notice that the treewidth is anti-monotone in the number of individuals. This is because the increase in the treewidth is achieved (paramterizing the simulator) by increasing the number of males in a population. Increasing the number of males has the side effect of decreasing the number of individuals.

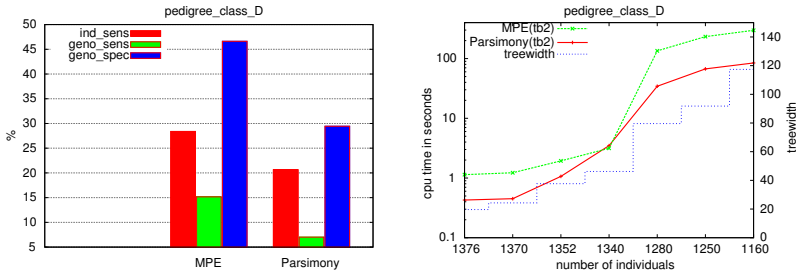


Fig. 2. Left: Histograms compare the sensitivities and specificities of MPE and Parsimony. Right: `toulbar2` CPU-time for both problems.

We tried several solvers and observed that as the problem size increases, `toulbar2` has the best performance. In fact only `toulbar2` is able to tackle real pedigree instances. For the parsimony problem, we were able to solve real pedigrees with up to 120,000 individuals in a few seconds. These problems were modeled as soft constraint networks with up to 9,500 variables and 17,000 constraints (after `toulbar2` preprocessing and variable elimination). All real pedigrees can be solved in a maximum of one minute of CPU time and have from 2 to 106 errors found. Solving such a large network is possible thanks to the powerful lower bounds provided by soft local consistencies, in particular EDAC extended to ternary constraints. As the size of instances increase EDAC is an order of magnitude better. The combination of the conflict heuristic and variable elimination has the best performance and corresponds to the combination used in all the previous experiments.

4 Conclusion

This paper deals with detecting Mendelian errors and providing an optimal correction. Compared to existing tools dedicated to this problem [13, 16, 2, 18], the novelty of our approach is to provide an optimal correction based on parsimony or maximum likelihood criterion for large pedigree data. This application lead us to the development of new algorithms, described in Section 2, for ternary constraints. We believe this result can be directly generalized to n-ary constraints, by considering all the intermediate arity levels. Using the parsimony model and the dedicated extension to ternary constraints we where able to solve large real pedigree instances of 9,500 variables that where unsolved up to now.

For MAP and MPE, we have shown on simulated data that MPE is a good approximation of MAP and is orders of magnitude faster to solve than MAP. However, on large real pedigrees, MPE could not be solved by `toulbar2`.

In the future, we will explore more complex probabilistic models in order to detect non Mendelian errors. It implies working on multi-locus models, where other interesting biological questions have been recently investigated by the AI community [11, 12].

References

1. ACETO, L., HANSEN, J. A., INGÓLFSDÓTTIR, A., JOHNSEN, J., AND KNUDSEN, J. The complexity of checking consistency of pedigree information and related problems. *Journal of Computer Science Technology 19*, 1 (2004), 42–59.
2. J. BENNEWITZ AND N. REINSCH AND E. KALM GENCHECK: A program for consistency checking and derivation of genotypes at co-dominant and dominant loci *Journal of Animal Breeding and Genetics* (2002), 119(5):350-360.
3. BISTARELLI, S., FARGIER, H., MONTANARI, U., ROSSI, F., SCHIEX, T., AND VERFAILLIE, G. Semiring-based CSPs and valued CSPs: Frameworks, properties and comparison. *Constraints 4* (1999), 199–240.
4. M. COOPER. High-order Consistency in Valued Constraint Satisfaction. *Constraints*, 10(3):283–305, 2005.
5. S. DE GIVRY AND T. SCHIEX AND G. VERFAILLIE Exploiting tree decomposition and soft local consistency in weighted CSP. In *Proc. of AAAI-06*, page 6p., Boston, MA, 2006.
6. DECHTER, R. *Constraint Processing*. Morgan Kaufmann Publishers, 2003.
7. LARROSA, J., DE GIVRY, S., HERAS, F., AND ZYTNICKI, M. Existential arc consistency: getting closer to full arc consistency in weighted CSPs. In *Proc. of the 19th IJCAI* (Aug. 2005).
8. LARROSA, J., AND SCHIEX, T. In the quest of the best form of local consistency for weighted CSP. In *Proc. of the 18th IJCAI* (Aug. 2003), pp. 239–244.
9. J. LARROSA, E. MORANCHO AND D. NISO On the practical applicability of Bucket Elimination: Still-life as a case study. *Journal of Artificial Intelligence Research* (2005), 23 :421-440.
10. C. LECOUTRE AND L. SAIS AND S. TABARY AND V. VIDAL Last Conflict based Reasoning. In *Proc. of ECAI-06* (2006), pp. 133–137, Trento, Italy.
11. I. LYNCE AND J.P. MARQUES SILVA Efficient Haplotype Inference with Boolean Satisfiability. In *Proc. of AAAI-06*, page 6p., Boston, MA, 2006.
12. R. MARINESCU AND R. DECHTER Memory Intensive Branch-and-Bound Search for Graphical Models. In *Proc. of AAAI-06*, page 6p., Boston, MA, 2006.
13. O'CONNELL AND WEEKS PedCheck: a program for identification of genotype incompatibilities in linkage analysis. *American Journal of Human Genetics 63*, 1 (1998), 259–66.
14. O'CONNELL AND WEEKS An optimal algorithm for automatic genotype elimination. *American Journal of Human Genetics 65*, 6 (1999), 1733–40.
15. PARK, J. D., AND DARWICHE, A. Complexity results and approximation strategies for map explanations. *Journal of Artificial Intelligence Research 21* (2004), 101–133.
16. SAITO, M. AND SAITO, A. AND KAMATANI, N. Web-based detection of genotype errors in pedigree data. *Journal of human genetics* (2002), 47(7):377-379.
17. SANCHEZ, DE GIVRY, SCHIEX Mendelian error detection in complex pedigrees using weighted constraint satisfaction techniques. *Constraints* Special issue on Bioinformatics, to appear.
18. THOMAS, A. GMCheck: Bayesian error checking for pedigree genotypes and phenotypes *Bioinformatics* (2005), 21(14):3187–3188.

Author Index