WCB'11

Workshop on Constraint Based Methods for Bioinformatics

September 12^{th}, 2011

Proceedings

Alessandro Dal Palù, Agostino Dovier, Andrea Formisano

Preface

The Workshop on Constraint-based methods for Bioinformatics has reached its seventh consecutive edition. This year's edition was held in the medieval town of Perugia, Italy on the 12^{th} of September as satellite workshop of the CP'11 conference. By looking at the corpus of contributions gathered in these years (more than fifty papers), we can draw some observations about trends and connections between the constraint area and the modern biology. In the *Omics* era, the availability of large amount of biological data is made possible by modern experimental techniques and recent advances in the biochemical research area. Non-trivial techniques are required in order to analyze, mine and combine this large piece of information. Constraint techniques showed to be a valid methodology to tackle the challenge presented by biology. The various papers presented throughout the years at WCB cover different topics that are challenging because of either the large amount of data involved (networks and genes) or the intrinsic complexity (structural problem). We can identify at least five main categories in which most of the contributions to WCB fall: sequence alignment (DNA classification, gene prediction, module identification, etc.), RNA secondary structure prediction, biological networks (pathways, metabolic and gene regulatory networks, analysis of biological systems, etc.), haplotype inference and related problems, and protein structure prediction (secondary structure prediction, tertiary structure prediction on-lattice and off-lattice, quaternary structure prediction). It is also interesting, from the Computer Science point of view, to note that different flavors of constraint-related techniques have been exploited during the recent years. For instance, we mention the use of different popular constraint solvers (such as CLP(FD), Gecode, Comet), as well as the proposals relying on other general logic programming frameworks (such as Answer Set Programming, SAT solvers and Integer Linear Programming solvers, Inductive Logic Programming). At the same time, several search heuristics have been proposed and adopted within approaches based on local search and local neighboring search. From the point of view of programming languages, various applications of Prolog have been employed: BProlog for PRISM applied to Hidden Markov Models, Prolog for Biochamm and Stochastics Concurrent Constraint Programming to model networks evolution. Moreover, some dedicated constraint solvers have been designed and implemented for specialized constraint handling (Chemera and COLA are significant examples).

Concerning this edition of the workshop, seven papers are collected in WCB'11 Proceedings (and listed in random order).

The first paper, *Declarative Merging of and Reasoning about Decision Diagrams*, by T. Eiter, T. Krennwallner, and C. Redl, proposes a modular framework to merge decision diagrams in a transparent and declarative way. An application of the proposed approach, that is based on Answer Set Programming, to the problem of DNA classification is addressed by the authors.

In the paper *Constraints and Global Optimization for Gene Prediction Overlap Resolution*, C. Theil Have shows how to apply constraints (in particular, Constraint Handling Rules are considered) and global optimization to the problem of restricting overlapping of gene predictions. The specific case of prokaryotic genomes is dealt with.

The third paper, *Introducing FIASCO: Fragment-based Interactive Assembly for protein Structure prediction with COnstraints* illustrating the work of a numerous research team, outlines a declarative constraint-based framework designed to support chemists, biologists, and computer scientists in their analysis of protein conformations.

The paper *Improving Multiple Sequence Alignments with Constraint Programming and Local Search* by M. Correia, F. Madeira, P. Barahona, and L. Krippahl, proposes a method for a repairing MSA by exploiting constraint programming and local search techniques.

Petri nets are exploited in the paper by A. Palinkas and A. Bockmayr, titled *Petri Nets for Integrated Models of Metabolic and Gene Regulatory Networks*, as a basis for a systematic method that supports the integrated modeling of metabolic and gene regulation networks.

The paper *A Constraint Program For Subgraph Epimorphisms with Application to Identifying Model Reductions in Systems Biology*, by S. Gay, F. Fages, T. Martinez, and S. Soliman, shows how a constraint-based approach to the subgraph epimorphism problem can be used in the biological context to identify meaningful relationships between biochemical reaction graphs.

Finally, the paper *A New Local Move Operator for Reconstructing Gene Regulatory Networks*, by J. Vandel and S. De Givry, deals with the problem of learning the structure of a Bayesian network. The application to the case of regulatory networks is addressed.

In conclusion, we would like to thank all colleagues that accepted to serve as members of the Program Committee or as external reviewers and that dedicated their precious time in the reviewing phase. We also would like to thank the Dipartimento di Matematica e Informatica in Perugia for hosting the event and the Committees of the CP conference. In particular, we mention Christian Schulte, the workshops and tutorials chair of CP'11, as well as the conference chair Stefano Bistarelli and the local organization team in Perugia. Finally, we express our gratitude to all the authors that submitted their papers to WCB'11 and to the invited speaker Alessandro Brozzi.

Perugia, September 12, 2011

Alessandro Dal Palù
Agostino Dovier
Andrea Formisano

Organization

WCB'11 has been organized by Alessandro Dal Palù, Agostino Dovier, and Andrea Formisano and was hosted by the Dipartimento di Matematica e Informatica of the Università di Perugia.

Program Committee

Rolf Backofen, Freiburg Universität
Pedro Barahona, Universidade Nova de Lisboa
Alexander Bockmayr, Freie Universität Berlin
Mats Carlsson, SICS, Uppsala
Alessandro Dal Palù, Università di Parma
Simon de Givry, INRA, Toulouse
Agostino Dovier, Università di Udine
Esra Erdem, Sabanci University
François Fages, INRIA Rocquencourt
Andrea Formisano, Università di Perugia
Inês Lynce, INESC-ID Lisboa
Neil Moore, University of St. Andrews
Enrico Pontelli, New Mexico State University
Sven Thiele, Potsdam Universität
Sebastian Will, Freiburg Universität

Additional Referee

David Fournier

Sponsoring institutions

A. Dal Palù, A. Dovier, and A. Formisano are partially supported by the grants GNCS-INdAM *Tecniche innovative per la programmazione con vincoli in applicazioni strategiche*, GNCS-INdAM *Nuova architettura parallela per la Programmazione Logica mediante GPGPU*, and by MIUR-PRIN'08 project *Innovative and multi-disciplinary approaches for constraint and preference reasoning.*

Table of Contents

Preface

Inferring Gene Pathways Controlling Clonal Outgrowth by High-throughput Insertional Mutagenesis Screens*

INVITED TALK

Alessandro Brozzi

Dipartimento di Oncologia Sperimentale
Istituto Europeo di Oncologia (IEO), Milano

Abstract. A pathway is a group of genes which act together in a coordinated manner to control a cellular phenotype. The characterization of each set of genes forming a pathway and their organization inside of it represents an hard task in biology.

Clonal outgrowth is a cellular phenotype resulting from a combination of acquired mutations affecting cellular genes.

The insertion of proviruses in the genome of the host cell can mutate cellular genes and it might confer a selective advantage to the host cell resulting in clonal outgrowth. When the proviruses are experimentally injected into cells it takes the name of insertional mutagenesis. By this genetic approach we wanted to discover genome-wide the set of genes involved in the pathway controlling clonal outgrowth.

We conducted insertional mutagenesis on mouse model of PML-RARalpha induced leukemia. Examining 48 leukemias we found a set of nearly 200 genes affected by the proviruses linked to clonal outgrowth.

To infer the organization of these genes inside a pathway we took advantage by Mutual Information (MI) reverse-engineering approach [1] to quantify the extent to which the expression profiles of two genes are related to each other across a dataset of 20000 gene expression profiles.

Out of the 200 genes, we discovered large and interconnected co-expressed modules consisting of genes strongly connected to each other. These co-expressed modules represent genes putatively sharing the same features and occupying the same position inside the pathway organization.

The co-expressed modules constitute the basis to detail the fine organization and each biological role of the pathway components.

We discuss the computational challenges and the future perspectives about insertional mutagenesis teqnique as an highly informative genetic approach to infer gene pathway linked to cellular proliferation.

Keywords: Gene pathways, inference, proliferation, high-throughput.

* Joint work with Chiara Ronchini, Dipartimento di Oncologia Sperimentale, Istituto Europeo di Oncologia (IEO), Milano.

References

[1] V. Belcastro, V. Siciliano, F. Gregoretti, P. Mithbaokar, G. Dharmalingam, S. Berlingieri, F. Iorio, G. Oliva, R. Polishchuck, N. Brunetti-Pierri, and D. Di Bernardo. Transcriptional gene network inference from a massive dataset elucidates transcriptome organization and gene function. *Nucleic acids research*, DOI:10.1093/nar/gkr593, 2011.

Declarative Merging of and Reasoning about Decision Diagrams*

Thomas Eiter, Thomas Krennwallner, and Christoph Redl

Institut für Informationssysteme, Technische Universität Wien
Favoritenstraße 9-11, A-1040 Vienna, Austria
{eiter,tkren,redl}@kr.tuwien.ac.at

Abstract. Decision diagrams (DDs) are a popular means for decision making, e.g., in clinical guidelines. Some applications require to integrate *multiple* related yet different diagrams into a single one, for which algorithms have been developed. However, existing merging tools are monolithic, application-tailored programs with no clear interface to the actual merging procedures, which makes their reuse hard if not impossible. We present a general, declarative framework for merging and manipulating decision diagram tasks based on a belief set merging framework. Its modular architecture hides details of the merging algorithm and supports pre- and user-defined merging operators, which can be flexibly arranged in merging plans to express complex merging tasks. Changing and restructuring merging tasks becomes easy, and relieves the user from (repetitive) manual integration to focus on experimenting with different merging strategies, which is vital for applications, as discussed for an example from DNA classification. Our framework supports also reasoning over DDs using answer set programming (ASP), which allows to drive the merging process and select results based on the application needs.

1 Introduction

Many medical decisions are based on Decision Diagrams (DDs), which support medical doctors and health care personnel in decision making like to determine the best medication or intervention for a patient depending on her medical history and examinations. Such diagrams are also used for diagnosis as part of computer supported decision systems, cf. [6]. A very common use case frequently found in clinical protocols is to quantify the degree of severity depending on the patient's condition, see eg. [14]; the suggested treatment depends then on the stage. Clearly, DDs are relevant beyond clinical practice and have become popular in economy (e.g. in liquidity rating), psychology (e.g. in tests for personality disorders), and basic life sciences; for example, [12]—which we will consider in more detail below—uses decision trees to classify given DNA sequences into protein coding and non-coding ones. More applications are listed in [2].

Multiple DDs often exist for the same issue, due to various reasons: different institutes working on similar projects, different views of correct decisions, statistical impreciseness, or simply human errors. Making a choice between several DDs, especially from authoritative sources, is notoriously difficult and ignoring expertise captured

* This research has been supported by the Austrian Science Fund (FWF) project P20840 and P20841, and by the Vienna Science and Technology Fund (WWTF) project ICT 08-020.

in such diagrams is a waste of resources. This requires to integrate multiple diagrams into a single one that should be concise and coherent. Several integration algorithms have been developed [7, 10], and implemented tools exist [13]. However, they are monolithic programs tailored for a particular application, without clear interface between the integration components and other programs parts; reusing the merging procedures is hard if not impossible. Moreover, existing approaches usually produce *one* output diagram, but merging can often be done in various ways, hence it is interesting to develop merging strategies that produce *multiple* diagrams. A diagram can then be selected according to the application needs. This naturally calls for *reasoning* about DDs.

In this paper we present a general, declarative approach which supports semi-automatic integration of multiple DDs into a single one, as well as reasoning about DDs. Both features can also be combined to include user-defined constraints or—more generally—rules that influence the further integration process. To this end, we encode decision diagrams to *belief sets* and transform the *integration of DDs into a belief set merging problem in the generic framework* of [11], which provides merging plans of abstract merging operators to accomplish complex belief merging tasks. *With the MELD system*, which implements the framework via answer set programming (ASP) [3], *we can exploit a rich infrastructure to realize a powerful declarative tool*. It facilitates a range of different DD integration algorithms, allows to formulate complex, operator-based integration tasks in a modular, flexible manner, and offers on top the possibility to use ASP for reasoning about DDs. Specifically, this can be exploited to compute properties of diagrams (like height or number of variables) that are used for filtering results.

We proceed as follows. After fixing a formal model of decision diagrams, we map DDs to belief sets and integration of DDs into a belief set merging in Sec. 3. Reasoning over DDs and support for it in our tool, the *DDM system*, is discussed in Sec. 4. Finally, we consider application for *DNA classification* similar as in [12], with the aim to stress flexibility and user friendliness, and capabilities beyond those of other systems (Sec. 6).

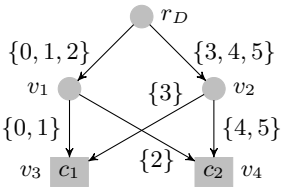
2 Decision Diagrams

We define a *classification function* $c\colon \mathcal{D} \to \mathcal{C}$ as a mapping of some domain $\mathcal{D}$ to a set of class labels $\mathcal{C}$. To represent such a function one can use, e.g., lookup tables, production rules, or decision diagrams. We focus on the latter as they have turned out to be very useful in practice not only because they are comprehensible and easy to explain.

Abstractly, we can define a decision diagram as follows.

Definition 1. *A* decision diagram (DD) *over $\mathcal{D}$ and $\mathcal{C}$ is a labelled rooted directed acyclic graph $D = \langle V, E, \ell_{\mathcal{C}}, \ell_E \rangle$, where V is the set of nodes with unique root node $r_D \in V$ and $E \subseteq V \times V$ is a nonempty set of directed edges. The labelling function $\ell_{\mathcal{C}}$ maps leaf nodes in D to elements from $\mathcal{C}$, and $\ell_E\colon E \to 2^{\mathcal{D}}$ assigns each edge a subset of domain $\mathcal{D}$. We call D a* decision tree *if every node has at most one incoming edge.*

Classifying an element $d \in \mathcal{D}$ is intuitively done by starting at node r_D and following an outgoing edge e iff $d \in \ell_E(e)$. This step is repeated until a leaf node v is reached. Then $\ell_{\mathcal{C}}(v)$ is the class label assigned to d. To guarantee that the classification by some DD D is deterministic and the result is unique, ℓ_E is required to satisfy the following two conditions. In this case we say that D is *valid*:



(a) Graphical representation

$root(r_D)$; $inner(r_D)$; $inner(v_1)$; $inner(v_2)$;
$leaf(v_3, c_1)$; $leaf(v_4, c_2)$;
$cond(r_D, v_1, z, <, 3)$; $else(r_D, v_2)$;
$cond(v_1, v_3, z, <, 2)$; $else(v_1, v_4)$;
$cond(v_2, v_3, z, <, 4)$; $else(v_2, v_4)$

(b) Formal encoding

Fig. 1: A valid decision diagram D

(a) for non-leaf nodes v of D, $\bigcup_{(v,w)\in E} \ell_E(v,w) = In(v)$, where $In(r_D) = \mathcal{D}$ and $In(u) = \bigcup_{(p,u)\in E} \ell_E(p,u)$ for $u \neq r_D$, and
(b) for a node v of D and any successors u, w of v, $\ell_E(v,u) \cap \ell_E(v,w) \neq \emptyset \Rightarrow u = w$.

Condition (a) states that, if a node is reached for element d, there must be an outgoing edge for d, i.e., computation can always continue, while (b) forces branching at internal nodes to be deterministic. In the following we consider valid decision diagrams only.

Example 1. Fig. 1a shows a valid decision diagram D over $\mathcal{D} = \{0,1,2,3,4,5\}$ and $\mathcal{C} = \{c_1, c_2\}$. The edges are marked with ℓ_E, and its leaves with $\ell_\mathcal{C}$. It represents the classification function c s.t. $c(d) = c_1$ if $d \in \{0,1,3\}$ and $c(d) = c_2$ if $d \in \{2,4,5\}$.

For practical purposes it is convenient to realize edge labels by *queries* over some query language. A query $Q(z)$ with free variable z is then a shortcut for all domain elements which satisfy it. E.g., if V is the set of positive integers and we want to distinguish between prime and non-prime numbers, we may label the according edge with $Q(z) = z > 1 \wedge \forall m, n\,(m > 1 \wedge n > 1 \supset z \neq m \cdot n)$ instead of $\{2,3,5,7,11,\dots\}$. In practice, simple queries of form $X \circ Y$ often suffice, where X and Y are constants or attribute values of the domain, and $\circ$ is a comparison operator. For example, if the domain consists of patient records with attributes such as blood values, the query $Q(z) = z.TSH > 4.5mU/l$ is true for all patients z that have a Thyroid-Stimulating Hormone level larger than 4.5 milli-units per liter. More complex queries involving logical connectives can easily be rewritten to this form by introducing intermediate nodes.

For the further development of our framework we assume that the query language of a DD is fixed. When developing our encoding, we will assume that queries are of the form $X \circ Y$, but it can easily be extended to more complex query languages.

3 Merging of Decision Diagrams

Some applications require to work with multiple DDs. The reasons for this are manyfold: statistical fluctuations, different expert opinions on correct decisions, or simply human errors. For an example merging, see Fig. 4, which is discussed in Section 6. We will see in Sec. 4 that merging operators (as introduced here) may produce multiple output diagrams. Picking one of them over another is not easy to justify, and sometimes it is shearly impossible to have any preference among a variety of diagrams.

Belief Merging. Redl *et al.* [11] developed the ASP-based MELD system for belief set merging tasks.[1] MELD can integrate multiple collections of belief sets using merging operators that are hierarchically arranged in trees, called *merging plans*. They are evaluated bottom-up, and the result is the one at the root.

More in detail, a *belief* $(\neg)p(c_1, \ldots, c_n)$ is a literal (atom or negated atom) where p is a predicate symbol of arity n from a set of predicate symbols Σ_P, and the c_i are constants from a set Σ_C of constant symbols. A *belief set* is any set B of beliefs (wrt. Σ). A *collection of belief sets* $\mathcal{B}$ is any set of belief sets (wrt. Σ); $\mathbb{B}_\Sigma$ denotes the set of all collections of belief sets. A *belief set merging operator* is a function $Op\colon (\mathbb{B}_\Sigma)^k \times \mathcal{A}_1 \times \cdots \times \mathcal{A}_m \to \mathbb{B}_\Sigma$ that assigns each tuple $\hat{\mathcal{B}} = (\mathcal{B}_1, \ldots, \mathcal{B}_k)$ of collections of belief sets $\mathcal{B}_i$ and arguments $A_1, \ldots, A_m$ from domains $\mathcal{A}_1, \ldots, \mathcal{A}_m$ a result collection of belief sets $Op(\hat{\mathcal{B}}, A_1, \ldots, A_m)$; we allow $k = 1$ (in abuse of terminology) to enable also transformations of collections of belief sets. A *merging plan* is any expression built using the operators over *belief bases*, which comprise facts and (optionally) logical rules; each belief base has an associated collection of belief set (its semantics), used for evaluation.

Example 2. We define operator $\circ_\cup^{2,0}$ for consistently integrating two collections $\mathcal{B}_1$ and $\mathcal{B}_2$ of belief sets

$$\circ_\cup^{2,0}(\mathcal{B}_1, \mathcal{B}_2) = \{B_1 \cup B_2 \mid B_1 \in \mathcal{B}_1, B_2 \in \mathcal{B}_2, \nexists A \text{ s.t. } \{A, \neg A\} \subseteq (B_1 \cup B_2)\} \quad .$$

The operator computes the pairwise union of two belief sets B_1 and B_2 from $\mathcal{B}_1$ and $\mathcal{B}_2$, respectively, where classically inconsistent pairs are skipped. Assume $\mathcal{B}_1 = \{\{a, b, c\}, \{\neg a, c\}\}$ and $\mathcal{B}_2 = \{\{\neg a, d\}, \{c, d\}\}$, we get that $\circ_\cup^{2,0}(\mathcal{B}_1, \mathcal{B}_2) = \{\{a, b, c, d\}, \{\neg a, c, d\}\}$. Let $\mathcal{B}_3 = \{\{\neg d, e\}, \{d, e\}\}$, then $\circ_\cup^{2,0}(\circ_\cup^{2,0}(\mathcal{B}_1, \mathcal{B}_2), \mathcal{B}_3)$ is a merging plan which evaluates to $\{\{a, b, c, d, e\}, \{\neg a, c, d, e\}\}$.

We instantiate the framework for decision diagram merging. We obtain an implementation, based on MELD, called the *DDM system*.[1] The basic idea is to

- *encode DDs as belief sets*, described by *belief bases*;
- define *merging operators* for MELD (implemented in C++), tailored to the integration and manipulation of encoded diagrams; and based on them
- declare *merging plans* to integrate and manipulate (convert, optimize, etc.) the encoded DDs.

Encoding. Let $D = \langle V, E, \ell_\mathcal{C}, \ell_E \rangle$ be a decision diagram over domain $\mathcal{D}$ and class labels $\mathcal{C}$. We assume that $\mathcal{D}$ is a set of tuples $(a_1, \ldots, a_n) \in \mathcal{D}_1 \times \cdots \times \mathcal{D}_n$, where $\mathcal{D}_i$, $1 \leq i \leq n$, is the domain of attribute a_i. Informally, $\mathcal{D}$ consists of composed objects, which are described by n attribute values. Then we use the following atoms to encode D:

- $root(r_D)$ for the root node r_D of D;
- $inner(v)$ for inner nodes $v \in V$;
- $leaf(v, c)$ for leaf nodes $v \in V$ with assigned class label $c = \ell_\mathcal{C}(v)$;
- $cond(v, w, a, Op, b)$ for an edge $(v, w) \in E$ with some condition $Q(\mathbf{z}) = a\ Op\ b$ such that $Q(\mathbf{z})$ holds iff $\mathbf{z} \in \ell_E((v, w))$, where a and b are constants or named attributes of $\mathbf{z}$ and Op is a comparison operator;

[1] `http://www.kr.tuwien.ac.at/research/systems/dlvhex/meld.html`

– $else(v, w)$ for an ***else-edge*** $(v, w) \in E$. It is optional but unique for v and encodes the set of domain elements $\mathcal{D} \setminus \bigcup_{(v,u)\in E, u\neq w} \ell_E((v, u))$.

Formally, the query language has expressions $a\ Op\ b$ and $else$ for optional else-edges, where $else(v, w)$ is viewed as the conjunction of the negated expressions on all other out-edges of v. Thus, a tuple $t \in \mathcal{D}$ belongs to $\ell_E((v, w))$ iff no condition of some other out-edge of v is true for t. An example encoding that corresponds to the diagram in Fig. 1a is shown in Fig. 1b. This basic encoding can be easily extended to provide more features and support enriched decision diagrams. For instance, we could allow leaf nodes to store additional information besides the class label (we will use this below).

Decision Diagram Merging using Merging Plans. Let $\urcorner_{\mathcal{D},\mathcal{C}}$ denote the set of all decision diagrams over domain $\mathcal{D}$ and classes $\mathcal{C}$. We then define (recall that 2^X is the powerset of a set X):

Definition 2. *An n-ary DD merging operator is a function*

$$\circ^n \colon (2^{\urcorner_{\mathcal{D},\mathcal{C}}})^n \times \mathcal{A}_1 \times \cdots \times \mathcal{A}_m \to 2^{\urcorner_{\mathcal{D},\mathcal{C}}}$$

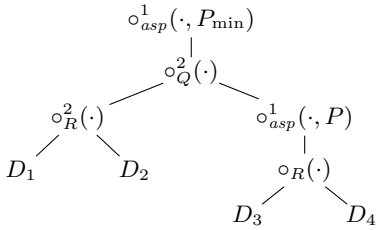
which maps each tuple $\Delta = (\Delta_1, \ldots, \Delta_n)$ of sets of DDs Δ_i (over $\mathcal{D}$ and $\mathcal{C}$) to a set of DDs $\circ^n(\Delta, A_1, \ldots, A_m)$, where $A_i \in \mathcal{A}_i$ are additional arguments from domains $\mathcal{A}_i$ for all $1 \leq i \leq m$.

In our examples, the domains of additional arguments will usually be either the natural numbers or the set of all ASP programs. Def. 2 allows for arranging operators hierarchically in so-called *merging plans* (for an example see Fig. 2a). The merging plan can be evaluated bottom-up, and the final result is the output of the topmost operator. Concrete operators $\circ^n(\Delta, A_1, \ldots, A_m)$ are given e.g. in [7, 10] and in Sec. 6. Allowing operators of arity $n = 1$ enables decision diagram transformations. It is often convenient to transform diagrams into a special form (e.g. trees) prior to integration; this may simplify the implementation of the actual merging operators ($n \geq 2$) enormously.

Each such operator produces an output decision diagram which behaves as if the input classifiers were consulted independently and the results were combined as described below for some predefined operators:

– *majority voting*: the majority of the input diagrams $D_1, \ldots, D_n$ decides;
– *user preference*: wrong decisions may be of different severity. In medical screening tests, e.g., one usually prefers false positives to false negatives: additional tests may refute the former, while the latter let the disease proceed. Thus a natural decision rule could be: "If the input classifiers vote differently, classify as X rather than Y";
– *average*: interpreting decision diagrams as decision boundaries in an n-dimensional feature space, it is natural to compute the (possibly weighted) "average boundary."

Technically, merging plans are declaratively specified in a user-friendly language and may be automatically evaluated by our prototype implementation. The set of predefined operators can be extended by custom ones by implementing an operator-API in C++. For more technical details we refer to the online documentation.



(a) Decision Diagram Merging Plan

$$
\begin{aligned}
cnt(I,C) \leftarrow\ & LC = \#count\{L : leaf_{in}(I,L,C)\},\\
& IC = \#count\{N : inner_{in}(I,N)\},\\
& root_{in}(I,R), C = LC + IC\\
c(I) \leftarrow\ & root_{in}(I,R), \text{not } \neg c(I)\\
\neg c(I) \vee \neg c(J) \leftarrow\ & root_{in}(I,R), root_{in}(J,S), I \neq J\\
leaf(L,C) \leftarrow\ & c(I), leaf_{in}(I,L,C)\\
& \vdots\\
else(N_1,N_2) \leftarrow\ & c(I), else_{in}(I,N_1,N_2)\\
\bot \leftarrow\ & M = \#min\{NC : cnt(I,NC)\},\\
& c(I), cnt(I,C), C > M
\end{aligned}
$$

(b) $P_{\min}$

Fig. 2: Node count minimization

4 Reasoning about Decision Diagrams

The second major benefit of our formal representation is the *possibility of reasoning about DDs*. Properties of a diagram (e.g., height, variable occurrences, redundancy, etc.) can be computed automatically from an encoding as in Sec. 3. This is particularly interesting when merging and reasoning operators are combined in merging plans. According to Def. 2, merging operators output *sets of* DDs. Hence, when a merging operator encounters a choice point, it may simply produce alternative diagrams. Such choice points can e.g. be leaves with (almost) uniform distributions, i.e., the best classification is not obvious. One can then select the most appropriate diagram by reasoning over the alternatives, resorting e.g. to properties as above. For example, take the redundancy measure defined as the number of indistinguishable nodes. Computationally optimal representations require an appropriate choice for the diagram with minimum redundancy. Another possibility is to prefer DDs with minimum height or minumum number of nodes. This is reasonable if the decision diagram is intended for being used by humans such as medical doctors applying classifiers published in medical guidelines.

To reason about DDs, answer set programs are well-suited for several reasons: (1) transitive closures allow to reason over paths in diagrams, (2) the multi-model semantics allows for producing multiple diagrams (one per answer set), and (3) constraints are useful to rule out inappropriate diagrams, or to account for a cost. Technically, we realize reasoning over diagrams (i.e. "applying" programs to diagrams) by instantiating Def. 2 as a special operator $\circ_{asp}(\Delta, P)$ which can be used as any other operator in the merging plan. The input is a set $\Delta \in 2^{\urcorner_{\mathcal{D},\mathcal{C}}}$ of DDs and an answer set program P. The operator $\circ_{asp}$ encodes all input diagrams in Δ, adds them as facts to the user-defined program P, and returns as result its answer sets. They are expected to contain the encoded output diagrams (one per answer set). ASP is well-suited for this purpose because of its multi model semantics which allows for producing multiple alternative results.

We slightly modify our encoding from Sec. 3: to handle *multiple* diagrams within one set of input facts, we add a diagram index I as first argument to all predicates $p \in \{root, leaf, inner, cond, else\}$ and call them p_{in}; to distinguish between program input diagrams and result diagrams, p_{in} are used for the input, and p denote output predicates.

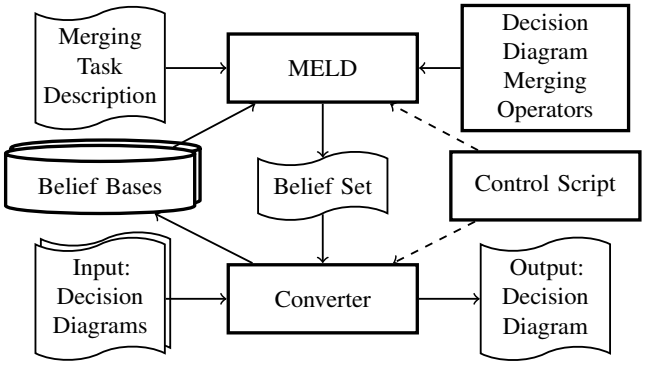


Fig. 3: DDM Architecture (data flow $\rightarrow$, control flow $\dashrightarrow$)

Use Case: Node Count Minimization. The merging plan in Fig. 2a shows four input DDs $D_1, \ldots, D_4$. First, we merge D_1 and D_2, as well as D_3 and D_4 using operator $\circ_R^2$ whose result is subsequently fed into a user-defined program P (potentially, any program can be used that is using the encoding as described above). The result of this and the merge of D_1, D_2 is passed to the next binary merging operator $\circ_Q^2$, which is eventually filtered by $P_{\min}$ as shown in Fig. 2b. Program $P_{\min}$ is intended to select among arbitrarily many input DDs the one with the minimal number of nodes. Let $V(D)$ denote the set of nodes in diagram D. We have the following result.

Proposition 1. *For a set Δ of input decision diagrams, we get that $\circ_{asp}(\Delta, P_{\min})$ yields a set of decision diagrams $\Delta_{\min} \subseteq \Delta$ such that for every $D \in \Delta_{\min}$ there is no $D' \in \Delta$ with $|V(D)| > |V(D')|$.*

Intuitively, $P_{\min}$ filters diagrams with minimal node number. It computes in cnt for each diagram (identified by its root) the total number of nodes (rule 1). Then it selects non-deterministically exactly one input diagram at a time (rule 2–3) and copies it to the answer set (rules 4–); to minimize the node count, answer sets representing non-minimal diagrams are eliminated in the last integrity constraint. In practice, the selection criteria might be more involved. In Sec. 6 we will (abstractly) propose a program P_{sel} which tests the input diagrams over some test set and selects the diagram with the best behavior.

5 Prototype Implementation of the DDM System

The architecture of our prototype (see Fig. 3) consists of the following parts.[2]

MELD. This is the underlying belief set merging system. It is implemented on top of the logic programming reasoner DLVHEX, which evaluates HEX programs; for details we refer to Redl et al. ([11]). Our DDM system extends MELD by two major components: a *converter* between different forms of decision diagram representation, and a *suite of*

[2] `http://www.kr.tuwien.ac.at/research/systems/dlvhex/ddm.html`

decision diagram merging operators, which are plugins for the MELD system. Further components are the Merging Task Description and the Control Script.

Converter. MELD expects decision diagrams in the belief set encoding from above; the Converter transforms human-readable input and output formats of machine learning tools (which realize hierarchical structures) into corresponding belief bases (sets of facts in HEX format). Our implementation, graphconverter, currently supports (1) a graph-based input format, (2) the output format of the machine learning tool RapidMiner (`http://rapid-i.com`), (3) a representation as logic program or as answer set. graphconverter takes two arguments that specify the input resp. output format; it reads from standard input and writes to standard output.

Merging Operators. At the core of our DDM system is a suite of merging operators which interpret their input as encoded decision diagrams. We provide some predefined operators, which are mostly considered to serve as examples for demonstrating the possibilities. Users may use them directly, refine them to make them suitable for a certain application, or develop completely new operators as plugins (in C++).

Merging Task Description. The merging plan, the decision diagrams and the merging operators used, is stored as a *merging task description* in an `.mt` file, say `task.mt`, formulated in MELD's *merging task language (MTL)*. The merging operators are tailored to decision diagrams only, i.e., they assume that belief sets associated with belief bases encode decision diagrams (otherwise an error is raised). The merging task can be initiated by invoking the command

```
$ dlvhex --merging task.mt > res.as
```

storing the output in file `res.as`. (Note that `dlvhex --merging` invokes MELD.) The result is another diagram represented by the facts in the belief set.

Control Script. A simple control script, as used in the examples of the system, manages the workflow of executing a merging task. It converts the input diagrams, stored in files `diagN.X` (the filename extension `X` tells the input diagram type) to belief bases in files `diagN.hex`. It then calls MELD as above, and finally converts the obtained diagram (represented by a set of facts given by `res.as`) into the input format `X`; e.g., for `dot` files it calls

```
$ graphconverter as dot < res.as > res.dot
```

Further details on system usage input and MTL format description is given at the accompanying homepage.[2]

6 Example: DNA Classification

A central task in (semi-)automatic generation of protein databases is to recognize genes in DNA sequences. Recall that DNA molecules are composed of the four bases (A)denine, (G)uanine, (C)ytosine and (T)hymine which are lined up in vast strings. In reproduction, only a minor part of the total DNA will be transcribed as most of it is *junk DNA* not encoding proteins. To construct protein databases like SWISSPROT (`http://expasy.`

`org/sprot/`) requires to automatically classify sequenced DNA into (protein) *coding* and *non-coding* (junk DNA) samples.

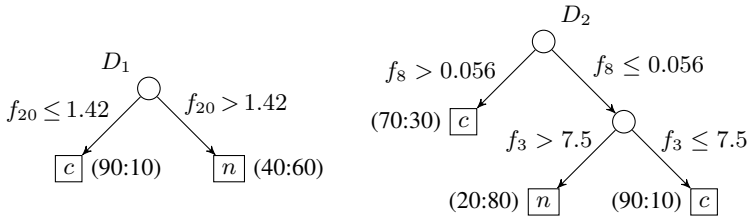
To this end, one usually computes *numeric features* for a set of annotated training sequences. They incorporate knowledge from molecular biology which allows to discriminate—with some level of uncertainty—between the two classes. For instance, it is known that the predominant bases at the first codon position in coding sequences are purines (A and G), while in non-coding sequences the distribution is rather random. Hence, a useful feature is the relative frequency of A and G on the first codon position with respect to the number of codons in a sequence. For details we refer to [8]. Feature vectors can then be used to train a classifier using machine learning techniques.

We instantiate Def. 1 with $\mathcal{C} = \{c, n\}$ (coding and non-coding) and use the query language in Sec. 3 for the edge labelling. We extend our basic DD model by an additional frequency distribution $\ell_F(v) = (c_v, n_v)$ at each node v, which tells the number of coding (c_v) and non-coding (n_v) samples (in fact, below we will use it only at leaf nodes). E.g., if in leaf l we have that 70 out of 100 training samples were coding, the classification is $\ell_{\mathcal{C}}(l) = c$ with frequency distribution $\ell_F(l) = (70, 30)$.

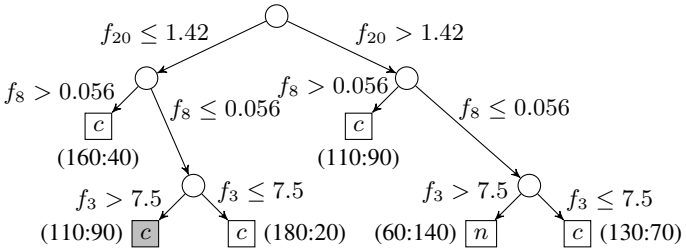
MORGAN Merging Operator. The MORGAN system [13] trains *multiple* decision trees $D_1, \ldots, D_n$ and merges them afterwards (see Fig. 4). The class of a yet unseen sample s is then determined as follows. First s is classified by each tree D_i, ending in leaf l_i. Then all frequency distributions $\ell_F(l_i)$ are summed up by component-wise vector addition, and the class with the largest count is the final classification. For instance, if we have two classifiers which yield the distributions $\ell_F(l_1) = (90, 10)$ and $\ell_F(l_2) = (20, 80)$, they are added to $(110, 90)$, and consequently the final classification is c (since $110 > 90$). The implementation of the binary case is straightforward as follows: Each leaf node of diagram D_1 is replaced by a copy of the diagram D_2. Then the frequency distribution of each new leaf node is recomputed, and the class label is set according to the highest component. More than two diagrams are integrated by iteration.

We implemented MORGAN's merging strategy as operator $\circ_M$ for our DDM system. It takes as input two singleton collections of belief sets (general diagrams must be converted to trees, for which our system provides operators). The output will be another classifier behaving as the suggested procedure, but represented by a new decision tree. Fig. 4c shows the diagram after application of another operator $\circ_{simp}$ from our system, which simplifies the input diagram by eliminating unnecesary branches and reusing equivalent subdiagrams.

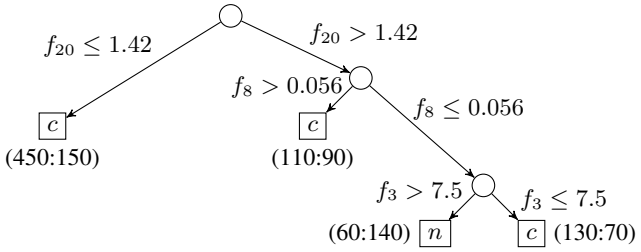
Experiments with Decision Diagram Merging. Concerning the accuracy increase, our most impressive results were achieved with three different decision trees, trained by the open-source machine learning tool RapidMiner. The variations concerned both the selected algorithm and the training samples. The training sets of only ten sequences per tree was drawn randomly from a set of $4,000$ sequences ($2,000$ coding and $2,000$ non-coding) from [5]. The intuition was to obtain trees that involve at most two attributes (with the greatest variance between coding and non-coding sequences). These attributes depend on the training set and the selected learning technique. The performance of these trees was tested with $2,000$ test instances ($1,000$ coding and $1,000$ non-coding) outside the training set. As expected, the results were very poor due to the very small training set. Table 1 shows the overall performance which is about 50%, or in other words, as



(a) Two decision diagrams D_1 and D_2



(b) Merged decision diagram $D_1 \circ_M D_2$



(c) Simplified decision diagram $\circ_{simp}(D_1 \circ_M D_2)$

Fig. 4: Classifiers for coding (c) and non-coding (n) DNA sequences, based on features f_i

good as random classification. An interesting observation is that the first classifier tends towards *non-coding* and the second towards *coding*; the third is slightly better balanced, i.e., ratio of false positives and false negatives is smaller.

The merged tree, produced by the described merging procedure, performs surprisingly good. The overall accuracy was 65.25%, which is much better than any of the source classifiers. Recall that we used only very few (ten) training examples to train the individual decision trees; in total we had 30 samples. In experiments we found that about $1,000$–$2,000$ training examples are needed to reach this accuracy with a single-set decision tree. Furthermore, such trees had depth $\approx$7, which is much larger than height 3 of the merged tree. This accuracy cannot be enhanced much by using more training samples or source classifiers. Empirical results for different algorithms show that $\approx 75\%$ is the best one can expect, which seems to be a limit of the statistical features [15].

Our findings in experiments with DNA data from the Human Genome Project largely confirm those of [13]. Compared to training single diagrams, the merging approach

Table 1: DNA Classification Results (Data from Human Genome Project)

	Input 1		Input 2		Input 3		Merged	
	TC	TN	TC	TN	TC	TN	TC	TN
PC	175	214	854	877	262	346	565	260
PN	825	786	146	123	738	654	435	740
A	48.05%		48.85%		45.80%		65.25%	

PC/PN: predicted coding/non-coding, TC/TN: true coding/non-coding, A: accuracy

– often yields a simpler diagram structure (in particular height);
– often gains the same accuracy with a smaller (overall) training set; and
– can use parallel training (also with different methods).

We stress that qualitative improvements (as targeted in machine learning) was *not* the primary goal of our research, and thus we omit detailed statistical results here. Instead, our contribution is the *methodology and tool support for flexible DD merging*: many experiments and trials are needed to obtain the above results, and this is only reasonably possible with a tool allowing to quickly restructure the modular merging plans. This makes our system more powerful than, e.g., MORGAN, which uses a hard-coded merging *procedure*. We can switch from MORGAN's merging strategy to majority voting by changing one line in the merging plan; more than two input diagrams can be merged hierarchically, possible using different operators. Furthermore, different from MORGAN operatores can be reused for other applications.

Extending the Scenario. In addition, a major advantage when using our declarative approach is the *possibility to reason about DDs* between the merging steps. While $\circ_M$ returns exactly one output diagram for two input diagrams [13], the following variant $\circ_{M'}$ seems reasonable (for space reasons, we omit formal details): when merging two leaf nodes l_1 and l_2 with frequency distributions (c_1, n_1) and (c_2, n_2), the merged node l gets $\ell_F(l) = (c_1 + c_2, n_1 + n_2)$. While $\circ_M$ classifies l as coding (c) if $c_1 + c_2 > n_1 + n_2$, and non-coding (n) otherwise, it makes sense to produce *both* alternatives if $\left|\frac{c_1+c_2-(n_1+n_2)}{c_1+c_2+n_1+n_2}\right| < \epsilon$ for some threshold $\epsilon > 0$, i.e., the numbers of coding and non-coding samples are almost equal.

This strategy intends to avoid overfitting by estimating and minimizing the generalization error of the classifier, which is known as the *model selection problem* in machine learning (for more information, see [1]). Moreover, it may even lead to different diagram structures after simplification. Applying $\circ_{simp}$ to the diagram in Fig. 4b gives the diagram in Fig. 4c. In contrast, when the label of the shaded leaf node is switched from c to n, the left and the right subtree of the root become equivalent. Therefore $\circ_{simp}$ will eliminate the unnecessary branching at the root, and D_2 is reproduced.

A declarative choice program P_{sel} can then select, one of the alternatives. This program may prefer the diagram which performs best over some test set, or it prefers diagrams with a simpler structure or lower number of nodes. Our contribution in this regard is *tool support for convenient generation and selection of best merges* by declara-

tive merging plans, e.g., $\circ_{asp}(\circ_{simp}(\circ'_M(\{D_1\},\{D_2\})), P_{sel})$, where $\circ'_M$ (or any other operator) constructs candidates, and $\circ_{asp}$ makes the final selection. In the DNA application, this strategy led to sensible differences in the resulting diagrams (yet not to a significant increase in best precision).

7 Related Work and Conclusion

The integration of several classifiers is known in machine learning as *ensemble learning*, for which well-working methods are available; see e.g. [4, 9] for an overview. However, these approaches *train* new classifiers using an existing one and training samples. In decision diagram merging, we *directly integrate them* without using training samples at all. This strategy was also discussed in [13] where an algorithm and a tool for integrating decision diagrams for DNA classification was developed. We have discussed this scenario (and extensions). Their system, however, is monolithic, hard-coded, and tailored to this application. Our DDM system, instead, is more general and can be used for different tasks as well. Its modular architecture simplifies the exchange, reusability and modification of merging strategies enormously. This is especially useful for experimenting with different strategies and evaluating their outcomes empirically.

The real strength of our system becomes visible when combining merging capabilities with *declarative reasoning* about decision diagrams between the merging steps. User-defined ASP programs may be used on a meta-level to constrain the further integration process. This allows merging operators to produce *multiple* alternative results. The ASP program can in turn select one which is appropriate for the application in mind. In particular, the high expressivity of HEX programs and the possibility to access other software from them offers support to declare involved criteria.

Concerning complexity issues, both the time complexity of merging and the size of the integrated diagram depends on the merging operators in use. The analysis of concrete operators remains for future work.

References

[1] Arlot, S., Celisse, A.: A survey of cross-validation procedures for model selection. Statist. Surv. 4, 40–79 (2010)

[2] Bahar, R., Frohm, E., Gaona, C., Hachtel, G., Macii, E., Pardo, A., Somenzi, F.: Algebraic decision diagrams and their applications. In: ICCAD'93. pp. 188–191. IEEE (1993)

[3] Brewka, G., Eiter, T., Truszczyński, M.: Answer set programming at a glance. Commun. ACM (2011), to appear

[4] Dietterich, T.G.: Ensemble methods in machine learning. In: Intl. Workshop Multiple Classifier Systems. pp. 1–15. Springer (2000)

[5] Fickett, J.W., Tung, C.S.: Assessment of protein coding measures. Nucleic Acids Res. 20(24), 6441–6450 (1992), `http://fruitfly.org/sequence/human-datasets.html`

[6] Mair, J., Smidt, J., Lechleitner, P., Dienstl, F., Puschendorf, B.: A decision tree for the early diagnosis of acute myocardial infarction in nontraumatic chest pain patients at hospital admission. Chest 108(6), 1502–1509 (1995)

[7] Naylor, B., Amanatides, J., Thibault, W.: Merging BSP trees yields polyhedral set operations. In: SIGGRAPH'90. pp. 115–124. ACM (1990)
[8] Peng, H., Long, F., Ding, C.: Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. IEEE Trans. Pattern Anal. Mach. Intell. 27(8), 1226–1238 (2005)
[9] Polikar, R.: Ensemble based systems in decision making. IEEE Circuits Syst. Mag. 6(3), 21–45 (2006)
[10] Quinlan, J.R.: Simplifying decision trees. Int. J. Hum.-Comput. St. 51(2), 497 – 510 (1999)
[11] Redl, C., Eiter, T., Krennwallner, T.: Declarative belief set merging using merging plans. In: PADL'11. pp. 99–114. Springer (2011)
[12] Salzberg, S.: Locating protein coding regions in human DNA using a decision tree algorithm. J. Comput. Biol. 2(3), 473–485 (1995)
[13] Salzberg, S., Delcher, A.L., Fasman, K.H., Henderson, J.: A decision tree system for finding genes in DNA. J. Comput. Biol. 5(4), 667–680 (1998)
[14] Sobin, L., Gospodarowicz, M., Wittekind, C. (eds.): TNM classification of malign tumors. Wiley-Blackwell, 7 edn. (2009), http://www.uicc.org
[15] Sree, P.K., Babu, I.R.: Identification of protein coding regions in genomic DNA using unsupervised FMACA based pattern classifier. Int. J. Comput. Sci. Network Secur. 8(1), 305–309 (2008)

Constraints and Global Optimization for Gene Prediction Overlap Resolution

Christian Theil Have

Research group PLIS: Programming, Logic and Intelligent Systems
Department of Communication, Business and Information Technologies
Roskilde University, P.O.Box 260, DK-4000 Roskilde, Denmark
E-mail: cth@ruc.dk

Abstract. We apply constraints and global optimization to the problem of restricting overlapping of gene predictions for prokaryotic genomes. We investigate existing heuristic methods and show how they may be expressed using Constraint Handling Rules. Furthermore, we integrate existing methods in a global optimization procedure expressed as probabilistic model in the PRISM language. This approach yields an optimal (highest scoring) subset of predictions that satisfy the constraints. Experimental results indicate accuracy comparable to the heuristic approaches.

1 Introduction

Traditionally, gene finding has been considered as a classification task which could be performed without much context [6]. This ignores the problem of the constraints between the set of predicted genes and their placement in the genome. A common problem occurs with overlapping genes. Overlapping genes are rare in prokaryotic genomes, but they do occur [12, 8].

The traditional intrinsic gene finding methods have a tendency to predict too many overlapping genes (particularly in GC rich genomes) because the feature patterns of a gene predicted in one reading frame give rise to similar feature patterns in other reading frames. This effect is known as shadow genes.

Several gene finders deal with the problem of overlapping genes by discarding some of the overlapping predictions in a post-processing step. In this paper we consider and compare such post-processing techniques and give unified presentation using Constraint Handling Rules [7]. We demonstrate how such rules can be formulated as constraints and integrated with a global optimization procedure implemented as a constrained Markov chain in the PRISM system [13].

We adopt a divide and conquer approach to gene finding, which can be seen as composed of two steps:

1. A gene finder supplies a set of candidate predictions $p_1 \dots p_n$, called the *initial set*.
2. The *initial set* is pruned according to certain rules or constraints. We call the pruned set the *final set*.

The present paper is concerned with methods for the second step. The purpose of this step is to repair effects of flawed assumptions in the first step, i.e. leading to over-prediction of overlapping genes, and more specifically to improve accuracy by pruning false predictions. We consider this step as a Constraint Satisfaction Problem (CSP).

Definition 1. *A Constraint Satisfaction Problem is a triplet* $\langle X, D, C \rangle$*. X is a set of n variables, $X = x_1, \ldots, x_n$, with domains $D = D(x_1), \ldots, D(x_n)$. The constraints C impose restrictions on possible assignments for sets of variables. A solution is an assignment of a value $v \in D(x_i)$ to each variable $x_i \in X$, consistent with C.*

We introduce variables $X = x_i \ldots x_n$ corresponding to each prediction $p_1 \ldots p_n$ in the initial set. All variables have boolean domains, $\forall x_i \in X, D(x_i) = \{true, false\}$ and $x_i = true \Rightarrow p_i \in$ **final set**.

If there are multiple solutions, then we are usually interested in the "best" one. We interpret "best" as meaning a solution that contains as many real genes as possible and as few incorrect predictions as possible. We do not know in advance which predictions are correct, but optimize the probability (or a similar measure) that the predictions are correct. This extends the problem as a constraint optimization problem.

Definition 2. *A Constraint Optimization Problem (COP) is a CSP where each solution is associated with a cost and the goal is to find a solution with minimal cost*[1]*.*

2 Local heuristic methods

An approach taken by many gene finders is to employ local heuristic pruning rules to post-process a set of gene predictions. These rules make pruning decisions based on the context of only a subset of the predictions. Typically, the rules consider overlapping predictions on a case by case basis and deletes inconsistent predictions based on various criteria. The rules essentially work as propagators that reduce the domains of variables, e.g. a deletion corresponds to reducing the boolean domain of the corresponding variable to $false$. The drawback is that the rules are generally not guaranteed to yield a globally optimal solution and that they may produce different solutions depending on the order in which they are applied.

These types of rules are conveniently expressed as *simplification* rules in the Constraint Handling Rules (CHR) language. Such rules work on a constraint store, which starts out as the initial set. The simplification rules remove predictions from the constraint store, until no more rules apply. Then, the constraint store represents the final set.

As example, consider the post-processing procedure of the Genemark frame-by-frame gene finder [11] expressed as a single rule in CHR:

[1] Or equivalently, a solution with maximal negative cost (utility).

```
prediction(Left1,Right1), prediction(Left2,Right2) <=>
        Left1 =< Left2,  Right1 >= Right2
        |  prediction(Left1,Right1).
```

The head of the rule — the part before `<=>` — matches two predictions in the constraint store. The rule replaces both predictions with the first prediction if the first prediction completely overlaps the second prediction. This condition is expressed in the guard of the rule – the part between the head and the `|` character. The rule is applied for all predictions matching the head and the guard, effectively removing all predictions which are completely overlapped by another prediction. With this rule it does not matter in which order the predictions are processed – the final set will be same. This is a consequence since the program consisting of the unique rule presented is *confluent* [1], i.e. it is not sensitive to the order of execution.

As an example of non-confluent rules, consider the scheme used in the ECOPARSE gene finder [9] which addresses partial overlaps and the score of the predictions:

```
prediction(Left1,Right1,Score1), prediction(Left2,Right2,Score2) <=>
    overlap_length((Left1,Right1),(Left2,Right2),OverlapLength),
    length_ratio((Left1,Right1),(Left2,Right2),Ratio),
    length(Left1,Right1,Length1), length(Left2,Right2,Length2),
    OverlapLength > 15, Score1 > Score2
    ((Length1 > 400, Length2 > 400) ; Ratio > 0.5),
    | prediction(Left1,Right1,Score1).

prediction(Left1,Right1,Score1), prediction(Left2,Right2,Score2) <=>
    overlap_length((Left1,Right1),(Left2,Right2),OverlapLength),
    length_ratio((Left1,Right1),(Left2,Right2),Ratio),
    length(Left1,Right1,Length1), length(Left2,Right2,Length2),
    OverlapLength > 15,  Ratio =< 0.5,  Length1 =< Length2
    | prediction(Left1,Right1,Score1).
```

If two predictions overlap by more than 15 bases, then one of them is removed. If the ratio between the longest and shortest of the predictions is more than 0.5, then the lowest scoring is removed (first rule) otherwise the shortest one is removed (second rule). Note how this may lead to different effects depending on the order in which predictions are considered, as illustrated in figure 1.

There are other approaches which employ more complex local heuristics. An example is heuristics of the RescueNet gene finder [10] which has rules considering scores, percent overlaps and local overlaps between up to three predictions. These heuristics can be implemented with nine CHR rules (not shown), but the resulting program is not confluent.

It is a general theme for the heuristics to be based on two central characteristics of overlapping predictions – the score of the predictions and the (relative) lengths of the predictions and the overlap.

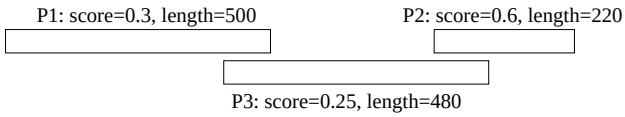


Fig. 1. ECOGENE post processing: We have two predictions $P1$ and $P2$ that overlap each end of a third prediction $P3$ by more than 15 bases. If $P1$ and $P3$ is are considered before $P2$ and $P3$ then $P3$ will be removed by the first rule. Consequently $P2$ does not overlap and is kept. If they are considered in opposite order, however, then $P2$ will be removed by the second rule and subsequently $P3$ is removed by the first rule.

3 Global optimization

We would like the final set to reflect the relative confidence scores in the predictions assigned by the gene finder and at the same time be consistent with the overlap constraints. To accomplish this we reformulate the problem as a constraint optimization problem.

Let the scores of $p_1 \dots p_n$ be $s_1 \dots s_n$ and $s_i \in \mathcal{R}^+$. The scores are the confidence scores given by the underlying gene finder, i.e. they reflect the supposed probability that a prediction constitutes a real gene. Such scores are commonly expressed as probabilities, but need not be.

We would like to maximize the sum of the scores $\sum_{i=1}^{n} s_i$ since it is directly related to the criteria of the model that produced the initial set. With this criteria, the inclination to prune a prediction in the final set is inversely proportional to the score which is expected to reflect the underlying models belief that the prediction is a real gene.

To perform global optimization with a set of constraints, we propose to use a constrained first-order Markov chain. We assume that a gene finder has produced initial set of predictions, $p_1 \dots p_n$, and further require these to be sorted by the position of their left-most base, such that $\forall p_i, p_j, i < j \Rightarrow \text{left-most}(p_i) \leq \text{left-most}(p_j)$. The variables $x_1 \dots x_n$ of the CSP are given the same ordering.

The Markov chain has a *begin* state, an *end* state and two states for each variable x_i corresponding to its boolean domain $D(x_i)$. The state corresponding to $D(x_i) = true$ is denoted α_i and the state corresponding to $D(x_i) = false$ is denoted β_i. In this model, a path from the begin state to the end state corresponds to a potential solution of the CSP. The Markov model is illustrated in figure 2. The *begin* state has transitions to α_1 with probability $P(\alpha_1|begin) = \sigma_1$ and β_1 with probability $P(\beta_1|begin) = 1 - \sigma_1$. The last two prediction states, α_n, β_n can only transit to the end state, i.e. $P(end|\alpha_n) = P(end|\beta_n) = 1$. For all other states, we have the transition probabilities,

$$P(\alpha_i|\alpha_{i-1}) = P(\alpha_i|\beta_{i-1}) = \sigma_i \text{ and } P(\beta_i|\alpha_{i-1}) = P(\beta_i|\beta_{i-1}) = 1 - \sigma_i$$

We normalize the scores to the interval $(0.5, 1]$, yielding the normalized probability scores $\sigma_1 \dots \sigma_n$, in the following way,

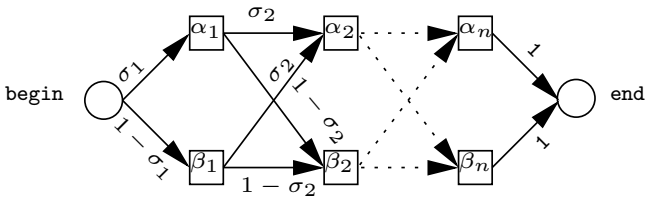


Fig. 2. Illustration of the Markov chain used. The transitions are marked with their corresponding probabilities. Only the first few and the last states are included - the dotted transition arrows symbolize the omitted $\alpha_3 \ldots \alpha_{n-1}$ and $\beta_3 \ldots \beta_{n-1}$ states and their transitions, which follows the same principle as the previous.

$$\sigma_i = 0.5 + \lambda + \frac{(0.5 - \lambda) \times (s_i - min(s_1 \ldots s_n))}{max(s_1 \ldots s_n) - min(s_1 \ldots s_n)}$$

λ is a small pseudo-count to ensure that all σ scores are above 0.5. Since α probabilities are always larger than 0.5, the model prefers α states over their corresponding β states. Hence, a most probable path from the *begin* state to the *end* state will not include any β states. The predictions that maximize the product of the σ scores will also maximize the sum of the original scores, since the normalized σ scores are monotonic to the original scores, $\sigma_i \geq \sigma_j \iff s_i \geq s_j$.

For inference with the model we use the Viterbi algorithm [16], which returns a most probable state sequence $\{begin, S_1, S_2 \ldots S_n, end\} | S_i \in \{\alpha_i, \beta_i\}$.

Constraints are defined on states that are not allowed to occur together in a path. These constraints force the Viterbi algorithm to choose a most probable path, consistent with the imposed constraints, i.e. this path may include β states. The constraints are formulated as CHR rules similar to those of the local heuristics, but instead of removing predictions they define conditions for inconsistency. We call these *inconsistency rules*. Inconsistency rules match predictions corresponding to α and β states in the head of the rule. The guard of the rule ensures that the additional criteria for rule application are met and the implication of the rules is always failure. Note that such rules are necessarily confluent.

As example, version 3 of the Glimmer gene finder [5] use a similar approach with a constraint that enforce a maximal length of overlaps (110 for *E.coli*). In our system, this constraint is formulated as,

```
alpha(Left1,Right1), alpha(Left2,Right2) <=>
    overlap_length((Left1,Right1),(Left2,Right2),OverlapLength),
    OverlapLength > 110
    | fail.
```

The Genemark heuristic rule is represented as two inconsistency rules,

```
alpha(Left1,Right1), alpha(Left2,Right2) <=>
    Left1 =< Left2, Right1 >= Right2 | fail.
beta(Left1,Right1), alpha(Left2,Right2) <=>
    Left1 =< Left2, Right1 >= Right2 | fail.
```

The first rule states that one prediction may not completely overlap another and the second says that we cannot include a prediction if a pruned prediction completely overlaps it. Since the heuristic is confluent it may also be applied to the initial set as a filtering algorithm before the process of global optimization. We can reformulate the two ECOGENE rules in the same fashion (guard is omitted, but it is the same as in the heuristic rules),

```
alpha(Left1,Right1), alpha(Left2,Right2) <=>  ... | fail.
beta(Left1,Right1), alpha(Left2,Right2) <=>  ... | fail.
```

Note that the `Score` arguments have been removed. They are now implicitly integrated in the optimization algorithm. The confluence issue is resolved due to the optimization procedure. In effect, the execution strategy that maximizes the score is applied.

3.1 Implementation in PRISM

A PRISM program that implements the constrained Markov chain is created from the initial set of predictions and constraints expressed as CHR rules. PRISM is an extension of Prolog with special goals representing random variables. A derivation of the PRISM program corresponds to a path through the Markov chain. The Markov chain is implemented as a recursive predicate, such that in the $i'th$ recursive call, the (random) variable x_i is assigned a value corresponding to a Markov chain state; α_i or β_i. After each recursion — an attempted transition in the Markov model — the constraints are checked.

Relevant recent states As part of a derivation we maintain a list of recent states (m_i) sorted by the right-most position of the corresponding predictions. Constraints are only checked for predictions corresponding to elements of m_i. In step i, we construct m_i as the maximal prefix of $x_i + m_{i-1}$, such that $x_j \in m_i \iff \text{right-most}(p_j) \geq \text{left-most}(p_i)$. If the constraints propagate `fail`, then the PRISM derivation fails and the (partial) path it represents is pruned from the solution space.

The most probable consistent path is found using PRISMs generic adaptation of the Viterbi algorithm for PRISM programs [14].

4 Evaluation

In lack of a true golden standard, we use an accepted reference set to define the set of "correct" genes. A slight complication of this approach is that the reference set itself may have incorrect and missing annotations. True positives

are gene predictions in the final set which are included (exactly) in the reference set and false positives are those predictions that are not.

Traditionally, in gene finding, accuracy is measured in terms sensitivity and specificity. Sensitivity measures the fraction of reference genes exactly predicted by the approach and specificity measures the fraction of predicted genes that are correct. Since the starting point is the initial set of predictions (which may omit some potential genes) we cannot improve on sensitivity. The goal of a pruning approach is then to improve on specificity with minimal impact to sensitivity.

We consider a pruning approach *successful* wrt. to an initial set when it prunes false positives at a higher rate than it prune true positives. This is reflected by the difference in sensitivity and specificity of the final set compared to the initial set.

We consider constraints *safe* when the constraints prune only false positives. Neither of the examined constraints are safe with respect the RefSeq annotation of *E.coli*, *NC_*000913. Three of the reference genes are completely overlapped by another reference gene. These would be removed by the genemark heuristic and hence it is not safe, although the negative impact of sensitivity would negligible. Similarly with the Glimmer constraint – the reference annotation have four overlaps longer than 110 bases which would be removed by this constraint. There are 93 overlaps longer than 15 bases. All of these would be removed by the ECOGENE constraints, which is therefore expected to have a noticeable negative sensitivity impact.

4.1 Experimental validation

We compare the different approaches using the predictions from a very simple codon preference based gene finder – the simplest model described in [4]. The gene finder has been trained on *E.coli NC_*000913 and applied to predict genes in the same genome. It overpredicts quite a lot – a total of 10799 predictions for the genome, which has 4145 known genes.

We ran the constrained Markov chain using the gene finder predictions as initial set, applying our adaptations of the both the Genemark constraint, the ECOGENE constraint and the Glimmer3 constraint. We also tested the local heuristic versions of the Genemark and ECOGENE constraints. The results are summarized in table 1.

Both the Genemark and ECOGENE heuristics achieve quite impressive improvement compared to the initial set. Our global optimization achieves better sensitivity than ECOGENE and better specificity than Genemark, but seen as a combination of the measures, the result is not significantly better.

Note that the optimal or highest scoring set of predictions subject to the constraints is not necessarily the most successful, but it is the one that most faithfully reflects the confidence scores assigned by the gene finder.

The purely declarative CHR implementations of genemark or ECOGENE rules are quite slow (hours), e.g. it essentially considers each pair of constraints resulting in $\mathcal{O}(n^2)$ complexity, n being the number of predictions in the initial

Method	#predictions	Sensitivity	Specificity	Time (seconds)
initial set	10799	0.7625	0.2926	na
Genemark rules	5823	0.7558	0.5379	1.4
ECOGENE rules	4981	0.7148	0.5947	1.7
global optimization	5222	0.7201	0.5714	75

Table 1. Accuracy of predictions using different overlap resolution approaches. Note that the results for the ECOGENE heuristic may vary depending on execution strategy - in case of above results, predictions with lower left position are considered first.

set. However, with proper control in place (using the relevant recent states optimization described in section 3.1), they can be made to run very fast (less than two seconds). The running time for the global optimization is slower – it takes a little more than one minute. This is still acceptable.

5 Conclusions

We presented a novel way to post-process gene prediction results based on constrained global optimization. Contrary to the heuristic approaches our approach provides an optimality guarantee – the final set of prediction will be the maximally scoring set that satisfies the imposed constraints. We have incorporated existing heuristic methods with the optimization procedure using inconsistency rules implemented in CHR. Currently, the approach has similar accuracy to the heuristic methods. The results indicate that maximizing the sum of scores have the effect of including more short predictions. This could be addressed weighting the scores by prediction length. We also plan to experiment with different constraints to achieve better accuracy. Our approach is limited to local overlap constraints and is not well-suited for global or long-distance constraints.

We are not the first to use dynamic programming based approaches to post-processing of gene predictions. Version 3 of Glimmer [5] use a custom dynamic programming algorithm which is similar to the present approach, but incorporates only the maximal overlap constraint. Another difference is that our approach is expressed as a declarative PRISM program and can therefore utilize the generalized Viterbi algorithm. Our approach is similar to constrained HMMs in PRISM, which has previously be applied to other biological sequence analysis tasks [2, 3]. A main difference is that we express constraints with CHR rules.

CHRiSM[15] already combines CHR and PRISM and is to our knowledge the first system to do so. CHRiSM assigns probabilistic semantics to CHR rules, which are interpreted as chance rules – e.g. even if a rule head is matched the rule is only applied with a certain probability. The main difference with our approach is that we use ordinary CHR rules in conjunction with a PRISM program, although ordinary CHR rules may be seen as a special case of CHRiSM rules, where the probability of invocation is one. Additionally, the form of the CHR rules we use is restricted (inconsistency rules) and they are only used in the constraint checking part of the PRISM program. It would be interesting to

use CHRiSM as a method of incorporating soft constraints with our approach, e.g. redefining the inconsistency rules as CHRiSM chance rules.

Acknowledgement This work is part of the project "Logic-statistic modeling and analysis of biological sequence data" funded by the NABIIT program under the Danish Strategic Research Council.

References

[1] S. Abdennadher, T. Fruehwirth, and H. Meuss. On confluence of constraint handling rules. *Lecture Notes in Computer Science*, 1118:1–15, 1996.

[2] Henning Christiansen, Christian Theil Have, Ole Torp Lassen, and Matthieu Petit. A constraint model for constrained hidden markov models: a first biological application. In *Proc. of the International Workshop on Constraint Based Methods for Bioinformatics*, pages 19–26, Lisbon, Portugal, September 2009.

[3] Henning Christiansen, Christian Theil Have, Ole Torp Lassen, and Matthieu Petit. Inference with constrained hidden markov models in PRISM. *TPLP*, 10(4-6):449–464, 2010.

[4] Henning Christiansen, Christian Theil Have, Ole Torp Lassen, and Matthieu Petit. Bayesian Annotation Networks for Complex Sequence Analysis. In John Gallagher and Michael Gelfond, editors, *Technical Communications of the 27th International Conference on Logic Programming (ICLP'11)*, volume 11 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 220–230, Dagstuhl, Germany, 2011. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[5] Arthur L. Delcher, Kirsten A. Bratke, Edwin C. Powers, and Steven L. Salzberg. Identifying bacterial genes and endosymbiont DNA with Glimmer. *Bioinformatics*, 23:673–679, 2007.

[6] James W. Fickett and Chang-Shung Tung. Assessment of protein coding measures. *Nucl. Acids Res.*, 20(24):6441–6450, 1992.

[7] Thom W. Frühwirth. Constraint handling rules. In Andreas Podelski, editor, *Constraint Programming*, volume 910 of *Lecture Notes in Computer Science*, pages 90–107. Springer, 1994.

[8] Yoko Fukuda, Yoichi Nakayama, and Masaru Tomita. On dynamics of overlapping genes in bacterial genomes. *Gene*, 323:181 – 187, 2003.

[9] Anders Krogh, I. Saira Mian, and David Haussler. A hidden Markov model that finds genes in E.coli DNA. *Nucl. Acids Res.*, 22(22):4768–4778, 1994.

[10] Shaun Mahony, James O. McInerney, Terry J. Smith, and Aaron Golden. Gene prediction using the self-organizing map: automatic generation of multiple gene models. *BMC Bioinformatics*, 5:23, 2004.

[11] Anton M.Shmatkov, Arik A.Melikyan, Felix L.Chernousko, and Mark Borodovsky. Finding prokaryotic genes by the frame-by-frame' algorithm: targeting gene starts and overlapping genes. *Bioinformatics*, 15(11):874–886, 1999.

[12] S. Normark, S. Bergstrom, T. Edlund, T. Grundstrom, B. Jaurin, F. P. Lindberg, and O. Olsson. Overlapping genes. *Annual Review of Genetics*, 17:499–525, 1983.

[13] Taisuke Sato. Generative Modeling by PRISM. *Proceedings of the International Conference on Logic Programming*, LNCS 5649:24–35, 2009.

[14] Taisuke Sato and Yoshitaka Kameya. A viterbi-like algorithm and EM learning for statistical abduction, June 16 2000.

[15] Jon Sneyers, Wannes Meert, Joost Vennekens, Yoshitaka Kameya, and Taisuke Sato. CHR(PRISM)-based probabilistic logic learning. *TPLP*, 10(4-6):433–447, 2010.

[16] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–269, 1967.

Introducing FIASCO: Fragment-based Interactive Assembly for protein Structure prediction with COnstraints

M. Best[1], K. Bhattarai[1], F. Campeotto[2], A. Dal Palù[3,1], H. Dang[1], A. Dovier[2], F. Fioretto[1], F. Fogolari[2], T. Le[1], and E. Pontelli[1]

[1] Depts. Computer Science and Biology, New Mexico State University
[2] Depts. Math. & Computer Science and Biomed. Sciences & Tech., Univ. Udine
[3] Dept. Mathematics, Univ. Parma

Abstract. The paper summarizes the recent developments in the creation of a constraint-based framework for the analysis of protein conformations. The framework is designed to be used by computational and life scientists to provide new insights in the structural analysis of proteins. The framework is composed of two parts: a graphical and web-based user interface and a parallel constraint solving engine. The solver is able to model the geometric and energetic properties of proteins and perform a conformational search among structure candidates. The paper reports the design and preliminary experimental considerations.

1 Introduction

In this paper, we present an overview of *FIASCO (Fragment Interactive Assembly for protein Structure with COnstraints)*, a new framework dedicated to protein structure analysis. The goal of our project is to deliver a tool that can be used by researchers interested in the analysis of protein behaviors and structure. The underlying core of the system is built on *constraint programming technology*, which has been shown to be effective in modeling and reasoning about complex systems, because of its modularity, simplicity, and its implicitly exploitable parallelism. This project is work in progress—in this report we highlight the overall structure and current development status. The key features and goals that guide the development of the system are:

- A simple and portable interface to allow a convenient protein modeling;
- The search engine technical details are hidden by the interface;
- It provides online access and seamlessly integrates with existing databases;
- It uses simplified spatial models in order to speed-up the search, while maintaining a good description of protein features;
- The geometrical description of protein structure is based on a set of simple physical constraints that are controlled by the user;
- The scoring of conformations is based on user-selectable energy functions and it is decoupled from the geometric modeling;

- It effectively uses constraint technology to explore the protein's degrees of freedom and to guide the generation of conformations;
- It exploits cluster-level parallelism to enhance computational speed.

For biological background and constraint modeling we refer to [3, 4].

FIASCO's capabilities range from ab-initio prediction on approximated models to conformation analysis and refinement.

For *ab-initio prediction*—i.e., search for a molecular structure that lies in the global minimum of a suitable cost function—a set of native state candidates are generated according to a user-selectable scoring function. A set of additional information, derived from homology, prediction and/or experimental data, can be included as constraints describing the molecule's properties. The results can be refined by molecular dynamics for further enhancements in accuracy [3].

In the *analysis of conformations*, a global enumeration of promising and geometrically feasible conformations is performed. This can be used for studying protein flexibility, loop closure, docking, long range transitions, and even as part of the folding process. Particularly for those scenarios that require a long time scale, which are beyond the current state-of-the-art all-atom simulations, FIASCO is able to provide a simplified yet insightful overview of the process.

The ability to generate a conformational space with the guidance of constraints is substantially different from traditional approaches, e.g., those based on Monte Carlo, Genetic and/or Simulated Annealing methods. Classical algorithms produce new offsprings/results based on the evolution of a previous generation and thus biased by the energy model used. The geometric description through constraints allows us, instead, to sample the space with different methods and heuristics that can be controlled with greater precision.

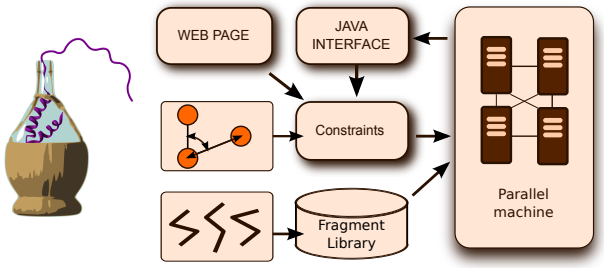


Fig. 1. FIASCO logo and structure

In Fig. 1, we depict the overall structure of the framework. The user interface allows scientists to define the protein and its geometric characteristics. These properties are encoded as a set of user-defined structural constraints, that add to the possible local arrangements stored in a *fragment* library. The Constraint Optimization Problem defined by the user through constraints and a cost function is solved by a parallel *constraint solver*. The results, e.g., protein structures, can be fed back to the interface for further analysis and refinement. In the remainder of the paper we provide some details about each single phase.

Related work. Several studies have been presented using constraint programming in the context of the protein folding problem. Most of them deal with lattice spatial models, where each amino acid is represented by its $C\alpha$ atom and the set of points allowed are the nodes of a discrete lattice. For example, [1] investigates the protein folding problem using a *face centered cubic lattice* and a Boolean energy function—i.e., an energy contribution (measured as -1) is present only when two hydrophobic amino acids are in contact, while other contacts do not provide any contributions. The authors are able to efficiently determine the optimal folding, with minimal energy, for relatively long proteins. In [3, 6], this approach is extended with a more refined energy function. In particular, in [6] the authors propose an ad-hoc parallel constraint solver, COLA, based on constraint solving over 3D domains for amino acid locations. Another line of research has been developed that makes use of constraint based technologies to investigate protein folding and protein docking problems [2].

In the context of fragment assembly, the most popular approach is the one used in the ROSETTA system [12]. Lee et al. [11] have recently extended the idea of fragment assembly to investigate loop modeling, i.e., predicting the loops that connect partially known parts of a protein structure.

Finally, the design of our graphical interface has been partially inspired by the *Fold It* video game (`http://fold.it/portal`).

2 Web Interface and Java GUI

We have developed the proof-of-concept of a graphical user interface that allows the interactive definition of a protein structure prediction problem. The user can identify the target primary sequence and manually define known fragments and constraints among fragments (e.g., spatial constraints), using 3D representations of structures. The GUI interacts with public databases, enabling the retrieval of homology and secondary structure prediction information. The GUI provides also a portal to the constraint solving engine, located on a dedicated server, allowing the user to submit all the collected information and constraints, and retrieve and graphically represent the output produced by the constraint engine. The GUI itself is integrated in a Web portal.

The GUI is based on the BioJava [10] and the Jmol (`http://www.jmol.org`) open-source libraries, that offer many features to process and visualize protein structures and to handle protein structure encoding formats. The tool is structured according to the Model-View-Controller pattern style. It allows the user to define fragments selected from homologous proteins and to introduce geometric constraints by placing the fragments in a 3D workspace.

The workflow is managed by three main panels (Fig. 2): **(1)** the *Target* panel, that displays the alignment of the selected fragments compared to the target, **(2)** the *Extraction* panel, which allows the user to extract fragments from a known protein (these fragments will be referred as *special fragments*), and **(3)** the *Assembling* panel, where the user can assemble, move, and impose the following types of constraints on the fragments:

- *Block constraint:* Fragments can be constrained in their relative coordinates as a single rigid unit (free to rotate and translate).
- *Volume constraint:* A specific point (amino acid) is forced into a specific box-shaped volume.
- *EQ/LEQ Distance constraint:* It imposes a constraint of exact/maximum Euclidean distance between two residues.
- *Exact_Coordinates constraint:* The specific fragment is blocked with specific coordinates in the space, in order to prevent rotations and translations.

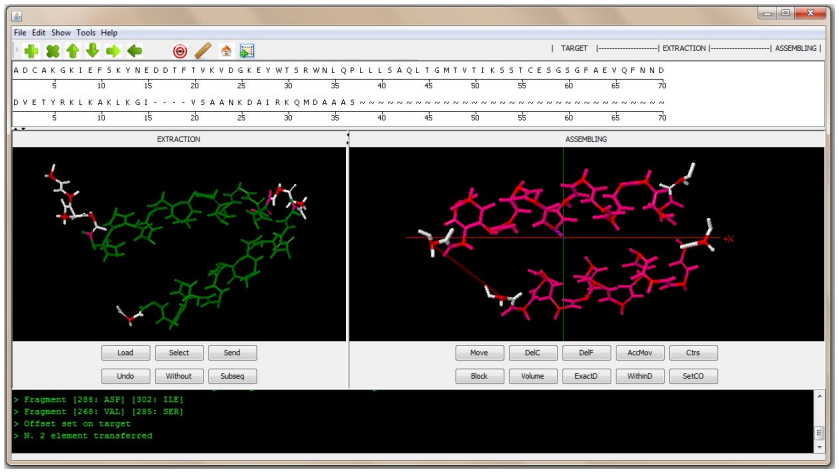


Fig. 2. A screenshot of the FIASCO Java interface

3 The constraint solver

In this section we describe the constraint framework addressed for encoding the problem of the Protein Structure Prediction.

Main geometrical constraints. Our approach is based on a protein fragment assembly technique [4, 5]. The key idea is to simplify the exploration of the conformational space of a protein by looking at a subset of probable and observed behaviors of several local substructures. Each part of the protein is associated to a number of *fragments*, i.e., possible arrangements in the space, that can be combined together while preserving typical structural constraints, derived from chemical properties and user-defined restrictions.

We start by devising a *fragment library* of structures observed in the Protein Data Bank (PDB, `http://www.pdb.org`) for fragments of short length (e.g., 4 amino acids), along with statistics concerning their occurrence frequencies. For the case of longer fragments, the PDB typically does not offer a statistically relevant coverage. However, it is possible to derive highly conserved *homologous* structures that resemble an hypothetical similar conformation with high probability. These *special* fragments can be retrieved by combining different types of primary sequence analysis, e.g., alignment for homology, secondary structure

prediction, etc. The interface provides methods to aid the user in identifying large fragments that can be used as building blocks of the protein.

The fragments are handled through a constraint that models the assembling of consecutive selected fragments. This constraint, in conjunction with `all_distant` global constraint [7], allows us to model geometrically feasible and probable structures. However additional constraints, coming from user knowledge, can restrict the feasible structures. For example, the prediction of secondary structure and the presence of similar known sequences with structure may be added to prune incompatible solutions. We also introduce a novel constraint to describe spatial interactions among known *special* fragments (the *pair constraint*). A fragment of type *special* represents a highly conserved polymer conformation. These conformations are likely to form strong local interactions among each other, according to the nature of their components and their specific shapes. In order to capture these local interactions, we design a constraint that models the relations among local conformations by identifying the possible placements of *fragments pairs*. These relationships are statically determined during preprocessing where, given the first fragment of the pair, multiple orientations and distances are tested for the second fragment. We maintain the k-best relative positions according to the contact and torsional angle contributions of a free energy model.

Modeling. Let us introduce the following notations. The primary sequence of a protein is denoted by $S = a_1, \ldots, a_n$, where a_i is the i^{th} amino acid. $\mathcal{F}_{std}$ is the set of *standard fragments* imported from the *Assembly DB* [4]. $\mathcal{F}_{spc}$ is the set of *special fragments*, containing the fragments created through the Extraction panel described in section 2. The Constraint Satisfaction Problem is defined over three set of variables: `Point`, `Fragment`, and `Pair`.

The variables in the set `Point` represent the 3D position of the corresponding $C\alpha$ atoms and their domains are described by pairs $[\boldsymbol{L}, \boldsymbol{U}]$, where $\boldsymbol{L}, \boldsymbol{U} \in \mathbb{R}^3$ represents the lower and the upper bound of the cube in which the variable can range, namely the $p \in \mathbb{R}^3$ such that $L_x \leq p_x \leq U_x$, $L_y \leq p_y \leq U_y$ and $L_z \leq p_z \leq U_z$. The points are subject to the `all_distant` property [7], i.e., each pair of amino acids must be separated by a minimal Euclidean distance. Another distance constraint restricts the maximal `diameter` between each pair of $C\alpha$ atoms. As argued in [3], a good diameter value is $5.68\, n^{0.38}$ Å.

Each variable $\mathbf{F}_i$ in `Fragment` is associated to fragments starting at amino acid i and it represents the possible choices of compatible fragments that can be placed over the positions $a_i, \cdots, a_{i+k}$ $(3 \leq k \leq n-3)$. The domain of the variable contains every standard and/or special fragment that is compatible with the specific amino acid types involved.

The information necessary to combine two fragments $\mathbf{F}_i$ and $\mathbf{F}_j$ consists of a rotation matrix $M_{i,j}$ and a translation vector $\boldsymbol{s}_j$, which provide the relative affine transformation to best fit the two fragments—by overlapping the 3 amino acid in common between the two fragments (the last 3 in $\mathbf{F}_i$ and the first 3 in $\mathbf{F}_j$). This information is computed in a pre-processing stage to ensure an efficient handling during the search.

In order to restrict the search space, when using special fragments, it is possible to pre-compute the most probable spatial relationships between pair of special fragments. The search can explore first these potential relative placements and therefore we introduce the set `Pair` of variables, that summarizes this information into the corresponding domains. Each variable in the set refers to a specific pair of special fragments.

Each domain element of a variable in `Pair` identifies a specific rotation matrix M and a translation vector $\boldsymbol{v}$ associated to the relative position of the two fragment involved. M and $\boldsymbol{v}$ describe the affine transformations to be applied to the second fragment of the pair, in order to couple the two fragments with a favorable energy contribution. A special value is added to the domain. This value is selected when no specific relationship is enforced over the pair.

Search. The search is guided by the instantiation of the fragment variables. These variables are selected according to the following strategy. We distinguish the following cases:

1. The leftmost $\mathbf{F}_i$, such that either there is no pair variable associated or every pre computed association has been tested before, is selected and assigned.
2. There exists a pair variable $\mathbf{P}$ associated to the i^{th} and j^{th} fragment variables $\mathbf{F}_i$ and $\mathbf{F}_j$, and a fragment f_a has already been selected from the domain $\mathbf{F}_i$. The choice of a specific element from the domain of $\mathbf{P}$ will propagate a specific $f_b \in \mathbf{F}_j$ and relative positions over the corresponding `Point` variables.

This selection strategy tries to label first all the special variables `Fragment`, possibly involved in some pair relation. The rationale behind this choice is the desire to produce a highly constrained search space.

Note that the selection strategy described above attempts to place variables `Pair` in a cascade effect: the selection of a pair $\langle f_a, f_b \rangle$ can trigger at the next step the selection of a pair $\langle f_b, f_c \rangle$ if such a relation exists. The intuition here is that, multiple pair relations among fragments might denote the presence of groups of structures in the target protein, characterized by strong local interactions. These local forces, providing high stability to the polypeptide, might give a strong hint what the final 3D protein structure looks like.

The values of the domains of the variables selected are assigned starting with the most likely choice, or with the one maximizing local interaction (when applicable). Energy contributions are encoded as constraints that link the coordinates associated to variable `Point` and the types of the amino acids involved.

From a computational point of view, we analyzed the computational complexity of the fragment assembly problem, and showed that finding a conformation with an energy fewer than an input limit is NP-complete even in a 2D space, HP energy model approximation of the problem.

Energy. The *energy function* employed is described in [5] and it is based on three components: *(1)* a contact potential for side chain and backbone contacts, *(2)* an energy component for each backbone conformation based on backbone conformational preferences observed in the database, and *(3)* a component that considers the relative orientation of spatially close triplets.

The first two components are described in [4]. The third component weighs the proper orientation of three consecutive amino acid fragments in order to form hydrogen bonds, following [9]. This energy contribution is introduced when the distance between two three-amino acid fragments is less than 5.8Å. Each fragment identifies a plane, and we are interested in those cases where the planes of the two fragments are almost co-planar and normal to the distance vector, i.e., the absolute product of the cosines of the angles between the normals to the two planes among themselves and with the distance vector is greater than 0.5.

4 Parallelism

The constraint solving engine has been implemented in C and parallelized using MPI and a multi-threaded design; the parallel version is capable of performing dynamic load balancing to address the problem of the irregular structure of the search trees. The search can be abstracted as a tree where each node (a choice point) is expanded according to available choices. Some of the choices can fail due to constraint inconsistencies detected by constraint propagation. In the parallel framework, an *agent* can process a partition of the search tree, e.g., a specific subtree or task. A naive static partition of the main task is not effective, since subtrees sizes may vary drastically, depending on the specific interactions of the variables analyzed. To ensure an effective load balance, we devised a dynamic rescheduling strategy that ensures the task reallocation and migration depending on the specific features of each task and agents status.

Each agent is implemented by two threads: a *worker* and a *scheduler*.

- The worker performs the actual constraint solving activities; it owns a queue of tasks to be processed and it is also able to generate new sub-tasks derived from a partition of its current task. Each task has a *weight* information, estimating its size (the estimate is computed by combining the depth of the subtree ad the average depth of the agent during its exploration).
- The scheduler is in charge of negotiating the *tasks* to pass around and/or to obtain. Tasks are encoded with minimal information (e.g., the description of the branch that leads to that subtree) in order to minimize network usage. The scheduling is decentralized and each agent is in charge to poll the others with minimal overhead (see Algorithm 1 for details).

The parallelization approach corresponds to the traditional search-based parallelism (e.g., [8, 13]); a novelty of our approach is the use of a multi-threaded engine, which allows us to separate the scheduling activities from the constraint solving process (which allows a more elegant overlapping between scheduling and computation). Work is in progress to explore different scheduling strategies.

5 Experimental tests

Table 1 summarizes a test set for FIASCO against two proteins: 3L2A (129 amino acids)–associated to the Ebola virus–and 3EMN (283 amino acids)–derived from

the study of the inner ear of the Xenopous laevis. The experimental tests have been performed on a parallel machine composed of 24 nodes, each composed of 2 quad core CPUs (Intel Xeon E5335), 288GB of RAM, and an Infiniband network. This test set is performed using 8 concurrent agents.

For every referred protein, we perform different computations, according to the number of *special* fragments ($\mathcal{F}_{spc}$) used to build the final conformation. We perform an exhaustive search with time limit set to 2 hours. The solutions reported correspond to the best conformation found (in terms of energy minimization) within the time limit. Note that the pair constraint has not been enabled for the above computations. Moreover, we perform additional tests targeted to minimizing the RMSD values. The best evaluation (w.r.t. the native, known, structure) are marked in the table with the "*" symbol.

Prot ID	$\lvert\mathcal{F}_{spc}\rvert$	Energy	RMSD	T(s)
3L2A	2	-697.5	11.1	478
3L2A*	2	-670.5	5.7	478
3L2A	3	-259.8	12.0	$2h$
3L2A*	3	-200.7	3.8	$2h$
3L2A	4	-333.2	13.6	$2h$
3L2A*	4	-241.3	7.6	$2h$
3EMN	3	-1402.3	15.5	$2h$
3EMN*	3	-1150.3	3.1	$2h$

Table 1. Computational results

As expected, increasing the number of gaps (part of the protein not covered by *special* fragments) the computational complexity arises. The gaps, indeed, are modeled through shorter fragments (of length 4) imported by the *Assembly DB*, that implicitly define a higher number in the degrees of freedom in those protein regions. The experiments underline that the way of partitioning the target sequence to produce special fragments plays an important role in this respect. It is our intention to automatize the system to produce/suggest the splitting site for possible special fragments candidates. This suggestion system would help in producing conformations that are structurally closer to the protein native state, yet, possibly, reducing the computational workload by avoiding some unnecessary degrees of freedom, in those area of the protein in which homology information can be used to generate special fragment candidates.

It is interesting to note that, the set of conformations that could potentially be generated by FIASCO, contains solutions close enough to the native conformations. The RMSD minimization tests ensure the feasibility of the methods presented, showing that a good fragment set, able to produce satisfactory candidates, can be generated.

Note that the energy function used in this test set is optimized for proteins that are completely immersed in fluid. In this respect we wish to stress that 3EMN is is a channel protein (beta-barrel eukaryotic membrane protein), thus it does not fall in the above category.

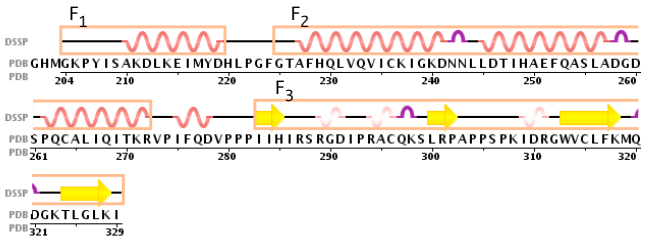


Fig. 3. 3L2A (3): secondary structure prediction and *special* fragments (red boxes)

6 Conclusions

In this paper, we provided a very brief overview of a novel experimental platform for the investigation of protein structures. The platform is based on the use of constraint solving techniques to implement a fragment assembly approach to prediction of tertiary protein structures. The fragments include statistically-ranked peptides extracted from the Protein Data Bank as well as longer fragments obtained through user observations, homology studies, or secondary structure predictions. The assembly process is modeled as resolution of different types of constraints, capturing the known fragments, estimated relative positions, geometric restrictions, and energetic considerations derived from a novel energy model. Parallelism and novel search strategies are employed to guarantee an efficient exploration of the search space of possible conformations.

The framework is currently under development—though several of its core components have already been completed. It is accessible through a web-based interface which integrates a sophisticated GUI, that allows the user to graphically provide the desired input (e.g., user defined fragments). We are currently completing the implementation work and applying it to study two sets of unknown proteins—one associated to the Ebola virus and one derived from the study of the inner ear of the Xenopous laevis.

Acknowledgments. This research is partially supported by NSF grants 0947465, 0812267, and 0420407, by a grant from AHPCRC, by INdAM GNCS 2011, and by PRIN 20089M932N.

References

[1] R. Backofen and S. Will. A Constraint-Based Approach to Fast and Exact Structure Prediction in 3-Dimensional Protein Models. *Constraints* 11(1):5–30, 2006.

[2] P. Barahona and L. Krippahl. Constraint Programming in Structural Bioinformatics. *Constraints* 13(1–2):3–20, 2008.

[3] A. Dal Palù, A. Dovier, and F. Fogolari. Constraint logic programming approach to protein structure prediction. *BMC Bioinformatics*, 5(186), 2004.

[4] A. Dal Palù, A. Dovier, F. Fogolari, and E. Pontelli. CLP-based protein fragment assembly. *Theory and Practice of Logic Programming*, 10(4–6):709–724, 2010.

[5] A. Dal Palù, A. Dovier, F. Fogolari, and E. Pontelli. Exploring Protein Fragment Assembly Using CLP. *Proc. of IJCAI*, pp. 2590–2595, AAAI Press, 2011.
[6] 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* 37(13):1405–1449, 2007.
[7] A. Dal Palù, A. Dovier, and E. Pontelli. Computing Approximate Solutions of the Protein Structure Determination Problem using Global Constraints on Discrete Crystal Lattices. *Int. Journal of Data Mining and Bioinformatics* 4(1):1–20, 2010.
[8] G. Gupta, E. Pontelli, M. Carlsson, M. Hermenegildo, K. Ali. Parallel Execution of Prolog Programs: a Survey. ACM TOPLAS, 23(4), 2001.
[9] T. X. Hoang, A. Trovato, F. Seno, J. R. Banavar, and A. Maritan. Geometry and symmetry presculpt the free-energy landscape of proteins. *PNAS*, 101(21):7960–7964, 2004.
[10] R.C.G. Holland; T. Down; M. Pocock; A. Prlic; D. Huen; K. James; S. Foisy; A. Drger; A. Yates; M. Heuer; M.J. Schreiber. *BioJava: an open-source framework for bioinformatics*. Bioinformatics 2008.
[11] J. Lee, D. Lee, H. Park, E.A. Coutsias, and C. Seok. Protein Loop Modeling by Using Fragment Assembly and Analytical Loop Closure. *Proteins*, 78(16):3428–3436, 2010.
[12] S. Raman, R. Vernon, J. Thompson, et al. Structure prediction for CASP8 with all-atom refinement using Rosetta. *Proteins* 77(S9):89–99, 2009.
[13] C. Schulte. Parallel Search Made Simple. *Techniques for Implementing Constraint Programming Systems Workshop*, 2000.

Algorithm 1 A general overview of the scheduler algorithm.

```
1: while ¬ global termination do
2:    decode incoming message (if any) according to their msg.type
3:    if timer elapsed then
4:       send msg.type=statinfo to next agent in the ring
5:    end if
6:    if worker.status=busy then
7:       process incoming message according to msg.type
8:    else
9:       if termination conditions then
10:         send msg.type=termination to next agent in the ring
11:      end if
12:      process incoming message according to msg.type
13:   end if
14: end while
15: Collect solutions
16: return best-k solution(s)
```

The procedure handles incoming communications asynchronously, and processes them according to the type of message and the actual status of the worker (idle/busy). Termination is guaranteed by a suitably modified *Dijkstra termination detection* algorithm in a token-ring fashion. Upon termination, the $k-$best solutions found by every agent are exchanged among agents, and processed, in turn, to return the global k-best solutions.

Improving Multiple Sequence Alignments with Constraint Programming and Local Search

Marco Correia, Fábio Madeira, Pedro Barahona, and Ludwig Krippahl

CENTRIA-DI {mc,fmmm,pb,ludi}@di.fct.unl.pt

Abstract. Sequence alignment is a core problem in Bioinformatics, and multiple sequence alignments (MSA) are an important tool for phylogenetics, motif and domain identification, physiological studies and even protein structure and interaction, since MSA provide information on the coevolution of amino acid residues. However, the complexity of simultaneously aligning multiple sequences is exponential on the number of sequences, and so MSA must be computed using heuristics that cut through this large search space, compromising the quality of the result and limiting the scoring functions that can be used. In this paper, we propose a constraint programming (CP) and local search based method for repairing MSA obtained with classical algorithms in order to improve the alignments and to allow greater flexibility in the scoring functions.

1 Introduction

The sequence alignment problem is at the origin of bioinformatics [1, 2] and is still of central importance. As the growth of sequence databases made more demands on the organization of these data, alignment algorithms that speed up the search by pruning the search space with heuristics became the norm [3, 4], dominating the field over the initial, dynamic-programming, approaches. This is even more evident in multiple sequence alignment (MSA), which is the harder problem of aligning, simultaneously, a large number of sequences.

A MSA can be understood as a graph where the symbols are joined by edges representing the correspondences in the alignment, with a complete n-partite graph with edges joining all elements of each sequence to all elements of every other sequence representing the set of all possible MSA [5, 6]. Alternatively, a MSA can be seen as a matrix where each row represents a sequence and each column represents a set of aligned elements from all sequences. In this case, gaps are added to the alignment matrix to adjust the position of the elements (gaps are not part of the sequences themselves). The ultimate goal of the MSA is to show the evolutionary relations between the sequences, indicating which parts of the molecules descend from common ancestors, and which were added or deleted during evolution.

Since the classical dynamic programming methods are exponential in the number of sequences, the practical solution is to limit the search. The Clustal family of MSA algorithms [7], for example, aligns all pairs of sequences and

then builds the MSA by progressively aligning the closest sequences with the consensus sequence obtained from previous alignments. With no possibility of backtracking, this greedy optimization is likely to stop at local optima, resulting in some misalignment. Furthermore, the scoring function cannot consider the complete MSA because the MSA is being built progressively. Some alternatives (e.g. MUSCLE [8], MAFFT [9] and PROBCONS [10]) can redo previous alignments, repeatedly divide an alignment into two groups of aligned sequences and then realigning the groups. Aside from these progressive methods in the Clustal family (including T-Coffee[11], which uses a similar approach), there are alternatives based on probabilistic models (e.g. hidden Markov models (HMM) [12]) and genetic algorithms (e.g. SAGA [13]). HMM methods are based on a variant of the character frequency profile matrices, taking into account position-specific insertion and deletion (indel) probabilities. GA methods stochastically combine and mutate candidate alignments through a directed evolutionary process by providing a measure of fitness for individual alignments within the population, but this is generally too slow for real applications [13].

However, in all cases there is the need to avoid most of the search space, compromising the quality of the final results. Also, the most used approaches are restrictive in the scoring functions that can be used. For instance, one problem of particular interest to us is the detection of protein coevolution by the correlation of mutations in different positions. A MSA can show these correlations, indicating that those residues coevolved because of an important interaction. This can have structural implications and is useful in predicting protein structure and interaction. However, typically the MSA alignment scores assume that all mutations are independent, an assumption that is false whenever there is coevolution, and, in general, the MSA algorithms depend on such assumptions. For example, it would be infeasible to simply add this score to a progressive algorithm like ClustalW, which assembles the MSA one sequence at a time, since we need the alignment to estimate the correlation.

Evidence for the need to improve the MSA generated automatically can be found in the manual fixing of misalignments ("by eye") often reported in the literature, where researchers adjust the MSA according to their own criteria. Manually refined alignments are generally considered superior to purely automated methods [14], taking into account structural and functional factors. The assessment of MSA generally include the effectiveness of a particular heuristic for the optimization of the scoring function and the accuracy relative to reference alignments [15]. One of the best databases of manually refined and curated MSA, specifically designed for the evaluation and comparison of MSA software, is BAliBASE [16]. BAliBASE provides manually refined alignments based on 3D structural superpositions and implements two different alignment scores. The sum-of-pairs (SP) score which is the percentage of correctly aligned pairs of residues in the test alignment, relative to the reference alignment, to determine the success in aligning some, if not all, of the sequences in an alignment. And the column score (CS), the percentage of correctly aligned columns, which tests

the ability of the programs to align all of the sequences correctly at any given position.

1.1 Our proposal

Some methods have been proposed to solve alignment problems using constraint programming (CP) or related approaches, such as a CP problem in order to introduce additional constraints in sequence alignments [17], as an integer linear programming problem [18] or as a SAT problem [19]. Our proposal differs in that we take advantage of the established methods, such as ClustalW, and then repair the MSA using a CP approach, focusing on those regions that, being less conserved, are more likely to include misalignments. In this way we can narrow down the search space based on constraints such as not lowering the alignment score, we can impose additional constraints, such as those based on structure comparisons, and we can use a greater range of scoring functions than those available to progressive methods. Furthermore, this approach is closer to the established practice in the biological community of obtaining the MSA using the automated methods and then refining it according to the additional considerations. The main contribution of this paper is this framework for improving the MSA by undoing mistakes made by the greedy heuristics and allowing other scoring functions that may need to consider the MSA as a whole, such as mutual information across different positions.

2 Method

The problem of correcting a region of a MSA may be formalized as a matrix of $n \times p$ cells representing amino acid codes, where n is the number of sequences to be aligned and p are possible positions for the amino acid residues. Gaps in the sequence are represented by the special character "-" (fig.1).

S	P	V	I	-	-	-	L
R	-	-	I	-	-	-	S
S	-	-	-	-	-	-	L
F	N	T	T	Q	G	G	P
T	-	-	-	-	-	-	-
F	S	K	N	-	-	-	-
E	T	F	G	Q	-	-	-
K	S	-	-	-	-	-	T

Table 1. Multiple sequence alignment problem

Let $a_{i,j}$ represent the residue (or gap) at sequence i, position j, and $\mathbf{s}_i$ represent the sequence of amino acid residues at row i, i.e. $\mathbf{s}_i = \langle a_{i,1}, \ldots, a_{i,p} \rangle$. Let the score associated with two residues $a_{1,j}$,$a_{2,j}$ in the same position j given by a

function $\sigma_A(a_{1,j}, a_{2,j})$. From the individual amino acid scores we may compute a score for the alignment of two sequences $\mathbf{s}_1$, $\mathbf{s}_2$,

$$\sigma_S(\mathbf{s}_1, \mathbf{s}_2) = \sum_{i=1}^{p} \sigma_A(a_{1,i}, a_{2,i}) - \gamma(\mathbf{s}_1, \mathbf{s}_2) \tag{1}$$

The term $\gamma(\mathbf{s}_1, \mathbf{s}_2)$ present in the formula above is called *gap penalty* and accounts for the number of consecutive gaps in the sequences. The score of the multiple alignment is then derived from the pairwise sequence alignment scores,

$$\sigma = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \sigma_S(\mathbf{s}_i, \mathbf{s}_j)$$

The alignment correction problem is to find the alignment with the best score by changing the positions of the gaps. Note that this procedure may increase or decrease the number of consecutive gaps (for example the number of consecutive gaps in the second sequence in fig.1 may be decreased from 2 to 1 by placing "I" next to "R" or before "S").

2.1 CP Model

The alignment correction problem may be naturally modeled in Constraint Programming. A straightforward approach assigns a finite domain variable $x_{i,j} \in X$ for each cell in the matrix, where its domain $D(x_{i,j}) = \{'-', 'A', 'C', 'D', \ldots\}$ is the set of all possible amino acids plus the gap. Each sequence $\mathbf{s}_i$ may be obtained by changing the position of the gaps. This constraint, which we call validSequence, may be modeled by the inTable constraint [20] as follows,

$$\text{validSequence}(\mathbf{s}_i) = \text{inTable}(\langle x_{i,1}, \ldots, x_{i,p} \rangle, T_i) \tag{2}$$

Each table T_i is created so that each row is obtained by placing the gaps in a distinct position. The number of rows in table T_i is therefore $C_{g_i}^{p}$ where g_i is the number of gaps in sequence $\mathbf{s}_i$. Since the complexity of the propagation algorithm for this constraint is linear on the size of the table this method works only for small p. Fortunately, we may propagate this constraint using an algorithm that is not exponential in p as follows.

For each sequence $\mathbf{s}_i$ we introduce a set X_i^G of auxiliary finite domain variables, where a variable $x_{i,k}^G \in X_i^G$ models the position of the k'th gap in the sequence $\mathbf{s}_i$, and hence $1 \leq k \leq g_i$. Similarly, we introduce a second set X_i^A of auxiliary finite domain variables, where a variable $x_{i,k}^A \in X_i^A$ represents the position of the k'th amino acid residue in the sequence $\mathbf{s}_i$, which we denote as $\mathbf{s}_i(k)$, with $1 \leq k \leq p - g_i$. Note that these two sets are strictly sorted, and partition the set $\{1, \ldots, p\}$. The validSequence constraint is therefore modeled as follows,

$$\begin{aligned}\textsc{validSequence}\left(\mathbf{s}_i\right) = & \forall_{2\leq k\leq g_i} x^G_{k-1} < x^G_k \\ & \wedge \forall_{2\leq k\leq p-g_i} x^A_{k-1} < x^A_k \\ & \wedge \textsc{distinct}\left(X^G_i \cup X^A_i\right) \\ & \wedge \forall_{1\leq k\leq g} X_i\left[x^G_{i,k}\right] =' -' \\ & \wedge \forall_{1\leq k\leq <p-g} X_i\left[x^A_{i,k}\right] = \mathbf{s}_i\left(k\right) \end{aligned} \tag{3}$$

For modeling the objective function we use formula 1, and apply function σ_A over all pairs of finite domain variables $x_{1,j}$, $x_{2,j}$ in the same column. This may be accomplished by means of a TABLE or ELEMENT constraint over the table of amino acid affinities. Finally, the gap penalty term $\gamma\left(\mathbf{s}_1, \mathbf{s}_2\right)$ may be integrated also using different models. If the VALIDSEQUENCE constraint is implemented by equation 2, then we can create an extra column in each row of a table T_i to specify the number of consecutive gaps corresponding to the given sequence, and project this number to a new finite domain variable c_i by changing equation 2 to

$$\textsc{validSequence}(\mathbf{s}_i)=\textsc{inTable}\left(\left\langle x_{i,1}, \ldots, x_{i,p}, c_i\right\rangle, T_i\right)$$

When using equation 3 to model the VALIDSEQUENCE constraint, c_i may be obtained using reification and a SUM constraint as follows,

$$c_i = \sum_{k=2}^{g_i} \left[x^G_{i,k} > x^G_{i,k-1} + 1\right]$$

2.2 Search

We used a greedy variable and value heuristics for directing search quickly towards a good solution. They are defined as

$$\begin{aligned}\textsc{var}\left(X\right) &= \arg\max_{x_{i,j}} \max_{v_1, v_2 \in D(x_{i,j})} q\left(x_{i,j}, v_1\right) - q\left(x_{i,j}, v_2\right) \\ \textsc{val}(x_{i,j}) &= \arg\max_{v\in D(x_{i,j})} q\left(x_{i,j}, v\right)\end{aligned}$$

where $q\left(x_{i,j}, v\right)$ is a function which estimates the cost of assigning value v to variable $x_{i,j}$,

$$q\left(x_{i,j}, v\right) = q^A\left(x_{i,j}, v\right) + nq^G\left(x_{i,j}, v\right) \tag{4}$$

and is composed of two terms. The first estimates the cost of this assignment based on the set of amino acid residues already placed in column j, using function σ_A,

$$q^A\left(x_{i,j}, v\right) = \sum_{k=1}^{n} \begin{cases} \sigma_A\left(x_{k,j}, v\right) & \Leftarrow \left|D\left(x_{k,j}\right)\right| = 1 \\ 0 & \Leftarrow \text{otherwise} \end{cases}$$

The second term estimates the impact that this assignment will have on the number of consecutive gaps and is -10 if it creates a new gap, 10 if it does not open a new gap, and 0 if it is not known.

The above heuristics were used to drive limited discrepancy search [21]. The allowed discrepancy begins at 1 and is iteratively increased each time the search space is exhausted so that completeness is still guaranteed. Additionally, since the search space appears to have some steep local optima, we introduced a small random perturbation term in formula 4, and restarted the solver after a geometrically increasing slice of time.

2.3 Local Search

The same model was tested in constrained local search (COMET), with a greedy hill-climbing heuristic optimizing the same score using the Gonnet substitution matrix (the objective function) starting from a randomized transformation of the MSA blocks. With a few exceptions the improvements on the score were not as good as those obtained with the CP model and heuristics, but (like for CP) better heuristics and meta-heuristics should improve the obtained scores. These results should thus be regarded as initial, and subject to further work.

2.4 Experiments

To test our approach, we measured how much we could improve the standard ClustalW MSA, both according to a scoring function and by comparing the alignments to the reference alignments in BALiBASE. The procedure was thus to start from the same set of sequences as those in a BALiBASE reference alignment, obtain the ClustalW MSA and attempt to improve the alignment on contiguous blocks of columns containing a mix of gaps and residues. This ruled out regions that are well conserved and thus more likely to be correctly aligned, and allowed us to focus on the regions of the MSA that could be adjusted by shifting the gap positions. Our scoring function to improve the alignments was the standard Gonnet substitution matrix [22], and the ClustalW alignments were calculated with this matrix and the default gap penalties. The next step was to apply the CP and Local Search algorithms to improving the alignment score. However, one problem is that ClustalW uses a highly optimized scoring function that, though based on the same substitution matrix we used, adjusts the relative weights given to each sequence and also the gap penalties depending on the neighboring residues [23]. This means that our improvement could come either from actually correcting errors in the ClustalW alignment or due simply to the slight differences in the scoring function.

To test this, we compared the ClustalW and corrected alignments with the BALiBASE set of highly accurate alignments, manually curated by experts and determined not only from sequence data but also from structural information. Given that ClustalW uses a more sophisticated scoring function than our current implementation, and since neither ClustalW nor our implementation is using structural data or other information available to the experts that created the BALiBASE alignments, if it were the case that our score improvements were only due to a difference in the scoring functions we would expect our corrections not

to improve the ClustalW alignments when compared to the BALiBASE alignments. In contrast, if, even with a simpler scoring function, our implementation was improving the alignments by making them more similar to the BALiBASE benchmark, this would mean that our approach was really correcting mistakes made by ClustalW. We applied this procedure to 22 alignments for the CP implementation and 24 for the local search implementation (a few were rejected because they provided less than 10 columns for adjustment over the whole alignment). Running times ranged from a few seconds per alignment in the local search implementation to several minutes for the CP implementation, which was set to time out at two minutes for each block.

3 Results and Discussion

For the CP implementation, with the Gonnet substitution matrix score the average improvement for each column changed in the alignments (up to 233 columns, in one case, but averaging 54 columns for the set of 22 MSAs) was slightly above the average score attributed to the match of identical residues (106% of the average identity match score in the substitution table). This means that our improvement was the equivalent of gaining one additional identity match for each column changed. In the comparison with the reference alignments using the BALiBASE sum-of-pairs score, our implementation improved 77% (17 out of 22) of the alignments tested tests using CP. This suggests that, even with a simpler scoring function and no additional information such as structural or functional data, the CP implementation improves the ClustalW alignment. The average improvement for all 22 alignments was 11% in the BALiBASE sum-of-pairs score. The local search implementation did not perform as well, with an insignificant average improvement when compared to BALiBASE alignments and improving only 14 out of 24 alignments. Nevertheless, our goal in this paper was to show that improving MSA by correcting mistakes in less conserved regions is a promising approach. At this stage CP seems to give better results, but there is still work to be done optimizing the scoring functions and heuristics, and the local search implementation takes only milliseconds to find solutions, against several minutes for CP, so there is much room for improvement. In addition, this gives us a flexible framework for using different scoring functions, not limited to the peculiarities of the underlying alignment algorithm. Of special interest is the inclusion of structural information, whether from determined structures or prediction algorithms, and also scoring functions adapted to coevolution studies, where the assumption of independent mutations does not hold.

Acknowledgements

This work was funded by Fundação para a Ciência e Tecnologia, MCTES, under project PTDC/EIA-CCO/115999/2009.

References

[1] Needleman SB and Wunsch CD. A general method applicable to the search for similarities in [...] proteins. J. Mol. Biology 48 (3): 443–53 (1970)

[2] Smith TF, Waterman MS.: Identification of Common Molecular Subsequences. J. Mol. Biol. 147, 195–197 (1981)

[3] Lipman, DJ; Pearson, WR. Rapid and sensitive protein similarity searches. Science 227 (4693): 1435–41 (1985)

[4] Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. Basic local alignment search tool. J Mol Biol 215 (3): 403–410 (1990)

[5] Kececioglu, JD. The maximum weight trace problem in multiple sequence alignment. In Apostolico et al (eds.), Proc. 4th Symp. Comb. Patt. Matching, pp 106–119 (1993).

[6] Backofen R., Gilbert D., Bioinformatics and Constraints. In: Rossi F., van Beek, P., Walsh T. (eds.) Handbook of Constraint Programming, Elsevier 2006.

[7] Chenna R, Sugawara H, Koike T et al. Multiple sequence alignment with the Clustal series of programs. Nucleic Acids Res 31 (13): 3497–3500 (2003)

[8] Edgar, R.C. MUSCLE: multiple sequence alignment with high accuracy and high throughput.Nucleic Acids Res. 32(5):1792-1797 (2004)

[9] Katoh K, Toh H. Recent developments in the MAFFT multiple sequence alignment program. Brief Bioinform 9: 286–298 (2008)

[10] Do CB, Mahabhashyam MS, Brudno M, Batzoglou S. ProbCons: Probabilistic consistency-based multiple sequence alignment. Genome Res 15: 330–340 (2005)

[11] Notredame C, Higgins DG, Heringa J. T-Coffee: A novel method for fast and accurate multiple sequence alignment. J. of mol. biol., 302(1), 205-17. (2000)

[12] Eddy, S. R. Profile hidden Markov models. Bioinformatics, 14(9), 755. Oxford Univ Press (1998)

[13] SAGA: Sequence Alignment by Genetic Algorithm, C. Notredame, D.G. Higgins, Nucleic Acid Research, Vol. 24, 1515-1524, (1996)

[14] Edgar, R. C., Batzoglou, S. Multiple sequence alignment. Current opinion in structural biology, 16(3), 368-73 (2006)

[15] Do, CB, Katoh K, Protein Multiple Sequence Alignment, Functional Proteomics Methods in Molecular Biology, 2008, Volume 484, IV, 379-413

[16] Thompson, J. D., Plewniak, F., Poch, O. BAliBASE: a benchmark alignment database [...]. Bioinformatics (Oxford, England), 15(1), 87-8 (1999)

[17] Will S., Bush A., Backofen R. Efficient Sequence Alignment with Side-Constraints by Cluster Tree Elimination. Constraints 13(1-2): 110-129 (2008)

[18] Reinert K, Lenhof HP, Mutzel P, Mehlhorn K, Kececioglu JD, A Branch-and-Cut Algorithm for Multiple Sequence Alignment, In Proc. of the 1st RECOMB (1997)

[19] Prestwich S, Higgins D. A SAT-Based Approach to Multiple Sequence Alignment. In 9th Int. Conf. Princ. and Pract. of CP, pp. 940–944 (2003)

[20] Bessière C and Régin JC. Arc Consistency for General Constraint Networks: Preliminary Results. IJCAI'97 pp 398—404 (1997)

[21] Harvey WD, Matthew LG. Limited Discrepancy Search, In C. S. Mellish (ed) Proceedings of IJCAI'95; Vol. 1, pages 607–615 (1995)

[22] Gonnet GH, Cohen MA, Benner SA. Exhaustive matching of the entire protein sequence database. Science, 256(5062):1443-5. (1992)

[23] Thompson JD, Higgins DG, Gibson TJ. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment [...]. Nucleic acids research, Vol. 22, No. 22, pp. 4673-4680. (1994)

Petri Nets for Integrated Models of Metabolic and Gene Regulatory Networks

Aljoscha Palinkas[1] and Alexander Bockmayr[2]

[1] DFG Graduiertenkolleg Computational Systems Biology,
[2] DFG Research Center Matheon,
Freie Universität Berlin, Arnimallee 6, D-14195 Berlin, Germany
Aljoscha.Palinkas@fu-berlin.de, Alexander.Bockmayr@fu-berlin.de

Abstract. Metabolic reactions and gene regulation are two closely related cellular processes. In integrated models, both are considered together. Metabolic and gene regulatory networks in isolation can be described by Petri nets. Although an integrated Petri net model for tryptophan synthesis has been proposed in the literature, a systematic and general method is not available so far. The goal of this paper is to present such a method, assuming that the stoichiometry of the metabolic reactions is known, the gene regulatory network is described in the Thomas formalism, and the interactions with the metabolism are given by logical conditions. For the gene regulatory network, the resulting Petri net shows exactly the asynchronous unitary dynamics given by the Thomas framework.

1 Introduction

Metabolism and gene regulation are closely related processes in the cell (see e.g. [1]). On the one hand, genes regulate the synthesis of enzymes that catalyze metabolic reactions. On the other hand, metabolites may modulate directly or indirectly gene expression and activities.

In mathematical modeling, metabolism and gene regulation are often treated separately. The reason is that methods available for the isolated systems usually cannot be directly extended to integrated models. On the continuous side, ordinary differential equation models face the problem of different time scales. On the discrete side, various methods have been proposed that extend flux balance analysis (FBA) for metabolic networks to incorporate effects from gene regulation (rFBA [2], iFBA [3], idFBA [4], SR-FBA [5]). However, these methods heavily depend on the steady-state assumption of FBA.

Among the different formalisms, Petri nets seem suitable for integrated models since they can naturally represent not only metabolic networks (e.g. [6, 7]), but also dynamic models of gene regulatory networks (GRNs) [8] based on the well-known Thomas formalism [9]. Siñao et al. [10] built an integrated Petri net model of tryptophan synthesis. They used a systematic translation of GRNs to Petri nets proposed by Chaouiya et al. [8], where the GRN was assumed to

be represented in a formalism called regulatory graphs. Next, biological knowledge was applied in order to add appropriate interactions with the metabolic part. However, no systematic method was given that would allow translating integrated models in general.

In this paper, we present a new translation of GRNs to Petri nets. It is based on the more common description of GRNs by Boolean expressions, as it has been used for example for integrated models of *E. coli* [11], *S. cerevisiae* [12], or *B. subtilis* [13]. Our translation has the advantage that the resulting Petri net is minimal and that it can easily be extended to translate integrated models which include also a metabolic part. The resulting Petri nets, however, are of the same kind, i.e., if a GRN is given in both descriptions, then our translation of the Boolean expressions would result in the same Petri net as the translation from the regulatory graph following [8] (minor differences that do not affect the behaviour are possible).

Regarding the size of the models, the Petri net of the GRN grows exponentially in the number of interactions between the genes. But, this number is usually limited and in most cases one gene from the GRN can be represented by less than 10 nodes in the Petri net. A more serious issue is the size of the reachability graph of the Petri net, which represents all the possible dynamics and is essential for most kinds of analysis, e.g. model checking. Its size can grow exponentially in the number of nodes as well as tokens.

So far, we used our algorithm only for rather small models, where the GRNs did not have more than 8 genes. In these cases, it was most convenient to pursue the translation by hand. A computer implementation for larger models is currently under development.

2 Preliminaries on Petri Nets

We start with a short introduction into the Petri net formalism (for additional details, see e.g. [14, 7]). The j-th column of a matrix A will be denoted by a^j, the i-th row by a_i, and the entry belonging to both by a_i^j. With $A^\top$ we denote the transpose of A.

A *Petri net* is a bipartite directed graph with $\mathbb{N}$-weighted edges. There are two disjoint sets of nodes $\{p(1), \ldots, p(k)\}$ and $\{t(1), \ldots, t(h)\}$, called *places* and *transitions* respectively. Directed edges connect places and transitions, but never two nodes of the same kind. Furthermore the places can hold tokens. The dynamics of a Petri net consists of tokens that are moved from one place to another via the transitions and along the directed edges.

The formal definition of the directed graph, called *Petri net structure*, is usually given by two $(k \times h)$-matrices, the *input matrix* R and the *output matrix* Q. In both matrices, the rows represent the places, the columns the transitions, and the entries the weight of the edges. In the input matrix R, edges lead from places to transitions, while in the output matrix Q, edges lead from transitions to places. Zero entries indicate missing edges (see Fig. 1 and Fig. 2).

The *marking* of the places with tokens can be described by a k-dimensional vector $m \in \mathbb{N}^k$. It changes when a transition *fires*, i.e., the transition consumes tokens from the places along the inward edges and produces tokens along the outward edges, according to the numbers given by the edge weights. If tokens that are to be consumed are lacking on the corresponding places, the firing cannot take place. Formally, a transition $t(j)$ is *enabled* if $m \geq r^j$ componentwise. If a multiset of transitions is fired, this multiset can be represented by a *Parikh-vector* $x \in \mathbb{N}^h$, counting the number of occurrences of each transition. Firing of this multiset leads to the new marking $m' = m + (R - Q)x$. The matrix $D := R - Q \in \mathbb{N}^{k \times h}$ is called the *incidence matrix*, which alone suffices to compute the marking m'. However, the input matrix R is still needed to determine which transitions are enabled. For our translation method, we will describe how to obtain the columns of the matrices R and D.

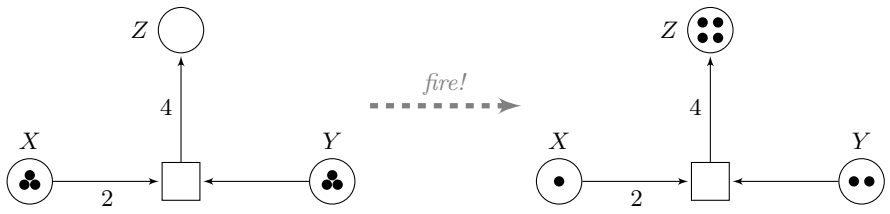


Fig. 1. Example for the firing of a transition. The matrices are $R = (2, 1, 0)^\top$ and $Q = (0, 0, 4)^\top$. After one firing the transition is not enabled anymore, since the new marking is $m' = m + d^1 = (3, 3, 0)^\top + (-2, -1, 4)^\top = (1, 2, 4)^\top \not\geq r^1 = (2, 1, 0)^\top$.

We need two additional notions [14, 8] that will play a central role further on. A *test-edge* is an abbreviation for a pair of equally weighted edges that connect the same two nodes in opposite directions. Test-edges implement further constraints on the dynamics because they add conditions for transitions to be enabled. In the incidence matrix they are invisible.

Complementary places is the name for a pair of places such that every token leaving one place moves to the other one (see Fig. 4). In other words, in the incidence matrix, the row for the first place is the negative of the row for the second place. The sum *Max* of the tokens on complementary places thus remains constant. Complementary places are needed to test for the absence of tokens: if there are more than k tokens on one place it follows that the other place has less than $Max - k$. We use this to represent inhibition in the GRN.

For the translation of a metabolic network, we only need the stoichiometric matrix, which will be interpreted as the incidence matrix D of the Petri net. The input and output matrix can then be obtained as $R = \frac{|D|-D}{2}$ and $Q = \frac{|D|+D}{2}$ ($|\cdot|$ is applied componentwise). While this is not true in general, it holds here because a metabolic Petri net does not contain test-edges. We will come back to this in Sect. 5 when dealing with integrated models where some reactions are

constrained by gene regulation. Note that this translation only works if reversible reactions are split into a forward and a backward reaction, see Fig. 2.

In the graphical representation, places correspond to circles, with small dots inside indicating the tokens. Transitions correspond to rectangles. Edges have the weight 1 unless another weight is indicated.

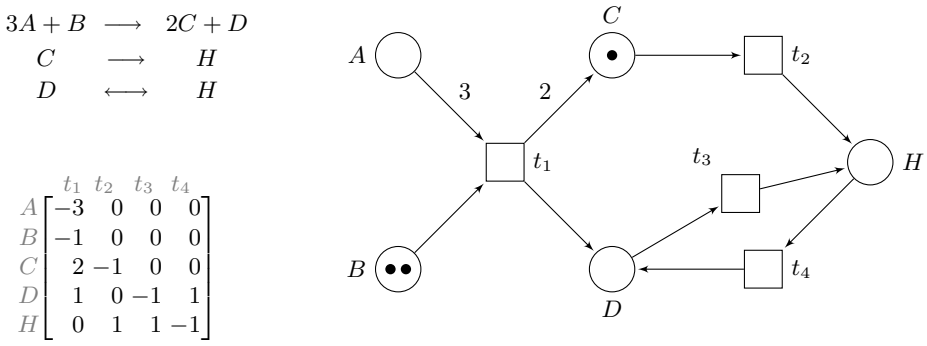


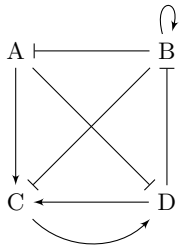
Fig. 2. Example of a metabolic network with three reactions. In the stoichiometric matrix, the reversible reaction is already split up, so we can directly interpret it as the incidence matrix of a Petri net structure (shown on the right).

3 Logical Networks

Gene regulatory networks are modeled as logical networks in the sense of the Thomas formalism [9]. Basically these can be described by regulatory components $1, \ldots, k$, which represent the genes. Each component i can take as value an integer from 0 to Max_i. An assignment of values from these ranges to all components is called a *state* of the network and the set $Z := \prod_{i=1}^{k} \{0, \ldots, Max_i\}$ is the *state space*.

To define the dynamical behavior of the network, for each component a *target function* $f_i \colon Z \to \{0, \ldots, Max_i\}$ determines a target value for the component i when the network is in state z. We assume that the target functions f_i are given by Boolean expressions. Therefore, we first describe the states using Boolean variables. For all $i = 1, \ldots, k$ and $w = 1, \ldots, Max_i$ we define the Boolean variables $x_w^i := [z_i \geq w]$. Any state $z \in Z$ gives an assignment of these variables, which we denote by $x(z)$. To describe a particular state it is not necessary to define the assignment of all variables, because some implications hold. Consider for example a network with two components and $Max_1 = Max_2 = 2$. Then $[z = (1, 2)] \Leftrightarrow x_1^1 \wedge \bar{x}_2^1 \wedge x_2^2$, where the variable x_1^2 is left out because it would be redundant (since $x_2^2 \Rightarrow x_1^2$). If all states z such that $f_i(z) = w$ are described each by a conjunction as above and then we take the disjunction of all these conjunctions, we get a Boolean expression which we call ψ_w^i. By construction

it has the property that $[f_i(z) = w] \Leftrightarrow \psi^i_w(x(z))$. The target functions f_i can thereby be represented by the Boolean expressions ψ^i_w, for all $i = 1, \ldots, n$ and $w = 0, \ldots, Max_i$.



A	B	C	D	f_A	f_B	f_C	f_D
0	0	0	0	1	1	0	0
0	0	0	1	1	0	1	0
0	0	1	0	1	1	0	1
0	0	1	1	1	0	1	1
0	1	0	0	0	1	0	0
0	1	0	1	0	1	1	0
0	1	1	0	0	1	0	1
0	1	1	1	0	1	1	1
1	0	0	0	1	1	1	0
1	0	0	1	1	0	1	0
1	0	1	0	1	1	1	0
1	0	1	1	1	0	1	0
1	1	0	0	0	1	0	0
1	1	0	1	0	1	1	0
1	1	1	0	0	1	0	0
1	1	1	1	0	1	1	0

$$Max_{A,B,C,D} = 1$$

$$\begin{aligned} \psi_1^A &= \overline{B} \\ \psi_1^B &= B \vee \overline{D} \\ \psi_1^C &= (A \wedge \overline{B}) \vee D \\ \psi_1^D &= \overline{A} \wedge C \end{aligned}$$

Fig. 3. Example of a logical network. The target function f is given by its table of values, from which the Boolean expressions ψ can be derived as described before. They are shown here in minimised DNF. For ψ_1^C the construction gives a DNF with ten minterms, one for each 1 in the f_C-column. But, this number can be reduced to two. In this case where all components are 2-valued, we have some simplifications: first, the Boolean variables x_1^A, x_1^B, x_1^C, x_1^D can be abbreviated by A, B, C, D resp., and second, the expressions ψ_0 can be omitted because they are given by $\overline{\psi_1}$.

Given such a network, we may define different dynamics (see e.g. Richard [15]). Here, we focus on the *asynchronous unitary dynamics* [9]. In each update, one single component changes its value by ± 1 such that it approaches the target value. If no component can change its value, we are in a fixpoint and no updates are possible. This dynamics can be represented in the state transition graph.

Definition 1. *The (asynchronous unitary) state transition graph (STG) of the network with target functions f_i has as nodes the states $z \in Z$ and there is an edge from z to z' iff there is an (asynchronous unitary) update that leads from z to z'.*

4 Petri Nets of Logical Networks

Chaouiya et al. [8] presented a method to translate so-called regulatory graphs into Petri nets. We follow them in the way the resulting Petri net represents the components of the logical network. Each gene is represented by a pair of complementary places. The transitions are executing an unitary update if the component is fired, i.e., they shift one token from one place to the other, thereby changing the value of the component by ± 1. Since the change of one component

depends on the values of other components, there are test-edges connecting these elementary building blocks with each other.

Most discrete models of regulatory networks are described by Boolean expressions. In integrated models, the dependencies of reactions on gene expression can be formulated with Boolean expressions, as done e.g. in the genome-scale *E. coli* model by Covert et al. [11]. Therefore, we propose here a new translation method that is working with Boolean expressions. These have the additional advantage that their disjunctive normal form (DNF) can be minimised using e.g. the classical Quine-McCluskey method. Thus, we can get a Petri net that is minimal in the number of transitions, which is not the case in the translation of Chaouiya et al. [8]. Note that Steggles et al. [16] also used the minimisation of DNF to build minimal Petri nets of logical networks, but of a very different kind (with synchronous update).

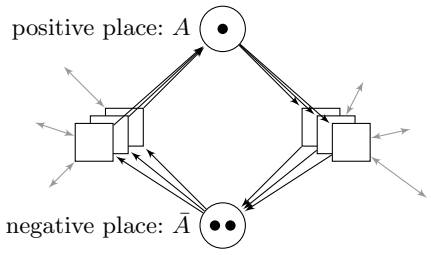


Fig. 4. Elementary building block: this pair of complementary places represents a component A of the logical network with $Max_A = 3$. Its current value is 1. In grey, the test-edges are shown that will connect the components and therewith implement the Boolean expressions defining the networks dynamics.

Suppose the logical network is described by the Boolean expressions ψ_w^i, for $i = 1, \ldots, k$ and $w = 0, \ldots, Max_i$. The target function f_i can take $Max_i + 1$ different values, but the update of component i is either an activation $(+1)$ or an inhibition (-1). Therefore, we consider the Boolean expressions

$$\Phi_{+1}^i := \bigvee_{w=1}^{Max_i} \psi_w^i \wedge \bar{x}_w^i \quad \text{and} \quad \Phi_{-1}^i := \bigvee_{w=0}^{Max_i - 1} \psi_w^i \wedge x_{w+1}^i$$

for activation and inhibition, respectively.

Next we need transitions that shift one token between the complementary places of component i (see Fig. 4) whenever the state of the network fulfills Φ_{+1}^i or Φ_{-1}^i. Evaluating these Boolean expressions will be achieved with test-edges that can test each of the Boolean variables x_w^i.

In order to have at most one transition enabled at each component (Fig. 4) and to obtain a minimal number of transitions, we apply some equivalence transformations. After, each conjunction (also called minterm) is translated into one

transition according to the pseudocode below. The numbering of the places starts with all positive places. The negative complementary places follow such that $p(k+j)$ is the negative complement of $p(j)$, for all $j = 1, \ldots, k$. For this algorithm it is convenient to construct the incidence matrix D and the input matrix R. As explained before, this pair gives a complete description of the Petri net structure. In D, test-edges are not visible, so we only have to encode the actual shifting of tokens which, in our case, is just one token that is shifted between the negative and positive place of one component. All other edges are test-edges. For these we have to translate every minterm of the DNF Φ into one transition and every literal of this minterm into one test-edge, connecting the transition with a place of another component. The test-edges at one transition combine as a conjunction of the literals, since the transition is only enabled if all test-edges return a positive result. Implementing a separate transition for each minterm yields the disjunction of these, since every transition can be enabled independently of the others. Negative literals $\bar{x}_w^j$ are tested by checking if there are more than $Max_j - w$ tokens on the negative place of component j, which would imply that the value of j is less than w.

Algorithm 1:

```
n ⟵ 1
for i ← 1 to k do
    for v ∈ {−1, +1} do
        for κ conjunction of Φ_v^i do
            create the 0-vectors r^n, d^n ∈ ℕ^{2k}
            d_i^n ⟵ v                  // edges that actually shift tokens
            d_{k+i}^n ⟵ −v                          // to change the value
            for y literal of κ do
                if y is positive literal x_w^j then
                    r_j^n ⟵ w              // test-edges to positive, and
                if y is negative literal x̄_w^j then
                    r_{k+j}^n ⟵ Max_j − w + 1       // to negative places
            n ⟵ n + 1
```

Dynamics of the Resulting Petri Net

Several transitions may be enabled at one moment, but only one transition can fire in each step. After the firing, some transitions might not be enabled anymore. Similar to the STG of logical networks, we can represent all possible dynamics in a directed graph. The Petri net consists solely of elementary building blocks (Fig. 4). When component i of the logical network has maximal value Max_i, then the tokens in the corresponding elementary building block should sum up to Max_i. Any marking that fulfills this is called *valid*.

Definition 2. *The global reachability graph of the constructed Petri net has as nodes all the valid markings and an edge from m to m' iff at marking m there is an enabled transition $t(j)$ such that firing leads to m' (formally: $m \geq r^j$ and $m + d^j = m'$, where r^j,d^j are the j-th column of the input and output matrix, respectively).*

Analysing the full translation algorithm shows that the firing of any transition leads to an asynchronous unitary update and that a transition is only enabled if the target function implies this update. Conversely, for every update that is prescribed by the target function, there is a transition that implements it. This can be formally verified and leads to the following statement.

Theorem 1. *The asynchronous unitary STG of a logical network and the global reachability graph of the Petri net obtained by Algorithm 1 are isomorphic as directed graphs.*

5 Integration of Metabolism and Gene Regulation

We now extend Algorithm 1 to integrated models. We start with a formal definition that includes all information necessary for the translation.

Definition 3. *An* integrated model *consists of metabolites $M(1), \ldots, M(l)$, reactions $R(1), \ldots, R(h)$, and genes $G(1), \ldots, G(k)$. It is described by:*

1. *a GRN with all genes and some metabolites as components. The target functions are given by Boolean expressions ψ_w^i for $i = 1, \ldots, N$ and $w = 0, \ldots, Max_i$ in the Boolean variables $x_w^j := [z_j \geq w]$.*
2. *the stoichiometric matrix S of the metabolic network.*
3. *for reaction $R(i)$ there is a Boolean expression ϱ_i in the variables x_w^j, defining the conditions under which the reaction is enabled.*

Some models for rFBA have exactly this form, see e.g. [11]. The Boolean expressions ϱ_i reflect the knowledge about the enzymes that catalyse a reaction and the genes that code for these enzymes. If information is incomplete or if we do not want to include the regulating genes in our model, we set $\varrho_i =$TRUE, thus leaving the reaction unconstrained.

There are now two kinds of components: genes and metabolites. The classical Petri net of metabolism as in Fig. 2 is not sufficient here, since metabolites can have regulatory effects. To implement regulation in the GRN, we always have to test for the absence of tokens. This can only be done with complementary places as explained in Sect. 2. Therefore, both kinds of components will be implemented as pairs of complementary places. For gene-components, the tokens on the positive place represent the level of gene expression. For the metabolic components, they represent the concentration.

For example, in the model of tryptophan synthesis by Siˆmao et al. [10], the genes of the operon trpEDCBA are expressed whenever neither tryptophan (TRP) nor transcription factor from the gene trpR are abundant. If all

components are Boolean ($Max_i = 1$) we can formulate this as $\psi_1^{trpEDCBA} = \overline{TRP} \wedge \overline{trpR}$.

For the logical network in isolation, we first set the places and then just apply Algorithm 1 to add transitions and edges. This implements the unitary asynchronous dynamics defined by the Boolean expressions. The procedure for the integrated model is given by the following extension:

1. A pair of complementary places is implemented for each component, whether gene or metabolite.
2. Algorithm 1 is applied to implement all regulatory dynamics.
3. The Boolean expressions ϱ_i are translated into transitions, the stoichiometry defines edges to other metabolites, whereas the conditions from ϱ_i implement test-edges to places of genes.

An example of an integrated model

In Fig. 5, we give an example of a tiny integrated model, with metabolites A, B, C and genes E, F, Z, Y. The fat arrows pointing to the reactions $R1$ and $R2$ stand for the enabling of these reactions by the genes which code for enzymes. For reaction $R1$, the genes E and F produce isozymes so that we have a logical OR in the corresponding reaction condition ϱ_1. The gene regulation is given by expressions ψ as before. Maximal values are defined for metabolites and genes. For the multivalued regulatory components B and C, we have the Boolean variables x_1^B, x_2^B, x_3^B and x_1^C, x_2^C, x_3^C, x_4^C. But only x_1^B and x_1^C play a role, so we abbreviate them as B, C respectively, as we did with the other, 2-valued components.

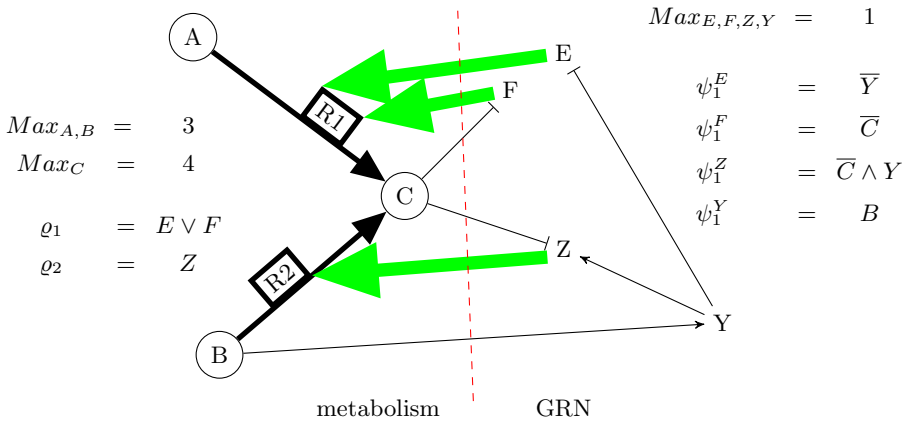


Fig. 5. An example of a tiny integrated model

Fig. 6 shows the Petri net obtained with our algorithm. Ignoring the dotted test-edges, we see the Petri net of the metabolism (with complementary

places) and the Petri net of the GRN as isolated systems. Reaction $R1$ has two transitions corresponding to the two minterms in ϱ_1.

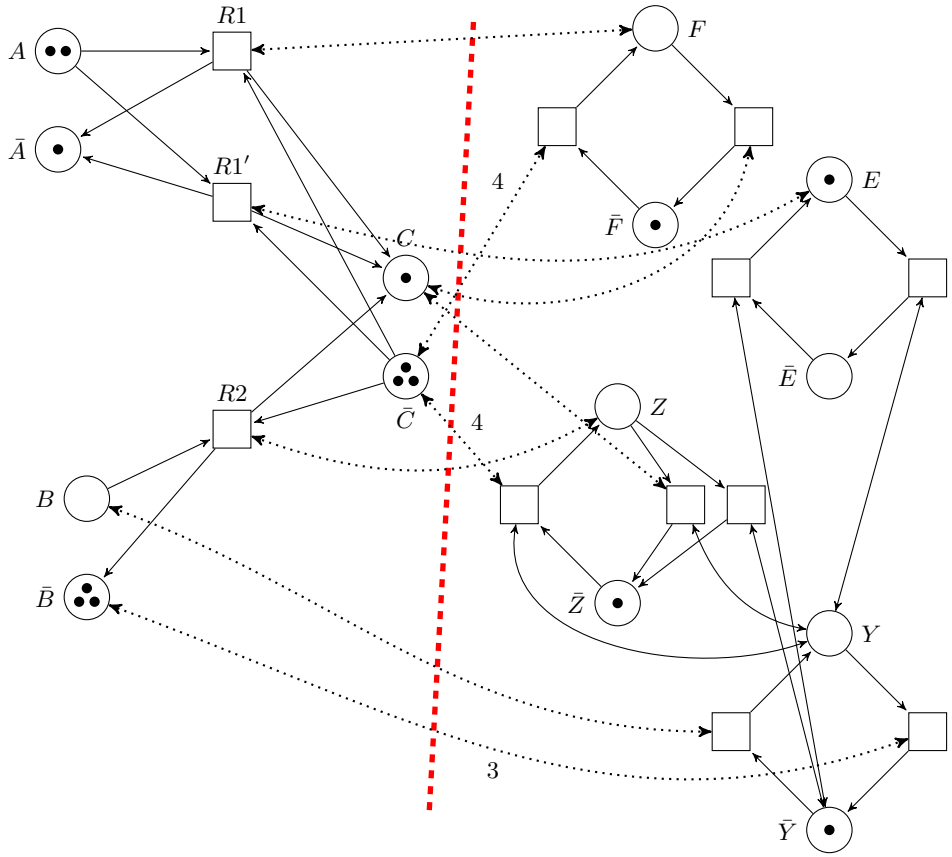


Fig. 6. Petri net of the tiny model

Pseudocode for the integrated Petri net

In the Petri net of an integrated model as in Def. 3, the total number of components is $N := l + k$. The Petri net thus has $2N$ places, where the first N places are the positive ones and $p(N + j)$ is the negative place complementary to $p(j)$, for all $j = 1, \ldots, N$. Again we assume all places to be already implemented. The code below adds the transitions and edges by creating columns d^n of the incidence and r^n of the input matrix. Note that the numbering of the genes starts with $l + 1$ because the metabolites come first. The first part of the code is again Algorithm 1. In the new part, the stoichiometric matrix is included. As before each minterm of the conditions ϱ_j is implemented by a transition. Test-edges are

translated as before. For shifting metabolite tokens according to the stoichiometry, the columns of the stoichiometric matrix S have to be included in D. For the complementary negative places the negative of the S-column is inserted. The input part of these shifts has to be implemented in R as well.

Algorithm 2:

```
n ⟵ 1
for i ← l + 1 to N do
    for v ∈ {−1, +1} do
        for κ conjunction of Φ_v^i do
            create the 0-vectors r^n, d^n ∈ ℕ^{2N}
            d_i^n ⟵ v                  // edges that actually shift tokens
            d_{N+i}^n ⟵ −v                        // to change the value
            for y literal of κ do
                if y is positive literal x_w^j then
                    r_j^n ⟵ w              // test-edges to positive, and
                if y is negative literal x̄_w^j then
                    r_{N+j}^n ⟵ Max_j − w + 1        // to negative places
            n ⟵ n + 1
for i ← 1 to h do
    for κ conjunction from ϱ_i do
        create the 0-vectors r^n, d^n ∈ ℕ^{2N}
        for j ← 1 to l do
            d_j^n ⟵ s_j^i                      // here the stoichiometric
            d_{N+j}^n ⟵ −s_j^i                      // matrix is inserted
            r_j^n ⟵ (|s_j^i| − s_j^i)/2                    // (R = (|D|−D)/2)
            r_{N+j}^n ⟵ −(|s_j^i| − s_j^i)/2
        for y literal of κ do
            if y is positive literal x_w^j then
                r_j^n ⟵ w                  // test-edges will implement
            if y is negative literal x̄_w^j then
                r_{N+j}^n ⟵ Max_j − w + 1           // the conditions ϱ_i
        n ⟵ n + 1
```

6 Use of the Petri Net Model and Further Research

Simão et al. [10] studied the reachability graphs of their Petri net starting from different initial markings. They identified fixpoints, cyclic attractors and interpreted them biologically. Their reachability graphs had the very small size of 120 markings.

In general, even for small Petri nets, the reachability graph will be huge and can be handled computationally only for limited model size. We experimented with a slightly larger tryptophan model than in [10], with seven regulatory components (+3) and two tryptophan pathways (+1). The reachability graphs had already about 10^5 nodes and identification of attractors seemed not sufficient to draw interesting conclusions.

Here, model checking seems to be a well suited tool to answer specific questions about the dynamics of the Petri net model. It is also powerful enough to handle much larger reachability graphs.

Petri nets are very flexible objects. For example, by assigning a rate to each transition, we get a continuous time Markov chain, which allows applying probabilistic model checking or stochastic simulation.

As already mentioned, the genome-scale model of *E. coli* [11] is given in a form that can be translated to a Petri net using our method. However, the reachability graph would be much too big to be computed in practice. Further research is needed to develop analysis methods that are suitable for such networks.

References

[1] C.H. Yeang. Integration of Metabolic Reactions and Gene Regulation. *Molecular Biotechnology*, 47:70–82, 2011.

[2] M.W. Covert, C.H. Schilling, and B. Palsson. Regulation of gene expression in flux balance models of metabolism. *Journal of theoretical biology*, 213(1):73–88, 2001.

[3] M.W. Covert, N. Xiao, T.J. Chen, and J.R. Karr. Integrating metabolic, transcriptional regulatory and signal transduction models in Escherichia coli. *Bioinformatics*, 24(18):2044, 2008.

[4] J.M. Lee, E.P. Gianchandani, J.A. Eddy, and J.A. Papin. Dynamic analysis of integrated signaling, metabolic, and regulatory networks. *PLoS computational biology*, 4(5):e1000086, 2008.

[5] T. Shlomi, Y. Eisenberg, R. Sharan, and E. Ruppin. A genome-scale computational study of the interplay between transcriptional regulation and metabolism. *Molecular systems biology*, 3(1), 2007.

[6] V.N. Reddy, M.N. Liebman, and M.L. Mavrovouniotis. Qualitative analysis of biochemical reaction systems. *Computers in biology and medicine*, 26(1):9–24, 1996.

[7] I. Koch. Petri Nets – A Mathematical Formalism to Analyze Chemical Reaction Networks. *Molecular Informatics*, 29(12):838–843, 2010.

[8] C. Chaouiya, A. Naldi, E. Remy, and D. Thieffry. Petri net representation of multi-valued logical regulatory graphs. *Natural Computing*, pages 1–24, 2010.

[9] R. Thomas and M. Kaufman. Multistationarity, the basis of cell differentiation and memory. II. Logical analysis of regulatory networks in terms of feedback circuits. *Chaos*, 11(1):180–195, 2001.

[10] E. Siñao, E. Remy, D. Thieffry, and C. Chaouiya. Qualitative modelling of regulated metabolic pathways: application to the tryptophan biosynthesis in E. Coli. *Bioinformatics*, 21(suppl 2), 2005.

[11] M.W. Covert, E.M. Knight, J.L. Reed, M.J. Herrgard, and B.O. Palsson. Integrating high-throughput and computational data elucidates bacterial networks. *Nature*, 429(6987):92–96, 2004.

[12] M.J. Herrgård, B.S. Lee, V. Portnoy, and B.Ø. Palsson. Integrated analysis of regulatory and metabolic networks reveals novel regulatory mechanisms in Saccharomyces cerevisiae. *Genome research*, 16(5):627, 2006.

[13] A. Goelzer, F.B. Brikci, I. Martin-Verstraete, P. Noirot, P. Bessières, S. Aymerich, and V. Fromion. Reconstruction and analysis of the genetic and metabolic regulatory networks of the central metabolism of Bacillus subtilis. *BMC systems biology*, 2(1):20, 2008.

[14] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 2002.

[15] A. Richard. Negative circuits and sustained oscillations in asynchronous automata networks. *Advances in Applied Mathematics*, 44(4):378–392, 2010.

[16] L.J. Steggles, R. Banks, O. Shaw, and A. Wipat. Qualitatively modelling and analysing genetic regulatory networks: a Petri net approach. *Bioinformatics*, 23(3):336, 2007.

A Constraint Program For Subgraph Epimorphisms with Application to Identifying Model Reductions in Systems Biology

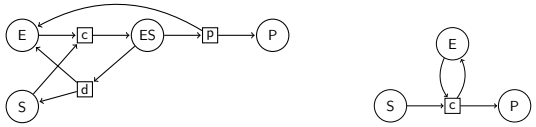
Steven Gay, François Fages, Thierry Martinez, Sylvain Soliman

EPI Contraintes, INRIA Paris-Rocquencourt,
BP105, 78153 Le Chesnay Cedex, France.
Steven.Gay@inria.fr

Abstract. This paper presents a constraint program for checking whether one graph can be obtained from another graph by using node deletions and node mergings. This NP-complete problem is equivalent to the existence problem of a subgraph epimorphism between two graphs, This differs from the well-known subgraph isomorphism problem by the existence of merge in addition to delete operations. Subgraph epimorphisms allow us to identify biologically meaningful reduction relationships between biochemical reaction graphs in large model repositories such as biomodels.net. This concept thus offers a computational tool for studying model reductions in systems biology by considering solely the structure of biochemical networks.

1 Introduction

Our interest in the concept of subgraph epimorphism stems from the study of model reductions in systems biology, where large systems of biochemical reactions can be naturally represented by bipartite digraphs of species and reactions [1, 2]. In this setting, one can define a very general notion of model reduction as a particular form of graph transformation and use it to compare models in systems biology model repositories [3]. For instance, the classical reduction of Michaelis-Menten consists in reducing a system of three reactions, where an enzyme E binds in a reversible manner to a substrate S to form a complex ES and release a product P, to a single reaction catalyzed by the enzyme, as depicted by the following graphs:



The reduced graph can be obtained from the detailed graph by a sequence of delete and merge operations on either species or reaction nodes. These transformations are typically justified in chemistry by considering: (i) reaction deletions

for slow reverse reactions, (ii) reaction mergings for reaction chains with a limiting reaction, (iii) molecular species deletions for species in excess and (iv) molecular mergings for quasi-steady state approximations.

This operational view of graph reduction is equivalent to the existence of an induced subgraph (corresponding to delete operations) epimorphism (i.e. surjective homomorphism, corresponding to merge operations) from a source graph to a reduced graph. Subgraph epimorphisms (SEPI) differ from minors [4] by the possibility to merge non adjacent nodes, by creating a loop when merging two adjacent nodes and by the impossibility to delete an arc without merging the nodes.

In this paper, after a presentation of the basic definitions and properties, we describe the constraint model and search strategy we use to compute subgraph epimorphisms in the biomodels.net repository. This benchmark consists in 241 curated models of up to several hundreds of molecular species.

2 Model Reduction

2.1 Reaction Graphs

One classical way of representing a systems biology model is to see it as a set of chemical reactions, which leads to the use of Petri Nets.

Here we chose to use the underlying directed graph. It is a bipartite graph, and we call it a *reaction graph*:

Definition 21 *Formally, a reaction graph G is a bipartite directed graph, that is a triple $G = (S, R, A)$, where S is the set of species nodes, R is the set of reaction nodes, and $A \subseteq S \times R \cup R \times S$.*

Now something a (biologist) modeller may want to do is to check whether some model is a *reduced* version of another model. On a larger scale, the modeller wants to get a *hierarchy* of models where each model is a refinement or a simplification of the surrounding ones.

One way to relate two models is to define graph editing operations which make it possible to transform one reaction graph into another.

A simple thing to do when trying to reduce models is to consider that two species are variants and treat them as equivalents, and to merge every interaction any of the two species had into a new species.

The same merging operation can be generalized for reactions.

Another natural operation is node deletion. It may be useful for instance to remove intermediate species, or species whose concentration is constant, or reactions that have become trivial after a molecular merging, or reverse reactions that occur in a much slower rate than their forward counterpart. Model refinement proceeds with the dual operations of node addition and splitting and is thus also covered by this approach.

2.2 Graph Edition Operations and Model Reduction

Here we generalize the reduction to arbitrary digraphs. There are two operations: node deletion and node merging.

Definition 22 (Delete) *Let $G = (V, A)$ be a graph and $u \in V$. The result of the deletion of u in G is the induced subgraph $d_u(G) = G_{\downarrow V \setminus \{u\}}$.*

The *merge* operation removes two nodes from a graph and replaces them with a new one inheriting all incident arcs.

Definition 23 (Merge) *Let $G = (V, A)$ be a graph and $u, v \in V$ such that $u \neq v$. The result of the merge of u and v in G is the graph $m_{u,v}(G)=(V', A')$ such that $V' = V \setminus \{u, v\} \cup \{uv\}$, and $A' = A \cap (V' \times V') \cup \{(uv, w) \mid (u, w) \in A \text{ or } (v, w) \in A\} \cup \{(w, uv) \mid (w, u) \in A \text{ or } (w, v) \in A\}$.*

We now define a *model reduction* as a finite string of delete/merge operations.

The problem of interest is now: given two graphs G and G', is there a reduction from G to G'?

First, let us find a way to express a reduction as a one-step operation.

2.3 Subgraph Epimorphisms

Notice that the subgraph isomorphism problem is equivalent to the existence of a string of deletions between two graphs.

There is a way to define a deletion/merging string as a kind of morphism:

Definition 24 *Let $G = (N, A)$ and $G' = (N', A')$ be two graphs.*

A morphism *from G to G' is a function μ from N to N', such that $\forall (x, y) \in A, (\mu(x), \mu(y)) \in A'$.*

An epimorphism *from G to G' is a morphism that is surjective on (both the nodes and the arcs of) G'.*

As shown below, graph epimorphisms relate graphs that can be obtained by only merge operations. To account for node deletions, we consider:

Definition 25 *Let $G = (N, A)$ and $G' = (N', A')$ be two graphs. A* subgraph morphism *μ from G to G' is a morphism from a subgraph induced by a subset of nodes of G, to G': $N_0 \longrightarrow N'$, with $N_0 \subseteq S$, such that $\forall (x, y) \in A \cap N_0 \times N_0, (\mu(x), \mu(y)) \in A'$.*

A subgraph epimorphism *(SEPI for short) from G to G' is a subgraph morphism that is surjective.*

Given these definitions, it can be proved that:

Theorem 26 ([3]) *Let $G = (N, A)$ and $G' = (N', A')$ be two graphs. There exists a* subgraph epimorphism *μ from G to G' if and only if there exists a finite sequence of* delete *and* merge *operations that, when applied to G, yield a graph isomorphic to G'.*

When testing for a reduction from G to G', the question is now to find whether there exists a SEPI from G to G'.

3 Constraint Program

In another paper with Christine Solnon being currently reviewed, we prove that the SEPI problem is NP-complete. This means that there does not exist an efficient algorithm for solving *all* problem instances in polynomial time, if we admit the conjecture $P \neq NP$.

Nevertheless, the practical instances of such problems may well be solved by efficient algorithms and it is the purpose of this section to describe a constraint program for the SEPI problem. For this work, we developed a GNU-prolog [5] program dedicated to our particular subgraph epimorphism problems, using finite domain constraints and a simple search strategy for enumerating all solutions by backtracking.

3.1 Constraint Model

The mathematical definition of subgraph epimorphisms given in the previous section can be encoded quite directly in an executable constraint model.

Graph morphisms can be modeled quite naturally by introducing one variable per node of the source graph, with, as domain, one (integer) value per node of the target graph. A variable assignment thus represents a mapping from the source nodes to the target nodes [6].

In this representation, the morphism condition itself, stating that the arcs must be preserved by the mapping, can be written using the primitive tabular constraint of GNU-Prolog

`fd_relation(integer_list_list, variable_list)`

This constraint states that the tuple of variables in the second argument (here two variables representing the image of an arc in the source graph) is equal to one element of the list in the first argument (representing all the arcs of the target graph).

The surjectivity property could be represented using the primitive cardinality constraint `fd_at_least_one` applied to the arcs of the target graph. However, a more efficient modeling was found by introducing antecedent variables for the target arc variables, i.e. one variable associated with each target arc with initial domain the set of source arcs, constrained to take as values only the source arcs that are antecedents of the associated target arcs. To achieve this, we use the constraint

`fd_element_var(Ante, ArcImagesList, TargetArc)`

which states that the Ante-th element in the ArcImagesList is equal to TargetArc. For each arc in the target graph, this constraint is actually used on the first and second nodes of the arc to state that the antecedent variables correspond to an arc in the source graph.

The subgraph condition can be modelled using a dummy value $\perp$ for deleted nodes, as formalized by the following property:

Proposition 31 ([3]) *Let $G = (N, A)$ and $G' = (N', A')$ be two graphs. Let $G'' = (N'', A'')$, with $N'' = N' \uplus \{\perp\}$, and $A'' = A' \uplus (\{\perp\} \times N') \uplus (N' \times \{\perp\}) \uplus (\{\perp\} \times \{\perp\})$.*

Then there is a SEPI from G to G' iff there is a graph morphism from G to G'' that is surjective on N' and A'.

For computing SEPI's in this modeling, a dummy value is thus added to the domain of the source graph node variables, and the surjectivity is enforced on the non-dummy arcs only.

3.2 Search Strategy

It may seem that given the model, we have to enumerate both source node variables and antecedent arc variables. Actually, enumerating only one of the sets is enough.

Suppose we have tried to enumerate the source node variables, and failed. Then, there is clearly no SEPI from the source graph to the target graph.

If on the contrary the enumeration succeeded, then there is obviously a morphism. Is it surjective ? Since every antecedent variable has a non-empty domain, we know that every pair of antecedents of the corresponding target pair of nodes comes from an arc in the source graph.

Thus, an enumeration of the source node variable is enough to enforce arc surjectivity. However, compared to enumerating the antecedents variables beforehand, this choice of variables checks the surjectivity quite late.

Now, suppose we have enumerated only the antecedent variables, and failed. Once again, it is obvious that there is no SEPI from the source graph to the target graph.

If the enumeration succeeded, then some source node variables have been *determined* by the process, i.e. their domain have a single value. Suppose we put the $\perp$ value (coded by 0) for every source node variable that has not been determined. Then, it can be proved that the valuation of the source node variables is a SEPI from the source graph to the target graph. Indeed, the valuation is always a morphism because of the way dummy values are used, and the antecedent arc variables being determined, it is surjective on the arcs.

This proves that enumerating antecedent arc variables is enough (provided we fill the remaining variables with $\perp$). This "antecedents first" strategy works best in practice.

3.3 GNU-Prolog Code

The original GNU-Prolog program for reaction graphs used in BIOCHAM [7] is too long to be shown here. Here is a simplified GNU-Prolog program for computing SEPIs according to the previous constraint model. For brevity, it supposes that the target graph has no isolated nodes. This allows us to enforce surjectivity on arcs only, as in this case it entails surjectivity on nodes.

```
epi_sub_graph(SourceNodeCount, SourceArcs,
              TargetNodeCount, TargetArcs, NodeImages) :-
   length(NodeImages, SourceNodeCount),
```

```
   fd_domain(NodeImages, 0, TargetNodeCount),
   findall(X,
     (X = [0, 0] ;
      for(Node, 1, TargetNodeCount),
        (X = [0, Node]; X = [Node, 0])),
     DummyArcs),
   append(TargetArcs, DummyArcs, AllTargetArcs),
   morphism_constraint(SourceArcs, NodeImages, AllTargetArcs),
   source_arcs(SourceArcs, NodeImages, ArcImages0, ArcImages1),
   surjectivity_constraint(TargetArcs, ArcImages0,
                           ArcImages1, Antecedents),
   fd_labeling(Antecedents, []), fd_labeling(NodeImages, []).

morphism_constraint([], _NodeImages, _AllTargetArcs).
morphism_constraint([[N0, N1] | Arcs], NodeImages, AllTargetArcs) :-
   nth(N0, NodeImages, I0), nth(N1, NodeImages, I1),
   fd_relation(AllTargetArcs, [I0, I1]),
   morphism_constraint(Arcs, NodeImages, AllTargetArcs).

source_arcs([], _NodeImages, [], []).
source_arcs([[N0, N1] | Arcs], NodeImages,
            [I0 | Arcs0], [I1 | Arcs1]) :-
   nth(N0, NodeImages, I0), nth(N1, NodeImages, I1),
   source_arcs(Arcs, NodeImages, Arcs0, Arcs1).

surjectivity_constraint([], _ArcImages0, _ArcImages1, []).
surjectivity_constraint([[N0, N1] | Arcs], ArcImages0,
                        ArcImages1, [I | Antecedents]) :-
   fd_element_var(I, ArcImages0, N0),
   fd_element_var(I, ArcImages1, N1),
   surjectivity_constraint(Arcs, ArcImages0,
                           ArcImages1, Antecedents).
```

4 Evaluation

The reduction relations between all pairs of models of the biomodels.net repository have been computed (with a time out of 20 minutes per problem) and the results reported in [3].

Some matchings between unrelated model classes were found. These biologically false positive matchings typically arise with small models that formally appear as reductions of large models without any biological meaning. These false positives arise in less than 9% of the total inter-class pairs, and in 1.2% of the tests after the removal of the small models.

Apart from these cases, biologically meaningful model reductions were automatically discovered between models of MAPK signaling, cell cycle, circadian

clock, calcium oscillation etc. For instance, the matchings found between the models of the MAPK cascade are depicted in Figure 1.

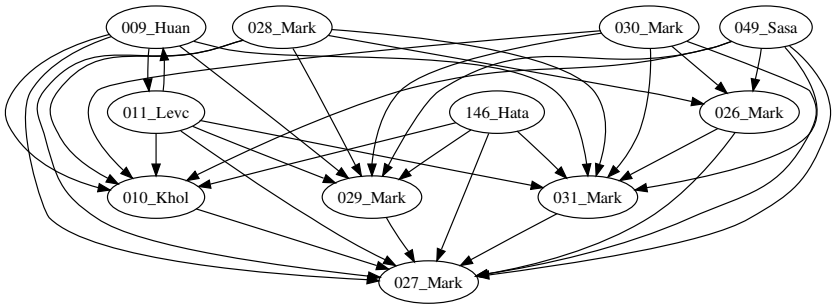


Fig. 1. Matchings found between all models of the MAPK cascade (Schoeberl's model 14 and Levchenko's model with scaffold 19 are not represented here, they do not map each other but can be mapped to small models).

This class contains the family of models of [8] numbered 26 to 31 in biomodels.net.In this family, models 27, 29 and 31 are the simpler ones: they have few molecules because the catalyses are represented with only one reaction. The epimorphism exhibited from model 31 to 27 corresponds to the splitting of two variants of MAPKK in 31. Model 29 distinguishes between the sites of phosphorylation of Mp, yielding a model with two molecules MpY and MpT. The subgraph epimorphism found from 29 to 27 corresponds to the deletion of one variant of Mp. Conversely, this distinction prevents the existence of an epimorphism from 31 or 27 to 29.

Models 26, 28 and 30 have more detailed catalysis mechanisms and differ as previously by the phosphorylation sites of Mp.

However, some epimorphisms from big models to small ones may have no biological meaning. This comes from the absence of constraint on the nodes that can be merged, and the relatively high number of arcs in Markevich's small models where most molecules are catalysts. Still, model 26 (with non-differentiated Mp) does not reduce to model 29 since that model indeed distinguishes MpY and MpT variants.

Now, concerning 3-step MAPK cascade models, the models 9 and 11 of [9] and Levchenko et al. respectively are detected as isomorphic. Indeed, they only differ by molecule names and parameter values. They do not reduce to 28 and 30, which are models that do not differentiate sites of phosphorylation. They do not reduce to 26 either, which uses a more detailed mechanism for dephosphorylations.

Model 10 is another 3-step MAPK with no catalysts for dephosphorylations. It has the particularity to be cyclic, that is, the last level's most phosphorylated

molecule catalyzes the phosphorylations of the first level. This is shown here as a reduction of the previous models obtained by merging the output of the third level with the catalyst of the first level.

Finally, models 49 and 146 are bigger than the others and can easily be matched by them. There were a few comparisons for which no result was found before the timeout.

5 Conclusion

The constraint model presented here to solve the SEPI NP-complete problem is very simple, and still it yields computation times of a few seconds for most cases in the biomodels.net repository of biochemical networks. However some optimizations, such as redundant constraints, should result in even better behaviour for bigger instances. Furthermore, the handling of labels and annotations attached to molecular species nodes would drastically reduce the search space for the labeling.

Future work includes another harder problem to tackle, the problem of greatest common epimorphic subgraph, i.e. model intersection, and its dual of smallest common epimorphic supergraph, i.e. model union. Given two graphs G and G', what are the greatest (smallest) graphs G'' such that both G and G' reduce to (resp. are reductions of) G'' by SEPIs ? Such graphs may however be not unique.

References

[1] Reddy, V.N., Mavrovouniotis, M.L., Liebman, M.N.: Petri net representations in metabolic pathways. In Hunter, L., Searls, D.B., Shavlik, J.W., eds.: Proceedings of the 1st International Conference on Intelligent Systems for Molecular Biology (ISMB), AAAI Press (1993) 328–336

[2] Hofestädt, R.: A petri net application to model metabolic processes. Systems Analysis Modelling Simulation **16** (October 1994) 113–122

[3] Gay, S., Soliman, S., Fages, F.: A graphical method for reducing and relating models in systems biology. Bioinformatics **26**(18) (2010) i575–i581 special issue ECCB'10.

[4] Lovász, L.: Graph minor theory. Bulletin of the American Mathematical Society **43**(1) (2006) 75–86

[5] Diaz, D.: GNU Prolog user's manual. (1999–2003)

[6] le Clément, V., Deville, Y., Solnon, C.: Constraint-based graph matching. In: 15th International Conference on Principles and Practice of Constraint Programming (CP 2009). Volume 5732 of Lecture Notes in Computer Science., Lisbon, Portugal, Springer-Verlag (2009) 274–288

[7] Fages, F., Jovanovska, D., Rizk, A., Soliman, S.: BIOCHAM v3.2 Reference Manual. INRIA. (2010)

[8] Markevich, N.I., Hoek, J.B., Kholodenko, B.N.: Signaling switches and bistability arising from multisite phosphorylation in protein kinase cascades. Journal of Cell Biology **164**(3) (February 2005) 353–359

[9] Huang, C.Y., Ferrell, Jr., J.E.: Ultrasensitivity in the mitogen-activated protein kinase cascade. PNAS **93**(19) (September 1996) 10078–10083

A New Local Move Operator for Reconstructing Gene Regulatory Networks

Jimmy Vandel and Simon de Givry

INRA - BIA, Toulouse, France,
jimmy.vandel@toulouse.inra.fr

Abstract. The discovery of regulatory networks is an important aspect in the post genomic research. Among structure learning approaches we are interested in local search methods in the Bayesian network framework. We propose a new local move operator to escape more efficiently from local maxima in the search space of directed acyclic graphs. This operator allows to overtake the acyclic constraint of Bayesian networks and authorizes local moves previously banned with classic operators. First results show improvements of learnt network quality. Our algorithm uses Comet language providing abstraction for local search and constraint programming.

Keywords: structure learning, Bayesian networks, local search, Comet language, gene regulation inference, genetical genomics.

1 Introduction

Inferring gene regulatory networks (GRN) from microarray data is a challenging problem, in particular because the sample size is typically small compared to the thousands of genes that compose the network. Currently, integrative approaches are developed to combine several sources of information in order to improve prediction quality. One of these approaches consists in using genetical genomic data combining gene expressions and sequence polymorphisms observed by genetic markers [1] [2](Chap. 4).

Among the many existing frameworks used to infer GRN, we choose probabilistic graphical models and more specifically *static* Bayesian Networks (BN) [3]. Learning BN structures from data is a NP-hard problem [4] and several approaches have been proposed to solve it. One of them consists in exploring the space of BN structures using local search methods and evaluating each structure with a specific scoring criterion in order to select the structure which maximizes the score.

In Section 2 we present Bayesian network and a new operator called "iterative swap cycle" (ISC) for local search algorithms. Then we report in Section 3 our preliminary work using this operator inside the Comet local search platform and give some positive results on simulated genetical genomic data.

2 Bayesian network and local search methods

A *Bayesian network* [3] denoted by $B = (\mathcal{G}, \mathbf{P}_{\mathcal{G}})$ is composed of a directed acyclic graph $\mathcal{G} = (\mathbf{X}, \mathbf{E})$ with nodes representing p random discrete variables $\mathbf{X} = \{X_1, \ldots, X_p\}$,

linked by a set of directed edges $\mathbf{E}$, and a set of conditional probability distributions $\mathbf{P}_{\mathcal{G}} = \{P_1, \ldots, P_p\}$ defined by the topology of the graph: $P_i = \mathbb{P}(X_i|Pa(X_i))$ where $Pa(X_i) = \{X_j \in \mathbf{X} \,|\, \overrightarrow{(X_j, X_i)} \in \mathbf{E}\}$ is the set of parent nodes of X_i in $\mathcal{G}$. A Bayesian network B represents a joint probability distribution on $\mathbf{X}$ such that:

$$\mathbb{P}(\mathbf{X}) = \prod_{i=1}^{p} \mathbb{P}(X_i|Pa(X_i)) \tag{1}$$

The conditional probability distributions $\mathbf{P}_{\mathcal{G}}$ are determined by a set of parameters, θ, via the equation:

$$\mathbb{P}(X_i = k|Pa(X_i) = j) = \theta_{ijk}$$

where k is a value of X_i, and j is a value configuration of the parent set $Pa(X_i)$. Given the structure G, parameters θ_{ijk} can be estimated by following the maximum likelihood principle.

Learning the structure of a Bayesian network consists in finding a DAG $\mathcal{G}$ maximizing $\mathbb{P}(\mathcal{G}|\mathbf{D})$ where $\mathbf{D}$ represents the observed data. We use in our study the popular Bayesian Dirichlet criterion to maximize the score:

$$BDeu(\mathcal{G}) = \prod_{i=1}^{p} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(n_{ij} + \alpha_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(n_{ijk} + \alpha_{ijk})}{\Gamma(\alpha_{ijk})}$$

with n_{ijk}, the number of occurrences of the configuration $(X_i = k, Pa(X_i) = j)$ in the n samples, $n_{ij} = \sum_{k=1}^{r_i} n_{ijk}$ and Dirichlet hyper-parameters $\alpha_{ijk} = \frac{\alpha}{r_i * q_i}$ where α is the *equivalent sample size* parameter, r_i is the domain size of variable X_i and $q_i = \prod_{X_j \in Pa(X_i)} r_j$, is the product of the parental domains of X_i.

In a GRN context with genetical genomic data the set of discrete random variables $\mathbf{X}$ is composed of one variable per gene-activity, denoted G_i, and one variable for each genetic marker, denoted M_i, $\forall i \in \{1, \ldots, p\}$ with p the number of genes. We assume each gene G_i is co-located with a single genetic marker M_i. Each marker may explain the variation of its associated gene activity or the variations of other regulated genes. An example is given in Figure 1.

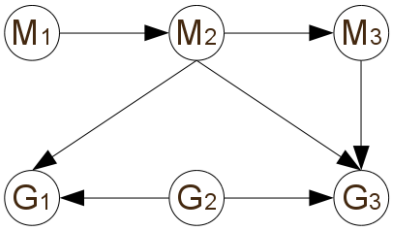


Fig. 1. Example of 3-genes network with regulations from gene 2 to gene 1 and 3. We assume a marker order (M_1, M_2, M_3) on a single chromosome.

Heuristic local search algorithms are widely used to learn Bayesian network structures as hill-climbing search, simulated-annealing, MCMC, genetic algorithms, ant colony optimization [5] and dozens more with additional refinements [6]. These methods are

often compared in previous papers with different datasets and results but if these methods tend to develop sophisticated algorithms to select at each step the best neighbor, only few of them tried new local operators to define this neighborhood [7, 8, 9]. Other approaches working on larger neighborhoods collapse a set of DAGs into a unique representative configuration. For instance, they explore the search space of total variable orderings (an optimal DAG compatible with the order is then easier to deduce) [10, 11], or the search space of Markov-equivalent partially-oriented DAGs [12, 13, 14]. Classical operators are addition, deletion and reversal of a directed edge, but these operators lead to reach quickly local maxima, even if some metaheuristic principles like Tabu list or simulated-annealing reduce this drawback. Furthermore the acyclic constraint of Bayesian networks is often considered as a hard constraint to define the neighborhood of a graph. We propose a new operator called ISC (Iterative Swap Cycle) to potentially overcome this constraint.

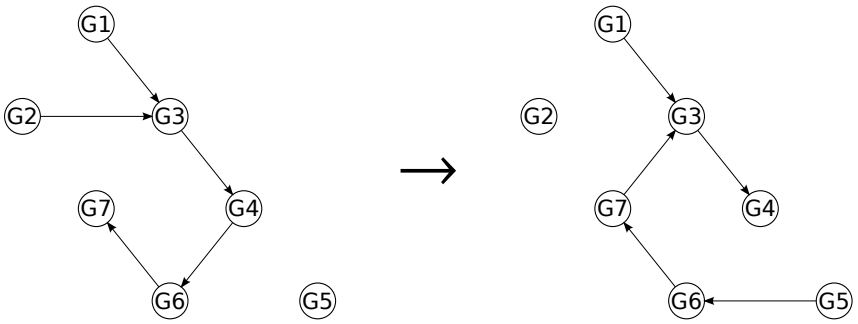


Fig. 2. Modification of 7-gene network structure after an ISC operation.

Let us consider the situation in Figure 2. We call Δ_{G_i,G_j} the BDeu score variation when we add variable G_i as parent of G_j with other parents fixed, this variation is positive if the score increases when we add the arc $G_i \rightarrow G_j$ and negative otherwise. We define the swap operator as follow, swapping an edge $G_i \rightarrow G_j$ with G_k produces the simultaneous deletion of $G_i \rightarrow G_j$ and the addition of $G_k \rightarrow G_j$. Given the initial structure in Figure 2, to swap $G_2 \rightarrow G_3$ with G_7 will be forbidden with classical operators because of the acyclic constraint even if Δ_{G_7,G_3} is high. The idea of ISC operator is to add $G_7 \rightarrow G_3$ anyway and if needed to break the cycle created by this addition. First we remove an edge of this cycle minimizing Δ value (in this example we assume $G_4 \rightarrow G_6$), if this deletion plus the swap operation do not decrease the score of the graph ($\Delta_{G_7,G_3} - \Delta_{G_2,G_3} - \Delta_{G_4,G_6} > 0$) the move is validated. Otherwise we choose another parent (not included in the considered cycle) for node G_6 maximizing Δ value (here we add $G_5 \rightarrow G_6$) that we can consider as a swap of $G_4 \rightarrow G_6$ with G_5. If another cycle is created during this swap, we iterate a new cycle break operation. Finally we iterate until the initial cycle is broken (means no sub-cycles were created) or the score of the modified graph cannot be greater than the initial score. We only validate local moves if all the cycles are broken and if the modified graph increases the score.

If several cycles are created by the same edge addition, we break each of them, one after another by applying ISC operator for each one.

The main idea of ISC operator is to try, each time we want to add a forbidden edge $G_i \rightarrow G_j$, to break the cycles by deleting or swapping a parent for one node of the cycles and to iterate this operation to solve potential cycles created during the swap. A forbidden edge could appear when we try to swap an edge as in our previous example but also after an addition or reversal of an edge. So ISC operator is applicable for add, reverse and swap operators.

3 On going work

We use the Comet software [15] to implement iterated hill-climbing search. Comet is a specific language which provides useful concepts for implementing local search methods like invariant, objective and constraint functions. In our implementation we encode BDeu score using cache for up to two parents and incremental cycle detection (using incremental topological ordering [16]) using user-defined invariants on a graph which means that each modification on the graph is automatically propagates on the score and cyclicity test, allowing an easier way to develop new neighborhood operators and new search algorithms. We do not use any constraint features of Comet but invariants in Comet offer incrementality for free. For each node we keep in memory the neighborhood defined by all operators applicable on this node with associated score variations and potential cyclic situations, which saves computations and helps to quickly update cyclic situations.

We did not implement ISC operator yet, but we developed its simplified version called nISC (non Iterative Swap Cycle). nISC is similar to ISC but does not iterate on potential cycle creations, it try to delete or swap only one edge of the cycle produced by classical operator to break it. If deleting or swapping an edge does not break the cycle or create another cycle, the classical operation is tagged as invalid and cannot be applied in this configuration. This operator is more simple and less time consuming than ISC but allows to overcome some cycle situations. We show in Figure 3 BDeu scores reached (with $\alpha = 1$) and time requirement of 1000 runs of hill climbing search starting from a random structure (2 randomly selected parents per node) in five configurations. These configurations differ by the set of authorized operators during the search among classical ones (A:addition, D:deletion and R:reversal) and the swap operator (S:swap). The last configuration represents the results with nISC extension (*:nISC) for addition, reversal and swap operators (deletion of an edge cannot create a cycle). Our test network is composed of 2 000 nodes (1 000 genes and 1 000 markers) and 300 samples from DREAM5 challenge (SysGenA300_Network1) [17]. In order to deal with large datasets, we restrict the list of candidate parents as done in [18].

We see in Figure 3 that the scores increase as more operators are used but in counterpart slows down the search.

If the reversal operation increases a little the mean score, we see a significant improvement when we use the swap operator. These results show that swapping an edge allows a deeper structure modification than reversing it although both are composed of an addition and a deletion. Furthermore variance of scores is reduced with the swap

operator. Efficiency of the swap operator is simply explained by the fact that initial structures could allocate for one node G_j, a medium quality parent G_i ($\Delta_{G_i,G_j} > 0$). However if a better parent G_k exists for this node $\Delta_{G_k,G_j} > \Delta_{G_i,G_j}$ but both G_i and G_k cannot be parents at the same time we would need to swap G_i by G_k. So we first need to remove G_i but this operation will decrease the score and will not be considered in a hill-climbing search or with low probability in a simulated-annealing algorithm.

Applying nISC with the four operators allows to increase once more the mean score reached and to reduce variance which suggest that trying to overcome cycles allows to escape from local optima more and to achieve similar quality structures. But this more intensive search is much more time consuming even if the current implementation could be improved. Trade off between search time and quality of the learnt structure still need to be investigated.

	A+D	A+D+R	A+D+S	A+D+R+S	A*+D+R*+S*
BDeu scores (*in* log_{10})					
mean	-360 415	-360 349	-358 885	-358 826	**-358 417**
variance (10^3)	25.936	26.681	4.446	4.763	**2.713**
Mean Time (seconds)	**22.2**	28.2	36.6	40.2	153

Fig. 3.

Our first results show large difficulty for local search algorithms to escape from local maxima with classical operators. A difficulty which can explain poor results we obtained with the simulated annealing method [19], even if a progressive decrease of the temperature can move the search in a promising area, but when temperatures become too low the algorithm quickly suffers from restrictive operators and falls in a local maximum. For this reason instead of developing highly complex metaheuristics, we decide to explore new operators to define larger neighborhoods.

References

[1] R. Jansen and J. Nap, "Genetical genomics : the added value from segregation," *Trends in genetics*, vol. 17, no. 7, pp. 388–391, 2001.

[2] S. Das, D. Caragea, W. H. Hsu, and S. M. Welch, eds., *Handbook of Research on Computational Methodologies in Gene Regulatory Networks*. Hershey, New York: IGI Global, 2010.

[3] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[4] D. Chickering and D. Heckermann, "Learning bayesian networks is NP-complete," *In learning from data: Al and Statistics*, 1996.

[5] R. Daly and Q. Shen, "Learning bayesian network equivalence classes with ant colony optimization," *Journal of Artificial Intelligence Research*, vol. 35, pp. 391–447, 2009.

[6] E. Salehi and R. Gras, "An empirical comparison of the efficiency of several local search heuristics algorithms for bayesian network structure learning," in *Learning and Intelligent OptimizatioN Workshop (LION 3)*, 2009.

[7] L. de Campos, J. Fernandez-Luna, and J. Puerta, "Local search methods for learning bayesian networks using a modified neighborhood in the space of dags," in *Advances in Artificial Intelligence IBERAMIA 2002*, vol. 2527, pp. 182–192, 2002.

[8] A. Moore and W. Wong, "Optimal reinsertion: A new search operator for accelerated and more accurate bayesian network structure learning," in *Proceedings of the 20th International Conference on Machine Learning (ICML '03)*, pp. 552–559, AAAI Press, 2003.

[9] A. Holland, M. Fathi, M. Abramovici, and M. Neubach, "Competing fusion for bayesian applications," in *Proc. of the 12th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2008)*, pp. 378–385, 2008.

[10] G. Cooper and E. Hersovits, "A bayesian method for the induction of probabilistic networks from data," *Machine Learning*, vol. 9, pp. 309–347, 1992.

[11] M. Teyssier and D. Koller, "Ordering-based search: A simple and effective algorithm for learning bayesian networks," in *Proceedings of the Twenty-first Conference on Uncertainty in AI (UAI)*, pp. 584–590, 2005.

[12] D. Chickering and D. Maxwell, "Learning equivalence classes of bayesian-network structures," *J. Mach. Learn. Res.*, vol. 2, pp. 445–498, 2002.

[13] S. Acid and L. M. de Campos, "Searching for bayesian network structures in the space of restricted acyclic partially directed graphs," *J. Artif. Int. Res.*, vol. 18, pp. 445–490, 2003.

[14] D. Fierens, J. Ramon, M. Bruynooghe, and H. Blockeel, "Learning directed probabilistic logical models: ordering-search versus structure-search," *Annals of Mathematics and Artificial Intelligence*, vol. 54, pp. 99–133, 2008.

[15] L. Michel and P. V. Hentenryck, "A constrained-based architecture for local search," in *17th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications*, 2002.

[16] D. J. Pearce and P. H. J. Kelly, "A dynamic topological sort algorithm for directed acyclic graphs," *J. Exp. Algorithmics*, vol. 11, 2007.

[17] "Dream5, systems genetics challenges." http://wiki.c2b2.columbia.edu/dream/index.php/D5c3, 2010.

[18] A. Goldenberg and A. Moore, "Tractable learning of large bayes net structures from sparse data," in *Proceedings of the twenty-first international conference on Machine learning*, ICML '04, pp. 44–51, 2004.

[19] D. Chickering, D. Heckerman, and D. Geiger, "Learning bayesian networks: search methods and experimental results," *AI + Stats*, 1995.

Author Index