Proceedings of WCB13
Workshop on
Constraint Based Methods for Bioinformatics

Alessandro Dal Palù and Agostino Dovier

September 16, 2013, Uppsala (Sweden)

Program Committee

- *Nicos Angelopoulos*, Netherlands Cancer Institute, Amsterdam, The Netherlands
- *Rolf Backofen*, Institut für Informatik, Albert-Ludwigs-Universität, Freiburg, Germany
- *Pedro Barahona*, CENTRIA, Univ. Nova de Lisbona, Portugal
- *Alexander Bockmayr*, Freie Universität Berlin, Germany
- *Alessandro Dal Palù* (co-chair), Dipartimento di Matematica, Univ. of Parma, Italy
- *Simon De Givry*, Centre de recherches de Toulouse, Station de biométrie et d'intelligence artificielle, INRA, France
- *Agostino Dovier* (co-chair), DIMI, Università di Udine, Italy
- *François Fages*, INRIA Paris-Rocquencourt, France
- *Inês Lynce*, Departamento de Engenharia Informática, INESC-ID Lisboa, Portugal
- *Enrico Pontelli*, Dept. of Computer Science, New Mexico State University, USA
- *Thomas Schiex*, Dept. de Mathématique et Informatique appliquées, INRA Toulouse, France
- *Sven Thiele*, INRIA Rennes, France.
- *Pascal van Hentenryck*, Optimization Research Group, NICTA, Australia
- *Sebastian Will*, Computer Science Department, Leipzig University, Germany

External referees

- *Sabine Peres*, Laboratoire de Recherche en Informatique, Université Paris-Sud, France
- *Steven Gay*, INRIA Paris-Rocquencourt, France

Preface

The *Workshop on Constraint Based Methods for Bioinformatics* has reached its 9th consecutive edition. As for its first edition (Sitges 2005) as well as for the two ancestor workshops in 1997 and 1999, this year's edition is hosted as satellite workshop of the International Conference on Constraint Programming, which is perhaps the most natural location for this workshop. Yet in 2005 Bioinformatics was a hot topic and it is still a very active area nowadays. Various problems, both challenging and of high relevance, can be formalized and solved by declarative methods. Moreover, a notable trend that characterized these latest years is that the intrinsic complexity of these problems favors the hybridization of several techniques (Constraint Programming, Logic Programming, Integer Linear Programming, SAT, Local Search, Tabling, and so on) in order to achieve more accurate results in reasonable time.

As in previous editions, the accepted papers range on a rich set of interesting problems. In particular, Fioretto and Pontelli study how to infer additional knowledge in the domain of Gene regulatory networks by modeling data integration by means of Constraint Programming; Kishimoto and Marinescu deal on haplotyping problems, namely Genetic linkage analysis based constraint networks. Soliman, Fages and Radulescu use Constraint Programming for implementing equilibrium conditions of perturbation theory for the reduction of biochemical models. Fages and Soliman, with Gay and Santini deal with the subgraph epimorphism problem in particular pointing out the application of the results in model reduction. Elsen, de Givry, Katsirelos, and Shumbusho model and compare different solvers on the problem of genomic selection design. Lesaint, Mehta, and O'Sullivan deal with the problem of finding *soft patterns* in already classified protein families. Mann and Thiel face the problem of molecule representation, in particular revisiting the Kelulè model, by means of constraint programming. David and Bockmayr develop new techniques for the analysis of genome-scale metabolic networks by means of ILP. Bau, Waldmann, and Will deal with the RNA secondary structure design problem using SAT techniques.

We are particularly grateful to Peter Stuckey who accepted to give an invited talk on a topic that is very close to editor's personal research interest, namely *What is the minimal dictionary of protein substructures of which all known proteins are made?* The talk summarizes a recent work by Peter in collaboration with Arun Konagurthu, Arthur M. Lesk, David Abramson, and Lloyd Allison.

A particular thank to the Association for Constraint Programming that accepted to host our workshop and in particular to the Workshop (and Tutorial) Chair Laurent Michel and to the CP Conference Chairs Mats Carlsson, Pierre Flener, and Justin Pearson, and, again to our invited speaker Peter Stuckey.

Alessandro Dal Palù and Agostino Dovier

Table of Contents

Constraint Programming in Community-based Gene Regulatory Network Inference

Ferdinando Fioretto[1,2] and Enrico Pontelli[1]

[1] Dept. Computer Science, New Mexico State University
[2] Depts. Math. & Computer Science, University of Udine
ffiorett,epontell@cs.nmsu.edu

Abstract. Gene Regulatory Network (GRN) inference is a major objective of Systems Biology. The complexity of biological systems and the lack of adequate data have posed many challenges to the inference problem. *Community networks* integrate predictions from individual methods in a "meta predictor", in order to compose the advantages of different methods and soften individual limitations. This paper proposes a novel methodology to integrate prediction ensembles using Constraint Programming, a declarative modeling paradigm, which allows the formulation of dependencies among components of the problem, enabling the integration of diverse forms of knowledge. The paper experimentally shows the potential of this method: the addition of biological constraints can offer improvements in the prediction accuracy.

1 Introduction

Within a cellular context, genes interact to orchestrate a multitude of important tasks. These interactions are regulated by different gene products, as proteins called *Transcription Factors (TFs)* and RNA, and they constitute an intricate machinery of regulation referred to as *Gene Regulatory Networks (GRNs)*. In turn *GRN inference* describes the process of inferring the topology of a particular GRN. GRN inference from high-throughput data is of central importance in computational system biology. Its use is crucial in understanding important genetic diseases, such as cancer, and to devise effective medical interventions.

The availability of a wealth of genomic data has encouraged the development of diverse methods for GRN inference. However, data sets are quite heterogeneous in nature, containing information which is limited and difficult to analyze [24]. This reverberates on performance of GRN inference methods, which tend to be biased toward the type of data and experiments. For instance, methods based on linear models perform poorly on highly non-linear data, such as the one produced in presence of severe perturbations like gene knock-outs [11]. To alleviate these difficulties several alternatives have been proposed, such as integrating heterogeneous data into the inference model [20], or integrating a collection of predictions across different inference methods in *Community Networks (CNs)* [13, 14]. The former is a promising research direction but it has to face several challenges which span from how to relate different types of data to

data sets normalization processes. The latter has the advantage of promoting the benefits of individual methods while smoothing out their drawbacks. Moreover it does not exclude the use of the former solution within the initial prediction set. The CN integration process poses many challenges, raising questions like: **(i)** how to take into account strengths and weaknesses of individual inference methods—e.g., the difficulty for Mutual Information (MI) or correlation based methods to discriminate TFs; and **(ii)** how to leverage additional information which cannot be taken into account by the individual methods.

In this paper, we propose a novel methodology based on *Constraint Programming (CP)* to integrate community predictions. CP is a declarative problem solving paradigm, where logical rules are used to model problem properties and to guide the construction of solutions. CP offers a natural environment where heterogeneous information can be actively handled. The use of constraint expressions allows the incremental refinements of a model. This is particularly suitable to take care of biological knowledge integration, when such knowledge cannot be directly handled by individual prediction methods.

We test our method on a set of 110 benchmarks proposed by the DREAM3 [14] and DREAM4 [17] challenges. We show increases in prediction accuracy with respect to a CN prediction based on the Borda count election method [13].

2 Related Work

A wide variety of GRN inference methods from expression data have been proposed [20]. These include: **(1)** Discrete models based on Boolean networks and Bayesian networks [11]; **(2)** Regression methods like *TIGRESS*—which imposes a regression problem to each gene; **(3)** Methods based on mutual information (MI) theory, such as *ARACNE* [15] statistical likelihood of MI values. *Ensemble learning* has been explored for example by *GENIE3*, which uses a Random Forest approach [10]. Meta approaches have also been explored, such as *Inferelator*, based on re-sampling combining *median-corrected z-scores(MCZ)*, *time-lagged CLR (tlCLR)*, and linear ODE models [8].

Community Networks (CNs) integrate multiple inference methods to obtain a common consensus prediction. They have been shown to achieve better average confidence across different datasets and produce more robust results with respect to the individual methods being composed [13]. A simple scheme for combining predictions in a community network has been proposed in [13], where each interaction is re-scored by averaging the ranks it obtained within each of all the employed predictions. In the rest of the paper we will refer to it with $\mathrm{CN}_{\mathrm{rank}}$.

Constraint Technologies have been recently successfully applied in the field of System Biology [23]. For example, Answer Set Programming has been adopted to address problems in network inconsistencies detection [7] and in metabolic network analysis [21]. CP has been investigated to reason over discrete network models, where GRNs are modeled using multi-valued variables and transition rules [4]. In particular, CP is exploited to represent GRNs' possible dynamics [6].

3 Methods

The CN approach adopted in this work is built by combining four GRN inference procedures and creating an *inference ensemble*. Three of them are top-ranking methods that have been presented in the past DREAM competitions [13]: *(i) TIGRESS* [9], *(ii) Inferelator* [8], and *(iii) GENIE3* [10]. The fourth is an "off-the-shelf" widely adopted MI-based method *(CLR)* [5]. *TIGRESS* is a regression method which imposes a regression problem to each gene, solving it using Lasso Regression [22] with stability selection [16] together with a bootstrapping technique [9]; *GENIE3* uses a Random Forest approach to rank interactions based on the importance of TFs for the prediction [10]. *Inferelator:* is a meta approach based on re-sampling combining *median-corrected z-scores(MCZ)*, to rank edges based on a z-score derived from TF-deletion data, *time-lagged CLR (tlCLR)*, for the analysis of time-series data, and a linear ODE model constrained by Lasso [8]. CLR [5] is a GRN inference method based on statistical likelihood of MI values, and hence is capable of capturing "non-linearity" features in the data. In particular CLR applies a correction step to eliminate indirect interactions [5]. The former methods employ bootstrapping whilest the latter does not. These methods have been selected to provide robustness and diversity, avoiding method redundancies that could potentially bias the inference ensemble. We use the *GP-DREAM* web platform (`http://dream.broadinstitute.org`) to generate the predictions from each of these methods.

The initial set of methods for the CN approach plays a central role in the final prediction and a robust analysis of strengths and weaknesses may greatly influence performance. Nevertheless, this analysis is beyond the scope of this paper.

3.1 Problem Formalization

Gene Regulatory Networks. A GRN can be described by a weighted directed graph $G = (V, E)$, where V is the set of regulatory elements of the network and $E \subseteq V \times V \times [0, 1]$ is the set of regulatory interactions. The presence of an edge $\langle s, t, w\rangle \in E$ indicates that an interaction between the regulatory elements s and t is present with *confidence* value w. The number $|V|$ of regulatory elements of the GRN is referred to as its *size*. If the GRN has no uncertainty, then each edge in E has weight 1. In the problem of *GRN inference*, we are given the set of vertices V and a set of experiments describing the behavior of the regulatory elements. The goal is to accurately detect the set of regulatory interactions E.

CSP modeling. Given a set of n genes, we describe a GRN inference problem as a CSP $\langle \mathcal{X}, \mathcal{D}, \mathcal{C}\rangle$ where $\mathcal{X} = \langle x_1, \ldots, x_{n^2-n}\rangle$, and each x_k describes a regulatory relation (excluding self regulations); $\mathcal{D} = \langle D_1, \ldots, D_{n^2-n}\rangle$, with each $D_k = \{0, \ldots, 100\}$ describing the set of possible confidence values associated with the regulatory relation modeled by x_k. Values close to 0 indicate high confidence about the absence of a regulatory relation (with 0 denoting the higest condicence), whereas values close to 100 indicate high confidence about the presence

of a regulatory relation (with 100 denoting the highest confidence); $\mathcal{C}$ is a k-tuple of constraints $\langle C_1, \ldots, C_k \rangle$, where a constraint C_j over set of variables $S_j \subseteq \mathcal{X}$ expresses a restriction for the joint assignments that can be given to the variables in S_j. Constraints expressing restrictions of peculiar network topologies will be discussed following in this Section. A variable x_i is said to be *assigned* when the possible choices for its value assignments in its associated domain D_i have been reduced to a single one. We adopt the notation $d(x_i)$ to indicate the value of an assigned variable x_i. For the sake of presentation, we denote with $x_{\langle s,t \rangle}$ the variable associated with the regulatory relation *"s regulates t"* and $D_{\langle s,t \rangle}$ its domain. A solution to the above CSP defines a GRN prediction $G = (V, E)$, with $V = \{1, \ldots, n\}$ and $E = \{\langle s, t, w \rangle \mid d(x_{\langle s,t \rangle}) > 0\}$, where $w = d(x_{\langle s,t \rangle})/100$.

Constraint Modeling. The proposed CSP solution leverages the collection of GRN predictions obtained employing all the methods described in Sec. 3 by: **(1)** considerably reducing the size of the solution search space[3] and **(2)** taking into account the discrepancies among the community predictions. Furthermore we analyze various constraints that can be exploited to enforce the satisfaction of GRNs' specific properties and to take into account collective strengths and individual weaknesses of the CN predictions.

Community Network Constraints.

Let us consider a set of predictions $\mathcal{G}$ of a GRN $G = (V, E)$. We denote with G_j each prediction in the inference ensemble, and we denote with E_j the edges of E predicted by G_j with a given confidence score. We also assume that each prediction has been normalized with respect to the ensemble itself. Furthermore, let θ_d ($0 \leq \theta_d \leq 1$) be a given disagreement threshold. The `community_network` constraint over a variable $x_{\langle s,t \rangle}$ ensures that the possible confidence values to be assigned during solution search are restricted to the average confidence score of the community network predictions (`w_rank` in Alg. 1) and its pseudo first and third quantiles (`w_d` in Alg. 1) when the edge confidence scores exhibit large variance. The propagation of the `community_network` constraint, described in Alg. 1, reduces the possible values for a CSP variable to at most three. We calculate the average confidence value, `w_rank`, according to the *Borda* count election method—as presented in [13]—averaging the ranked edge confidence values assigned by each prediction. The discrepancy value, `w_d`, captures the ensemble prediction disagreement for a given edge, averaging the pairwise differences of the edge ranks associated to each prediction of the ensemble. The average confidence value and the discrepancy values within $\mathcal{G}$ are computed in line (3) of Alg. 1. Both measures are normalized in the $[0, 1] \subseteq \mathbb{R}$ interval. If the discrepancy value exceeds the discrepancy threshold θ_d and the average confidence value is not strongly informative (line 5), we force the domain $D_{\langle s,t \rangle}$ to take account of the prediction disagreement by adding a variation of `w_d`$/2$ to the average confidence value. `fd` is the nearest integer function which converts a prediction confidence value into an integer domain encoding, and it is defined as: `fd`$(x) = \lfloor 100\,x + 0.5 \rfloor$. Line 4 ensures the presence of the value `w_rank` in

[3] An upper bound for the search space of a GRN inference problem of size n is 101^{n^2}.

$D_{\langle s,t\rangle}$. For a given prediction G_j, $\omega_j^{\#}(s,t) : V \times V \to [0,1] \subseteq \mathbb{R}$ is the function ranking the prediction confidence for the edge (s,t) within the confidence values in E_j. The rank is normalized in the $[0,1] \subseteq \mathbb{R}$ interval.

Algorithm 1 `community_network`($x_{\langle s,t\rangle}$, $\mathcal{G}$, θ_d) filtering algorithm

Require: normalized $G_j \in \mathcal{G}, x_{\langle s,t\rangle}, \theta_d$

1: $J \leftarrow |\mathcal{G}|$
2: $B \leftarrow \emptyset$
3: $(\texttt{w\_rank}, \texttt{w\_d}) \leftarrow \left(\frac{1}{J}\sum_{j=1}^{J} \omega_j^{\#}(s,t), \quad \frac{1}{\binom{J}{2}} \sum_{j=1}^{J} \sum_{i=j+1}^{J} \left|\omega_j^{\#}(s,t) - \omega_i^{\#}(s,t)\right| \right)$
4: $B \leftarrow B \cup \{\texttt{fd}(\texttt{w\_rank})\}$
5: **if** $\texttt{w\_d} \geq \theta_d \ \wedge \ 0.1 < \texttt{w\_rank} < 0.9$ **then**
6: $\quad B \leftarrow B \cup \left\{ \max\left(0, \texttt{fd}\left(\texttt{w\_rank} - \frac{\texttt{w\_d}}{2}\right)\right), \min\left(100, \texttt{fd}\left(\texttt{w\_rank} + \frac{\texttt{w\_d}}{2}\right)\right) \right\}$
7: **end if**
8: $D_{\langle s,t\rangle} \leftarrow D_{\langle s,t\rangle} \cap B$

Sparsity Constraints.

It is widely accepted that the GRN machinery is controlled by a relatively small number of genes. Several state-of-the-art methods for reverse engineering GRNs encourage sparsity in the inferred networks [13]. Nevertheless, when combining predictions in a community based approach, no guarantees on the sparsity of the resulting prediction can be provided. To address this issue we introduce a sparsity constraint, which is built from two more general constraints: `atleast_k_ge` and `atmost_k_ge`. They both enforce a relation among a set of variables and ensure that among the variables involved at least (resp. at most) k of them have values greater or equal than a threshold. Formally, the constraint:

$$\texttt{atleast\_k\_ge}(k, X, \theta) : \quad \left|\{x_i \in X \mid d(x_i) > \theta\}\right| \geq k \tag{1}$$

enforces a lower bound (k) on the number of variables in X whose confidence value is greater than θ; the constraint:

$$\texttt{atmost\_k\_ge}(k, X, \theta) : \quad \left|\{x_i \in X \mid d(x_i) > \theta\}\right| \leq k \tag{2}$$

limits to at most k the variables in X with confidence value greater than θ.

The propagation of the `atmost_k_ge` constraint is exploited during the solution search to enforce the property (2) by the following:

$$\texttt{atmost\_k\_ge}(k, X, \theta) : \frac{S = \{x_i \in X \mid d(x_i) > \theta\},\ |S| = k}{\bigwedge\limits_{x_j \in X \setminus S} D_{\langle x_j\rangle} = D_{\langle x_j\rangle} \cap \{0, \ldots, \theta\}} \tag{3}$$

For the `atleast_k_ge` early failures can be detected during the solution search by checking the upper bound on the number of variables not yet instantiated which satisfy property (1).

The sparsity constraint `g-sparsity` is a global constraint over the variables in X. It enforces lower and upper bounds on the number of edges whose confidence

value is outside a given threshold. Formally, given $k_l, k_m, \theta_l, \theta_m$:

$$\texttt{atleast\_k\_ge}(k_l, X, \theta_l) \ \cap \ \texttt{atmost\_k\_ge}(k_m, X, \theta_m) \tag{4}$$

Redundant Edge Constraints.

Several state-of-the-art inference methods rely on MI or correlation techniques; the community approach adopted for this work employs *CLR* an MI-based method (see Sec. 3). One of the disadvantages of such methods is the difficulty in speculating on the directionality of a given prediction. We define a constraint that has been effective in our experiments in detecting the edge directionality based on the collective decision of the CN predictions, among the non MI- or correlation-based methods.

Let us consider a collection of predictions $\mathcal{G} = \{G_1, \ldots, G_n\}$ for a GRN $G = (V, E)$, and a non-empty set of MI- or correlation-based methods $\mathcal{H} \subseteq \mathcal{G}$. An edge (t, s) is said to be *redundant* if:

$$\forall\, G_i \in \mathcal{G} \setminus \mathcal{H}\,. \quad \omega_i(s,t) > \omega_i(t,s) + \beta \tag{5}$$

where $\omega_i(s,t) : V \times V \to [0,1] \subseteq \mathbb{R}$ expresses the confidence value of the edge (s,t) in the prediction G_i, and $\beta > 0$ is a positive real value. Given a redundant edge (t,s) we call the edge (s,t) the *required* edge. The `redundant_edge` constraint enforces a relation between two variables $x_{\langle s,t\rangle}$ and $x_{\langle t,s\rangle}$. Let X_R be the set of all the required and redundant variables.[4] For a pair of variables $x_{\langle s,t\rangle}, x_{\langle t,s\rangle} \in X_R$ the constraint:

$$\texttt{redundant\_edge}(x_{\langle s,t\rangle}, x_{\langle t,s\rangle}, \theta_e, L): \quad x_{\langle s,t\rangle} > \theta_e \wedge \max(D_{\langle t,s\rangle}) < L \tag{6}$$

ensures that the confidence value assigned to the required variable $x_{\langle s,t\rangle}$ is greater than a given threshold value $\theta_e \in \mathbb{N}$, with $0 \leq \theta_e \leq 100$, and that the domain of the redundant edge variable $x_{\langle t,s\rangle}$ contains no values greater than L. The propagation of the `redundant_edge` constraint is exploited during the solution search to enforce property (6):

$$(x_{\langle s,t\rangle}, x_{\langle t,s\rangle}, \theta_e, L) : \frac{\min(D_{\langle s,t\rangle}) > \theta_e,\ \max(D_{\langle t,s\rangle}) \geq L}{D_{\langle t,s\rangle} = \ D_{\langle t,s\rangle} \cap \{0, \ldots, L-1\}} \tag{7}$$

Transcriptor Factor Constraints.

Often, GRN specific information, such as sequence DNA-binding TFs or functional activity of a set of genes, is available from public sources (e.g., DBD [12]). Moreover, several studies show that similar mRNA expression profiles are likely to be regulated via the same mechanisms [1]. Not every method may be designed to handle such information, or this information can become available in an incremental fashion, and hence not suitably usable by prediction methods. We propose constraints that can directly incorporate such information in the CN model.

[4] $x_{\langle s,t\rangle}$ is required/redundant if the corresponding edge (s,t) is required/redundant.

A regulatory element is a *Transcription-factor (TF)* if it regulates the production of other genes. This property is described through a relation on the out-degree of the involved gene for those edges with an adequate confidence value. The `transc-factor` constraint over a gene s is enforced by an $\texttt{atleast\_k\_ge}(k, X_s, \theta)$ constraint with $X_s = \{x_{\langle s,u\rangle} \in \mathcal{X} \mid u \in V\}$, and k representing the co-expressing degree, i.e., the number of genes targeted by the TF.

Multiple TFs can cooperate to regulate the transcription of specific genes; these are referred to as *Co-regulators*. When this information is available it can be expressed by a `coregulator` constraint. The latter involves two TFs, s' and s''; it enforces a relation over a set of variables X, to guarantee the existence of at least k elements that are co-regulated by both s' and s'' for which an interaction is predicted with confidence value greater than θ ($0 < \theta \leq 1$). Formally:

$$\texttt{coregulator}(k, X, \theta): \qquad \forall x_{\langle s',t'\rangle}, x_{\langle s'',t''\rangle} \in X$$
$$\left| \{(s', s'', t') \mid s' \neq s'' \wedge t' = t'' \wedge d(x_{\langle s',t'\rangle}) > \theta \wedge d(x_{\langle s'',t''\rangle}) > \theta\} \right| \geq k \qquad (8)$$

Search Strategy. The proposed modeling of GRN prediction allows a great degree of flexibility in exploring the solution space. We implement two search strategies: **(1)** a classical prop-labeling tree exploration (DFS), where constraint propagation phases are interleaved with non-deterministic branching phases used to explore different value assignments to variables [2], and **(2)** a Monte Carlo (MC)-based prop-labeling tree exploration, which performs a random value assignment to each variable. We set a trial limit for the MC-based solution and a solution number limit for both strategies.

GRN Consensus. A challenge in GRN inference is the absence of a widely accepted objective function to drive the solution search. We decided to generate an ensemble of m solutions and propose three criteria to compute the final GRN prediction. Given a set of m solutions $S = \{S_1, \ldots, S_m\}$, where each $S_i = \langle a_1^i, \ldots, a_{n^2-n}^i \rangle$, let $S|_{x_k} = \bigcup_{i=1}^{m} \{a_k^i\}$ be the set of values assigned to the variable x_k in the different solutions, and $\texttt{freq}(a, k)$ be the function counting the occurrences of the value a among the assignments to x_k in the solution set. The consensus value a_k^* associated with the variable x_k is computed by:

- *Max Frequency:* $a_k^* = \arg\max_{a \in S|_{x_k}} (\texttt{freq}(a, k))$. This estimator rewards the edge confidence value appearing with the highest frequency in the solution set. The intuition is that edge-specific confidence values appearing in many solutions may be important for the satisfaction of the constraints.
- *Average:* $a_k^* = \frac{1}{m} \sum_{i=1}^{m} a_k^i$. It computes the average edge consensus among all solution in order to capture recurring predictive trends.
- *Weighted average:* $a_k^* = \frac{1}{\sum_{a \in S|_{x_k}} \texttt{freq}(a,k)^2} \sum_{a \in S|_{x_k}} \texttt{freq}(a, k)^2 a$. This estimator combines the intuitions of the two above by weighting the average edge confidence by the individual quadratic value frequencies.

We also investigated some potential *global* measures—i.e., acting collectively on the prediction values of all edges—in terms of the solution which minimizes the

Hamming distance among all edge prediction values. These global measures were always outperformed by the three estimators discussed above.

3.2 A Case Study

We provide an example to illustrate our approach. We adopt the "E.coli2" network from the 10-node DREAM3 subchallenge [14] (Fig. 1). The target network has two co-regulators (G_1 and G_5) which are in turn regulated by gene G_9. The network has 15 interactions.

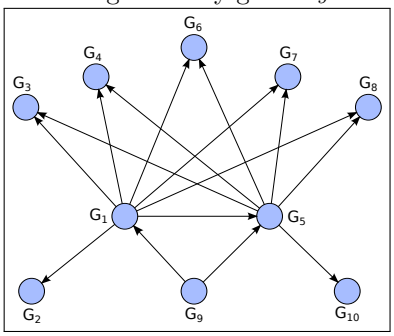


Fig. 1: An extract of E.coli GRN

Phase 1: CN Predictions. The inference ensemble was generated by feeding the datasets provided within the DREAM3 challenge to each of the four methods adopted in the community network schema (see Sec. 3). In addition, we generate a CN_{rank} as done in [13], and used it to restrict the values to be assigned to the variables subjected to the `community_network` constraint (see Sec. 3.1), and as a comparison network for evaluation.

Phase 2: Modeling the CSP. We apply a `community_network` constraint to each variable of the CSP with disagreement threshold $\theta_d = 0.20$. Its application reduced the possible values to assign to each variable to 1 for 64 cases, and to 3 for the others. As the inference ensemble adopted employs methods that may suffer from the *edge redundancy* problem, we impose a `redundant_edge` constraint for all the edge pairs $(s,t),(t,s)$ that satisfy the definition with $\beta = 0.15$ as:

$$\texttt{redundant\_edge}(x_{\langle s,t\rangle}, x_{\langle t,s\rangle}, 75, 50). \qquad (r)$$

This constraint was able to reduce the value uncertainty for two additional variables—only one element in their domains can possibly satisfy the conditions above for any value choice of the required edge variable.
A sparsity constraint was imposed at a global level as:

$$\texttt{g-sparsity}: \quad \texttt{atleast\_k\_ge}(10, \mathcal{X}, 65) \cap \texttt{atmost\_k\_ge}(25, \mathcal{X}, 65). \qquad (s)$$

Phase 3: Generating the Consensus. We performed $1,000$ Monte Carlo samplings and return all the solutions found, which we refer to as *Constrained Community Networks (CCNs)*. To illustrate the effect of constraints integration on the CCNs we consider the best prediction returned by each CSP exhibiting a different combinations of the imposed constraints. We plot it as a graph containing all and only the edges of highest confidence necessary to make such graph weakly connected. These resulting predictions are illustrated in Fig. 2, together with the CN_{rank} (top-right). In each network the green edges (thick with filled arrows) denote the true positive predictions, the red edges (with empty arrows) denote the false positive predictions, and the gray (dotted) edges denote the false negatives.

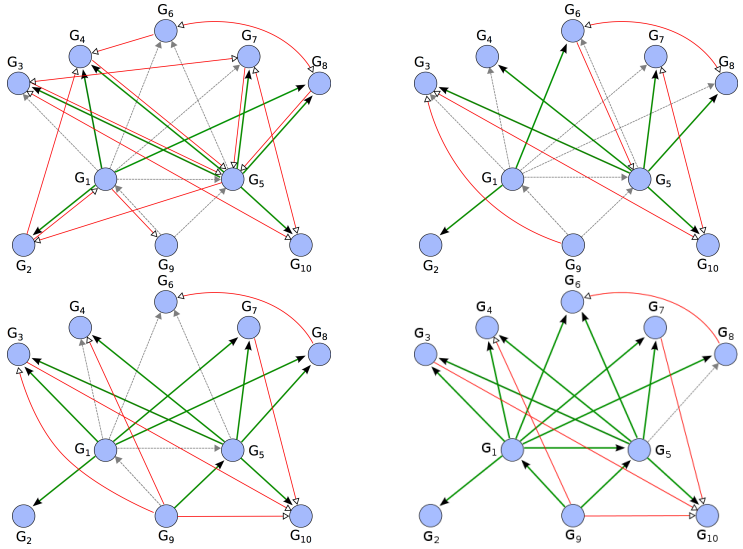


Fig. 2: The CN_{rank} consensus (top-right) and the CCN prediction after the integration of the redundant edge and sparsity constraints (top-left), the TF constraints (bottom-left) and Co-factor constraint (bottom-right).

The results are also summarized in Table 1, where we report the AUC scores [3] for the best prediction (CCN_{best}) generated and for each CCN generated by the evaluation criteria presented in in Sec. 3.1.

Phase 4: Employing network specific information. Let us now model some specific information about the target network. The target network includes three TFs: G_1, G_5, G_9, which can be modeled via three `transc_factor` constraints as:

$$\texttt{atleast\_k\_ge}(2, N_1, 85), \texttt{atleast\_k\_ge}(2, N_5, 85), \texttt{atleast\_k\_ge}(2, N_9, 85) \qquad (t)$$

with $N_i = \{x_{\langle i,s\rangle} \,|\, (\forall G_j \in \mathcal{G})\ \omega_j(i,s) > 0.10\}$. Note that $\omega_j(i,s)$ is the prediction confidence assigned to edge (i,s) by the inference method J in the prediction G_j. Fig. 2 and Table 1 show the improvements using the latter formalization.

Finally, speculation about the activity of genes G_1 and G_2 as co-regulators can be captured via a `coregulator` constraint expressed by:

$$\texttt{coregulator}(1, V, 75) \qquad (c)$$

with V defined as in (8) with $s' = 1, s'' = 5$. As shown in Fig. 2 and in Table 1, the application of this additional constraint produces further improvements. A discussion on the parameter selection is presented in Section 4.

4 Results and Discussions

Benchmark Networks & Datasets. The proposed approach has been tested using benchmarks from the DREAM3 and DREAM4 competitions [14, 17]. In

Constr.	CN_{rank}	CCN_{best}	$\text{CCN}_{\text{max-f}}$	CCN_{avg}	$\text{CCN}_{\text{w-avg}}$
r	0.7271	0.8036	0.7556	0.7644	0.7751
s	0.7271	0.8044	0.7529	0.7164	0.7591
r, s	0.7271	0.8453	0.7778	0.7609	0.7760
r, s, t	0.7271	0.9209	0.7458	0.8489	0.8587
r, s, t, c	0.7271	0.9378	0.7929	0.8622	0.8729

Table 1: The effects of constraint integration on the AUC scores for the "Ecoli2" CCNs.

both challenges the network topologies were obtained by extracting subnetworks from transcriptional regulatory networks of E. coli [19] and S. cerevisiae [18], including parts of the network with cycles and removing auto-regulatory interactions. The dynamics of the networks were simulated using a detailed kinetic model wich simulates both transcription and translation (see [14] and [17] for a more detailed description). The datasets adopted to produce the predictions via the methods composing the CN ensemble include the steady state expression levels for wild type and for knockouts of every gene and the time-series data (a variable number of trajectories, depending on the size of the network). We generate 110 predictions: 50 of size 10, 25 of size 50, and 50 of size 100. For each problem we generate four consensus from each of the community methods described in Sec. 3 together with a consensus network constructed by averaging individual edges ranks (CN_{rank}).

Validation. To measure prediction accuracy against the corresponding reference network we adopted the AUC score [3], which relates the ratio between the *true positive* rate and the *false positive* rate. An AUC value of 0.5 corresponds to a random prediction, whereas a value of 1.0 indicates perfect prediction.

Settings. The experiments were performed on a generic Finite Domain CSP solver, which explores the space of the possible solutions via a classic prop-labeling tree [2]. The system explores the queue of constraints using techniques based on the notion of event (a change in the domain of a variable) and it is implemented in C++. We emphasize that the focus of this paper is on the formalization of the GRN inference in a CP context and its feasibility study. Therefore we leave the performance comparison of the CSP model on other well known constraint solvers (e.g., Gecode, JaCoP, etc.[5]) as object of further studies.

For each experiment we perform a $1,000$ Monte Carlo samplings and generate the CCNs using all the solution found. We observed that the DFS was always outperformed by the MC search and therefore not reported. The `community_network` constraint was enabled by default on every variable of the problem. The disagreement threshold θ_d was set to be as the average of the discrepancy values `w_d` across all the edges of the network (see Alg. 1).

The `g_sparsity` (s) and `redundant_edge` (r) constraints have been enabled for all the experiments. To guide the parameter selection for the sparsity constraint, we set thresholds θ_l and θ_m (see Eq. (4)) to be respectively the n-th

[5] see `http://www.minizinc.org/`

and the $n\,log(n)$-th highest values in the $\mathrm{CN}_{\mathrm{rank}}$ edge list. Hence, we identify the bounds k_l and k_m which would make the constraint unsatisfiable and use them to set the sparsity parameters. In this way, k_l and k_m are set so that they are bounded, respectively, above by $|\{x_i|x_i \in \mathcal{X} \wedge \max(D_{x_i}) > \theta_l\}|$, and below by $|\{x_i|x_i \in \mathcal{X} \wedge \min(D_{x_i}) > \theta_m\}|$, provided that $k_l < k_m$. The closer are their values to the respective bounds, the more restrictive is the constraint. For the `redundant_edge` constraints we impose the threshold θ_e as the average of all the required edges confidence values composing the reduced ensemble set $\mathcal{G} \setminus \mathcal{H}$, L to be the minimum confidence value among all the ones associated to the redundant edges in $\mathcal{G} \setminus \mathcal{H}$ (see Eq. (6)) and the value β was set to be the mean among all the differences of the confidence values of the pair of edges (s,t) and (t,s) in each of the network of the CN ensemble $\mathcal{G}$ (see Eq. (5)). We observed that such settings, for both search and constraints parameters, produced stable results across the whole benchmark set, which in turn was designed to capture a variety of network topologies to assess GRN inference algorithms. We generate four CN consensus (CCNs), one for each estimator described in Sec. 3.1 ($\mathrm{CCN}_{\text{max-f}}$, $\mathrm{CCN}_{\text{avg}}$, $\mathrm{CCN}_{\text{w-avg}}$) and $\mathrm{CCN}_{\text{best}}$, as best prediction with respect to the AUC score, and compare them against $\mathrm{CN}_{\text{rank}}$. The estimators-based CCNs may outperform the $\mathrm{CCN}_{\text{best}}$ as they are not elements of the set of solutions returned. We experimentally verified their constraints consistency, which was always satisfied.

Experiments. We first focused on examining the predicted CCNs using the sparsity and redundant edge constraints to leverage community-method features and networks properties. We categorize the benchmarks by DREAM edition and size (n), and average their respective AUC scores. Table 2 reports the percentage of the average AUC improvements for the best $\mathrm{CCN}_{\text{best}}$ and best $\mathrm{CCN}_{\text{w-avg}}$ with respect to $\mathrm{CN}_{\text{rank}}$ across all the benchmarks (first two rows). Our choice of reporting only the weighted average estimator, among all those defined in Sect. 3.1, is driven by the observation that the former offers higher stability to parameter tuning and in general outperforms the other two. The CCNs achieved higher average prediction accuracy with respect to $\mathrm{CN}_{\text{rank}}$ for small and medium size networks, while performance improvements decreased for bigger networks. This is probably due to the high permissiveness of the CSP model for bigger networks. We show next that the application of additional constraints overcomes such effect.

We extended the set of constraints to include specific knowledge about individual networks. We enabled the transcriptor-factor constraint over a set of randomly selected genes which were verified TFs in the target networks. The TFs set sizes were chosen to be at most 30%, 15% and 10%, respectively, for the networks of size 10, 50 and 100; the co-expressing degree was set as $\lfloor k{=}log(n) \rfloor$ and θ as the n-th highest value in the $\mathrm{CN}_{\mathrm{rank}}$ edge list. We performed 5 repetitions and for each TF t the set of possible regulators X has been chosen among the variables $x_{\langle t,s \rangle}$ such that $\omega^{\#}(t,s) > 0.25$. Moreover to promote such constraint we increased the uncertainty for the regulation $x_{\langle t,s \rangle}$ such that $\max D_{\langle t,s \rangle} \leq 50$. These parameters were chosen in accordance to the study presented in [1] show-

	Dream3 10	**Dream4 10**	**Dream3 50**	**Dream3 100**	**Dream4 100**
$\text{CCN}^{sr}_{\text{best}}$	+10.52	+7.01	+3.63	+1.75	−0.17
$\text{CCN}^{sr}_{\text{w-avg}}$	+3.01	+1.96	+1.49	+0.43	+0.05
$\text{CCN}^{srt}_{\text{best}}$	+15.02	+8.43	+8.49	+4.13	+2.29
$\text{CCN}^{srt}_{\text{w-avg}}$	+5.42	+2.48	+6.32	+3.21	+4.21

Table 2: Average AUC score improvements (in percentage) with respect to CN_{rank}

ing that genes with high correlated mRNA expression profiles are more likely to share a common TF binder.

The integration of additional knowledge produced improvements of the AUC scores for both the *best* and the *weighted average* measures—see the last two rows of Table 2. A complete summary of the results is reported in Table 3, where we report the AUC scores for the benchmark test set categorized by DREAM challenge and network size. In order from left to right, Table 3 shows the AUC scores for the networks predicted by each of the methods employed in the CNs—CLR, GEN (Genie3), TIG (Tigress), INF (Inferelator); the CN_{rank} predictions; the CCNs results computed by enabling the redundant edge and the sparsity constraint; and the CCNs returned by extending the model to include the transcriptor factor constraints. The table reports the best predictions found by each estimator. The best average AUC scores among the four individual methods composing the CNs are underlined. The average CCNs AUC scores outperforming CN_{rank} are highlighted in bold.

The CCNs outperformed in general CN_{rank}, and $\text{CCN}_{\text{w-avg}}$ offers larger improvement for the bigger networks with respect to the version without the TF constraint. This supports our hypothesis that the addition of biological knowledge can better guide the predictions even if re-adopting the same inference ensemble. From a preliminary analysis of the incorrect predicted regulations supported by the TF constraint we observed that many of the erroneous inferences relate genes located in different regions of the graph. This effect could be attenuated by clustering the consensus graph for different connectivities, and targeting the TF constraint on the same cluster (if no prior knowledge on the specific TF is given). We plan to investigate this direction as future work.

5 Conclusions

In this paper we introduced a novel approach based on CP to infer GRNs by integrating a collection of predictions in a CN. Our approach does not impose any hypothesis on the datasets adopted nor on the type of inference methods. We introduced a class of constraints able to (1) enforce the satisfaction of GRNs' specific properties and (2) take account of the community prediction collective agreements on each edge, and of method-specific limitations. Experiments over a set of 110 benchmarks proposed in past editions of the DREAM challenges show that our approach can consistently outperform the consensus networks constructed by averaging individual edges ranks, as proposed in [13] (up to 15.02% for small networks and 4.13% for big networks). We have shown how knowledge specific about target networks could provide further improvements in the AUC

measure. This was possible as our model encourages the modular integration of biological knowledge, in form of logical rules, and proposes a set of candidate solutions satisfying the imposed constraints rather than an arbitrary one chosen among many. We introduced three estimators to compute a consensus from the set of consistent candidates and verified their consistency among the imposed constraints. We plan to investigate new optimization measures by taking into account local and global network properties, e.g., the number of specific network motifs in a target GRN region, or the scale free degree in a given a portion of the graph. This can be achieved by including soft constraints in our model. On the CP side, we will extend existing constraints, for instance by studying the most likely set where a TF constraint could be targeted, and model new constrains and propagators to capture different type of biological knowledge, such us information about cell conditions at the time of the experiments.

Network	CLR	GEN	TIG	INF	CN_{rank}	CCN^{sr}_{best}	$CCN^{sr}_{\text{max-f}}$	CCN^{sr}_{avg}	$CCN^{sr}_{\text{w-avg}}$	CCN^{srt}_{best}	$CCN^{srt}_{\text{max-f}}$	CCN^{srt}_{avg}	$CCN^{srt}_{\text{w-avg}}$
	DREAM3 Size 10												
Ecoli1	0.6801	0.7399	0.7560	0.6565	0.7192	0.8101	0.7443	0.7422	0.7319	0.8642	0.7756	0.7664	0.7664
Ecoli2	0.7080	0.7537	0.6009	0.7049	0.7271	0.8453	0.7778	0.7778	0.7778	0.9209	0.8444	0.8676	0.8456
Yeast1	0.6637	0.7212	0.7950	0.5762	0.7413	0.8550	0.7625	0.7350	0.7637	0.8613	0.7325	0.7688	0.7685
Yeast2	0.6054	0.6086	0.6831	0.5532	0.6191	0.7145	0.6560	0.6123	0.6640	0.7606	0.6794	0.6646	0.6557
Yeast3	0.5231	0.5869	0.5842	0.5294	0.5428	0.6507	0.5742	0.5588	0.5622	0.6935	0.6457	0.5822	0.5842
Avg	0.6360	0.6821	0.6838	0.6040	0.6699	**0.7700**	**0.7023**	**0.6852**	**0.7000**	**0.8201**	**0.7355**	**0.7299**	**0.7241**
	DREAM4 Size 10												
Net1	0.7560	0.8000	0.6640	0.6258	0.7493	0.7858	0.7244	0.7324	0.7324	0.8471	0.7680	0.7600	0.7644
Net2	0.6664	0.6993	0.6241	0.6541	0.6943	0.7981	0.7618	0.7188	0.7188	0.8218	0.7331	0.7365	0.7348
Net3	0.7284	0.6862	0.6471	0.8502	0.8018	0.8622	0.8396	0.8329	0.8364	0.8649	0.8062	0.8329	0.8338
Net4	0.8162	0.7632	0.7692	0.8441	0.8501	0.9171	0.8771	0.8601	0.8601	0.9201	0.8511	0.8701	0.8721
Net5	0.8632	0.8761	0.8173	0.7473	0.8718	0.9541	0.9006	0.9006	0.9038	0.9348	0.8974	0.8921	0.8857
Avg	0.7660	0.7649	0.7043	0.7443	0.7934	**0.8635**	**0.8207**	**0.8090**	**0.8103**	**0.8777**	**0.8112**	**0.8183**	**0.8182**
	DREAM3 Size 50												
Ecoli1	0.6326	0.6631	0.6750	0.6124	0.6678	0.7317	0.6801	0.6871	0.6991	0.7919	0.7195	0.7724	0.7740
Ecoli2	0.6609	0.7227	0.6566	0.6719	0.7010	0.7214	0.7083	0.7075	0.7064	0.8205	0.7481	0.7954	0.7954
Yeast1	0.5880	0.7227	0.6434	0.6009	0.6539	0.6817	0.6545	0.6496	0.6586	0.7205	0.6520	0.6782	0.6842
Yeast2	0.5725	0.6672	0.6049	0.5942	0.6273	0.6609	0.6466	0.6429	0.6442	0.6866	0.6780	0.6712	0.6725
Yeast3	0.5996	0.6236	0.5934	0.6065	0.6181	0.6536	0.6236	0.6236	0.6340	0.6731	0.6360	0.6580	0.6579
Avg	0.6107	0.6799	0.6347	0.6172	0.6536	**0.6899**	**0.6626**	**0.6621**	**0.6685**	**0.7385**	**0.6867**	**0.7150**	**0.7168**
	DREAM3 Size 100												
Ecoli1	0.6937	0.7624	0.7359	0.7353	0.7704	0.7831	0.7647	0.7716	0.7746	0.8131	0.7921	0.8128	0.8128
Ecoli2	0.6534	0.7427	0.6507	0.6843	0.7152	0.7353	0.7179	0.7186	0.7226	0.7826	0.7479	0.7693	0.7692
Yeast1	0.6056	0.7632	0.6158	0.6879	0.6975	0.7141	0.7031	0.6984	0.7014	0.7337	0.7086	0.7181	0.7187
Yeast2	0.5578	0.6548	0.5493	0.6074	0.6116	0.6371	0.6116	0.6164	0.6134	0.6524	0.6201	0.6394	0.6438
Yeast3	0.5581	0.5823	0.5321	0.5629	0.5596	0.5723	0.5605	0.5629	0.5642	0.5794	0.5684	0.5633	0.5704
Avg	0.6137	0.7010	0.61676	0.6556	0.6709	**0.6884**	**0.6716**	**0.6736**	**0.6752**	**0.7122**	**0.6874**	**0.7006**	**0.7030**
	DREAM4 Size 100												
Net1	0.6710	0.7495	0.6744	0.7548	0.7829	0.7797	0.7523	0.7788	0.7785	0.7975	0.7606	0.8273	0.8273
Net2	0.6812	0.7083	0.6338	0.7107	0.7511	0.7396	0.7228	0.7549	0.7535	0.7773	0.7637	0.8085	0.8085
Net3	0.6301	0.6945	0.6203	0.7240	0.7158	0.7254	0.6956	0.7191	0.7200	0.7455	0.7288	0.7454	0.7492
Net4	0.6532	0.6853	0.6121	0.7559	0.7408	0.7380	0.7200	0.7429	0.7429	0.7604	0.7230	0.7652	0.7742
Net5	0.6583	0.7047	0.6278	0.6719	0.7222	0.7220	0.7054	0.7198	0.7205	0.7466	0.7241	0.7626	0.7643
Avg	0.6588	0.7085	0.6337	0.7235	0.7426	0.7409	0.7192	**0.7431**	**0.7431**	**0.7655**	0.7400	**0.7818**	**0.7847**

Table 3: AUC scores comparison.

References

1. D. Allocco et al. Quantifying the relationship between co-expression, co-regulation and gene function. *BMC Bioinformatics*, 5(1):18+, Feb. 2004.
2. K. Apt. *Principles of Constraint Programming*. Cambridge University Press, 2009.
3. P. Baldi et al. Assessing the accuracy of prediction algorithms for classification: an overview. *Bioinformatics*, 16(5):412–424, 2000.
4. F. Corblin, E. Fanchon, and L. Trilling. Applications of a formal approach to decipher discrete genetic networks. *BMC Bioinformatics*, 11:385, 2010.
5. J. J. Faith et al. Large-scale mapping and validation of escherichia coli transcriptional regulation from a compendium of expression profiles. *PLoS Biol*, 5(1), 2007.
6. J. Fromentin et al. Analysing gene regulatory networks by both constraint programming and model-checking. *Conf Proc IEEE Eng Med Biol Soc.*, 4595–8, 2007.
7. M. Gebser et al. Detecting inconsistencies in large biological networks with answer set programming. *CoRR*, abs/1007.0134, 2010.
8. A. Greenfield et al. Dream4: Combining genetic and dynamic information to identify biological networks and dynamical models. *PLoS ONE*, 5(10):e13397, 10 2010.
9. A. Haury et al. Tigress: Trustful inference of gene regulation using stability selection. *BMC Syst Biol*, 6(1):145, 2012.
10. V. A. Huynh-Thu et al. Inferring regulatory networks from expression data using tree-based methods. *PLoS ONE*, 5(9):e12776, 09 2010.
11. S. Kim et al. Dynamic Bayesian network and nonparametric regression for modeling of GRNs from time series gene expression data. *Biosystems*, 104–113, 2003.
12. S. K. Kummerfeld and S. A. Teichmann. DBD: a transcription factor prediction database. *Nucl. Acids Res.*, 34(suppl_1):D74–81, 2006.
13. D. Marbach et al. Wisdom of crowds for robust gene network inference. *Nat Meth*, 9(8):796–804, Aug. 2012.
14. D. Marbach et al. Revealing strengths and weaknesses of methods for gene network inference. *Proc Natl Acad Sci U S A*, pages 6286–6291, 2010.
15. A. A. Margolin et al. Aracne: An algorithm for the reconstruction of gene regulatory networks in mammalian cellular context. *BMC Bioinformatics*, 7(S1), 2006.
16. N. Meinshausen et al. Stability selection *J, of the Royal Stat. Soc., S. B* 72(4), 417–473, 2010.
17. R. J. Prill et al. Towards a rigorous assessment of systems biology models: The dream3 challenges. *PLoS ONE*, 5(2):e9202, 02 2010.
18. Reguly et al. Comprehensive curation and analysis of global interaction networks in Saccharomyces cerevisiae. *Journal of biology* 4(5):11+, 2006
19. Shen-Orr et al. Network motifs in the transcriptional regulation network of Escherichia coli. *Nature Genetics* 1(31):1061–4036, 2002.
20. A. Sîrbu et al. Integrating heterogeneous gene expression data for gene regulatory network modelling. *Theory in Biosciences*, 131(2):95–102, 2012.
21. T. Soh and K. Inoue. Identifying necessary reactions in metabolic pathways by minimal model generation. In *ECAI*, pages 277–282, IOS Press, 2010.
22. R. Tibshirani Regression Shrinkage and Selection Via the Lasso *J. of the Royal Stat. Soc., S. B*, 58, 267–288, 1994.
23. S. Videla et al. Revisiting the training of logic models of protein signaling networks with asp. In *CMSB*, pages 342–361, 2012.
24. X. Zhou et al. *Genomic Networks: Statistical Inference from Microarray Data*. Wiley, 2006.

Recursive Best-First AND/OR Search with Overestimation for Genetic Linkage Analysis

Akihiro Kishimoto and Radu Marinescu

IBM Research – Ireland
akihirok,radu.marinescu@ie.ibm.com

Abstract. The paper presents and evaluates the power of limited memory best-first search over AND/OR search spaces for genetic linkage analysis. We propose *Recursive Best-First AND/OR Search with Overestimation* (RBFAOO), a new algorithm that explores the search space in a best-first manner while operating with restricted memory. In addition, we develop a simple overestimation technique aimed at minimizing the overhead associated with re-expanding internal nodes. Our preliminary experiments show that RBFAOO is often superior to current state-of-the-art approaches, especially on very hard problem instances.

1 Introduction

Graphical models such as belief or constraint networks are a widely used representation framework for reasoning with probabilistic and deterministic information. These models use graphs to capture conditional independencies between variables, allowing a concise knowledge representation as well as efficient graph-based query processing algorithms. Optimization tasks such as finding the most likely state of a belief network or finding a solution that violates the least number of constraints can be defined within this framework and are typically tackled with either search or inference algorithms [1]. The most common search scheme is the *depth-first branch and bound*. Its use for finding exact solutions was studied and evaluated extensively, especially in the context of AND/OR search spaces that are sensitive to the underlying problem structure [2, 3].

Meanwhile, *best-first* search algorithms, despite their better time efficiency than depth-first search [4], are largely ignored in practice primarily due to their inherently enormous memory requirements [3]. Furthermore, an important best-first search property, avoiding the exploration of unbounded paths, seems irrelevant to optimization in graphical models where all solutions are at the same depth (ie, the number of variables).

This paper aims at inheriting advantages of both depth-first and best-first search schemes in graphical models. We introduce RBFAOO, a new algorithm that explores the context-minimal AND/OR search graph associated with a graphical model in a best-first manner (even with non-monotonic heuristics) while operating within restricted memory. RBFAOO extends Korf's Recursive Best First Search (RBFS) [5] to graphical models and thus uses a threshold controlling technique to drive the search in a depth-first like manner while using the available memory to cache and reuse partial search results. In addition, RBFAOO employs an overestimation technique designed to further reduce the high overhead caused by re-expanding internal nodes. RBFAOO is also related to

the AND/OR search algorithms based on Allis et al.'s proof/disproof numbers [6] (eg, df-pn$^+$ [7]) which are very popular for solving two-player zero-sum games [8]. However, one crucial difference between games and graphical models is that game solvers ignore the solution cost. The latter algorithms, therefore, do not come with optimality guarantees.

We focus the empirical evaluation on the task of finding the maximum likelihood haplotype configuration of a pedigree in genetic linkage analysis. Our preliminary results on a set of difficult pedigree networks show that RBFAOO improves dramatically over the most competitive state-of-the-art solvers, especially on very hard problem instances and when guided by relatively inaccurate heuristics.

2 Background

2.1 Belief Networks and their AND/OR Search Spaces

Belief Networks provide a formalism for reasoning about partial beliefs under conditions of uncertainty. A belief network is a tuple $\mathcal{B} = \langle \mathbf{X}, \mathbf{D}, G, \mathbf{P} \rangle$, where $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a set of random variables, $\mathbf{D} = \{D_1, \ldots, D_n\}$ is the set of the corresponding discrete valued domains, G is a directed acyclic graph over $\mathbf{X}$ and $\mathbf{P} = \{p_1, \ldots, p_n\}$, where $p_i = P(X_i|pa(X_i))$ ($pa(X_i)$ are the parents of X_i in G) denote *conditional probability tables* (CPTs). The belief network represents a joint probability distribution over $\mathbf{X}$ having the product form $P_{\mathcal{B}}(\bar{x}) = \prod_{i=1}^{n} P(x_i|x_{pa_i})$, where an assignment $(X_1 = x_1, \ldots, X_n = x_n)$ is abbreviated to $\bar{x} = (x_1, \ldots, x_n)$, where $pa(X_i) = x_{pa_i}$, and where x_S denotes the restriction of a tuple x over a subset of variables S. An evidence set e is an instantiated subset of variables. The primary optimization task over belief networks is finding the *most probable explanation* (MPE), namely finding a complete assignment to all variables having maximum probability, given the evidence. In practice, the MPE is typically solved as a minimization task by taking the negative log of the probability values. Namely, it is equivalent to finding $\mathrm{argmin}_{\mathbf{X}} \sum_{i=1}^{n} -\log P(X_i|pa(X_i))$.

The concept of AND/OR search spaces for graphical models has been introduced to better capture the problem structure [9]. Given a belief network $\mathcal{B} = \langle \mathbf{X}, \mathbf{D}, G, \mathbf{P} \rangle$ and a pseudo tree $\mathcal{T}$ [10] of G that captures problem decomposition, an AND/OR search graph based on $\mathcal{T}$ has alternating levels of OR nodes corresponding to the variables and AND nodes corresponding to the values of the OR parent's variable, with edges weighted according to $\mathbf{P}$. Identical subproblems, identified by their *context* (the partial instantiation that separates the subproblem from the rest of the network), can be merged, yielding an *AND/OR search graph* [9]. Merging all context-mergeable nodes yields the *context-minimal* AND/OR search graph. The state of the art algorithms are AND/OR Branch-and-Bound (AOBB) and AND/OR Best-First (AOBF) [2, 3] that utilize the mini-bucket heuristics which are admissible and consistent [11].

2.2 Genetic Linkage Analysis

In human genetic linkage analysis [12], the *haplotype* is the sequence of alleles at different loci inherited by an individual from one parent. The two haplotypes (maternal

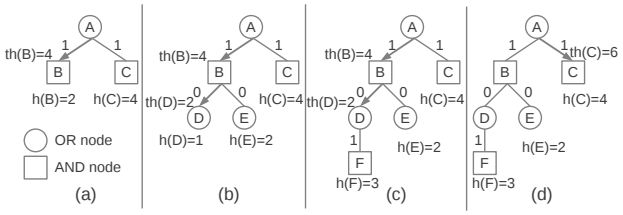


Fig. 1. Snapshot of RBFAOO search with no overestimation

and parental) of an individual constitute this individual's *genotype*. When genotypes are measured by standard procedures, the result is a list of unordered pairs of alleles, one pair for each locus. The *maximum likelihood haplotype* problem consists of finding a joint haplotype configuration for all members of the pedigree which maximizes the probability of data. The pedigree can be represented as a belief network with three types of random variables: *genetic loci* variables, which represent the genotypes of the individuals in the pedigree (two loci variables per individual per locus, one for the parental allele and one for maternal allele), *phenotype* variables, and *selector* variables which are auxiliary variables used to represent the gene flow in the pedigree. The CPTs that correspond to the selector variables are parametrized by the *recombination ratio* θ [13]. The remaining tables contain only deterministic information. It can be shown that given the pedigree data, the haplotyping problem is equivalent to computing the most probable explanation of the corresponding belief network (see also [13, 14] for more details).

3 Recursive Best-First AND/OR Search with Overestimation

Our RBFAOO uses a threshold controlling technique presented in [5, 7] to transform best-first AO* [15] into depth-first search. Although RBFAOO may be regarded as a special case of df-pn$^+$ [7], RBFAOO additionally uses an overestimation technique to possibly find a suboptimal solution and then refine it to an optimal one. Overestimation plays an essential role in enhancing RBFAOO's performance by avoiding a high overhead of re-expanding internal nodes, while theoretically guaranteeing solution optimality.

RBFAOO inherits AOBB and AOBF's advantages while overcoming their limitations. As in AOBB, in practice, RBFAOO requires a smaller amount of memory than AOBF and can allocate the remaining memory to a cache table to enhance search. When RBFAOO fills up the cache table, an effective batch-based replacement scheme (eg, [16]) discards useless table entries so that RBFAOO can save new results there. As in AOBF, it tries to perform best-first search and tends to expand fewer nodes than AOBB.

Before explaining RBFAOO in detail, we give an overview of the threshold controlling scheme that makes RBFAOO behave similarly to AO*. Assume that the weight from an OR node to an AND node is 1, the weight from an AND node to an OR node is 0, and a heuristic function h returns values as shown in Figure 1. Let $q(n)$, called *q-value*, be a lowerbound of the solution cost at node n and $th(n)$ be RBFAOO's threshold

at n. RBFAOO keeps examining the subtree rooted at n until either $q(n) > th(n)$ or the subtree is solved optimally. In Figure 1(a), RBFAOO selects B to expand, because $w(A,B)+q(B) = w(A,B)+h(B) = 3 < w(A,C)+q(C) = w(A,C)+h(C) = 5$. It sets $th(B) = w(A,C) + q(C) - w(A,B) = 4$ to indicate that C, which currently has the second smallest q-value, becomes the best child (ie, $w(A,B) + q(B) > w(A,C)+q(C)$ holds) if $q(B) > th(B)$. Then, RBFAOO expands B and updates $q(B)$ by using the q-values of B's children (Figure 1(b)). Because $q(B) = q(D) + q(E) = h(D) + h(E) = 3$, $q(B) \leq th(B)$ still holds. Hence, RBFAOO examines B's descendants with no backtracks to A. Assume that D is chosen to examine. RBFAOO sets $th(D) = th(B) - q(E) = 2$ to indicate that C becomes best if $q(D) > th(D)$ holds, which is equivalent to $q(B) = q(D) + q(E) > th(D) + q(E) = th(B)$. Next, RBFAOO expands D and updates $q(D) = w(D,F) + h(F) = 4$ (Figure 1(c)). Because $q(D) > th(D)$, the subtree rooted at D contains no best leaf in terms of AO*'s strategy. RBFAOO backtracks to A by updating $q(B) = q(D) + q(E) = 6$ and examines C (Figure 1(d)) with $th(C) = w(A,B) + q(B) - w(A,C) = 6$ to be able to select B when B becomes best.

RBFAOO revisits n to update $q(n)$. Caching $q(n)$ significantly reduces the overhead of reexamining n's subtree. Contexts of nodes in [3] are also used to merge different nodes with the same outcome into one, thus enabling to avoid duplicate search effort.

Algorithm Description Figure 2 shows the pseudo-code of RBFAOO. Let ϵ be a small number and assume $\infty - \epsilon < \infty$. By introducing ϵ, RBFAOO can return the solution cost of ∞ if the root node has no solution. In practice, a finite real number is used to represent ∞. Let δ be an empirically tuned parameter that determines the amount of overestimation. $HasNoChildren$ checks whether a node has no children (ie, terminal leaf or dead-end) or not. $Evaluate$ evaluates a terminal leaf/dead-end n and returns a pair of the cost (ie, 0 or ∞) and a Boolean flag indicating whether n is solved or a dead-end. $UnsolvedChild$ returns an unproven child. $SaveInCache$ saves in the cache table a q-value and a flag indicating whether a node is solved optimally or not. $RetrieveFromCache$ retrieves them from the cache table. $Context$ (see [3]) calculates the context of a node.

When RBFAOO starts solving a problem, the threshold of the root node is set to $\infty - \epsilon$. If RBFAOO exceeds this threshold, the problem is proven to have no solution. Otherwise, RBFAOO returns the optimal solution cost to the problem.

The $RBFS$ function traverses the subtree in a depth-first manner. At each node, RBFS calculates either an optimal solution cost or a lowerbound by using $BestChild$ or Sum and checks if the termination condition is satisfied. If the solution optimality is guaranteed at node n, $n.solved$ is set to **true**.

At an OR node, RBFS may find a suboptimal solution. In this case, $n.solved$ is still set to **false** and RBFS continues examining other children until it finds an optimal solution at n. Because the solution cost found so far (ub in Figure 2) is an upperbound of the optimal one, RBFS uses ub to prune branches that never lead to optimal solutions.

When RBFS selects c_{best}, it examines c_{best} with a new threshold. At OR nodes, $c_{best}.th$ is set to subtracting the weight between n and c_{best} from the minimum of: (1) The current threshold for n. (2) The second smallest lowerbound q_2 to solve n's child with considering the weight from n to that child among a list of such lowerbounds of n's

```
// Set up for the root node
double RBFAOO(node root) {
  root.th = ∞ − ϵ;
  q = RBFS(root);
  return q;
}
// Depth-first search with a threshold
double RBFS(node n) {
  // Terminal leaf/dead-end check
  if (HasNoChildren(n)) {
    // Calculate the probability
    // for a terminal leaf or dead-end
    (q, s) = Evaluate(n);
    // Store search results
    SaveInCache(Context(n),q,s);
    return q;
  }
  GenerateChildren(n);
  // Continue search until satisfying
  // the termination condition
  if (n is an OR node)
    loop {
      (c_best, q, q_2, ub) = BestChild(n);
      if (n.th < q || n.solved = true)
        break;
      // Update the threshold
      c_best.th = min(n.th,
                      q_2 + δ,
                      ub) − w(n, c_best);
      RBFS(c_best);
    }
  else
    loop { // AND node
      q = Sum(n);
      if (n.th < q || n.solved = true)
        break;
      (c_best, q_{c_best}) = UnsolvedChild(n);
      // Update the threshold
      c_best.th = n.th − (q − q_{c_best});
      RBFS(c_best);
    }
  // Store search results
  SaveInCache(Context(n),q,n.solved);
  return q;
}
```

```
// Select the best child
double BestChild(node n) {
  q = q_2 = ub = ∞;
  n.solved=false;
  foreach (n's child c_i) {
    ct = Context(c_i);
    if (ct is in the cache table)
      (q_{c_i}, s) = RetrieveFromCache(ct);
    else {
      q_{c_i} = h(c_i);
      s=false;
    }
    q_{c_i} = w(n, c_i) + q_{c_i};
    if (s=true) // c_i is solved
      ub = min(ub, q_{c_i});
    if (q_{c_i} < q ||
        (q_{c_i} = q && n.solved=false)) {
      q_2 = q;
      n.solved = s;
      q = q_{c_i};
      c_best = c_i;
    } else if (q_{c_i} < q_2)
      q_2 = q_{c_i};
  }
  return (c_best, q, q_2, ub);
}
// Calculate the total value
double Sum(node n) {
  q = 0;
  n.solved = true;
  foreach (n's child c_i) {
    ct = Context(c_i);
    if (ct is in the cache table)
      (q_{c_i}, s) = RetrieveFromCache(ct);
    else {
      q_{c_i} = h(c_i);
      s = false;
    }
    q = q + q_{c_i};
    n.solved = n.solved ∧ s;
  }
  return q;
}
```

Fig. 2. Pseudo-code of RBFAOO

children. This indicates when the current second best child becomes best. Additionally, a parameter δ that allows for returning a suboptimal solution cost is added to q_2 to avoid an excessive number of backtracks to n. (3) The upperbound of the optimal solution at n. At AND nodes, $c_{best}.th$ is set to the sum of c_{best}'s q-value and the gap between $n.th$ and the total q-value of n's children. If $q(c_{best}) > c_{best}.th$, $q(n) > n.th$ also holds.

Let N be the number of nodes in the search space. If the search space fits into memory, AO* expands $O(N)$ nodes in the worst case. In contrast, due to node re-expansions, RBFAOO's worst-case scenario is $O(N^2)$. However, in practice, by introducing δ, RBFAOO avoids such a high node re-expansion overhead.

The following theorem guarantees correctness of RBFAOO:

Theorem 1 (correctness). *Given a belief network $\mathcal{B}$ and an admissible heuristic function h, if RBFAOO finds a solution $\bar{x}$ to the MPE task over $\mathcal{B}$, then $\bar{x}$ is optimal.*

Proof. We assume that RBFAOO tries to solve a minimization task. Let $v(n)$ be the optimal solution cost for node n. We first prove that for any value q for node n in the cache table, $q \leq v(n)$ holds. Since different nodes with the same context are proven to be equivalent in directed acyclic graphs [3], we denote n as Context(n) when a search result at node n is saved in the cache table. Additionally, we assume $h(n) = h(n')$ if Context$(n) = Context(n')$.

Let $Cache_t$ be the state of the cache table immediately after the t-th save is performed in the cache table. Let $Q_t(n)$ be the value saved in $Cache_t$ for n if that value exists in $Cache_t$ or $h(n)$ if n is not preserved in $Cache_t$. By induction on t, we prove that all the entries in the cache table contain values that do not overestimate optimal ones.

1. *Because no result is stored in $Cache_0$, the above property holds for $t = 0$.*
2. *Assume that the above property holds for $t = k$. Then, $Q_{k+1}(n)$, which is saved in $Cache_{k+1}$, is calculated as:*
   - *If n is a terminal leaf, Evaluate(n) in the pseudo code always returns $v(n)$. $Q_{k+1}(n) = v(n)$ therefore holds.*
   - *If n is an internal OR node, $Q_{k+1}(n) = w(n, c_{best}) + Q_k(c_{best}) = \min_i(w(n, c_i) + Q_k(c_i))$ holds where c_i is n's child. Additionally, because $Q_k(c_i) \leq v(c_i)$, $Q_{k+1}(n) \leq \min_i(w(n, c_i) + v(c_i)) = v(n)$ holds.*
   - *If n is an internal AND node, $Q_{k+1}(n) = \sum_i Q_k(c_i)$ where c_i is n's child. Because $Q_k(c_i) \leq v(c_i)$, $Q_{k+1}(n) \leq \sum_i v(c_i) = v(n)$ holds.*

   Hence, $Q_t(n) \leq v(n)$ holds in case of $t = k + 1$.

Let $Q(root)$ be a value that is about to be saved in the cache table with satisfying the termination condition of $root.solved = true$. $Q(root) \leq v(root)$ holds from the above. Additionally, because RBFAOO has traced a solution tree with the cost of $Q(root)$, $v(root) \leq Q(root)$ holds. Therefore, $Q(root) = v(root)$ holds.

4 Preliminary Experiments

In this section, we evaluate empirically RBFAOO best-first AND/OR search scheme on benchmark problems derived from genetic linkage analysis[1]. The algorithms were implemented in C++ (64-bit) and the experiments were run on a 2.6GHz quad-core processor with 80GB of RAM. We consider the following solving alternatives:

- **AOBB**(i) - the depth-first AND/OR Branch-and-Bound algorithm with full caching and pre-compiled mini-bucket heuristics [3], where i is the mini-bucket i-bound.[2]
- **AOBF**(i) - the best-first AND/OR search algorithm exploring the context-minimal AND/OR search graph (full caching) and using mini-bucket heuristics [3].[3]

[1] Available at *http://graphmod.ics.uci.edu*

[2] Implementation available at *http://github.com/lotten/daoopt*

[3] Implementation available at *http://graphmod.ics.uci.edu*

– **RBFAOO**(i) - the recursive best-first AND/OR search algorithm from Section 3 also guided by the mini-bucket heuristics. The parameter δ used to control the overestimation was set to 1, which was determined by performing several preliminary experiments. RBFAOO(i) was implemented on top of the AOBF(i) implementation.
– **BTD** - the depth-first Branch-and-Bound exploring a tree decomposition of the primal graph and enforcing soft local consistency, called EDAC (existential directional arc-consistency), to generate a heuristic function [17]. We used the `toulbar2`[4] implementation with default parameters.

The algorithms using pre-compiled mini-bucket heuristics, AOBB(i), AOBF(i) and RBFAOO(i), respectively, were restricted to a static variable ordering obtained as a depth-first traversal of the guiding pseudo tree which was computed using a min-fill heuristic iteratively and stochastically [18]. Algorithms AOBB(i) and AOBF(i) order the subproblems rooted at node n in the search space in increasing order of their corresponding heuristic value. On the other hand, algorithm RBFAOO(i) orders the OR children of an AND node in increasing order of their disproof numbers while the AND children of an OR node are ordered in increasing order of their q-values.

To maintain a relatively fair comparison, BTD constructed its guiding tree decomposition along the same variable ordering used by the other AND/OR search algorithms. However, BTD orders the variables within a cluster dynamically using a last-conflict backjumping based heuristic [19]. The algorithm breaks ties lexicographically.

Table 1 displays the results obtained for experiments with 10 pedigree networks. We report the CPU time in seconds and the number of nodes expanded for solving the problems. We also specify the problems parameters such as the number of variables (n), maximum domain size (k), the depth of the pseudo tree (h) and the induced width of the graph (w^*). The best time for each i-bound is shown in bold and underlined, while the overall best performance points are boxed. The columns are indexed by the mini-bucket i-bound, $i \in \{2, 4, 10, 16, 18\}$. All algorithms were allotted a 1 hour time limit. Algorithm AOBF(i) was allowed a maximum of 80GB of RAM and its internal cache table allocated memory dynamically, when needed. In contrast, algorithm RBFAOO(i) used a cache table with 134,217,728 entries which pre-allocated 10 to 20GB of RAM.

When looking at the algorithms using mini-bucket heuristics, we see that RBFAOO(i) outperforms dramatically its competitors, especially for relatively weak heuristics (which are typically obtained for relatively small i-bounds) and on the most difficult problem instances. For example, RBFAOO(2) using the weakest heuristic ($i = 2$) was able to solve `pedigree50` in about 15 minutes and expanding over 118 million nodes, while AOBB(2) and AOBF(2) exceeded the time and memory limits, respectively. Moreover, RBFAOO(16) was the only solver able to close `largeFam2-10-52`, a previously unsolved instance, after about 36 hours and 12.8 billion node expansions, respectively. The time limit for the latter instance was set to 100 hours.

In terms of the size of the search spaces explored, we see that AOBF(i) typically expands the smallest number of nodes, as expected. RBFAOO(i) expands more nodes than AOBF(i), due to re-expansions, but in many cases it expands significantly fewer

[4] Available at: *https://mulcyber.toulouse.inra.fr/projects/toulbar2/*

instance	algorithm	i = 2		i = 4		i = 10		i = 16		i = 18		BTD	
(n, k, w^*, h)		time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes
pedigree7	AOBB												
	AOBF									1311	30545960		
	RBFAOO							**818**	144733023	**295**	47907073		
pedigree9	AOBB							2302	394887679	665	108980968		
	AOBF					1846	30506650	263	7682927	104	3187533		
	RBFAOO			**3216**	547471865	**522**	97410715	**60**	10634230	**39**	4369301		
pedigree13	AOBB												
	AOBF												
	RBFAOO							**2629**	364037130	**1490**	207723571		
pedigree19	AOBB												
	AOBF									2400	17666687		
	RBFAOO					**1753**	319268527	**378**	69780223	**643**	18761971		
pedigree20	AOBB			2839	366982095	9	1254460	6	359262	29	12266		
	AOBF			905	25302204	21	823742	9	249215	29	11160	955	15658549
	RBFAOO	**558**	106610582	**170**	34812813	**4**	965520	**5**	298133	**28**	12297		
pedigree23	AOBB	1828	194090018	71	7897530	11	1343660	**3**	8698	**5**	1891		
	AOBF	289	4204220	43	1384921	10	292587	**3**	8284	**5**	1951	**0**	4880
	RBFAOO	**34**	6745554	**7**	1766837	**1**	382167	**3**	8210	**5**	1868		
pedigree30	AOBB					244	34182326	**3**	107437	**7**	10374		
	AOBF			726	7518886	45	1717523	**3**	30794	8	6682	896	12453725
	RBFAOO	**424**	88306931	**76**	18897478	**14**	3840692	**3**	60479	**7**	4548		
pedigree31	AOBB												
	AOBF									2758	56131507		
	RBFAOO									**949**	130673223		
pedigree41	AOBB												
	AOBF												
	RBFAOO							**1517**	210630024	**2017**	277459272		
pedigree50	AOBB					32	203925						
	AOBF					34	148878						
	RBFAOO	**931**	118042401	**198**	27289416	**30**	135836						
largeFam3-10-52	AOBB												
	AOBF												
	RBFAOO							**129633**	12826083707				

Table 1. Results for pedigree networks. CPU time (in seconds) and number of nodes expanded. Time limit 1 hour. RBFAOO(i) ran with a 10-20GB cache table (134,217,728 table entries).

nodes than AOBB(i) which translates into important time savings. For example, on `pedigree23`, RBFAOO(2) is almost two orders of magnitude faster than AOBB(2), while expanding almost two orders of magnitude fewer nodes. We also notice that RBFAOO(i) and AOBB(i) have a relatively small overhead per node expansion. In contrast, the computational overhead of AOBF(i) is much larger. It is caused primarily by maintaining an extremely large search space in memory and, secondly, because the node values are typically updated all the way up to the root. Finally, BTD is competitive only on one instance (`pedigree23`), because the EDAC based heuristics are weaker than the mini-bucket ones in this case, which causes the algorithm to explore a much larger search space. In summary, we noticed that RBFAOO(i) was superior to its competitors especially for relatively inaccurate heuristics (smaller i-bounds) and on the hardest problem instances. This is important because it is likely that for these types of problems it may only be possible to compute rather weak heuristics given limited resources.

5 Conclusion

The paper rests on two important contributions. First, we introduce a limited memory best-first AND/OR search algorithm that traverses an AND/OR search graph for solving optimization tasks defined over graphical models. It belongs to the RBFS family of algorithms, thus using a threshold controlling mechanism to guide the search, and employs in addition a flexible caching scheme to reuse partial search results as well as an overestimation mechanism to further reduce the node re-expansion rate. Second,

our preliminary empirical evaluation on a set of difficult pedigree networks showed that RBFAOO is often superior to the current state-of-the-art solvers.

Our approach leaves room for further improvements. RBFAOO can be extended to use dynamic variable heuristics as well as soft arc-consistency based heuristics. Since many interesting real-world problems are still too hard to solve exactly, we also plan to convert the algorithm into an anytime best-first search scheme.

References

1. R. Dechter. *Constraint Processing*. Morgan Kaufmann Publishers, 2003.
2. R. Marinescu and R. Dechter. AND/OR branch-and-bound search for combinatorial optimization in graphical models. *Artificial Intelligence*, 173(16-17):1457–1491, 2009.
3. R. Marinescu and R. Dechter. Memory intensive AND/OR search for combinatorial optimization in graphical models. *Artificial Intelligence*, 173(16-17):1492–1524, 2009.
4. R. Dechter and J. Pearl. Generalized best-first search strategies and the optimality of A*. *In Journal of ACM*, 32(3):505–536, 1985.
5. R. Korf. Linear-space best-first search. *Artificial Intelligence*, 62(1):41–78, 1993.
6. L. V. Allis, M. van der Meulen, and H. J. van den Herik. Proof-number search. *Artificial Intelligence*, 66(1):91–124, 1994.
7. A. Nagai. *Df-pn Algorithm for Searching AND/OR Trees and Its Applications*. PhD thesis, The University of Tokyo, 2002.
8. A. Kishimoto, M. Winands, M. Müller, and J.-T. Saito. Game-tree search using proof numbers: The first twenty years. *ICGA Journal, Vol. 35, No. 3*, 35(3):131–156, 2012.
9. R. Dechter and R. Mateescu. AND/OR search spaces for graphical models. *Artificial Intelligence*, 171(2-3):73–106, 2007.
10. E. C. Freuder and M. J. Quinn. Taking advantage of stable sets of variables in constraint satisfaction problems. In *IJCAI*, pages 1076–1078, 1985.
11. K. Kask and R. Dechter. A general scheme for automatic generation of search heuristics from specification dependencies. *Artificial Intelligence*, 129(1-2):91–131, 2001.
12. Jurg Ott. *Analysis of Human Genetic Linkage*. The Johns Hopkins University Press, 1999.
13. M. Fishelson and D. Geiger. Exact genetic linkage computations for general pedigrees. *Bioinformatics*, 18(1):189–198, 2002.
14. M. Fishelson, N. Dovgolevsky, and D. Geiger. Maximum likelihood haplotyping for general pedigrees. *Human Heredity*, 2005.
15. N. J. Nilsson. *Principles of Artificial Intelligence*. Tioga Publishing Co, Palo Alto, CA, 1980.
16. A. Nagai. A new depth-first search algorithm for AND/OR trees. Master's thesis, Department of Information Science, The University of Tokyo, 1999.
17. S. de Givry, T. Schiex, and G. Verfaillie. Exploiting tree decomposition and soft local consistency in weighted csp. In *AAAI*, pages 22–27, 2006.
18. K. Kask, A. Gefland, and R. Dechter. Pushing the power of stochastic greedy ordering schemes for inference in graphical models. In *AAAI*, pages 54–60, 2011.
19. A. Favier, S. de Givry, A. Legarra, and T. Schiex. Pairwise decomposition for combinatorial optimization in graphical models. In *IJCAI*, pages 2126–2132, 2011.

A Constraint Solving Approach to Tropical Equilibration and Model Reduction

Sylvain Soliman[1], François Fages[1], and Ovidiu Radulescu[2]

[1] EPI Contraintes, Inria Paris-Rocquencourt, France
[2] University of Montpellier, France

Abstract. Model reduction is a central topic in systems biology and dynamical systems theory, for reducing the complexity of detailed models, finding important parameters, and developing multi-scale models for instance. While perturbation theory is a standard mathematical tool to analyze the different time scales of a dynamical system, and decompose the system accordingly, tropical methods provide a simple algebraic framework to perform these analyses systematically in polynomial systems. The crux of these tropicalization methods is in the computation of tropical equilibrations. In this paper we show that constraint-based methods, using reified constraints for expressing the equilibration conditions, make it possible to numerically solve non-linear tropical equilibration problems, out of reach of standard computation methods. We illustrate this approach first with the reduction of simple biochemical mechanisms such as the Michaelis-Menten and Goldbeter-Koshland models, and second, with performance figures obtained on a large scale on the model repository `biomodels.net`.

1 Preliminaries on Model Reduction by Tropicalization

We consider networks of biochemical reactions with mass action kinetic laws. Each reaction is defined as

$$\sum_i \boldsymbol{\alpha}_{ji} A_i \to \sum_k \beta_{jk} A_k.$$

The stoichiometric vectors $\alpha_j \in \mathbb{N}^n$, $\boldsymbol{\beta}_j \in \mathbb{N}^n$ have coordinates $\boldsymbol{\alpha}_{ji}$ and β_{jk} and define which species are consumed and produced by the reaction j and in which quantities.

The mass action law means that reaction rates are monomial functions of the species concentrations x_i and reads

$$R_j(\boldsymbol{x}) = k_j \boldsymbol{x}^{\boldsymbol{\alpha}_j}. \tag{1}$$

where $k_j > 0$ are kinetic constants, $\boldsymbol{\alpha}_j = (\alpha_1^j, \ldots, \alpha_n^j)$ are multi-indices and $\boldsymbol{x}^{\boldsymbol{\alpha}_j} = x_1^{\alpha_1^j} \ldots x_n^{\alpha_n^j}$.

The network dynamics is described by the following differential equations

$$\frac{dx_i}{dt} = \sum_j k_j(\beta_{ji} - \alpha_{ji})\boldsymbol{x}^{\boldsymbol{\alpha}_j}. \tag{2}$$

In what follows, the kinetic parameters do not have to be known precisely, they are given by their orders of magnitude. A convenient way to represent orders is by considering that

$$k_j = \bar{k}_j \epsilon^{\gamma_j}, \tag{3}$$

where ϵ is a positive parameter much smaller than 1, γ_j is an integer, and $\bar{k}_j$ has order unity. An approximate integer order can be obtained from any real positive parameter by

$$\gamma_j = \text{round}(\log(k_j)/\log(\epsilon)), \tag{4}$$

where round stands for the closest integer. For instance, if $\epsilon = 1/10$, γ_j will represent the logarithmic value of the parameter rounded to the nearest decade. Notice that in this representation, small quantities have large orders. Furthermore, the smaller ϵ, the better the separation between quantities of different orders, indeed $\lim_{\epsilon\to 0} \frac{k_i}{k_j} = \infty$ if $\gamma_i < \gamma_j$. We are also interested in the orders of the species concentrations, therefore we introduce a vector of orders $\boldsymbol{a} = (a_1, \ldots, a_n)$, such that $\boldsymbol{x} = \bar{\boldsymbol{x}}\epsilon^{\boldsymbol{a}}$. Orders $\boldsymbol{a}$ are unknown and have to be calculated. To this aim, the network dynamics can be described by a rescaled system of ordinary differential equations

$$\frac{d\bar{x}_i}{dt} = (\sum_j \epsilon^{\mu_j} k_j(\beta_{ji} - \alpha_{ji})\bar{\boldsymbol{x}}^{\boldsymbol{\alpha}_j})\epsilon^{-a_i}, \tag{5}$$

where

$$\mu_j = \gamma_j + < \boldsymbol{a}, \boldsymbol{\alpha}_j >, \tag{6}$$

and $<,>$ stands for the vector dot product. The r.h.s. of each equation in (5) is a sum of monomials in the concentrations, with positive and negative signs given by the stoichiometries $\beta_{ji} - \alpha_{ji}$. Generically, these monomials have different orders (given by μ_j) and there is one monomial that dominates the others. In this case, the corresponding variable will change rapidly in the direction imposed by this dominating monomial. However, on sub-manifolds of the phase space, at least two monomials, one positive and one negative may have the same order. This situation was called tropical equilibration in [6]. Tropical equilibration is different from equilibrium or steady state in many ways. Firstly, steady state means equilibration of all species, whereas tropical equilibration may concern only one or a few rapid species. Secondly, steady state means that forces are rigorously compensated on all variables that are at rest, whereas tropical equilibration means that only the dominant forces are compensated and variables may change slowly under the influence of uncompensated, weak forces. Compensation of dominant forces constrains the dynamics of the system to a low dimensional manifold named invariant manifold [7, 5]. As discussed in [6], tropical equilibrations encompass the notions of quasi-steady state and quasi-equilibrium from

singular perturbation theory of biochemical networks, but are more general. Let us provide a formal definition of tropical equilibration (see [6] for more details).

Definition 1. *Two reactions j, j' are tropically equilibrated on the species i iff:*
i) $\mu_j = \mu_{j'}$,
ii) $(\beta_{ji} - \alpha_{ji})(\beta_{j'i} - \alpha_{j'i}) < 0$ *(meaning that the effects of the reactions j and j' on the species i are opposite),*
iii) $\mu_k \geq \mu_j$ *for any reaction $k \neq j, j'$, such that* $\beta_{ki} \neq \alpha_{ki}$.

According to (6) and Definition 1, the equilibrations correspond to vectors $\boldsymbol{a} \in R^n$ where the minimum in the definition of the piecewise-affine function $f_i(\boldsymbol{a}) = \min_j(\gamma_j + < \boldsymbol{a}, \boldsymbol{\alpha}_j >)$ is attained at least twice. Tropical equilibrations are used to calculate the unknown orders $\boldsymbol{a}$. The solutions have a geometrical interpretation. Let us consider the equality $\mu_j = \mu_{j'}$. This represents the equation of a $n-1$ dimensional hyperplane of $\mathbb{R}^n$, orthogonal to the vector $\boldsymbol{\alpha}_j - \boldsymbol{\alpha}_{j'}$:

$$\gamma_j + < \boldsymbol{a}, \boldsymbol{\alpha}_j > = \gamma_{j'} + < \boldsymbol{a}, \boldsymbol{\alpha}_{j'} > \tag{7}$$

For each species i, we consider the set of reactions $\mathcal{R}_i$ that act on this species, namely the reaction k is in $\mathcal{R}_i$ iff $(\boldsymbol{\beta}_k - \boldsymbol{\alpha}_k)_i \neq 0$. The finite set $\mathcal{R}_i$ can be characterized by the corresponding set of stoichiometric vectors $\boldsymbol{\alpha}_k$. The set of points of $\mathbb{R}^n$ where at least two reactions equilibrate on the species i corresponds to the places where the function f_i is not locally affine (the minimum in the definition of f_i is attained at least twice). For simplicity, we shall call this locus tropical manifold [6, 9].

A simple example of biochemical network is the Michaelis-Menten mechanism of an enzymatic reaction. This network consists of two reactions:

$$S + E \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} ES \overset{k_2}{\to} P + E,$$

where S, E, ES, P represent the substrate, the enzyme, the enzyme-substrate complex and the product, respectively.

The system of polynomial differential equations reads:

$$\begin{aligned} x_1' &= -k_1 x_1 x_3 + k_{-1} x_2, \\ x_2' &= k_1 x_1 x_3 - (k_{-1} + k_2) x_2, \\ x_3' &= -k_1 x_1 x_3 + (k_{-1} + k_2) x_2, \\ x_4' &= k_2 x_2. \end{aligned} \tag{8}$$

where $x_1 = [S]$, $x_2 = [SE]$, $x_3 = [E]$, $x_4 = [P]$.

There are two conservation laws: $x_2 + x_3 = e_0$ and $x_1 + x_2 + x_4 = s_0$ The rescaled variables are $x_i = \bar{x}_i \epsilon^{a_i}$, $1 \leq i \leq 4$, $k_1 = \bar{k}_1 \epsilon^{\gamma_1}$, $k_{-1} = \bar{k}_{-1} \epsilon^{\gamma_{-1}}$, $e_0 = \bar{e}_0 \epsilon^{\gamma_e}$, $s_0 = \bar{s}_0 \epsilon^{\gamma_s}$. Let us notice that the last equation can never be equilibrated because it contains only one monomial. The tropical equilibration equations for

the remaining variables read:

$$\begin{aligned}
\gamma_1 + a_1 + a_3 &= \gamma_{-1} + a_2, \\
\gamma_1 + a_1 + a_3 &= \min(\gamma_{-1}, \gamma_2) + a_2, \\
\gamma_1 + a_1 + a_3 &= \gamma_2 + a_2, \\
\min(a_2, a_3) &= \gamma_e, \\
\min(a_1, a_2, a_4) &= \gamma_s.
\end{aligned} \tag{9}$$

The set of integer orders endowed with the minimum and sum operations is a semiring, called min-plus algebra [2] where the minimum is noted $\oplus$ and the sum $\otimes$. Our tropical equilibration problem is solving a set of polynomial equations in this semi-ring.

Let us emphasize an important difference between the calculation of tropical equilibrations and calculation of exact equilibria of systems of polynomial differential equations. If there are exact conservation laws, the set of exact equilibrium equations are linearly dependent, therefore one can eliminate some of them from the system. Because elements in a min-plus semiring do not generally have additive inverses, elimination is not automatically possible in systems of tropical equations. In this case, one should keep all the tropical equilibrium equations for all the variables and add to them the tropical conservation relations.

2 Example of Golbeter-Koshland Switch

A slightly more complicated network is the Goldbeter-Koshland mechanism. This consists of two coupled Michaelis-Menten equations. The mechanism is important because it plays the role of a switch, allowing the propagation of information in signal transduction networks. The detailed mechanism is represented by four mass action reactions

$$S + E_a \underset{k^a_{-1}}{\overset{k^a_1}{\rightleftharpoons}} E_a S \overset{k^a_2}{\to} S^* + E_a, S^* + E_b \underset{k^b_{-1}}{\overset{k^b_1}{\rightleftharpoons}} E_b S^* \overset{k^b_2}{\to} S + E_b.$$

where S and S^* are, for instance, the un-phosphorylated and phosphorylated forms of a substrate, E_a, E_b, are kinase and phosphatase enzymes, respectively.

This mechanism leads to the following system of differential equations:

$$\begin{aligned}
x_1' &= k^a_2 x_5 - k^a_1 x_1 x_3, \\
x_2' &= k^b_2 x_6 - k^b_1 x_2 x_4, \\
x_3' &= k^a_{-1} x_5 + k^b_2 x_6 - k^a_1 x_1 x_3, \\
x_4' &= k^a_2 x_5 + k^b_{-1} x_6 - k^b_1 x_2 x_4, \\
x_5' &= k^a_1 x_1 x_3 - (k^a_{-1} + k^a_2) x_5, \\
x_6' &= k^b_1 x_2 x_4 - (k^b_{-1} + k^b_2) x_6.
\end{aligned} \tag{10}$$

where $x_1 = [E_a]$, $x_2 = [E_b]$, $x_3 = [S]$, $x_4 = [S^*]$, $x_5 = [E_a S]$, $x_6 = [E_b S^*]$.

This system has three conservation laws:

$$\begin{aligned} x_1 + x_5 &= E_0^a, \\ x_2 + x_6 &= E_0^b, \\ x_3 + x_4 + x_5 + x_6 &= S_0. \end{aligned} \tag{11}$$

Equilibrating each equation of (10) and taking into account (11) leads to the following tropical equations:

$$\begin{aligned} \gamma_2^a \otimes a_5 &= \gamma_1^a \otimes a_1 \otimes a_3, \\ \gamma_2^b \otimes a_6 &= \gamma_1^b \otimes a_2 \otimes a_4, \\ (\gamma_{-1}^a \otimes a_5) \oplus (\gamma_2^b \otimes a_6) &= \gamma_1^a \otimes a_1 \otimes a_3, \\ (\gamma_2^a \otimes a_5) \oplus (\gamma_{-1}^b \otimes a_6) &= \gamma_1^b \otimes a_2 \otimes a_4, \\ \gamma_1^a \otimes a_1 \otimes a_3 &= (\gamma_{-1}^a \oplus \gamma_2^a) \otimes a_5, \\ \gamma_1^b \otimes a_2 \otimes a_4 &= (\gamma_{-1}^b \oplus \gamma_2^b) \otimes a_6, \\ a_1 \oplus a_5 &= \gamma_e^a, \\ a_2 \oplus a_6 &= \gamma_e^b, \\ a_3 \oplus a_4 \oplus a_5 \oplus a_6 &= \gamma_s. \end{aligned} \tag{12}$$

The corresponding CSP, described in the next section, is solved instantly and gives the unique solution: $a_1 = 5, a_2 = 4, a_3 = 3, a_4 = 4, a_5 = 7$ for parameter values consistent with the literature: $k_1^* = 1000, k_2^* = 150, k_{-1}^* = 150$.

3 Tropical Equilibration as a Constraint Satisfaction Problem

Given a biochemical reaction system with its Mass-Action kinetics, and a small ϵ, the problem of tropical equilibration is to look for a rescaling of the variables such that the dominating positive and negative term in each ODE *equilibrate* as per Definition 1, i.e., are of the same degree in ϵ.

Note that there are supplementary constraints related to this rescaling when some conservation laws exist for the original system. Finding these conservation laws is another CSP which can be solved efficiently with constraint methods [8]. Here we will assume that the conservation laws are given in input. In our prototype implementation, both the computation of conservation laws and the following equilibration are performed for a given system.

For each original equation dx_i/dt, $1 \leq i \leq n$ is introduced a variable $a_i \in \mathbb{Z}$ that is used to rescale the system by posing $x_i = \epsilon^{a_i} \bar{x_i}$. These are the variables of our CSP. Note that they require a solver handling $\mathbb{Z}$ like for instance SWI-Prolog [11, 10] with the `clpfd` library by Markus Triska.

The constraints are of two kinds. For each differential equation that should be equilibrated is a list of positive monomials M_i^+, and a list of negative monomials M_i^-. The degrees in ϵ of all these monomials are integer linear expressions in the a_i. Now, to obtain an equilibration one should enforce for each i

that the minimum degree in M_i^+ is equal to the minimum degree in M_i^-. This will ensure that we find two monomials (i of Definition 1) of opposite sign (ii) and of minimal degree (iii). This corresponds to the first six tropical equations of (12). We will see how they can be implemented with reified constraints, but for now, let us assume a constraint `min(L, M)` that enforces that the FD variable `M` is the minimum value of a list `L` of linear expressions over FD variables. We have in our CSP, for each $1 \leq i \leq n$, `min(PositiveMonomialDegrees, M)` and `min(NegativeMonomialDegrees, M)`.

The second kind of constraint comes from conservation laws. Each conservation law is an equality between a linear combination of the x_i and a constant c_i. By rescaling, we obtain a sum of rescaled monomials equal to $\epsilon^{\log(c_i)/\log(\epsilon)}\bar{c_i}$. We want this equality to hold when ϵ goes to zero, which implies that the minimal degree in ϵ in the left hand side is equal to (the round of) the degree of the right hand side. Since once again the degrees on the left are linear combinations of our variables a_i, this is again a constraint of the form: `min(ConservationLawDegrees, K)` where `K` is equal to $\text{round}(\log(c_i)/\log(\epsilon))$. This corresponds to the last three tropical equations of (12).

Furthermore, if the system under study is not at steady state, the minimum degree should not be reached only once, which would lead to a constant value for the corresponding variable when ϵ goes to zero, but at least twice. This is the case for the example treated in [5]. The constraint we need is therefore slightly more general than `min/2`: we need the constraint `min(L, M, N)` which is true if `M` is smaller than each element of `L` and equal to `N` elements of that list. Note that using CLP notation, we have:

```
min(M, L) :-  C#>=1, min(M, L, C).
```

In order to enforce that the minimum is reached at least a required number of times, one obvious solution is to try all pairs of positive and negative monomials and count the successful pairs [7]. However, this is not necessary, the `min(L, M, N)` constraint directly expresses the cardinality constraint on the minimums. and can be implemented using *reified constraints* to propagate information between `L`, `M` and `N` in all directions, without enumeration. Using SWI-Prolog notations, the implementation of `min/3` by reified constraints is as follows:

```
min([], _, 0).
min([H | T], M, C) :- M#=<H, B #<==> M#=H, C#=B+CC,
                      min(T, M, CC).
```

This concise and portable implementation will probably improve when the `minimum` and `min_n` global constraints are available (see [1] for a reference). However it already proves very efficient as demonstrated in the next section.

4 Computation Results on Biomodels.net

To benchmark our approach, we applied it systematically to all the dynamical models of the BioModels[1] repository [4] of biological systems, with ϵ set arbitrarily to 0.1. We used the latest release (*r24* from 2012-12-12) which includes 436 curated models.

Among them, only 55 models have non-trivial purely polynomial kinetics (ignoring *events* if any). Our computational results on those are summarized in the following table, where the first column indicates whether a complete equilibration was found, and the times are in seconds.

Found	# models	Variables (avg/min/max)	Time (avg/min/max)
yes	23	17.348/3/ 86	0.486/0.004/2.803
no	32	17.812/1/194	0.099/0.000/1.934

We managed to avoid timeouts by using an iterative domain expansion: the problem is first tried with a domain of $[-2, 2]$, i.e., equilibrations are searched by rescaling in the $10^{-2}, 10^{2}$ interval. If that fails, the domain is doubled and the problem tried again (until a limit of $10^{-128}, 10^{128}$). This strategy coupled with a domain bisection enumeration (`bisect` option in SWI-Prolog) allowed us to gain two orders of magnitude on the biggest models.

Only one of the models (number 002) used values far from 0 in the equilibration (up to ϵ^{40}) and has no complete equilibration if the domain is restricted to $[-32, 32]$. This is because all kinetics are scaled by the volume of the compartment, which in that case was 10^{-16}, translating the search accordingly. We thus do not believe that enlarging the domains even more would lead to more equilibrations. Nevertheless, choosing a smaller ϵ might increase the number of equilibrations.

18 of the 23 models for which there is a complete equilibration are actually underconstrained and appear to have an infinity of such solutions (typically linear relations between variables). For the 5 remaining ones, we computed all complete equilibrations:

Model	# equilibrations	Total time (s)
BIOMD0000000002	36	109
BIOMD0000000122	45	291
BIOMD0000000156	7	0.008
BIOMD0000000229	7	0.7
BIOMD0000000413	29	3.3

5 Discussion

One of the limits of this approach, is that it is not well suited to equilibration problems with an infinite number of solutions. For those, symbolic solutions depending on free parameters are necessary, as done in [6].

[1] `http://biomodels.net`

It is also possible to reduce a system using its conservation laws, and to apply tropical equilibration directly on the reduced system. However, the resulting equilibrations might be slightly different, apparently due to the possible loss of positivity of certain variables. We want to investigate this question further.

In many cases, it makes sense biologically to only look for partial equilibrations. Strategies to decide when such decision has to be made remain unclear. Nevertheless the framework of partial constraint satisfaction and more specifically Max-CSP [3] would allow us to easily handle the maximization of the number of equilibrated variables.

In this paper we discussed only the calculation of the tropical equilibrations and of the unknown orders of the variables. Once the orders of the variables are known, the rapid variables can be identified and the system reduced to a simpler one. The details of the reduction procedure, involving pruning of dominated terms and pooling of fast variables into fast cycles will be presented elsewhere. A simple reduction procedure, involving only pruning is described by Theorem 3.6 of [6].

Acknowledgements. This work has been supported by the French ANR BioTempo, CNRS Peps ModRedBio, and OSEO Biointelligence projects.

References

1. N. Beldiceanu, M. Carlsson, S. Demassey, and T. Petit. Global constraints catalog. Technical Report T2005-6, Swedish Institute of Computer Science, 2005.
2. G. Cohen, S. Gaubert, and J.P. Quadrat. Max-plus algebra and system theory: where we are and where to go now. *Annual Reviews in Control*, 23:207–219, 1999.
3. Eugene C Freuder and Richard J Wallace. Partial constraint satisfaction. *Artificial Intelligence*, 58:21–70, 1992.
4. Nicolas le Novère, Benjamin Bornstein, Alexander Broicher, Mélanie Courtot, Marco Donizelli, Harish Dharuri, Lu Li, Herbert Sauro, Maria Schilstra, Bruce Shapiro, Jacky L. Snoep, and Michael Hucka. BioModels Database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic Acid Research*, 1(34):D689–D691, January 2006.
5. Vincent Noel, Dima Grigoriev, Sergei Vakulenko, and Ovidiu Radulescu. Tropical geometries and dynamics of biochemical networks application to hybrid cell cycle models. In Jérôme Feret and Andre Levchenko, editors, *Proceedings of the 2nd International Workshop on Static Analysis and Systems Biology (SASB 2011)*, volume 284 of *Electronic Notes in Theoretical Computer Science*, pages 75–91. Elsevier, 2012.
6. Vincent Noel, Dima Grigoriev, Sergei Vakulenko, and Ovidiu Radulescu. Tropicalization and tropical equilibration of chemical reactions. arXiv:1303.3963, in press Contemporary Mathematics, 2013.
7. O. Radulescu, A.N. Gorban, A. Zinovyev, and V. Noel. Reduction of dynamical biochemical reaction networks in computational biology. *Frontiers in Bioinformatics and Computational Biology*, 3:131, 2012.
8. Sylvain Soliman. Invariants and other structural properties of biochemical models as a constraint satisfaction problem. *Algorithms for Molecular Biology*, 7(15), May 2012.

9. O. Viro. From the sixteenth Hilbert problem to tropical geometry. *Japanese Journal of Mathematics*, 3(2):185–214, 2008.
10. Jan Wielemaker. *SWI-Prolog 6.3.15 Reference Manual*, 1990–2013.
11. Jan Wielemaker, Tom Schrijvers, Markus Triska, and Torbjörn Lager. SWI-Prolog. *Theory and Practice of Logic Programming*, 12(1-2):67–96, 2012.

Optimizing the reference population in a genomic selection design

Jean-Michel Elsen[1], Simon de Givry[2], George Katsirelos[2], and Felicien Shumbusho[1]

[1] SAGA, UR 631, INRA, F-31320 Castanet Tolosan, France
[2] MIA-T, UR 875, INRA, F-31320 Castanet Tolosan, France
{Jean-Michel.Elsen,degivry,george.katsirelos,Felicien.Shumbusho}@toulouse.inra.fr

Abstract. In genomic selection, when candidate animals for reproduction are selected on an estimate of their breeding value from genomic information (using Single Nucleotide Polymorphims (SNP) chips), it is needed to build a reference population whose members are both genotyped on the SNPs and phenotyped for the economical trait(s) to be improved. We studied, with numerical simulations of such genomic selection plan, how to optimize the design of this reference population. The problem is summarized as minimizing a quadratic function on Boolean variables with a cardinality constraint. Integer linear/quadratic/constraint programming and weighted Max-SAT and CSP solvers are compared on a few examples.

1 Introduction

Thanks to the discovery of very abundant Single Nucleotide Polymorphisms (SNP) and availability of high throughput genotyping technologies, *genomic selection*, as described by [8] more than ten years ago, became realistic and rapidly turned to be the new standard in Dairy cattle breeding schemes [16]. Its application to other species is still a matter of discussion, as described for instance by [18] in pig or [17] in sheep. Genomic selection schemes comprise two steps. The estimation step, performed from phenotypes and genotypes recorded in a *reference population*, provides estimations of SNPs effects on the quantitative trait of interest. Different models were proposed for these estimations, the simplest, *Genomic Best Linear Unbiased Prediction* (GBLUP), modeling the performance as the sum of fixed nuisance effects and all SNPs random effects with a prior in a Gaussian distribution of known variance [8]. The selection step comprises an estimation of *Genomic Breeding Values* (GBV) merging the genotypic information about each candidate and the SNP effects previously estimated.

Amongst other factors, the efficiency of genomic selection largely depends on the design of the reference population [1, 11]. There are increasing evidence that closer the reference population to the selected population is, more precise the genomic evaluation will be. As an example, between breeds designs with SNPs estimated in a breed and selection candidates belonging to another breed (e.g. Jersey and Holstein breeds in dairy cattle) are efficient only with very dense SNP chips [14].

The present work aims at providing a tool for optimizing the reference population design. Populations displaying realistic linkage disequilibrium structures were simulated. Efficiency of different reference population designs were evaluated from the mean correlation between true and GBLUP estimated breeding values. As in [13], this criterion was used as an objective function to be maximized given a constraint of the reference population size. This paper describes a new approach to perform this optimization using a Taylor approximation in the framework of integer linear/quadratic/constraint programming and weighted Max-SAT/CSP.

2 The genomic selection design problem

The phenotyped population has n_p individuals. Among them, we want to select n_r individuals, forming the reference population, to be genotyped on m markers. The candidate population has n_c individuals, different of those in the phenotyped population. These candidate individuals are assumed to be already genotyped.

We assume a GBLUP linear mixed model [19] for the observed phenotypes of the reference population with the genetic effects modeled as random effects (and no fixed effects for the purpose of this study). In matrix notation, we have:

$$y = Xq + e$$

where $y = (y_1, \ldots, y_{n_r})$ is the column vector of observed (single value) phenotypes for the reference population, $X = (\forall l \in [1, n_r], \forall i \in [1, m] \quad x_{li})$ the matrix of recentered genotypes for the reference population with n_r rows and m columns, $q = (q_1, \ldots, q_m)$ is the column vector of m random genetic effects, and e is a vector of independent and identically distributed random error terms representing an environmental deviation. For each genotype, we have $x_{li} = a_{li} - 2f_i$, where $a_{li} \in \{0, 1, 2\}$ is the number of alleles A_i possessed by individual l at marker i (having two possible alleles A_i, B_i), and f_i is the frequency of A_i in the population.

q and e follow a normal distribution with zero mean and different variances: $\forall i \in [1, m], q_i \sim \mathcal{N}(0, \sigma_q^2)$ and $e \sim \mathcal{N}(0, \sigma_e^2)$. We denote $\lambda = \frac{\sigma_e^2}{\sigma_q^2}$, a known parameter value in our simulation. It can be shown that λ is related to heritability h^2 of the observed phenotypes: $\lambda = \frac{(1-h^2)2\sum_i^m f_i(1-f_i)}{h^2}$.

The estimation of the random genetic effects $\hat{q} = (\hat{q}_1, \ldots, \hat{q}_m)$ is obtained by the following formula [19]:

$$\hat{q} = (X^T X + \lambda I)^{-1} X^T y$$

We define the quality of this estimation on the candidate population by the mean square Pearson correlation $r^2_{g,\hat{g}} = \frac{1}{n_c} \sum_k^{n_c} r^2_{g_k,\hat{g}_k}$, by marginalizing the phenotypes, where $g_k = w_k q$ is the genotypic value of individual k and $\hat{g}_k = w_k \hat{q}$ its estimate, with $w_k = (w_{k1}, \ldots, w_{km})$ is the row vector of recentered genotypes of individual k in the candidate population.

Using standard calculus we get:

$$r^2_{g_k,\hat{g}_k} = \frac{cov^2(g_k,\hat{g}_k)}{var(g_k)var(\hat{g}_k)} = \frac{var(\hat{g}_k)}{var(g_k)} = 1 - \lambda \frac{w_k(X^TX+\lambda I)^{-1}w_k^T}{w_k {w_k}^T}$$

Our goal is to maximize the quality of the estimation, that is to minimize:

$$\begin{aligned} D(X) &= \lambda \sum_k^{n_c} \frac{w_k(X^TX+\lambda I)^{-1}w_k^T}{w_k {w_k}^T} \\ &= \lambda \sum_k^{n_c} \tilde{w}_k(X^TX+\lambda I)^{-1}\tilde{w}_k^T \end{aligned}$$

with $\forall k \in [1, n_c], \forall i \in [1, m] \quad \tilde{w}_{ki} = \frac{w_{ki}}{\sqrt{\sum_j^m w_{kj}^2}}$, the normalized genotypes in the candidate population.

For $m = 2$, we have:

$$\begin{aligned} D(X) &= \lambda \sum_k^{n_c} (\tilde{w}_{k1}, \tilde{w}_{k2})(X^TX+\lambda I)^{-1}(\tilde{w}_{k1}, \tilde{w}_{k2})^T \\ &= \lambda \sum_k^{n_c} \frac{\tilde{w}_{k1}^2(v_2+\lambda) + \tilde{w}_{k2}^2(v_1+\lambda) - 2\tilde{w}_{k1}\tilde{w}_{k2}c}{(\tilde{w}_{k1}^2+\tilde{w}_{k2}^2)((v_1+\lambda)(v_2+\lambda)-c^2)} \end{aligned}$$

with $v_1 = \sum_l^{n_r} x_{l1}^2$, $v_2 = \sum_l^{n_r} x_{l2}^2$, and $c = \sum_l^{n_r} x_{l1}x_{l2}$.

For the general case, we will approximate the matrix inversion by using a Taylor approximation. In the case of a Taylor approximation of order 1, we have:

$$\begin{aligned} D(X) &= \lambda \sum_k^{n_c} \tilde{w}_k(X^TX+\lambda I)^{-1}\tilde{w}_k^T \\ &= \sum_k^{n_c} \tilde{w}_k(\frac{X^TX}{\lambda} + I)^{-1}\tilde{w}_k^T \\ &\approx \sum_k^{n_c} \tilde{w}_k(I - \frac{X^TX}{\lambda})\tilde{w}_k^T \\ &\approx \sum_k^{n_c}\sum_i^m \tilde{w}_{ki}^2 - \frac{1}{\lambda}\sum_k^{n_c}\sum_i^m \tilde{w}_{ki}\sum_j^m \tilde{w}_{kj}\sum_l^{n_r} x_{li}x_{lj} \end{aligned}$$

We can rewrite this objective function by introducing Boolean variables $\delta_l \in \{0, 1\}$ for all individuals in the phenotyped population ($l \in [1, n_p]$). We denote z_{li} the recentered genotype of individual l at marker i in this population (whereas x_{li} is on the reference population).

We have:

$$D(X) \approx \sum_{k}^{n_c}\sum_{i}^{m} \tilde{w}_{ki}^2 - \frac{1}{\lambda}\sum_{k}^{n_c}\sum_{i}^{m}\tilde{w}_{ki}\sum_{j}^{m}\tilde{w}_{kj}\sum_{l}^{n_p}\delta_l z_{li} z_{lj}$$

$$D(X) \approx D_1(X) = \underbrace{\sum_{k}^{n_c}\sum_{i}^{m}\tilde{w}_{ki}^2}_{a} - \frac{1}{\lambda}\sum_{l}^{n_p}\underbrace{\sum_{k}^{n_c}\left(\sum_{i}^{m}\tilde{w}_{ki} z_{li}\right)^2}_{b_{ll}}\delta_l$$

In the case of a Taylor approximation of order 2, we have:

$$\begin{aligned}
D(X) \approx D_2(X) &= \sum_{k}^{n_c}\tilde{w}_k(I - \frac{X^TX}{\lambda} + \frac{(X^TX)^2}{\lambda^2})\tilde{w}_k^T \\
&= \sum_{k}^{n_c}\sum_{i}^{m}\tilde{w}_{ki}^2 - \frac{1}{\lambda}\sum_{k}^{n_c}\sum_{i}^{m}\tilde{w}_{ki}\sum_{j}^{m}\tilde{w}_{kj}\sum_{l}^{n_r}x_{li}x_{lj} \\
&\quad + \frac{1}{\lambda^2}\sum_{k}^{n_c}\sum_{i}^{m}\tilde{w}_{ki}\sum_{j}^{m}\tilde{w}_{kj}\sum_{h}^{m}(\sum_{l}^{n_r}x_{li}x_{lh})(\sum_{l}^{n_r}x_{lh}x_{lj}) \\
&= \sum_{k}^{n_c}\sum_{i}^{m}\tilde{w}_{ki}^2 - \frac{1}{\lambda}\sum_{k}^{n_c}\sum_{i}^{m}\tilde{w}_{ki}\sum_{j}^{m}\tilde{w}_{kj}\sum_{l}^{n_p}\delta_l z_{li}z_{lj} \\
&\quad + \frac{1}{\lambda^2}\sum_{k}^{n_c}\sum_{i}^{m}\tilde{w}_{ki}\sum_{j}^{m}\tilde{w}_{kj}\sum_{h}^{m}(\sum_{l}^{n_p}\delta_l z_{li}z_{lh})(\sum_{o}^{n_p}\delta_o z_{oh}z_{oj})
\end{aligned}$$

Finally we reorganize the terms depending on the different combinations of δ_l variables.

$$\begin{aligned}
D_2(X) &= a - \frac{1}{\lambda}\sum_{l}^{n_p}b_{ll}\delta_l + \frac{1}{\lambda^2}\sum_{l}^{n_p}\sum_{o}^{n_p}\left(\sum_{h}^{m}z_{lh}z_{oh}\right)\underbrace{\left[\sum_{k}^{n_c}\left(\sum_{i}^{m}\tilde{w}_{ki}z_{li}\right)\left(\sum_{j}^{m}\tilde{w}_{kj}z_{oj}\right)\right]}_{b_{lo}}\delta_l\delta_o \\
&= a - \frac{1}{\lambda}\sum_{l}^{n_p}b_{ll}\delta_l + \frac{1}{\lambda^2}\sum_{l}^{n_p}\sum_{o}^{n_p}\underbrace{\left(\sum_{h}^{m}z_{lh}z_{oh}\right)}_{c_{lo}}b_{lo}\,\delta_l\delta_o
\end{aligned}$$

To conclude we are going to minimize a quadratic objective function with $n_p(1+n_p)$ terms, n_p Boolean variables ($\delta_l\,\forall l \in \{1,\ldots,n_p\}$), and an additional linear *cardinality constraint* $\sum_l^{n_p}\delta_l = n_r$. Note that the time for computing the objective function coefficients is already $O(n_p^2 n_c m)$. Depending on the size of this minimization problem, it can be solved by complete search methods (*e.g.,* best-first or depth-first Branch and Bound) or by local search methods (*e.g.,* simulated annealing or Tabu search) in the framework of integer linear/quadratic/constraint programming and weighted Max-SAT/CSP.

3 Integer linear/quadratic/constraint programming models

We add n_p^2 extra variables γ_{lo} in order to linearize the quadratic objective function. For every pair of Boolean variables (δ_l, δ_o), there is a Boolean variable γ_{lo} that is equal to 1 iff $\delta_l = \delta_o = 1$. We have the following 0/1 linear programming (01LP) formulation:

$$\min \sum_{l}^{n_p} \sum_{o}^{n_p} c_{lo}\gamma_{lo} - \lambda \sum_{l}^{n_p} b_{ll}\delta_l$$

$$\text{s.t.} \sum_{l}^{n_p} \delta_l = n_r$$
$$\delta_l + \delta_o \leq 1 + \gamma_{lo} \quad (\forall l \in \{1,\ldots,n_p\}, o \in \{1,\ldots,n_p\})$$
$$\gamma_{lo} \leq \delta_l \quad (\forall l \in \{1,\ldots,n_p\}, o \in \{1,\ldots,n_p\})$$
$$\gamma_{lo} \leq \delta_o \quad (\forall l \in \{1,\ldots,n_p\}, o \in \{1,\ldots,n_p\})$$

By removing the last three inequations and replacing γ_{lo} by $\delta_l * \delta_o$, we get a 0/1 quadratic programming (01QP) formulation. The same 01QP formulation can be used by constraint programming (CP) languages such as MiniZinc [6]. By removing the cardinality constraint, we get a pure boolean quadratic optimization (BQO) formulation.

4 Weighted CSP and weighted Max-SAT models

A Weighted Constraint Satisfaction Problem (WCSP) [7] P is a triplet $P = (X, F, k)$ where X is a set of variables and F a set of cost functions. Each variable $x \in X$ has a finite domain of values that can be assigned to it. A cost function $f(S) \in F$, with scope S a sequence of distinct variables of X, is a function which associates to every assignment t of its variables a positive integer in $[0, k]$ where k is a maximum integer cost used for representing forbidden assignments.

The Weighted Constraint Satisfaction Problem is to find a complete assignment t minimizing the total cost $\mathcal{W} = \sum_{f(S)\in F} f(t[S])$ where $t[S]$ denotes the projection of t over variables S. This optimization problem has an associated NP-complete decision problem.

The genomic selection cost minimization problem has $X = \{\delta_1, \ldots, \delta_{n_p}, x_1, \ldots, x_{n_p+1}\}$, all δ_l (resp. x_l) domains are equal to {0,1} (resp. $[0, n_r]$), $F = \{f(\delta_l)\,\forall l \in \{1,\ldots,n_p\}\} \cup \{f(\delta_l, \delta_o)\,\forall l \times o \in \{1,\ldots,n_p\}^2, l \neq o\} \cup \{f(x_1), f(x_{n_p+1})\} \cup \{f(x_l, \delta_l, x_{l+1})\,\forall l \in \{1,\ldots,n_p\}\}$, and $k = +\infty$.

We define:

$$\forall l \in \{1,\ldots,n_p\} \quad f(\delta_l) = \lfloor 0.5 + M(\lambda b_{ll}(1-\delta_l) + c_{ll}\delta_l)\rfloor \text{ if } c_{ll} \geq 0$$

$$
\begin{array}{rrcl}
 & & = & \lfloor 0.5 + M(\lambda b_{ll} - c_{ll})(1-\delta_l) \rfloor \quad \text{if } c_{ll} < 0 \\
\forall l \times o \in \{1,\ldots,n_p\}^2, l \neq o & f(\delta_l, \delta_o) = & \lfloor 0.5 + M c_{lo} \delta_l \delta_o \rfloor & \text{if } c_{lo} \geq 0 \\
 & & = & \lfloor 0.5 + M c_{lo} (\delta_l \delta_o - 1) \rfloor \quad \text{if } c_{lo} < 0 \\
 & f(x_1) = & 0 & \text{if } x_1 = 0 \\
 & f(x_1) = & k & \text{if } x_1 \neq 0 \\
 & f(x_{n_p+1}) = & 0 & \text{if } x_{n_p+1} = n_r \\
 & f(x_{n_p+1}) = & k & \text{if } x_{n_p+1} \neq n_r \\
\forall l \in \{1,\ldots,n_p\} & f(x_l, \delta_l, x_{l+1}) = & 0 & \text{if } x_l + \delta_l = x_{l+1} \\
 & & = & k \quad \text{if } x_l + \delta_l \neq x_{l+1}
\end{array}
$$

with M a large value used to convert real numbers into integers (rounding to the nearest integer). We have $\mathcal{W} \simeq D_2(X) + C$, where C is a positive constant shift value used in order to keep all cost functions positive. Cost functions $f(x_l, \delta_l, x_{l+1})$ are used to decompose the cardinality constraint $\sum_l^{n_p} \delta_l = n_r$ into an equivalent set of low arity cost functions, by introducing extra counting variables $\{x_1, \ldots, x_{n_p+1}\}$.

By removing the part for encoding the cardinality constraint, we get a formulation ready for Max-SAT solvers.

5 Preliminary results

5.1 Simulation of genomic data

A population with a linkage disequilibrium (LD) extent comparable to one found in a real sheep population (*Manech Tête Rousse* breed) was simulated with the `QMSim` software [15]. For that, a historical population of $20,000$ individuals was simulated for $1,050$ generations by considering an equal number of individuals from both sexes, discrete generations, random matings, no selection and no migration to create an initial LD, and establish a mutation-drift equilibrium state. For the first $1,000$ generations, the population size was decreased to $2,000$ individuals and then increased to $16,000$ individuals within the last 50 generations to create a bottleneck and eventual decrease in effective population size as known in domestic animals. Furthermore, $15,000$ females and 350 males from the last historical generation were used as founders of the selected population. From the founder population, 10 overlapping generations of selection (with 20% and 30% replacement rate for females and males, respectively) and random mating were simulated as contemporary born animals. For the purpose of this study, females from generations 8 and 9 served as the phenotyped population, *i.e.,* $n_p \leq 20,928$, where to select the reference population, and males from generation 10 were used as the candidate population, *i.e.,* $n_c \leq 10,453$. The simulated genome consisted of $m = 10,000$ SNP markers, equally spaced across 5 chromosomes of 100 cM each and $2.5 * 10^{-5}$ mutation rate per marker.

5.2 Comparison of 01LP, 01QP, 01BQO, CP, Max-SAT, WCSP solvers

We compare the models described in Section 3 and 4, in terms of CPU-time, for solving the Taylor approximation of order 2. We vary the problem size n_p from 20 to 200, and experiment with different ratios $\frac{n_r}{n_p}$ from 0.25 to 0.5. We also compare with an unconstrained model where the cardinality constraint $\sum_l^{n_p} \delta_l = n_r$ has been discarded.

We compare the 01LP solver `SCIP` (version 1.2.0), the 01LP and 01QP solver IBM ILOG `cplex` (version 12.4.0.0), the semidefinite programming based BQO tool `BiqMac` [12], the pseudo-Boolean solvers `clasp` (version 2.0.4) and `SAT4J` (version 2.3.4), the CP solver `mistral` (version 1.3.40), the Max-SAT solvers `minimaxsat` [5] and `maxhs` [3] (both using the *tuple* encoding as described in [2]), all solvers using default options, and the WCSP solver `toulbar2` (version 0.9.6[3]) using default options except an initial limited discrepancy search phase [4] with a maximum discrepancy of 2 (option `-l=2` and no initial upper bound). `SCIP`, `toulbar2`, and `mistral` are accessed via the Python multi-solver modeling interface offered by `NumberJack`[4]. All real value coefficients in the models are multiplied by $M = 0.01$ and rounded to the nearest integer, ensuring completeness of the solvers. We measured the search effort for finding the optimum and proving optimality as reported in Table 1.

For the smallest instances ($n_p \in [20, 100]$), the quadratic programming solver QP/`cplex` and the semidefinite programming based boolean quadratic optimization tool `BiqMac`, used in the unconstrained case only, clearly dominate the other solvers. For the largest instances ($n_p \in \{200\}$), all the approaches failed to solve the problem in less than 10 hours.

In order to solve large problems (up to $n_p = 200$), we use a two-step procedure. First, we apply a local search method, called `ID Walk` for *Intensification / Diversification Walk* [10], available as a library [9][5] integrated in `toulbar2`. Due to its neighborhood structure (changing only one variable assignment per move), `ID Walk` can only be applied to the unconstrained problem. We perform 1 run of `ID Walk` with 10,000 iterations, selecting at random among 200 candidate neighbors. The best solution found by the local search method is then used as a pre-selection of the individuals[6] such that the second step is done by a complete search method (using `SCIP`) to satisfy the cardinality constraint. The resulting two-step procedure is called `ID Walk&SCIP`.

For the smallest instances solved optimally by complete search methods ($n_p \in [20, 100]$), `ID Walk&SCIP` always found the optimum for the unconstrained

[3] `http://mulcyber.toulouse.inra.fr/projects/toulbar2`

[4] `http://numberjack.ucc.ie/` and `http://github.com/eomahony/Numberjack/tree/fzn`.

[5] INCOP version 1.1 `http://www-sop.inria.fr/coprin/neveu/incop/presentation-incop.html`

[6] Either by discarding the remaining unselected individuals if too many individuals have been selected by the local search method, or by fixing the selected individuals if they are less than the required number n_r.

problems. The distance to the optimum increases slightly when the required number n_r is (very) different than the one found for the unconstrained case, *e.g.,* being up to 34% for $n_p = 100, n_r = 50$ as reported in Table 2. The overall time of the two-step procedure is clearly dominated by its second step, *e.g.,* unfinished after 10 hours for $n_p = 200, n_r = 100$, which means that the proposed approach should scale to larger problems only if n_r is close to the optimal unconstrained number of selected individuals.

Table 1. Time in seconds of complete search methods (−: unsolved after 10 hours, N/A: non applicable for `BiqMac`,`minimaxsat`, and `maxhs`, which were applied only in the unconstrained case). For unconstrained instances, the number of selected individuals (n_r) in the optimal solution is given in parentheses.

	`SCIP`	`cplex`	QP/`cplex`	`BiqMac`	`clasp`	`SAT4J`	`mistral`	`minimaxsat`	`maxhs`	`toulbar2`
n_p	$n_r = 25\%$									
20	0.7	0.3	0.02	N/A	0.01	0.7	1.3	N/A	N/A	0.16
40	15.8	6.7	0.51	N/A	16,452	–	–	N/A	N/A	48.7
60	942.8	1,089	12.3	N/A	–	–	–	N/A	N/A	–
100	–	–	223.2	N/A	–	–	–	N/A	N/A	–
200	–	–	–	N/A	–	–	–	N/A	N/A	–
n_p	$n_r = 50\%$									
20	2.5	1.0	0.02	N/A	0.8	3.4	19.3	N/A	N/A	0.14
40	101.1	22.7	0.65	N/A	–	–	–	N/A	N/A	77.2
60	–	26,853	9.3	N/A	–	–	–	N/A	N/A	–
100	–	–	1,031	N/A	–	–	–	N/A	N/A	–
200	–	–	–	N/A	–	–	–	N/A	N/A	–
n_p	n_r unconstrained (found $n_r = (9, 15, 21, 25)$ resp. for $n_p = (20, 40, 60, 100)$)									
20	1.9	1.1	0.02	0.86	1.1	5.3	19.8	2.1	14.7	0.04
40	94.5	68.3	1.1	13.7	–	–	–	4,781	–	5.0
60	–	20,249	22.4	29.8	–	–	–	–	–	11,062
100	–	–	348.1	87.8	–	–	–	–	–	–
200	–	–	–	–	–	–	–	–	–	–

6 Conclusion

We have presented an optimization problem occuring in the context of genomic selection design. Finding the optimal reference population can be approximated by a quadratic minimization problem on Boolean variables with a cardinality constraint. Preliminary results showed that only quadratic programming solvers such as `cplex` and the semidefinite programming based boolean quadratic optimization tool `BiqMac`, in the unconstrained case, are able to solve optimally

Table 2. Relative distances between the best solutions found by the local search method `ID Walk` followed by `SCIP` post-processing and by a complete search method (QP/`cplex`). CPU-times in seconds for `ID Walk` and `SCIP` are given in parentheses when appropriate.

	`ID Walk&SCIP`		
	n_r/n_p		
n_p	25%	50%	Unconstr.
20	$0.17\%(0.3+0.1)$	$0\%(0.3+0.03)$	$0\%(n_r=9)$
40	$0.32\%(0.6+0.39)$	$4.17\%(0.6+1.17)$	$0\%(n_r=15)$
60	$0.59\%(0.9+0.64)$	$4.56\%(0.9+9.17)$	$0\%(n_r=21)$
100	$0\%(1.4+2.47)$	$34.2\%(1.4+18,684)$	$0\%(n_r=25)$
200	$14.32\%(2.8+22,746)$	$55.16\%(2.8+36,000)$	$0\%(n_r=35)$

the Taylor approximation of order 2 for a phenotyped population up to 100 individuals. Also, performances of all the solvers vary based on the tightness of the cardinality constraint. These results are useful to assess the quality of local search methods, which are able to tackle much larger problems. Moreover, we have shown how to combine a local search and a complete method in a simple two-step procedure, while degrading the solution quality when the desired number of selected individuals differs significantly from the local search solution. More experiments remain to be done to better distinguish the quality of the two Taylor approximations, and to analyze the performance of local search methods on realistic datasets ($n_p \approx 10,000$) and the properties of the resulting reference population structures.

References

1. Albrecht, T., Wimmer, V., Auinger, H.J., Erbe, M., Knaak, C.: Genome-based prediction of testcross values in maize. Theor. Appl. Genet. 123, 339–350 (2011)
2. Bacchus, F.: Gac via unit propagation. In: Principles and Practice of Constraint Programming–CP 2007. pp. 133–147. Springer (2007)
3. Davies, J., Bacchus, F.: Solving maxsat by solving a sequence of simpler sat instances. In: Principles and Practice of Constraint Programming–CP 2011, pp. 225–239. Springer (2011)
4. Harvey, W.D., Ginsberg, M.L.: Limited discrepency search. In: Proc. of the 14^{th} IJCAI. Montréal, Canada (1995)
5. Heras, F., Larrosa, J., Oliveras, A.: Minimaxsat: An efficient weighted max-sat solver. J. Artif. Intell. Res.(JAIR) 31, 1–32 (2008)
6. Marriott, K., Nethercote, N., Rafeh, R., Stuckey, P., de la Banda, M.G., Wallace, M.: The design of the zinc modelling language. Constraints 13(3), 229–267 (2008)
7. Meseguer, P., Rossi, F., Schiex, T.: Soft constraints processing. In: Rossi, F., van Beek, P., Walsh, T. (eds.) Handbook of Constraint Programming, chap. 9. Elsevier (2006)
8. Meuwissen, T.H., Hayes, B.J., Goddard, M.E.: Prediction of total genetic value using genome-wide dense marker maps. Genetics 157, 1819–1829 (2001)
9. Neveu, B., Trombettoni, G.: INCOP: An Open Library for INcomplete Combinatorial OPtimization. In: Proc. of CP-03. pp. 909–913. Cork, Ireland (2003)

10. Neveu, B., Trombettoni, G., Glover, F.: Id walk: A candidate list strategy with a simple diversification device. In: CP. pp. 423–437 (2004)
11. Pszczola, M., Strabel, T., Mulder, H., Calus, M.: Reliability of direct genomic values for animals with different relationships within and to the reference population. J. Dairy Sci. 95, 389–400 (2012)
12. Rendl, F., Rinaldi, G., Wiegele, A.: Solving Max-Cut to optimality by intersecting semidefinite and polyhedral relaxations. Math. Programming 121(2), 307 (2010)
13. Rincent, R., Laloë, D., Nicolas, S., Altmann, T., Brunel, D., Revilla, P., Rodríguez, V., Moreno-Gonzalez, J., Melchinger, A., Bauer, E., Schoen, C.C., Meyer, N., Giauffret, C., Bauland, C., Jamin, P., Laborde, J., Monod, H., Flament, P., Charcosset, A., Moreau, L.: Maximizing the reliability of genomic selection by optimizing the calibration set of reference individuals: Comparison of methods in two diverse groups of maize inbreds (zea mays l.). Genetics 192, 715–728 (2012)
14. de Roos, A., Hayes, B., Spelman, R., Goddard, M.: Linkage disequilibrium and persistence of phase in holstein-friesian, jersey and angus cattle. Genetics 179(3), 1503–1512 (2008)
15. Sargolzaei, M., Schenkel, F.S.: Qmsim: a large-scale genome simulator for livestock. Bioinformatics 25, 680–681 (2009)
16. Schaeffer, L.: Strategy for applying genome-wide selection in dairy cattle. J Anim Breed Genet 123(4), 218–223 (2006)
17. Shumbusho, F., Raoul, J., Astruc, J., Palhiere, I., Elsen, J.: Potential benefits of genomic selection on genetic gain of small ruminant breeding programs. J Anim Sci in press (2013)
18. Tribout, T., Larzul, C., Phocas, F.: Efficiency of genomic selection in a purebred pig male line. J Anim. Sci 12, 4164–4176 (2012)
19. West, B.T., Welch, K.B., Galecki, A.T.: Linear mixed models: A practical guide to using statistical software. Chapman & Hall/CRC (2007)

Soft Pattern Discovery in Pre-Classified Protein Families through Constraint Optimization

David Lesaint[1], Deepak Mehta[2], and Barry O'Sullivan[2]

[1] LERIA, Université d'Angers, F-49045 Angers, France
david.lesaint@univ-angers.fr
[2] University College Cork, 4C, Cork, Ireland
d.mehta,b.osullivan@4c.ucc.ie

Abstract. A considerable effort has been invested in discovering patterns amongst protein sequences. This paper introduces the notion of *soft pattern* to characterize existing classes in a dataset. A soft pattern for a class is an exclusive set of subsequences, called c-blocks, which are common to the class and whose embeddings feature consistent mismatches. Soft patterns are less verbose than regular expressions and are not restricted to be linear as opposed to class signatures produced by multiple sequence alignment methods. We formalize a general soft pattern computation problem and present two variants enforcing either sequencing or non-replication of c-blocks in a pattern. We cast these problems into a lexicographic constraint optimization framework and present a two-stage procedure to solve the replication-free problem variant.

1 Introduction

Detecting patterns in amino acid sequences is critical to understand the relationship between the function and structure of proteins. This problem has been thoroughly investigated in Bioinformatics, notably via multiple sequence alignment (MSA) and pattern recognition approaches [1–6]. MSA methods insert gaps in input sequences (interpreted as amino acid indels) in order to align common blocks that have a limited amount of mismatch (interpreted as point mutations). By design, these methods only detect sequences of common blocks which is restrictive when analyzing proteins featuring different block orderings. Besides, MSA does not address the problem of enforcing pattern exclusivity when dealing with pre-classified sequences. The same applies to pattern recognition methods, and while the latter are less sensitive to block orderings, they may yield verbose or loose class signatures.

This paper proposes a constraint optimization approach to detect exclusive patterns in pre-classified protein families. The key requirements are addressed separately with constraints which gives the ability to use dedicated algorithms but also to integrate and reason about other dimensions of the problem (e.g., chemico-physical properties). The approach relies on the notion of *soft pattern* to accommodate point mutations and variable block orderings. A soft pattern for a class is an exclusive set of common subsequences where each subsequence features

the same point mutations across the class. We call *c-blocks* such subsequences and refer to point mutations as *hashes*. In order to facilitate biological interpretation and limit computational complexity, subsumption (*coverage*) between the c-blocks of a pattern is prohibited. Lexical subsumption and overlapping may also be prohibited to prevent *replication* or enforce *sequencing* of c-blocks.

On this basis, we formulate a core soft pattern discovery problem which we specialize into two different variants. The general problem is to detect exclusive and coverage-free patterns. The first variant detects replication-free patterns that are minimally exclusive and maximally refined while the second variant detects exclusive sequential patterns. All problems incorporate parametric constraints that bound the slack (i.e., the maximum number of consecutive hashes in c-blocks) and width of patterns (i.e., the minimum span of c-blocks) in order to discard degenerate solutions. Note that MSA problems may be cast as particular cases of the sequential pattern discovery variant.

The replication-free problem variant yields strong class characterisations since solution patterns are stripped of rendundant c-blocks while the remaining c-blocks cannot be specialized further, nor extended. We restrict our attention to this variant which we model as a lexicographic constraint optimization problem (COP). This COP addresses subsequence matching, commonality, hash-consistency, slack and width bounding, exclusivity, non-coverage and non equivalence of c-blocks through separate constraints. In particular, exclusivity is reducible to a set covering problem. The lexicographic objective function prioritizes minimum pattern cardinality over length maximality which ensures solutions are minimally exclusive and maximally refined soft patterns. We present a two-stage procedure to solve this COP. The procedure composes maximal c-blocks from minimal blocks before computing exclusive solution patterns.

The paper is organized as follows. Sec.2 discusses the notion of soft pattern and motivates the different pattern discovery tasks. Sec.3 formalizes the problems and introduces the constraint optimization model for computing replication-free patterns. Sec.4 presents the two-stage procedure for this problem. Sec.5 concludes. Due to lack of space, we refer the reader to [7] which includes proofs and presents a declarative implementation in Minizinc [8].

2 Soft Patterns

This section motivates the three pattern discovery problems and introduces the underlying concepts. Given a dataset composed of classes of protein sequences, the objective is to characterize a class by a *soft pattern*. A soft pattern is a set of *c-blocks* where a c-block corresponds to a subsequence that is common to the sequences of the class and whose adjacent characters are separated by the same number of mismatching characters in each sequence. We call *block* the embedding of a c-block in a sequence, and *hash* any mismatch inside a c-block. Note that a c-block cannot start nor end with a hash but no further constraint on hashes are assumed (allowed amino-acid substitutions and c-blocks starting or ending with hashes are not discussed here). Fig.1 shows a dataset made of

	Class A		*Class B*		
	protein	*protein*	*protein*	*protein*	*protein*
	ACADEEC	EECAEA	CADA	ADAAEEC	CAEEC
	1234567	123456	1234	1234567	12345
c-block	A#A	A#A	A#A	A#A	
c-block	CA	CA	CA		CA
c-block	EEC	EEC		EEC	EEC

Fig. 1. A soft pattern {A#A,CA,EEC} for class A.

two classes A and B. Class A includes proteins ACADECC and EECAEA and a soft pattern of three c-blocks is shown for it. The first c-block corresponds to the common subsequence AA with embeddings $\{1, 3\}$ in the first protein and $\{4, 6\}$ in the second, both sharing a hash in second position. This c-block is given signature A#A where # indicates a hash. The other two c-blocks are CA and EEC.

Beyond subsequence matching, commonality and hash-consistency, a key requirement is that patterns discriminate classes, that is, a pattern for a class should not match any "foreign" protein (i.e., any protein not in the class). *Pattern exclusivity* is defined differently based on the constraints one wishes to impose between the c-blocks of a pattern. We consider three constraints, namely, *non-coverage*, *non-replication* and *sequentiality*. Non-coverage means that no c-block embedding should contain another in a protein which is legitimate from biological and computational standpoints. In this case, a pattern is exclusive if any set of blocks in a foreign protein matching the c-blocks is itself coverage-free.

Optionally, a further restriction is to prohibit c-block *replication*. A c-block may indeed be repeated in a coverage-free pattern (i.e., c-blocks with the same signature) or replicated within larger c-blocks (e.g., a pattern containing AC#A and C#A). Replication subsumes coverage and determining exclusivity for replication-free patterns boils down to proving that every foreign protein is incompatible with one c-block. This is illustrated in Fig.1 where the pattern is replication-free and exclusive to class A since no protein of class B matches its three c-blocks.

Another alternative is to search for patterns whose c-blocks occur in the same order in each protein of the class. Since block embeddings may be abstracted as intervals, interval relations (e.g., Allen relations) may be used to characterize an ordering between blocks. The simplest and most intuitive relation is the precedence relation which guarantees that intervals do not overlap nor meet. Exclusivity holds in this case if any set of blocks in a foreign protein matching the c-blocks cannot be sequenced consistently with the pattern.

Different objectives may be pursued on top of these features. One such objective is to make patterns *minimally exclusive* and devoid of redundancy from a classification viewpoint. Minimal exclusivity is achieved by discarding c-blocks that do not contribute to making a pattern exclusive (notably, c-blocks that do not exclude any protein). This is the case for the pattern of Fig.1 since dropping any c-block yields a non-exclusive pattern. It would not be so if class B did not contain protein CAEEC, A#A being redundant in this case. Another objec-

tive is to try refining c-blocks as much as possible by substituting hashes with matching characters (*specialization*) or adding matching characters left or right, possibly introducing new hashes in doing so (*extension*). The pattern of Fig.1 is *maximally refined* in this sense.

Three indicators may also be used to assess pattern quality: *slack* (maximum number of consecutive hashes in c-blocks), *width* (minimum span of c-blocks) and *length* (total number of matching characters in the c-blocks). The width of a c-block is the sum of its slack and length while there is no such correlation for patterns as increasing the length of a pattern may increase the slack or decrease the width. Consistently with the need for maximality, preference goes to patterns with lower slack, greater width or length, all things being equal.

It is challenging though for domain experts to come up with a preference model aggregating all these features, criteria and objectives. On this basis, we formulate a general *soft pattern discovery problem* (SPD) and two variants. The SPD consists in computing exclusive and coverage-free patterns for a class. Note that SPD solutions may feature replicated c-blocks. The SPD enforces two parametric constraints, namely, an upper bound on slack to discard loose patterns and a lower bound on width to discard short patterns. The first variant, called *replication-free soft pattern discovery problem*, consists in computing SPD solution patterns that are minimally exclusive, maximally refined and non equivalent for replication. Such solutions are replication-free and minimal exclusivity boils down to a minimal set covering problem. The second variant, called *sequential soft pattern discovery problem*, consists in searching for sequential SPD solution patterns. Minimal exclusivity and c-block maximality may conflict under a precedence ordering which is why neither feature is imposed in this variant.

3 Soft Pattern Discovery Problems

We use the following notations. For $n \in \mathbb{N}$, $[n]$ denotes the *range* $\{i \in \mathbb{N} \mid 1 \leqslant i \leqslant n\}$, $|t|$ denotes the number of elements of a tuple or a set t and $[t]$ denotes the range $[|t|]$, t_i denotes the i-th element of a tuple t for all $i \in [t]$, and $f(A)$ denotes the image of a function $f : A \to B$ (i.e., $f(A) = \{f(i) \mid i \in A\}$). Σ denotes a finite *alphabet* and Σ^* the set of finite strings that are constructed from the characters of Σ by concatenation. Tuple notations will be used for strings.

A *class over* Σ is a tuple of strings belonging to Σ^* and a *dataset over* Σ is a tuple of classes over Σ. A string x has *length* n or is *n-long* if it consists of n, not necessarily distinct, characters from Σ. A string y is a *substring* of a string x if there exist not necessarily distinct and possibly empty strings $v_1, v_2 \in \Sigma^*$ such that $v_1 y v_2 = x$. A k-long string $y = y_1 \dots y_k$ is a *subsequence* of a string x if there exist $k+1$ not necessarily distinct and possibly empty strings $v_1, \dots v_{k+1} \in \Sigma^*$ such that $v_1 y_1 \dots v_k y_k v_{k+1} = x$. We denote this fact by $y \preceq x$. Let $y \preceq x$, an embedding of y in x is a strictly increasing function $\mu : [y] \to [x]$ such that $y_i = x_{\mu(i)}$ for all $i \in [y]$. Note that a subsequence may have multiple embeddings. A string y is a *common subsequence* of a class x if $y \preceq x_i$ for all $i \in [x]$.

A *block* for a string x is a triple $\langle \mu, y, x \rangle$ such that $y \preceq x$ and μ is an embedding of y in x. We associate to a block $b = \langle \mu, y, x \rangle$ a *hash function* $\gamma(b) : [y] \to [|x| - |y|]$ defined by $\gamma(b)(|y|) = 0$ and $\gamma(b)(i) = \mu(i+1) - \mu(i) - 1$ $(1 \leqslant i < |y|)$. For instance, `CDA` is a subsequence of `CACDEAC` that determines two blocks, the left-most one `C##D#A` verifying $\mu([y]) = \{1, 4, 6\}$ and $\gamma([y]) = \{2, 1, 0\}$. Let $b = \langle \mu, y, x \rangle$ and $b' = \langle \mu', y', x' \rangle$ be two blocks, we say that b and b' are *compatible* if $y = y'$ and $\gamma(b) = \gamma(b')$. In other words, compatible blocks embed a common subsequence with identical hashes. Let $b = \langle \mu, y, x \rangle$ and $b' = \langle \mu', y', x \rangle$ be two blocks for the same string x, we say that b *covers* b' if $\mu([y]) \supseteq \mu'([y'])$ and that b *replicates* b' if there exists a block b'' for x, not necessarily distinct from b', compatible with b' and covered by b. Coverage is a particular case of replication. For instance, block `ACD` covers and replicates the right-most block `CD` in string `CDACD` but only replicates the left-most block `CD`.

A *c-block* for a class x is a $|x|$-tuple t of compatible blocks such that $t_i = \langle \mu_i, y_i, x_i \rangle$ for all $i \in [x]$. That is, a c-block determines a common subsequence in a class whose embeddings have identical hashes. We say that a c-block is *compatible* with a block b if its blocks are compatible with b; and that it is *incompatible* with a string z if there is no block for z compatible with it. Let t and t' be two c-blocks for a class x, we say that t *covers* t' (respectively, t *replicates* t'), denoted $t \geqslant t'$ (resp., $t \succeq t'$), if there exists $i \in [x]$ such that t_i covers t'_i (resp., t_i replicates t'_i). Replication is a partial order that subsumes coverage which is itself non-transitive. Both relations preserve incompatibility, i.e., if $t \succeq t'$ and t' is incompatible with string z then t is incompatible with z. We introduce the following relations: $t < t' \Leftrightarrow (t \leqslant t' \wedge \neg(t \geqslant t'))$, $t \cong t' \Leftrightarrow (t \preceq t' \wedge t \succeq t')$, and $t \prec t' \Leftrightarrow (t \preceq t' \wedge \neg(t \succeq t'))$. $\prec$ is a strict partial order as opposed to $<$ and $\cong$ is an equivalence relation (equivalent c-blocks have signatures over $\Sigma \cup \{\#\}$).

A *pattern* for a class x is a set of c-blocks for x. We say that a pattern is compatible with a string z if one can replicate its c-blocks in z without coverage. Formally, let c be a dataset and p be a pattern for class c_i for some $i \in [c]$, p is *compatible* with a string z if there exists a set q of blocks for z such that (1) for all $t, t' \in p$, $t \neq t'$, there exists $b \in q$ and $b' \in q$ such that $b \neq b'$, t is compatible with b and t' is compatible with b', and (2) for all $b, b' \in q$, $b \neq b'$, b does not cover b'. If there is no replication in p, it suffices to show that each c-block of p is compatible with z. We say that p is *exclusive* to c_i wrt. c if for all $j \in [c]$ such that $j \neq i$, for all $k \in [c_j]$, p is incompatible with c_{j_k}. We say that p is *minimally exclusive* for a class c_i wrt. a dataset c if any pattern strictly included in p is not exclusive for c_i wrt. c.

We denote by λ, σ and ω the length, slack and width functions used for blocks, c-blocks or patterns. Let $b = \langle \mu, y, x \rangle$ be a block, $\lambda(b) = |y|$, $\sigma(b) = \max_{i \in [y]} \gamma(b)(i)$ and $\omega(b) = \mu(|y|) - \mu(1) + 1$. Let t be a c-block, $\lambda(t) = \lambda(t_i)$, $\sigma(t) = \sigma(t_i)$ and $\omega(t) = \omega(t_i)$ where $i \in [t]$. Let p be a pattern, $\lambda(p) = \sum_{t \in p} \lambda(t)$, $\sigma(p) = max_{t \in p} \sigma(t)$ and $\omega(p) = \min_{t \in p} \omega(t)$. Let $k \in \mathbb{N}$ and x denote a block, c-block or pattern, we say that x is *k-loose* if $\sigma(x) \leqslant k$; x is *strict* if it is 0-loose; and x is *k-wide* if $\omega(x) \geqslant k$. Let $l \in \mathbb{N}$ and $w \in \mathbb{N}$, a c-block t for a class x such that $\sigma(t) \leqslant l$ and $\omega(t) \geqslant w$ is $<^l_w$*-minimal over* x (resp. $<^l_w$*-maximal*,

$\prec_w^l$*-minimal*, $\prec_w^l$*-maximal*) if there is no c-block t' for x such that $\sigma(t') \leqslant l$, $\omega(t') \geqslant w$ and $t > t'$ (resp., $t < t'$, $t \succ t'$, $t \prec t'$).

The soft pattern discovery problem (SPD) consists in determining slack- and width-bounded patterns that are exclusive and coverage-free. The replication-free SPD (RSPD) requires that c-blocks be maximal for $\prec_w^s$ and non equivalent to prevent replications (non-equivalence may be dropped to allow repetitions). The sequential SPD (SSPD) requires that c-blocks be consistently and totally ordered over the class based on the following relation over embeddings: block $b = \langle \mu, y, x \rangle$ *precedes* block $b' = \langle \mu', y', x \rangle$ if $\mu(|y|) + 1 < \mu'(1)$. Let t and t' be two c-blocks over a class x, we say that t *precedes* t' if for all $i \in [x]$, t_i precedes t'_i, and that a pattern p is *sequential* if *precedes* is a total ordering over p.

Definition 1 (SPD, RSPD, SSPD). *Let C be a dataset over an alphabet A, $i \in [C]$, $s \in \mathbb{N}$, and $w \in \mathbb{N}^*$. A solution to a SPD $\langle A, C, i, s, w \rangle$ is an exclusive, coverage-free, s-loose and w-wide pattern for C_i. A solution to a RSPD $\langle A, C, i, s, w \rangle$ is a minimally exclusive pattern of $\prec_w^s$-maximal and non-equivalent c-blocks for C_i. A solution to a SSPD $\langle A, C, i, s, w \rangle$ is a sequential, exclusive, s-loose and w-wide pattern for C_i.*

We propose a lexicographic constraint optimisation model for the RSPD that computes solution patterns of minimum cardinality and, amongst those, of maximum length. The model substitutes maximality and minimality constraints with the requirement that c-blocks be maximal according to the lexicographic ordering prioritizing minimum cardinality over maximum length.

Lemma 1 (RSPD as a COP). *Let $P = \langle A, C, i, s, w \rangle$ be a RSPD, Π the set of patterns for C_i, and $p \in \Pi$. p is a minimum cardinality solution to P if and only if it satisfies the following conditions:*

1. *slack and width bounds: $\sigma(p) \leqslant s$ and $\omega(p) \geqslant w$;*
2. *exclusivity: p is exclusive to C_i wrt. C;*
3. *non-equivalence: for all $t \in p$, $u \in p$ s.t. $t \neq u$, $t \not\cong u$;*
4. *non-coverage: for all $t \in p$, $u \in p$ s.t. $t \neq u$, $\neg(t < u)$;*
5. *maximality for $\prec_{lex}$: for all $q \in \Pi$ s.t. q satisfies (1-4), $\neg(p \prec_{lex} q)$ where $p \prec_{lex} q$ iff $(|q| \geqslant |p| \;\Rightarrow\; (|q| = |p| \;\wedge \lambda(q) > \lambda(p)))$ holds true.*

The above result generalizes to the case where we include slack minimality as the least preferred criterion in the lexicographic objective function. Since RSPDs prohibit replication, the maximum number of solutions to a RSPD is (loosely) bounded by the maximum number of blocks in the smallest protein of the class C_i. The following result formulates the bound in the case of 1-loose patterns.

Lemma 2. *Let $\phi = \frac{1+\sqrt{5}}{2}$, $\psi = \frac{1-\sqrt{5}}{2}$, $n \in \mathbb{N}$ and $\beta(n)$ be the number of 1-loose blocks for a n-long string. $\beta(n) = \frac{\phi^{n+4} - \psi^{n+4}}{\sqrt{5}} + n - 3$ and $\lim_{n\to\infty} \beta(n) = \frac{\phi^{n+4}}{\sqrt{5}}$.*

More generally, the number of blocks of slack k that span a string of length n is the Fibonacci sequence of order k (where each element is the sum of the

previous k elements). The closed-form for the n-th element of the sequence is $[\frac{r^{n-1}(r-1)}{(k+1)r-2k}]$ where [] denotes the nearest integer function and r is the limit of the ratio between successive terms as n increases. r corresponds to the root of equation $x + x^{-k} = 2$ near to 2 and it approaches 2 as k increases.

4 A Two-Step Approach to Solving (R)SPD

As (R)SPD involves finding a pattern consisting of one or more maximal c-blocks for a class C_k, we propose a two-step approach where first we compute all the maximal c-blocks for C_k and then compute an exclusive pattern of minimum cardinality. We describe these two steps briefly.

Computing maximal c-blocks. To compute the set of maximal c-blocks, we first compute all blocks of length 2 that are common to the proteins of class C_k. This is done by first selecting the protein having the minimum size and then verifying for each valid block of length 2 whether it is common to all the proteins or not. Once done, we know all the minimal length blocks and their starting positions in the smallest protein of the class. We then compute all maximal blocks of the protein by composing the minimal length blocks while ensuring that each of them is involved in at least one c-block. Finally, we compute the set of all maximal c-blocks based on the maximal blocks and ensure that none of them covers another.

Computing optimal patterns. Once we have all the maximal c-blocks, we formulate a constraint optimization problem for computing an optimal pattern, i.e., an exclusive pattern of minimum cardinality. Let A denote the set of the maximal c-blocks for class C_k and F denote the set of foreign proteins (i.e., proteins not in C_k). For each protein $i \in F$, we compute the subset of A containing the maximal c-blocks whose signature is not matched by protein i. This set is denoted by $E_i \subseteq A$. For each combination of a protein $i \in F$ and a maximal c-block $j \in E_i$ a Boolean variable x_{ij} is created which denotes whether j is used to exclude i. Another Boolean variable y_j is created that denotes whether the maximal c-block j is part of the pattern or not. For each protein $i \in F$, we want to select at least one maximal c-block whose signature is not matched by i, i.e., $\sum_{j \in E_i} x_{ij} \geqslant 1$. A maximal c-block is selected if it is used to exclude a protein i, i.e., $y_j \geqslant max_{i \in F} x_{ij}$. The objective is to minimize the number of maximal c-blocks that are selected for class C_k, i.e., $\min \sum_{j \in A} y_j$.

Notice that the formulation is equivalent to that of a minimum set covering problem. This model only works for RSPD as it does not use the number of times a c-block signature is matched by a foreign protein to enforce exclusivity. We remark that it is possible to have a multiple c-blocks with same signature. Indeed, it is possible that a c-block signature is matched by a foreign protein but not as many times as there are maximal c-blocks in A sharing this signature. In this case, the model will consider each of these c-blocks as compatible whereas they are incompatible as a whole.

Let s_n be the number of maximal c-blocks associated with a signature s, and let $\{c_{s_1}, \ldots, c_{s_n}\} \subseteq A$ be some permutation of those maximal c-blocks. For solving SPD, we additionally associate each maximal c-block c_{s_j} with a number j, and the constraint that if c_{s_j} is selected then at least j number of c-blocks associated with the same signature must be selected. These constraints are enforced through a set of dependencies.

Our preliminary results using SPD are shown in Table 1 which suggest that the presented approach is indeed scalable for handling large size instances. We investigated with two databases: Late Embryogenesis Abundant Proteins (LEAP) and Small Heat Shock Proteins (sHSP). The number of classes and the total number of proteins in these classes is mentioned in the columns labelled as `nclasses` and `nproteins` respectively. For LEAP 6 out of 12 classes and for sHSP 3 out of 23 classes were unsatisfiable as they do not have any exclusive SPD patterns. The maximum (and the minimum) number of maximal c-blocks, slack and length of an optimal exclusive patterns associated with the classes of each database is shown in the columns labelled as `cardinality`, `slack` and `length` respectively.

Table 1. Results of LEAP and sHSP instances obtained using SPD

name	nclasses	nproteins	#unsat	cardinality	slack	length
LEAP	12	1066	6	5 (1)	4 (0)	13 (4)
sHSP	23	2244	3	10 (1)	25 (0)	20 (4)

5 Conclusion

We have introduced the notion of soft pattern to characterize and discriminate classes of protein sequences. Three pattern discovery problems have been formalized to prevent c-block coverage, replication or overlapping. We have shown that minimal exclusion and maximal refinement are compatible objectives when computing replication-free patterns. The principles of a lexicographic constraint optimization model have been laid out and a two-stage procedure has been sketched. Future work involves carrying out experiments on two existing datasets of unstructured and highly structured proteins [9–12].

References

1. Gusfield, D.: Algorithms on Strings, Trees and Sequences. Computer Science and Computational Biology. Cambridge University Press (2008)
2. Seiler, M. et al.: The 3of5 web application for complex and comprehensive pattern matching in protein sequences. BMC Bioinformatics, pp. 7–144 (2006)

3. Bailey, T.L. et al.: MEME SUITE: tools for motif discovery and searching. Nucleic Acids Research, 37:W202W208 (2006)
4. Uversky, V.N., Dunker, A.K.: Understanding protein nonfolding. Biochim. Biophysi. Acta 1804, pp. 1231–1264 (2010)
5. Grant, C.E., Bailey, T.L., Noble, W.S.: FIMO: Scanning for occurrences of a given motif. Bioinformatics, vol. 27, pp. 1017–1018 (2011)
6. Dinkel et al.: ELM the database of eukaryotic linear motifs. Nucleic Acids Res., vol. 40: D242-D251 (2012)
7. Lesaint, D., Mehta, D., O'Sullivan: Soft Pattern Discovery in Pre-Classified Protein Families through Constraint Optimization. Technical report (2013)
8. Nethercote, N. et al.: MiniZinc: towards a standard CP modelling language, Princ. and Pract. of Constraint Programming (CP'07), pp. 529–543, Springer-Verlag (2007)
9. Hunault, G., Jaspard, E.: The Late Embryogenesis Abundant Proteins DataBase. `http://forge.info.univ-angers.fr/~gh/Leadb/index.php` (2013)
10. Hunault, G., Jaspard, E.: LEAPdb: a database for the late embryogenesis abundant proteins. pp. 11–221, BMC Genomics (2010)
11. Jaspard, E., Macherel, D., Hunault, G.: Computational and statistical analyses of amino acid usage and physico-chemical properties of the twelve late embryogenesis abundant protein classes. PLoS ONE 7:e36968 (2012)
12. Hunault, G., Jaspard, E.: The Small Heat Shock Proteins Database. sHSPdb. `http://forge.info.univ-angers.fr/~gh/Shspdb/index.php` (2013)

Kekulé structure enumeration yields unique SMILES

Martin Mann[1] and Bernhard Thiel[2]

[1]Bioinformatics, Department for Computer Science, University of Freiburg, George-Köhler-Allee 106, 79106 Freiburg, Germany,
[2]Institute for Theoretical Chemistry, University of Vienna, Währingerstrasse 17, 1090 Vienna, Austria
mmann@informatik.uni-freiburg.de

Abstract. A standard representation of molecules is based on graphs where atoms correspond to vertices and covalent bonds are represented by a number of edges according to the bond order. This depiction reaches its limitations for aromatic molecules where the aromatic ring can be encoded by different bond order layouts, i.e. Kekulé structures, since electrons are shared within the ring rather than fixed to a specific bond. Thus, several Kekulé structures are possible for aromatic molecules. Here, we propose a new constraint programming based approach to enumerate all Kekulé structures for a given molecule. Furthermore, the ambiguity information derived is used to enable a unique Kekulé-based SMILES encoding of the molecule independent of any aromaticity detection algorithm. This is of importance, since there is no generally accepted aromaticity definition available that covers all cases.

1 Introduction

Molecules are often depicted as undirected graphs representing atoms as vertices and covalent single, double, or triple bonds by an according number of edges as given in Fig. 1, known as structural formula. This works well as long as it is possible to specifically assign electron pairs shared between two atoms to individual bonds. In that case, a unique graph representation can be given. But the depiction fails as soon as electrons are not uniquely assignable, a phenomenon named mesomerism. A classic example is benzene shown in Fig. 1 a). Two different graph representations, i.e. bond assignments, can be given and these were first identified and introduced by August Kekulé in 1872 [6]. Since that time, such explicit structural formula for molecules with ambivalent rings (different single-double-bond assignments) are refered to as resonance or *Kekulé structures*. In the following, we will focus only on mesomerism of rings within molecules, other forms are discussed later. This ambiguity usually poses no problem for most applications but gets crucial as soon as a unique representation of a molecule is needed, e.g. within chemical compound databases [2, 5] or when atom mappings for reactions are to be identified [9].

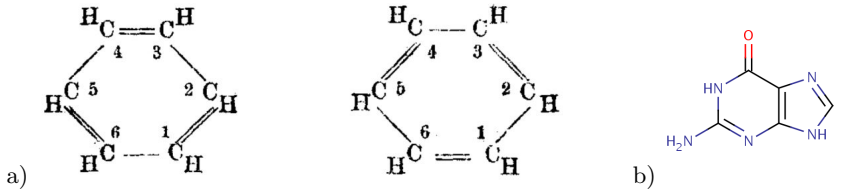


Fig. 1. a) The two isomeric structures of benzene identified by Kekulé (taken from [6]) and b) the Kekulé structure of guanine.

Whether or not a molecule gives rise to several Kekulé structures usually depends on the existence of (hetero) aromatic rings within the molecule. Within aromatic rings, bond electrons are shared within the ring and no unique single-double-bond assignment is possible, resulting in multiple Kekulé representations. The number of Kekulé structures is therefore combinatoric in the number of ambiguous aromatic rings part of the molecule. It was reasoned that the thermodynamic stability of a molecule is to some extent linked to its number of Kekulé structures [15, 3]. Most algorithms to enumerate Kekulé structures are based on graph theoretical studies and a lot of work was done in the early 80s. A thorough review of the early methods is given in [13] on pages 50-52. Therein, most algorithms were tailored to specific hydrocarbone molecule classes usually only covering benzene-like ring layouts and conjugations, e.g. [3, 1].

Within this contribution, we introduce a new constraint programming (CP) based method to enumerate all Kekulé structures for a given molecule. This encodes all possibibly ambiguouos edges and enumerates all valid bond assignments and thus all Kekulé structures. This is of importance for instance to parse molecules given in SMILES format [14] (later discussed in detail) or to provide all Kekulé variants where needed. For instance, we have recently introduced a CP-based approach for the computation of valid atom mappings for chemical reactions [8, 9]. Therein, the reaction's educt and product molecules are mapped onto each other revealing the bond breakings and formations occuring during the reaction. To this end, all Kekulé structures of all participating molecules have to be known and considered, since it is not known in advance, what specific Kekulé structure participates in the reaction. The approach is generic and not tailored to specific classes of molecules. As an input a single structural formula for each molecule has to be provided and all Kekulé structures are enumerated.

Beside the enumeration of Kekulé structures, we use the approach to enable the generation of unique SMILES strings without the need for aromaticity perception. SMILES is a standard format to represent molecules as strings [14]. The string is generated from a treelike-decomposition of the molecule, where ring closures are marked by according number pairs. For instance benzene depicted in Fig. 1 a) can be represented by `[H]C1=C([H])C([H])=C([H])C([H])=C([H])1` when hydrogens are explicitly encoded by `[H]`. Usually, hydrogens deducable from the structure are ommited leaving the SMILES `C1=CC=CC=C1`. Note, this

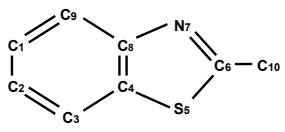
encoding is the same for both benzene Kekulé structures, since a SMILES does not encode any node indexing.

The SMILES language copes with ring ambiguity by a special treatment of aromatic rings. Therein, bonds and atoms part of an aromatic ring are given a special lowercase label marking their ambiguity. It is left to the SMILES parser to pick one of the encoded Kekulé structures, to enumerate them all, etc. The benzene example from Fig. 1 would be encoded by `c1ccccc1` in contrast to the Kekulé structure encoding `C1=CC=CC=C1` discussed above.

The central problem for the standard SMILES approach is the lack of a decent definition of aromaticity that covers all cases of aromatic molecules [11]. Furthermore, aromaticity cannot be simply used interchangeably with mesomerism, i.e. the existence of several Kekulé structures. A simple example is guanine depicted in Fig. 1 b). Both rings of the molecule are usually assumed to be aromatic. Still guanine features only a single Kekulé structure and does not show the usual aromatic ambiguity. It is therefore generally hard to decide whether or not a ring within a molecule is aromatic or not and thus if it is to be treated ambiguous or not, which is central to generate unique SMILES [14]. Within our approach, we use a variant of the presented CP approach to identify all edges that enable ambiguity instead of annotating whole rings. Only these edges are treated differently in the SMILES generation, which results in a slightly different but aromaticity-independent SMILES encoding. The new SMILES encoding is only encoding ambiguity information where needed and results in a general, unique molecule string encoding.

2 Preliminaries

Given a structural formula of a molecule, it can be represented by an undirected graph (V, E) with vertex set V representing the atoms of the molecule and edge set $E = \{ \{v, v'\} \mid v, v' \in V\}$ representing the covalent bonds between these atoms. The bond order, i.e. the number of electron pairs shared within the bond, is given by the input adjacency matrix A where each entry $A_{v,v'} \in \{0, 1, 2, 3\}$ denotes the according bond order between v and v'. An example is given in Fig. 2.



A	1	2	3	4	5	6	7	8	9	10
1		1							2	
2	1		2							
3		2		1						
4			1		1			2		
5				1		1				
6					1		2			1
7						2		1		
8				2			1		1	
9	2							1		
10						1				

Fig. 2. An example molecular graph (without hydrogens) with $V = \{1 \ldots 10\}$ and the according adjacency matry A.

For such a graph (V, E), we identify the subgraph (V°, E°) covering only vertices and edges participating in rings since we want to enumerate Kekulé structures for ring ambiguity. To enumerate all rings, we apply the ring perception algorithm by Hanser [4], which first removes all vertices with degree one and successively decomposes the remaining ring structure into single rings in an iterative fashion. Since triple bonds form very strong and inflexible atom interactions, we ignore all triple bond containing rings. For the example in Fig. 2, the Hanser algorithm identifies the three rings $1-2-3-4-8-9-1$, $4-5-6-7-8-4$, and $1-2-3-4-5-6-7-8-9-1$, resulting in $V^\circ = \{1 \dots 9\}$ (leaving out node 10).

Eventually, each vertex $v \in V^\circ$ participates in at least two edges and all edges $\{v, v'\} \in E^\circ$ are single or double bonds, i.e. $A_{v,v'} \in \{1, 2\}$, that might give rise to different Kekulé structures. All other "non-ring bonds" are assumed to be isomorphic between different Kekulé structures. Therefore, the problem of enumerating all Kekulé structures based on ring ambiguity reduces to the enumeration of all valid single-double bond assignments of the ring bonds in E°.

3 Enumerating all Kekulé structures

As introduced above, given the ring-covering subgraph (V°, E°) of a molecule's structure graph (V, E), it is sufficient to enumerate all valid single-double bond assignments for the bonds in E°. To this end, we formulate a constraint satisfaction problem as follows. For each edge $\{v, v'\} \in E^\circ$, we introduce a variable $X_{v,v'}$ with domain $D_{v,v'} = \{1, 2\}$. For each atom vertex $v \in V^\circ$, we add a linear constraint $\sum_{\{v,v'\} \in E^\circ} X_{v,v'} = \sum_{v' \in V^\circ} A_{v,v'}$, i.e. the sum over all bond orders for each atom has to be preserved by any assignment.

The example from Fig. 2 would result in 10 edge variables, e.g. $X_{1,2}, X_{2,3}, \dots$ and 9 linear constraints, e.g. for vertex 4: $X_{3,4} + X_{4,5} + X_{4,8} = 4$.

Note, while given here in terms of integer domains that were implemented using the Gecode library v4.0 [12], an equivalent CSP can be formulated using Boolean variables and domains. In such a formulation, the boolean encoding would cover whether or not a bond is e.g. a double bond and the applied linear constraints would limit the number of double bonds to $\sum_{v' \in V^\circ} \max(0, A_{v,v'} - 1)$, i.e. the overall ring bond order minus the number of ring bonds.

Given such a CSP for a certain molecule, we can simply apply a standard first-fail depths-first-search to enumerate all valid single-double bond assignments and thus according Kekulé structures. This reveals two Kekulé structures for the discussed example molecule; both are presented in Fig. 3.

We have applied the procedure to molecules from the ChEBI database [2]. From the 15,944 molecules in the database, we derived a subset of 10,920 ring-containing molecules for which full atom information was available (68.5% of the database). For each molecule, we applied the given procedure to enumerate all Kekulé structures. In Tab. 1, we report the resulting statistics where the dataset was further clustered according to the number of rings present in a molecule.

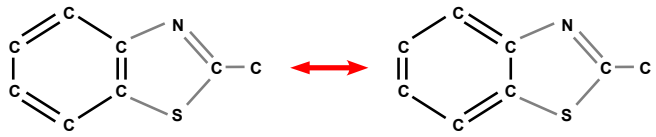


Fig. 3. The two Kekulé structures of the molecule depicted in Fig. 2 whereby both rings are aromatic.

		#ambiguous rings			#overlaps		#Kekulé structures		
#rings	#mols	median	mean	max	median	mean	median	mean	max
1	2871	0	0.4	1	0	0	1	1.4	2
2	2275	1	1	2	0	0.4	2	2.0	4
3	2261	2	1.6	3	1	0.9	2	2.5	8
4	1621	0	1.4	4	2	1.9	1	2.7	16
5	806	1	1.7	5	3	3.0	2	3.1	24
6-10	909	0	2.1	9	4	5.7	1	6.7	288
> 10	177	0	3.8	49	27	122.2	1	7.2	256

Table 1. Statistics on the number of ambivalent rings, the number of shared bonds between rings (#overlaps), and the number of distinct Kekulé structures within the ChEBI data set clustered by the number of rings per molecule.

When investigating the median of the number of ambiguous rings it becomes clear that most rings are non-ambiguous (median ~ 0) while there is on average at least one ring with ambiguity. Their average number only slightly increases with the number of rings a molecule features. The median of the number of bonds shared between rings is given in column #overlaps. Inspecting the numbers, most molecules in our data set seem to sport individual rings instead of ring fusions as e.g. for guanine in Fig. 1 b). Only for larger molecules with multiple rings, fused ring systems become more common.

5,816 molecules (53.3%) show multiple Kekulé structures, highlighting the need for appropriate ambiguity handling and enumeration. For 3,459 molecules, all present rings were ambiguous. Table 2 gives statistics on the number of ambiguous bonds for molecules with multiple Kekulé structures. On average, about half of all ring participating bonds are ambiguous. This is mainly due to ring fusions, where e.g. an ambiguous benzene ring is fused with a non-ambiguous ring. In such a case, all non-shared bonds of the second ring are non-ambiguous resulting in the presented statistics.

When averaging over the whole ChEBI data set, a "mean ring molecule" features about 3 rings where one is ambiguous with about 5 ambiguous bonds, which results in 2-3 Kekulé structures on average. This gives rise to the need for canonicalization to enable unique molecule representations within databases as discussed in the next section.

#rings	#mols ambi/all	#ringBonds mean(ambi) / mean(all)
1	1276/2871	6.0 / 6.0 = 100%
2	1320/2275	6.0 / 11.0 = 54%
3	1506/2261	9.8 / 16.2 = 60%
4	798/1621	11.7 / 21.3 = 55%
5	405/806	12.7 / 25.3 = 50%
6-10	435/909	16.1 / 32.8 = 49%
> 10	76/177	17.2 / 35.4 = 49%

Table 2. Statistics on the number of ambivalent ring bonds (#ringBonds ambi) and the overall number of bonds participating in rings (#ringBonds all) for all molecules with at least two distinct Kekulé structures (first number in column #mols vs. overall number) within the ChEBI data set clustered by the number of rings per molecule.

4 Unique SMILES with ambiguous bond encoding

The previous study on the ambiguity when representing molecules with specific bond assignments highlights the need for a unique canonical molecule representation, e.g. for database lookups etc. As discussed in the introduction, the SMILES encoding was introduced for this purpose with according canonicalization algorithms [14]. Therein, atoms are represented by according standard abbreviations like "C", "H", "Br", etc. (all starting upper case and enclosed in brackets if longer than one character), and bonds formed by more than one electron pair are encoded by the special characters "=" and "#" for double or triple bonds. The ambiguity resulting from aromatic rings was handled using special character encodings for atoms participating in such rings, i.e. using lower case characters as "c", "o", "n", ... for the common aromatic-ring atoms "C", "O", "N", ... respectively, as discussed for benzene in the introduction. Furthermore, an aromatic bond label ":" was introduced, which encodes the uncertainty if the bond is a single or a double bond. This encoding works well for simple standard cases of aromatic compounds. But the central problem is the decision whether or not a ring is aromatic or not, a question still not successfully solved in chemistry [11].

For instance, given the example molecule from Fig. 2. Depending on the aromaticity annotation, there are various possibilities to encode the molecule:

both rings aromatic : `c12ccccc2sc(C)n1`
large ring aromatic : `c12ccccc2SC(C)=N1`
small ring aromatic : `c12C=CC=Cc2sc(C)n1`
Kekulé: left Fig. 3 : `C12C=CC=CC=2SC(C)=N1`
Kekulé: right Fig. 3 : `C12=CC=CC=C2SC(C)=N1`

The SMILES notation thus mixes the problem of defining a unique and compact string represention for molecules with the even harder problem of aromaticity perception. Here, we will try to disentangle the two problems and to provide a

solution for the first, namely the generation of unique canonical SMILES without the need for aromaticity perception.

To this end, we simply fall back to the previous problem of Kekulé structure ambiguity, which poses the true problem for canonicalization. Currently, such ambiguity is intrinsically connected with aromaticity in the SMILES encoding, but there exist many counter-examples as e.g. guanine in Fig. 1 b). In contrast, we want to encode only for variation where it occurs, i.e. the ambiguous bonds within rings.

Given the CSP formulation from above, we only perform a constraint propagation until arc-consistency is reached. *No search* is performed. The bond-encoding variables $X_{v,v'}$ that are still unassigned $|D_{v,v'}| > 1$ encode for bonds $\{v, v'\} \in E$ can either be single or double bond, i.e. ambiguous bonds.

Once this subset of ambiguous bonds is identified, we can apply a variant of the canonical SMILES algorithm from [14], where

1. all atoms are treated non-aromatic (since no aromaticity perception was done),
2. ambiguous ring-bonds (non-assigned variables) are represented by the label ":",
3. non-ambiguous ring-bonds are represented by by single ("-") or double bond label ("=") based on the according variable assignment, and
4. non-ring bonds are labeled according to the initial molecule representation encoded by the adjacency matrix A with "-", "=", or "#" for $A_{v,v'} = 1, 2,$ or 3, respectively.

Given this special treatment, we can derive unique canonical SMILES without aromaticity perception using the standard SMILES canonicalization implementation as e.g. available in the Graph Grammar Library GGL [7].

In Figure 3, only the bonds of the 6-ring given in black are ambiguous leaving 4 of the 5 bonds of the smaller 5-ring (in gray) unambiguous. This results in the new non-aromatic but ambiguity-encoding SMILES representation `C12:C:C:C:C:C:1SC(C)=N2` instead of the SMILES with aromaticity annotation `c12ccccc2sc(C)n1`, which does not easily reveal the two Kekulé structures and the source of ambiguity.

5 Future work

The current approach is restricted to mesomerism of molecular ring systems based on the ambiguity of single-double bond assignments. Still, there are further sources of mesomerism that result in multiple resonance structures. A common form is the shift of unbound (valence) electrons of atoms, that define its charge, to neighbored atoms, which directly results in ambiguity. Another source for different representations of basically the same molecule is a phenomenon called tautomerism, where adjacent hydrogens are shifted to neighbored atoms resulting in a changed bond order pattern of the molecule. Furthermore, combination

of both can occur. Finally, ionizations of some atoms are possible, depending on the physical conditions. Sayle gives a detailed overview of the problem in [10].

Thus, we are planning to extend the described approach to further cases of mesomerism to enable a full enumeration of resonance structures for a given molecule. When applied to the presented ChEBI data set, this might reveal even stronger abundance of ambiguity when representing molecules as structural formula.

6 Conclusions

We have introduced a constraint programming based approach to enumerate the possible Kekulé structures for a molecule that result from ring-mesomerism. The approach was used to assess the abundance of such ambiguity in the ChEBI data base, revealing that half of the data set shows at least two Kekulé structures. Furthermore, it became obvious that this ambiguity results only from a fraction of the involved ring-bonds.

Given that such ambiguity is problematic when deriving unique molecule representations, we have extended the approach to yield canonical SMILES without need for aromaticity perception. The latter was the base for the standard SMILES approach to identify and deal with ambiguity. Since aromaticity is hard to define, the fact that not all aromatic rings are ambiguous, and given our statistics on ambiguous bonds, we proposed an approach that is independent of aromaticity assignment. To this end, we identify ambiguous bonds using our CP approach. Only these bonds are treated special during the standard canonical SMILES generation. Thus, we derive unique graph-based molecule representations.

References

1. John L. Bergan, Sven J. Cyvin, and Bjorg N. Cyvin. Number of kekulé structures of single-chain corona-condensed benzenoids (cycloarenes). *Chemical Physics Letters*, 125(3):218–220, 1986.
2. Kirill Degtyarenko, Paula de Matos, Marcus Ennis, Janna Hastings, Martin Zbinden, Alan McNaught, Rafael Alcántara, Michael Darsow, Mickaël Guedj, and Michael Ashburner. ChEBI: a database and ontology for chemical entities of biological interest. *Nucleic Acids Research*, 36(suppl 1):D344–D350, 2008.
3. B. Džonova-Jerman-Blažič and N. Trinajsticí. Computer-aided enumeration and generation of the Kekulé structures in conjugated hydrocarbons. *Computers Chemistry*, 6(3):121–132, 1982.
4. T. Hanser, P. Jauffret, and G. Kaufmann. A new algorithm for exhaustive ring perception in a molecular graph. *J. Chem. Inf. Comp. Sci.*, 36(6):1146–1152, 1996.
5. M. Kanehisa, S. Goto, Y. Sato, M. Furumichi, and M. Tanabe. KEGG for integration and interpretation of large-scale molecular data sets. *Nuc. Acids Res.*, 40(Database issue):D109–14, 2012.
6. August Kekulé. Ueber einige Condensationsproducte des Aldehyds. *Justus Liebigs Annalen der Chemie*, 162(1):77–124, 1872.

7. M. Mann, H. Ekker, and C. Flamm. The graph grammar library - a generic framework for chemical graph rewrite systems. In Keith Duddy and Gerti Kappel, editors, *Theory and Practice of Model Transformations, Proc. of ICMT 2013*, volume 7909 of *LNCS*, pages 52–53. Springer, 2013. Extended abstract at ICMT, long version at arXiv http://arxiv.org/abs/1304.1356.
8. M. Mann, H. Ekker, P.F. Stadler, and C. Flamm. Atom mapping with constraint programming. In R. Backofen and S. Will, editors, *Proceedings of the Workshop on Constraint Based Methods for Bioinformatics WCB12*, pages 23–29, Freiburg, 2012. Uni Freiburg. http://www.bioinf.uni-freiburg.de/Events/WCB12/proceedings.pdf.
9. M. Mann, F. Nahar, H. Ekker, P.F. Stadler, and C. Flamm. Atom mapping with constraint programming. In *Proceedings of the 19th International Conference on Principles and Practice of Constraint Programming, CP'13*, LNCS. Springer, 2013. Accepted for publication.
10. Roger A. Sayle. So you think you understand tautomerism? *Journal of Computer-Aided Molecular Design*, 24(6-7):485–496, 2010.
11. Amnon Stanger. What is... aromaticity: a critique of the concept of aromaticity-can it really be defined? *Chem. Commun.*, 0:1939–1947, 2009.
12. Gecode Team. Gecode: Generic constraint development environment, 2013. Available as an open-source library from http://www.gecode.org.
13. N. Trinajstić. *Chemical Graph Theory*, volume 1. CRC Press, 1983.
14. D. Weininger. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *J. Chem. Inf. Comp. Sci.*, 28(1):31–36, 1988.
15. G. W. Wheland. The number of canonical structures of each degree of excitation for an unsaturated or aromatic hydrocarbon. *J. Chem. Phys.*, 3(6):356–361, 1935.

Solving Subgraph Epimorphism Problems using CLP and SAT

Steven Gay, François Fages, Francesco Santini, Sylvain Soliman

INRIA Paris-Rocquencourt

Abstract. In this work, we compare CLP and SAT solvers on the NP-complete problem of deciding the existence of a subgraph epimorphism between two graphs. Our interest in this variant of graph matching problem 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. In this setting, model reduction can be formalized as the existence of a sequence of vertex, species or reaction, deletion and merge operations which transforms a first reaction graph into a second graph. This problem is in turn equivalent to the existence of a subgraph (corresponding to delete operations) epimorphism (i.e. surjective homomorphism, corresponding to merge operations) from the first graph to the second. We show how subgraph epimorphism problems can be modeled as Boolean constraint satisfaction problems, and we compare CLP and SAT solvers on a large benchmark of reaction graphs from systems biology.

1 Subgraph Epimorphisms

Subgraph epimorphisms (SEPI) can be seen as a variant of subgraph isomorphism (SISO). Our interest in this particular graph relation comes from reaction graphs in systems biology:

Definition 1 (Graph). *A* graph G *is a pair* $G = (V, A)$*, where* $A \subseteq V \times V$*.*

Definition 2 (Reaction Graph). *A* reaction graph G *is a triple* $G = (V, A, t)$*, where* $t : N \longrightarrow \{s, r\}$ *labels the type of nodes:* $S = t^{-1}(s)$ *is the set of* species *nodes,* $R = t^{-1}(r)$ *is the set of* reaction *nodes, and* $A \subseteq S \times R \cup R \times S$*.*

Example 1. The reaction graph on the left of Fig. 1 expresses an enzymatic mechanism, usually noted $E + M \leftrightharpoons F \rightarrow E + P$.

The species are represented here by ellipse nodes: $S = \{E, M, F, P\}$. The reactions the rectangle nodes: $R = \{c, d, p\}$. The arcs are $A = \{(M, c), (E, c), (c, F), (d, M), (d, E), (F, d), (p, P), (p, E), (F, p)\}$.

When reactions are equipped with kinetics and species with concentrations, it yields a reaction model, which can be simulated. Then simulation can be compared to real-life data, and the model can be modified so that the simulation fits the data, which is the final goal.

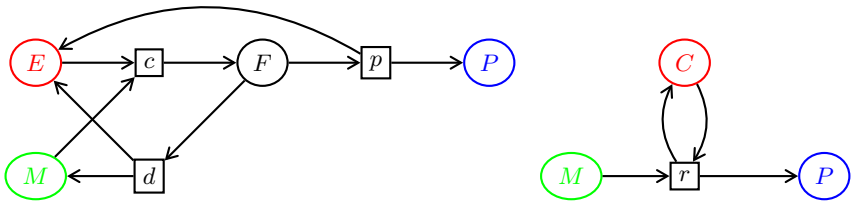


Fig. 1. On the left, an enzymatic mechanism. On the right, the Michaelis-Menten reduced version.

Modelers are interested in having the simplest model that behaves as the real-life data, so they apply mathematical reductions to their models.

These reductions induce transformations on the underlying reaction graph, which can be captured using graph operations:

Example 2. Applying a Michaelis-Menten reduction to the reaction mechanism on the left in Fig. 1 yields the reaction graph on the right, usually written $M + C \to P + C$:

Definition 3 (Delete, Merge). *Let $u, v \in V$. The graph $d_v(G)$ is defined as (V', A'), where $V' = V \setminus \{v\}$ and $A' = A \cap (V' \times V')$.*

The graph $m_{u,v}(G)$ is defined as (V', A'), where $V' = V \setminus \{u, v\} \uplus \{uv\}$, $A' = \{(s_{u,v}(x), s_{u,v}(y)) \mid (x, y) \in A\}$, uv is a fresh symbol, and $s_{u,v} : [u \longrightarrow uv, v \longrightarrow uv, x \notin \{u, v\} \longrightarrow x]$.

In the case of reaction graphs, vertices can be merged only if they are both species or both reactions: a reaction can not be merged with a species.

Example 3. In Fig. 1, take the graph on the left. Delete d and F, then merge c with p. The resulting graph is isomorphic to the graph on the right.

We write $G \to^*_{md} G'$ when a string of delete and/or merge operations from G yields G'. As strings of delete operations correspond to SISO, strings of delete/merge operations correspond to SEPI:

Definition 4 (Subgraph Epimorphism). *A subgraph epimorphism from G to G' is a function $f : V \to V'$ such that $\forall (u, v)$ s.t. $f(u)$ and $f(v)$ defined, $(u, v) \in A \Rightarrow (f(u), f(v)) \in A'$, f surjective (onto) on V' and A'.*

Theorem 1 *There exists a subgraph epimorphism from G to G' iff $G \to^*_{md} G'$.*

Example 4. The SEPI corresponding to the example 3 is $m : [M \longrightarrow M, E \longrightarrow C, P \longrightarrow P, c \longrightarrow r, p \longrightarrow r]$.

Deciding SISO is NP-complete, this is also the case for SEPI:

Theorem 2 ([6]) *Deciding the subgraph epimorphism problem is NP-complete.*

This results justifies using approaches such as Constraint Programming and SAT solving to solve SEPI problems.

2 Constraint Program

In this section, we describe how to decide the existence of a SEPI between two graphs using Constraint Programming (CP).

To differentiate mathematical variables and CP variables, we write CP variables in bold font (as in $\mathbf{X}$ opposed to X) ; $[a, b, c]$ denotes the list of the three elements a, b, c ; π_1 and π_2 are the first and second projection functions.

Let G and G' be two graphs, with $G = (V, A)$, $G' = (V', A')$, and $V = \{v_1 \dots v_n\}$, $A = \{a_1 \dots a_k\}$, $V' = \{v'_1 \dots v'_{n'}\}$, $A' = \{a'_1 \dots a'_{k'}\}$, $A'_\perp = A' \uplus \{(x, y) \in (V' \cup \{\perp\})^2 | x = \perp \vee y = \perp\} = \{a'_1 \dots a'_{k'}, a'_{k'+1} \dots a'_{k'_\perp}\}$.

2.1 CP Model

The existence of a SEPI from G to G' can be modeled using CP as follows.

Variables are associated with the vertices and edges of G and G' :

- Morphism variables
  - $\mathbf{X}_v$ for $v \in V$, with $D(\mathbf{X}_v) = V' \cup \{\perp\}$.
  - $\mathbf{A}_a$ for $a \in A$, with $D(\mathbf{A}_a) = \{1, \dots, |A'_\perp|\}$.
- Antecedent variables
  - $\mathbf{X'}_{v'}$ for $v' \in V'$, with $D(\mathbf{X'}_{v'}) = V$.
  - $\mathbf{A'}_{a'}$ for $a' \in A'$, with $D(\mathbf{A'}_{a'}) = A$.

Constraints to enforce the role of morphism and antecedent variables:

I. Morphism constraints
   i. $\forall a \in A, \texttt{element}(\mathbf{A}_a, [\pi_1(a'_1) \dots \pi_1(a'_{k'_\perp})], \mathbf{X}_{\pi_1(a)})$
   ii. $\forall a \in A, \texttt{element}(\mathbf{A}_a, [\pi_2(a'_1) \dots \pi_2(a'_{k'_\perp})], \mathbf{X}_{\pi_2(a)})$

II. Minimal antecedent constraints
   i. $\forall v \in V, \forall v' \in V', \mathbf{X'}_{v'} = v \Rightarrow \mathbf{X}_v = v'$
   ii. $\forall v \in V, \forall v' \in V', \mathbf{X}_v = v' \Rightarrow \mathbf{X'}_{v'} \leq v$
   iii. $\forall a \in A, \forall a' \in A', \mathbf{A'}_{a'} = a \Rightarrow \mathbf{A}_a = a'$
   iv. $\forall a \in A, \forall a' \in A', \mathbf{A}_a = a' \Rightarrow \mathbf{A'}_{a'} \leq a$

III. Global surjection constraints
   i. $\texttt{gsurjection}([\mathbf{X}_{v_1} \dots \mathbf{X}_{v_n}], V')$
   ii. $\texttt{gsurjection}([\mathbf{A}_{a_1} \dots \mathbf{A}_{a_k}], A')$

This model uses reified constraints and the usual `element` constraint.

It also uses a global constraint `gsurjection` that works as follows. Let $\mathbf{D} = [\mathbf{D}_1 \dots \mathbf{D}_d]$ be a list of variables and $T = [T_1 \dots T_t]$ of sorted list of integers. Let $\text{covered}(\mathbf{D}, T) = \{T_i \mid \exists j, \text{dom}(\mathbf{D}_j) = \{T_i\}\}$ be the elements of T that are taken by some variable, and $\text{committed}(\mathbf{D}, T) = \{\mathbf{D}_j \mid \text{dom}(\mathbf{D}_j) \subseteq \text{covered}(\mathbf{D}, T)\}$ be the variables which can not cover any uncovered variable.

Then $\texttt{gsurjection}(\mathbf{D}, T)$ enforces $|T| - |\text{covered}(T)| \leq |\mathbf{D}| - |\text{committed}(\mathbf{D})|$. When all $\mathbf{D}_j$ are ground, $\mathbf{D}$ has to be a surjection on the elements of T. Implementing a linear time propagator for this constraint is straightforward.

While constraints IIi and IIiii introduce dual variables for surjectivity, constraints IIii and IIiv break representation symmetries by choosing minimal antecedents. These constraints are redundant with `gsurjection`.

Proposition 3 *The CP model $\mathcal{P}$ associated with graphs G, G' has a solution if and only if there exists a subgraph epimorphism from G to G'.*

In order to specialize the CP model to reaction graphs, the domains of reaction (species) node variables can be restricted to reaction (species) nodes.

2.2 Search Strategy

We tried different search strategies, and the best we found is to enumerate first the ${\mathbf{A}'}_{a'}$, then the ${\mathbf{X}'}_{x'}$, and finally the morphism variables.

The following proposition sheds some light on this choice:

Proposition 1. *The SEPI CP model above yields a solution iff variables $({\mathbf{X}'}_{v'})_{v' \in V'}$ and $({\mathbf{A}'}_{a'})_{a' \in A'}$ can be successfully instantiated.*

Proof. Obviously, if enumerating antecedent variables fails, there is no SEPI from the source graph to the target graph.

Conversely, if the enumeration on antecedent variables succeeded, then the corresponding $(\mathbf{X}_v)_{v \in V}$ and $(\mathbf{A}_a)_{a \in A}$ have singleton domains, thanks to domain-arc-consistency of `element` constraints II. The induced subgraph formed by the source vertices and arcs that correspond to these variables are sufficient to cover G', and the morphism constraints I ensure that the variables code a morphism. Giving the $\perp$ value for every remaining morphism variable yields a SEPI from the source graph to the target graph. □

Therefore enumerating morphism variables last ensures there will be no backtracking on these variables: this could explain the good relative performance of this strategy.

3 SAT model

Coding problems into SAT instances and using a SAT solver to find whether it is satisfiable or not is another successful approach to solve NP-complete problems.

A SAT instance can be described as a pair $(X,\ C)$, where X is a set of variables, and C is a set of clauses $c_1 \dots c_r$ with $c_i = \bigvee l_{i,j}$, and finally $l_{i,j}$ is either x or $\bar{x}$, with $x \in X$. A SAT instance can be described more shortly as a boolean formula in conjunctive normal form.

In this section, we will describe, for a given SEPI problem (G, G'), an encoding of the problem as a CNF. This encoding has been implemented, and the evaluation will be made in the next section.

The boolean formulae given in this section are transformed into a CNF using an obvious normalization procedure : implications $a \to (b \wedge c)$ are broken into $a \to b$ and $a \to c$, implication $a \to b$ is coded in $\neg a \vee b$; no further transformations are done. We write clause(f), where f is a boolean function, to denote the clauses passed to the SAT solver.

We split the description of the coding into two main parts: first how to code a partial surjective function, then adding graph constraints to code a subgraph epimorphism.

3.1 Partial Surjective Function Coding

A SEPI m from G to G' is also a partial surjective function from V to V'.

Definition 5 (Partial Surjective Function). *A binary relation $m \subseteq E \times E'$ is a* partial surjective function *if the following conditions are fulfilled:*

- $\forall x \in E, x'_1 \in E', x'_2 \in E', ((x, x'_1) \in m \wedge (x, x'_2) \in m) \Rightarrow x'_1 = x'_2$
- $\forall x' \in E', \exists x \in E, (x, x') \in m$

The elements $x \in E$ do *not* have to be covered by some $(x, x') \in m$, hence the qualifier *partial*; we write $m(x) = x'$ when $x \in E$ is covered by x', $m(x) = \perp$ when x is not covered.

Variables. m is encoded as a binary relation on $V \times (V' \cup \{\perp\})$. The elements of $V' \cup \{\perp\}$ are put in a total order $v'_0 = \perp < v'_1 < \ldots < v'_{n'}$.
- $\forall (v, v') \in V \times (V' \cup \{\perp\}), \mathbf{m}_{v,v'} = 1$ iff $m(v) = v'$.
- $\forall (v, v') \in V \times (V' \cup \{\perp\}), \mathbf{m}^{<}_{v,v'} = 1$ iff $m(v) < v'$.

Clauses. The following clauses enforce the mathematical description of the variables:

I. Left Totality. $\forall v \in V$, clause$(\bigvee_{v' \in V' \cup \{\perp\}} \mathbf{m}_{v,v'})$
II. Functionality. $\forall (v, v'_j) \in V \times (V' \cup \{\perp\})$,
  i. clause$(\mathbf{m}_{v,v'_j} \Rightarrow \mathbf{m}^{<}_{(v,v'_{j+1})})$
  ii. clause$(\mathbf{m}^{<}_{v,v'_j} \Rightarrow \mathbf{m}^{<}_{(v,v'_{j+1})})$
  iii. clause$(\mathbf{m}^{<}_{v,v'_j} \Rightarrow \neg \mathbf{m}_{(v,v'_j)})$
III. Right Totality. $\forall v' \in V'$, clause$(\bigvee_{v \in V} \mathbf{m}_{v,v'})$

The encoding is self-explanatory, except for functionality. Functionality could be encoded directly as well, with something like:
$\forall (v, v'_1, v'_2) \in V \times (V' \cup \{\perp\})^2$ with $v'_1 \neq v'_2$, clause$(\neg(\mathbf{m}_{v,v'_1} \wedge \mathbf{m}_{v,v'_2}))$
This encoding has $|V| \cdot |V' \cup \{\perp\}|^2$ clauses, which is a problem in practice. The coding above, achieved by using the order on $|V' \cup \{\perp\}|$ to force the image of $v \in V$ to be minimal, only has $O(|V| \cdot |V' \cup \{\perp\}|)$ clauses.

3.2 Subgraph Epimorphism Coding

Let us build on the previous part to constrain the function to represent a SEPI.
Variables. Additional variables are used to constrain SEPI:
- Non deleted arcs. $\forall (a, a') \in A \times A', \mathbf{m}_{a,a'}$ iff $m(a) = a'$.
- Deleted arcs. $\forall a \in A, \mathbf{is\_dummy}(\mathbf{m}_a) = 1$ iff $m(a) = \perp$

Clauses.
- Left Totality on Arcs. $\forall a \in A$, clause$(\mathbf{is_dummy}(\mathbf{m}_a) \vee \bigvee_{a' \in A'} \mathbf{m}_{a,a'})$
- Right Totality on Arcs. $\forall a' \in A'$, clause$(\bigvee_{a \in A} \mathbf{m}_{a,a'})$
- Graph Morphism. $\forall ((u, v), (u', v')) \in A \times A'$,

i. clause$(\mathbf{m}_{(u,v),(u',v')} \Rightarrow \mathbf{m}_{u,u'})$
ii. clause$(\mathbf{m}_{(u,v),(u',v')} \Rightarrow \mathbf{m}_{v,v'})$

iii. clause$((\mathbf{m}_{v,v'} \wedge \mathbf{m}_{v,v'}) \Rightarrow \mathbf{m}_{(u,v),(u',v')})$

- Subgraph Morphism. $\forall (u, v) \in A$,

i. clause$(\mathbf{is_dummy}(\mathbf{m}_{(u,v)}) \Rightarrow \mathbf{m}_{u,\perp} \vee \mathbf{m}_{v,\perp})$
ii. clause$(\mathbf{m}_{u,\perp} \Rightarrow \mathbf{is_dummy}(\mathbf{m}_{(u,v)}))$
iii. clause$(\mathbf{m}_{v,\perp} \Rightarrow \mathbf{is_dummy}(\mathbf{m}_{(u,v)}))$

Once again the encoding follows the definition closely. The model can be specialized to reaction graphs by restricting domains, i.e. by setting $\mathbf{m}_{v,v'}$ to false when v and v' are not of the same type.

3.3 Surjectivity and Sorting Networks

The `gsurjection` propagation idea can be imitated with boolean clauses. This improves performance a little. The idea is to introduce minimal antecedents, and then use cardinality networks to force the number of minimal antecedents to be greater than the number of targets.

Cardinality networks use boolean sorting networks to have some consistency using only unit clause propagation.

For a full exposition, [5] compares different approaches to coding integers in boolean clauses, [2] uses such networks on decompositions of cardinality-related constraints, [3] shows that Parberry's odd-even networks behave better than Batcher's merge networks for this purposes, [8] efficiently solves MAXSAT by coding the cardinality part in boolean clauses.

4 Performance Evaluation

We implemented the CLP model using GNU Prolog [4] 1.4.4, and the SAT model is solved with Glucose [1] 2.2.

To test and compare GNU Prolog and Glucose performance on subgraph epimorphism problems, some of the System Biology models in the BioModels repository [9] have been used; in particular, the same models adopted in [7]. A thematic clustering has been accomplished, using information available from the notes of SBML models. The four most populated classes are: *i)* mitogen-activated protein kinase (abbreviated as *mapk*, 11 models), *ii)* circadian clock (*circ*, 11 models), *iii)* calcium oscillations (*caoscill*, 11 models), and *iv)* cell cycle (*ccycle*, 9 models).

In the experiments reported in Tab. 1, the computation time was limited with a timeout of 20 minutes. Performance has been evaluated on an Intel Core 2 Duo 2.4Ghz processor. The four macro-columns respectively show the number of intra-class comparisons, the number of relations found between models (i.e., of reductions), and the number of no-relations found, and, finally, the number of no-results (where timeout occurs). Each sub-column respectively reports performance for Glucose, GNU Prolog, and the methods combined together, using the same timeout for both (20min + 20min).

Clearly, in order to evaluate the two methods, the interesting value is the number of timeouts: here the efficacy of Glucose can be appreciated, in particular

Table 1. Solvers performance collected in 20min.

Class(Files)	Relations			Nonrelations			Timeouts		
	GNU	Glucose	Union	GNU	Glucose	Union	GNU	Glucose	Union
mapk (110)	38	38	42	60	63	63	12	9	5
circ (110)	17	37	37	60	73	73	33	0	0
caoscill (110)	38	38	38	72	72	72	0	0	0
ccycle (72)	9	12	12	43	51	51	20	9	9

Table 2. Solvers performance collected in 10s.

Class(Files)	Relations			Nonrelations			Timeouts		
	GNU	Glucose	Union	GNU	Glucose	Union	GNU	Glucose	Union
mapk (110)	36	35	41	59	60	60	15	15	9
circ (110)	15	33	33	59	68	68	36	9	9
caoscill (110)	38	38	38	72	72	72	0	0	0
ccycle (72)	9	6	10	42	49	49	21	17	13

on class *circ* (from 33 no results to nil), but also on class *ccycle* (from 20 no results to 9), and, lastly, a marginal improvement on class *mapk* (from 12 to 9). Class *caoscill* shows no improvement because it is very easy to match even with GNU Prolog (0 no results). These results have been further investigated in detail, discovering that *mapk* is the only class among the four where the GNU Prolog set of relations is not a subset of the one found with Glucose. This difference set (equivalent to $49 \rightarrow 9$, $49 \rightarrow 11$, $49 \rightarrow 28$, $49 \rightarrow 30$) also corresponds exactly to the difference set of no-results between glucose and GNU Prolog. From this the reader can deduce that from a merging of GNU Prolog and SAT implementations, only 4 no-results less on *mapk* can be gained (which would correspond to 4 relations more). Nevertheless, it also possible to deduce that on some models our GNU Prolog version can run more efficiently: in this case, model 49 is better reduced with GNU Prolog than with SAT.

The lists of comparisons for which no result can be obtained with either SAT or GNU are respectively, $\{49 \rightarrow 146, 146 \rightarrow 9, 146 \rightarrow 11, 146 \rightarrow 28, 146 \rightarrow 30\}$ on *mapk*, and $\{56 \rightarrow 7, 56 \rightarrow 111, 56 \rightarrow 144, 109 \rightarrow 7, 109 \rightarrow 111, 109 \rightarrow 144, 144 \rightarrow 111, 144 \rightarrow 169, 144 \rightarrow 196\}$ on *ccycle*. Few of the models seem to represent a bottleneck, due to the high frequency of the same models in these two lists.

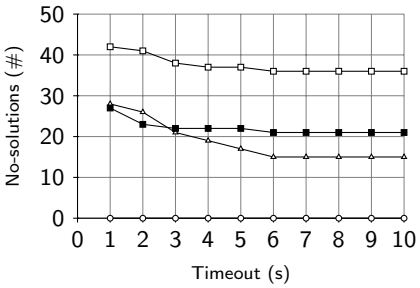


Fig. 2. No-solutions in GNU Prolog.

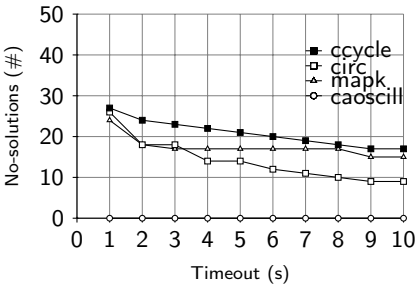


Fig. 3. No-solutions in Glucose.

Moreover, Tab. 2 shows that both implementations also perform well within a short timeout of 10 seconds. This is particularly true with our GNU Prolog implementation (only 7 no-results less over the four classes, from 10sec to 20min), while more debatable with Glucose (23 no-results less in total). Fig. 2 and 3 show how the number of no-solutions decreases by increasing the timeout from 1 up to 10 seconds (GNU Prolog and Glucose respectively).

5 Conclusion

We have compared CLP and SAT approaches for deciding the existence of a subgraph epimorphism from one graph to another.

The main application is to determine the feasibility of computing model reduction hierarchies of real-world model repositories. Section 4 has shown the efficiency of both methods, especially for SAT. In particular, with long timeouts (e.g., 20 minutes), the results found by solving our CNF model with Glucose almost totally subsumes those found by solving out CP model with GNU Prolog.

However, our CLP model can be improved by adding global constraints such as *alldifferent* on antecedents. In addition, we will refine the notion of reduction by adding labels on nodes (e.g., on molecular species), in order to only match related entities; this will also lead to a performance improvement. Moreover, we are currently working on the notion of maximal common subgraph for SEPI, still using both a CLP and a SAT model. This can be used to measure a distance between two biological models.

References

1. Gilles Audemard and Laurent Simon. Predicting learnt clauses quality in modern sat solvers. In *IJCAI*, volume 9, pages 399–404, 2009.
2. Christian Bessiere, George Katsirelos, Nina Narodytska, Claude-Guy Quimper, and Toby Walsh. Decompositions of all different, global cardinality and related constraints. In Craig Boutilier, editor, *Proceedings of the 21st International Joint Conference on Artificial Intelligence, IJCAI'09*, pages 419–424, 2009.
3. Michael Codish and Moshe Zazon-Ivry. Pairwise cardinality networks. In *Logic for Programming, Artificial Intelligence, and Reasoning*, pages 154–172. Springer, 2010.
4. Daniel Diaz, Salvador Abreu, and Philippe Codognet. On the implementation of GNU Prolog. *Theory and Practice of Logic Programming*, 12(1-2):253–282, 2012.
5. Niklas Eén and Niklas Sörensson. Translating pseudo-boolean constraints into sat. *Journal on Satisfiability, Boolean Modeling and Computation*, 2(1-4):1–26, 2006.
6. Steven Gay, François Fages, Thierry Martinez, Sylvain Soliman, and Christine Solnon. On the subgraph epimorphism problem. *Discrete Applied Mathematics*, 2013. to appear.
7. Steven Gay, Sylvain Soliman, and François Fages. A graphical method for reducing and relating models in systems biology. *Bioinformatics*, 26(18):i575–i581, 2010. special issue ECCB'10.
8. Miyuki Koshimura, Tong Zhang, Hiroshi Fujita, and Ryuzo Hasegawa. Qmaxsat: A partial max-sat solver system description. *Journal on Satisfiability, Boolean Modeling and Computation*, 8:95–100, 2012.
9. Nicolas le Novère et al. BioModels Database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic Acid Research*, 1(34):D689–D691, January 2006.

Constrained Flux Coupling Analysis

Laszlo David and Alexander Bockmayr

FB Mathematik und Informatik, Freie Universität Berlin
DFG Research Center Matheon
Arnimallee 6, D-14195 Berlin, Germany

Abstract. Constraint-based modeling has become a widely used approach for the analysis of genome-scale metabolic networks. It includes a variety of methods such as flux balance analysis (FBA), flux variability analysis (FVA), or the computation of elementary flux modes (EFM). Flux coupling analysis (FCA) identifies dependencies between the activity of reaction fluxes in a metabolic network at steady-state. It can be used for exploring a large range of biological questions such as network evolution, gene essentiality, or gene regulation. In this paper, we generalize the concept of FCA by allowing additional linear constraints on the reactions. We show that these constraints can be modeled as lower and upper bounds on the reactions of an alternative metabolic network. We introduce constrained flux coupling analysis (CFCA) and prove that, in many cases, knowing tight bounds of the reactions uniquely determines the coupling relationships. Finally, we present a new method to perform CFCA efficiently.

Keywords: constraint-based modeling, metabolic networks, flux coupling analysis, linear programming

1 Introduction

Constraint-based modeling has become a widely used approach for the analysis of genome-scale metabolic networks [1, 2]. It includes a variety of methods such as flux balance analysis (FBA), flux variability analysis (FVA), or the computation of elementary flux modes (EFM). Flux coupling analysis (FCA) [3] identifies dependencies between the activity of reaction fluxes in a metabolic network at steady-state. It can be used for exploring a large range of biological questions such as network evolution, gene essentiality, or gene regulation.

Given a metabolic network with m metabolites and n reactions, classical FCA works with the steady-state flux cone $C = \{v \in \mathbb{R}^n \mid Sv = 0, v_{Irr} \geq 0\}$. Here $S \in \mathbb{R}^{m \times n}$ is the stoichiometric matrix, $Irr \subseteq \{1, \ldots, n\}$ the set of irreversible reactions, and $v \in \mathbb{R}^n$ denotes a (steady-state) flux vector. While this has found many applications, it turns out that in some cases the concept of flux cone is too general, describes too many metabolic behaviours, and lacks certain desired details. With standard FCA, we find a coupling relationship between two reactions only if these reactions are coupled to each other in all flux vectors belonging to the cone. Often however, biologists are interested in a

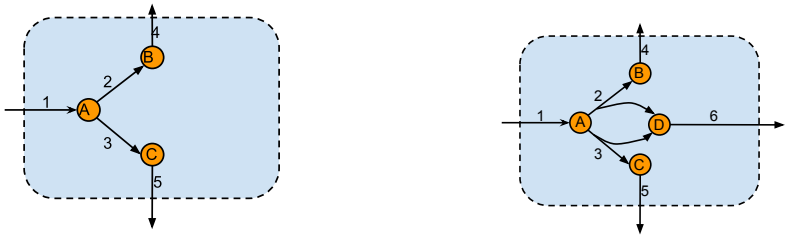


Fig. 1. The intuitive interpretation of additional constraints

particular subspace of the flux cone (e.g. the optimal flux space [4]) or would like to impose additional constraints on the reactions.

In this paper, we generalize the concept of FCA by allowing additional linear constraints on the reactions. We show that these constraints can be modeled as lower and upper bounds on the reactions of an alternative metabolic network. Based on this, we introduce *constrained flux coupling analysis* (CFCA) and prove that, in many cases, knowing tight bounds of the reactions uniquely determines the coupling relationships. Finally, we present a new method to perform CFCA efficiently.

2 Methods

In constrained FCA, we would like to impose additional linear constraints on the reactions, which are of the form $\alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_n v_n \geq \beta$, with $\alpha_1, \ldots, \alpha_n \in \mathbb{R}$ not all zero, and $\beta \in \mathbb{R}$. For algorithmic reasons, we will realize such a constraint by considering an alternative metabolic network. It is derived from the original one by introducing an artificial metabolite $m+1$ and an artificial reaction $n+1$. Every reaction $i \in \{1, \ldots, n\}$ for which $\alpha_i > 0$ (resp. $\alpha_i < 0$) will additionally produce (resp. consume) the metabolite $m+1$ with stoichiometric coefficient α_i. The reaction $n+1$ will be an exchange reaction that takes the metabolite $m+1$ out of the system. The mass-balance equation for metabolite $m+1$ is thus given by $\alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_n v_n - v_{n+1} = 0$. The additional linear constraint is now realized by imposing the lower bound $v_{n+1} \geq \beta$ in the alternative network. The concept is illustrated in Fig. 1. Assume we start with the metabolic network on the left and want to impose the additional constraint $v_2 + v_3 \leq 10$. Considering the network on the right and imposing $v_6 \leq 10$ will give the desired result. A similar transformation can also be done for equality constraints, by imposing β both as a lower and upper bound of v_{n+1}. We conclude that additional linear constraints may be realized by imposing lower and upper bounds on suitably defined reactions in an extended network.

2.1 Definitions

For a metabolic network with n reactions (possibly after reconfiguration), let L (resp. U) $\subseteq \{1, \ldots, n\}$ be the set of reactions i for which a finite lower bound $l_i \in \mathbb{R}$ (resp. upper bound $u_i \in \mathbb{R}$) has been defined. We will consider all flux vectors in the *(steady-state) flux polyhedron* P, defined by

$$P := \{v \in \mathbb{R}^n \mid Sv = 0,\ v_i \geq l_i,\ \text{for all } i \in L,\ v_i \leq u_j,\ \text{for all } j \in U\}. \quad (1)$$

Note that the standard thermodynamic constraints $v_{Irr} \geq 0$ have not been included here, because irreversible reactions can be defined by having the lower bound 0. Given the flux polyhedron P, we can construct an associated flux cone by defining the irreversible reactions. Let $Irr_P := \{i \in L \mid l_i \geq 0\}$. The *flux cone* C_P *associated with* P is defined as

$$C_P := \{v \in \mathbb{R}^n \mid Sv = 0,\ v_i \geq 0 \text{ for all } i \in Irr_P\}. \quad (2)$$

In the special case $U = \emptyset$ and $l_i = 0$ for all $i \in L$, the flux polyhedron P is equal to the flux cone C (with $Irr = L$). However, in general $P \subseteq C_P$ holds.

Blocked reactions are defined similarly to classical FCA [3]. Intuitively, these are the reactions that never participate in a steady-state flux vector.

Definition 1 (Blocked reaction) *Given the flux polyhedron P, a reaction $i \in \{1, \ldots, n\}$ is blocked in P if $v_i = 0$, for all $v \in P$. Otherwise, i is unblocked in P.*

Next we define the constrained coupling relations.

Definition 2 (Constrained coupling relations) *Given the flux polyhedron P, let i, j be two unblocked reactions in P. The constrained (un-)coupling relationships $\underset{c}{\longrightarrow}, \underset{c}{\longleftrightarrow}, \underset{c}{\Longleftrightarrow}$ and $\underset{c}{\not\longrightarrow}$ are defined by:*

- $i \underset{c}{\longrightarrow} j$ *(equiv. $j \underset{c}{\longleftarrow} i$) if for all $v \in P$, $v_i \neq 0$ implies $v_j \neq 0$.*
- $i \underset{c}{\longleftrightarrow} j$ *if for all $v \in P$, $v_i \neq 0$ is equivalent to $v_j \neq 0$.*
- $i \underset{c}{\Longleftrightarrow} j$ *if there exists $\lambda \neq 0$ such that for all $v \in P, v_j = \lambda v_i$.*
- $i \underset{c}{\not\longrightarrow} j$ *if there exists $v \in P$ such that $v_i \neq 0$ and $v_j = 0$.*

Reactions i and j are fully (resp. partially, directionally) coupled if the relation $i \underset{c}{\Longleftrightarrow} j$ (resp. $i \underset{c}{\longleftrightarrow} j$, $i \underset{c}{\longrightarrow} j$) holds. Otherwise, i and j are uncoupled, i.e., $i \underset{c}{\not\longrightarrow} j$ and $j \underset{c}{\not\longrightarrow} i$.

With $i \Longleftrightarrow j$, $i \longleftrightarrow j$, $i \longrightarrow j$, and $i \not\longrightarrow j$ we denote the corresponding (unconstrained) coupling relations in C_P, where P is replaced with C_P.

In the unconstrained case, fully coupled reactions belong to the same enzyme subset as introduced in [5].

2.2 Preprocessing

In a first preprocessing step, we aim to find the blocked reactions in P. As shown by computational experiments with FFCA [6] or F2C2 [7], genome-scale metabolic networks may contain several hundreds of blocked reactions. Eliminating these considerably reduces the network size.

Observation 1 *If a reaction is blocked in C_P, then it is also blocked in P.*

Observation 1 holds since $P \subseteq C_P$. Thus, non-existence of $v \in C_P$ with $v_i \neq 0$ implies the non-existence of such a vector in P. This allows us to detect and remove some blocked reactions in P by studying C_P. In particular, we can use the concept of dead-end metabolites from F2C2 [7] and perform a stoichiometric matrix-based search to find some of the blocked reactions in P.

Finding blocked reactions relates back to a more general problem. Even if a reaction is unblocked, there is no guarantee that it is able to display all the fluxes specified by its bounds. The network structure might further constrain reaction fluxes and prohibit them to attain their limit. This is different from the unconstrained case, where fluxes through unblocked reactions are scalable by any positive number. Therefore, we will compute tight bounds for every reaction, which we call the *true bounds* of the reaction.

Definition 3 (True bounds) *$l_k^* \in \mathbb{R} \cup \{-\infty\}$ (resp. $u_k^* \in \mathbb{R} \cup \{\infty\}$) is called the true lower (resp. upper) bound of reaction k if $u_k^* \geq v_k \geq l_k^*$, for all $v \in P$, and if for all $c \in]l_k^*, u_k^*[$, there exists $v \in P$ with $v_k = c$.*

Computing the true lower and upper bound of a reaction k can be done by solving the following two linear programs:

$$\begin{aligned} l_k^* &= \min \{v_k : Sv = 0,\ v_i \geq l_i,\ \text{for all } i \in L,\ v_i \leq u_j,\ \text{for all } j \in U\}, \\ u_k^* &= \max \{v_k : Sv = 0,\ v_i \geq l_i,\ \text{for all } i \in L,\ v_i \leq u_j,\ \text{for all } j \in U\}. \end{aligned} \tag{3}$$

Having computed the true bounds for every reaction, we can equivalently characterize the flux polyhedron P as

$$P = \{v \in \mathbb{R}^n \mid Sv = 0,\ u_i^* \geq v_i \geq l_i^*,\ \text{for all } i \in \{1, \ldots, n\}\}. \tag{4}$$

Trivially, if $l_k^* = u_k^* = 0$, then k is blocked in P. Since blocked reactions can be removed from the network without altering the coupling relationships between other reactions, we will assume for the rest of this section that the constrained metabolic network does not contain such reactions.

Based on the true bounds (l_k^*, u_k^*), we will distinguish eight classes of (unblocked) reactions:

- type 1: $(-\infty, \infty)$
- type 2: $(-a, \infty)$ with $a > 0$.
- type 3: $(-a, b)$ with $b \geq a > 0$.
- type 4: $(0, \infty)$

– type 5: $(0, b)$ with $b > 0$.
– type 6: (a, ∞) with $a > 0$.
– type 7: (a, b) with $b > a > 0$.
– type 8: (a, a) with $a > 0$.

Different types could appear, but by conveniently reversing the direction of such reactions (i.e., by multiplying the corresponding column in the stoichiometric matrix with -1), all cases can be reduced to the previous eight. For example, by multiplying the column of a reaction that would be of type $(-\infty, 0)$, we get the type $(0, \infty)$.

2.3 Algorithmic considerations

We now describe the connection between coupling relations in the flux polyhedron P and the associated flux cone C_P. As our results will show, in many cases it is enough to compute the flux coupling relations in C_P (using classical FCA) to obtain the constrained FCA relations for P. For the proofs of the following propositions, we refer to [8].

Proposition 2 *Consider a flux polyhedron P with no blocked reactions. If for two reactions i and j, $i \Longleftrightarrow j$ (resp. $i \longleftrightarrow j$, $i \longrightarrow j$) in C_P holds, then $i \underset{c}{\Longleftrightarrow} j$ (resp. $i \underset{c}{\longleftrightarrow} j$, $i \underset{c}{\longrightarrow} j$) also holds in P.*

Prop. 2 asserts that if a coupling relation exists in the unbounded network, the same relation will be carried over to the constrained case. Thus, we only need to check whether $i \not\longrightarrow j$ in C_P becomes $i \underset{c}{\longrightarrow} j$ in P. If both $i \underset{c}{\longrightarrow} j$ and $j \underset{c}{\longrightarrow} i$ hold in P, then we also have to check whether $i \underset{c}{\Longleftrightarrow} j$ holds in P.

Proposition 3 *Consider a flux polyhedron P with no blocked reactions. Assume every reaction is of type $1-7$. Then for any two reactions i and j, we have*

a) $i \underset{c}{\Longleftrightarrow} j$ in P if and only if $i \Longleftrightarrow j$ in C_P.
b) $i \underset{c}{\longrightarrow} j$ in P if and only if ($i \longrightarrow j$ in C_P or j is of type 6 or 7).

Together Prop. 2 and Prop. 3 imply that if every reaction in the metabolic network shows variability (i.e., is not of type 8), then it is enough to compute the unconstrained coupling relationships in C_P (using classical FCA). All constrained coupling relationships then can be obtained without solving additional linear programs.

Thus, the only case to be further considered is the one where at least one reaction is of type 8. While we can easily deduce the constrained coupling relationship between a type 8 reaction and any other reaction in the network, the effect these reactions will have on the remaining constrained coupling relationships is not trivial.

Observation 4 *For a flux polyhedron P, let i be of type 6, 7 or 8, and let j be an unblocked reaction in P. Then $j \underset{c}{\longrightarrow} i$ holds.*

Table 1. Summary of the deducible constrained coupling relationships based on the reaction type

i \ j	1	2	3	4	5	6	7	8
1	$i \underset{c}{\nleftrightarrow} j$ $i \underset{c}{\Longleftrightarrow} j$	$i \underset{c}{\nleftrightarrow} j$	$i \underset{c}{\nleftrightarrow} j$	$i \underset{c}{\nleftrightarrow} j$ $i \underset{c}{\longrightarrow} j$	$i \underset{c}{\nleftrightarrow} j$	$i \underset{c}{\longrightarrow} j$	$i \underset{c}{\longrightarrow} j$	$i \underset{c}{\longrightarrow} j$
2		$i \underset{c}{\nleftrightarrow} j$ $i \underset{c}{\Longleftrightarrow} j$	$i \underset{c}{\nleftrightarrow} j$	$i \underset{c}{\nleftrightarrow} j$ $i \underset{c}{\longrightarrow} j$	$i \underset{c}{\nleftrightarrow} j$	$i \underset{c}{\longrightarrow} j$	$i \underset{c}{\longrightarrow} j$	$i \underset{c}{\longrightarrow} j$
3			$i \underset{c}{\nleftrightarrow} j$ $i \underset{c}{\Longleftrightarrow} j$	$i \underset{c}{\nleftrightarrow} j$ $i \underset{c}{\longrightarrow} j$	$i \underset{c}{\nleftrightarrow} j$ $i \underset{c}{\longrightarrow} j$	$i \underset{c}{\longrightarrow} j$	$i \underset{c}{\longrightarrow} j$	$i \underset{c}{\longrightarrow} j$
4				any	$i \underset{c}{\nleftrightarrow} j$ $j \underset{c}{\longrightarrow} i$	$i \underset{c}{\longrightarrow} j$	$i \underset{c}{\longrightarrow} j$	$i \underset{c}{\longrightarrow} j$
5					any	$i \underset{c}{\longrightarrow} j$	$i \underset{c}{\longrightarrow} j$	$i \underset{c}{\longrightarrow} j$
6						$i \underset{c}{\longleftrightarrow} j$ $i \underset{c}{\Longleftrightarrow} j$	$i \underset{c}{\longleftrightarrow} j$	$i \underset{c}{\longleftrightarrow} j$
7							$i \underset{c}{\longleftrightarrow} j$ $i \underset{c}{\Longleftrightarrow} j$	$i \underset{c}{\longleftrightarrow} j$
8								$i \underset{c}{\Longleftrightarrow} j$

Proposition 5 *For a flux polyhedron P, let i be a reaction with true lower bounds $l_i^* < 0$, and j a reaction of different type. Then $j \underset{c}{\nrightarrow} i$ always holds.*

Corollary 6 *For a flux polyhedron P, let i and j be two reactions of different type with $l_i^* < 0$ and $l_j^* < 0$. Then $i \underset{c}{\nleftrightarrow} j$ holds.*

Proposition 7 *For a flux polyhedron P, let i and j be two reactions with true upper bounds $u_i^* = \infty$ and $u_j^* \neq \infty$. Then either $j \underset{c}{\nrightarrow} i$ holds or $l_i^* > 0$.*

Proposition 8 *For a flux polyhedron P, let i and j be two unblocked reactions in P. Then $i \underset{c}{\Longleftrightarrow} j$ holds only if i and j are of the same type.*

Proposition 9 *For a flux polyhedron P, let i and j be two reactions both of type 1, both of type 2 or both of type 3. Then $i \underset{c}{\longrightarrow} j$ holds only if $j \underset{c}{\longrightarrow} i$ holds.*

The previous results are summarized in Tab. 1. White cells mark the cases where the constrained flux coupling relationship are uniquely determined by the reaction types, whereas gray cells represent the cases where additional linear programs have to be solved. Note that there are only two cases where any coupling can appear. In all other cases, the coupling is either uniquely determined

or corresponds to one out of two options. The lower diagonal part of the table was left blank for simplicity as it is reverse symmetrical to the upper diagonal part.

For the entries in Tab. 1 where the coupling is not uniquely determined, solving additional linear programs (LPs) is necessary. We denote by $min(i, j, c_i)$ the optimum of the following LP, where i and j are two reactions and c_i is some constant with $c_i \in]l_i^*, u_i^*[\setminus \{0\}$:

$$min(i, j, c_i) = \min \{|v_j| : Sv = 0,\ v_i = c_i,\ u_k^* \geq v_k \geq l_k^*, \forall k \in \{1, \ldots, n\}\} \quad (5)$$

Based on Tab. 1, we distinguish between the following cases:

1. Only $i \underset{c}{\nleftrightarrow} j$ or $i \underset{c}{\Longleftrightarrow} j$ is possible (entries (1, 1), (2, 2) and (3, 3)). To find which coupling applies, it is enough to solve two LPs: $min(i, j, c_i^1)$ and $min(i, j, c_i^2)$, where $c_i^1 > 0$ and $c_i^2 < 0$. If the optimal solution to either LP is 0, then $i \underset{c}{\nleftrightarrow} j$, otherwise $i \underset{c}{\Longleftrightarrow} j$.
2. Only $i \underset{c}{\nleftrightarrow} j$ or $i \underset{c}{\longrightarrow} j$ is possible (entries (1, 4), (2, 4), (3, 4), (3, 5), (4, 5)). In this case, we can decide between the two options by solving $min(i, j, c_i)$. If $min(i, j, c_i) > 0$ then $j \underset{c}{\longrightarrow} i$, otherwise $i \underset{c}{\nleftrightarrow} j$.
3. Only $i \underset{c}{\longleftrightarrow} j$ or $i \underset{c}{\Longleftrightarrow} j$ is possible (entries (6, 6), (7, 7)). We can decide between the two cases by solving $min(i, j, c_i)$ and $min(j, i, c_j)$. $i \underset{c}{\Longleftrightarrow} j$ holds if and only if $min(i, j, c_i) \cdot min(j, i, c_j) = c_j \cdot c_i$.
4. Any coupling is possible (entries (4, 4), (5, 5)). Determining $min(i, j, c_i)$ and $min(j, i, c_j)$ suffices. If $min(i, j, c_i) > 0$ then $j \underset{c}{\longrightarrow} i$. If $min(j, i, c_j) > 0$ then $i \underset{c}{\longrightarrow} j$. If both LPs have a positive optimum, then $i \underset{c}{\longleftrightarrow} j$. $i \underset{c}{\Longleftrightarrow} j$ holds if additionally $min(i, j, c_i) \cdot min(j, i, c_j) = c_j \cdot c_i$.

In the F2C2 algorithm, the most important single improvement was obtained by the feasibility rule (Observation 6 in [7]). A similar observation can be made in the constrained case.

Observation 10 (Constrained feasibility rule) *For any $v \in P$ let $I = \{i \mid v_i \neq 0\}$ and $J = \{j \mid v_j = 0\}$. Then $i \underset{c}{\nrightarrow} j$ for all $(i, j) \in I \times J$.*

The transitive nature of flux coupling is preserved by Def. 2. Thus similar transitivity rules can be derived as for F2C2 [7]. For three reactions i, j and k, it is sometimes possible to derive a coupling relationship between i and k, based on the coupling relationship for reactions i and j, and reactions j and k. We refer to [8] for more details.

The main steps of CFCA are summarized in Tab. 2. In the worst case, the algorithm has to solve $2n + n(n - 1)$ linear programs.

Table 2. Main steps of the CFCA algorithm

Step	Rule
1.	Iteratively remove dead-end metabolites and incident reactions to them.
2.	Classify the reactions based on true lower and upper bounds; remove the remaining blocked reactions
3.	Compute the flux coupling relationships of C_P with F2C2.
4.	If every reaction is of type 1-7, use Prop. 2 and Prop. 3 to deduce all constrained coupling relationships, STOP.
5.	If the coupling relationship for every pair of reaction has been computed, STOP.
6.	Select a pair of reactions i and j for which a coupling has not yet been determined.
7.	Use Tab. 1 and solve corresponding LPs to determine the coupling between i and j.
8.	For every feasible vector computed in step 7, use the constrained feasibility rule to derive additional constrained uncoupling relationships.
9.	For every new constrained (un-)coupling relation computed in steps 7 and 8, use the transitivity rules to derive additional coupling relations.
10.	Goto step 5.

Table 3. Coupling relations in the *E. coli* core metabolism

Flux coupling relations	Directional	Partial	Full	Blocked reactions
CFCA	401	4	40	2
FCA	10	0	38	0

3 Results and conclusion

The CFCA algorithm has been implemented in Matlab, with CLP [9] as linear programming solver. We applied CFCA to the *E. coli* core metabolism [10], a network with 76 reactions and a biomass function. The glucose uptake was set to $10\,mmol/(gDW \cdot hr)$, and a minimum ATP production of $7.6\,mmol/(gDW \cdot hr)$ was required, which corresponds to the associated non-growth maintenance cost [10]. All other irreversible reactions were constrained to the interval $[0, 1000]$, while reversible reactions were limited to $[-1000, 1000]$. Tab. 3 summarizes the coupling relations of the constrained network. For comparison, we also performed standard FCA on the unbounded network using F2C2.

Many of the new directional coupling relations result from the fact that the glucose uptake and ATP producing reactions were required to have a non-zero flux. Hence, every other reaction will be coupled to these. After subtracting the corresponding 2×76 directional couplings from the total number of 401, there still remain 249 new directional couplings between other pairs of reactions.

We conclude that CFCA is a promising new tool for studying coupling relations in metabolic networks under more general conditions than classical FCA is able to do.

References

1. Terzer, M., Maynard, N.D., Covert, M.W., Stelling, J.: Genome-scale metabolic networks. Wiley Interdiscip Rev Syst Biol Med **1**(3) (2009) 285–297
2. Lewis, N.E., Nagarajan, H., Palsson, B.: Constraining the metabolic genotype-phenotype relationship using a phylogeny of in silico methods. Nat Rev Microbiol **10**(4) (2012) 291–305
3. Burgard, A.P., Nikolaev, E.V., Schilling, C.H., Maranas, C.D.: Flux coupling analysis of genome-scale metabolic network reconstructions. Genome Res. **14**(2) (2004) 301–312
4. Kelk, S.M., Olivier, B.G., Stougie, L., Bruggeman, F.J.: Optimal flux spaces of genome-scale stoichiometric models are determined by a few subnetworks. Scientific reports **2** (2012) 580
5. Pfeiffer, T., Sánchez-Valdenebro, I., Nuño, J.C., Montero, F., Schuster, S.: METATOOL: for studying metabolic networks. Bioinformatics **15** (1999) 251–257
6. David, L., Marashi, S.A., Larhlimi, A., Mieth, B., Bockmayr, A.: FFCA: a feasibility-based method for flux coupling analysis of metabolic networks. BMC Bioinformatics **12**(1) (2011) 236
7. Larhlimi, A., David, L., Selbig, J., Bockmayr, A.: F2C2: a fast tool for the computation of flux coupling in genome-scale metabolic networks. BMC Bioinformatics **13**(75) (2012) 57
8. David, L.: Algorithms for the constraint-based analysis of metabolic networks. Doctoral thesis, Freie Universität Berlin, In preparation.
9. COIN-OR: Computational Infrastructure for Operations Research `http://www.coin-or.org/`.
10. Palsson, B.O.: Systems Biology: Properties of Reconstructed Networks. Cambridge University Press, New York (2006)

RNA Design by Program Inversion via SAT Solving

Alexander Bau[1], Johannes Waldmann[1], and Sebastian Will[2]

[1] HTWK Leipzig, Fakultät IMN
Gustav-Freytag-Str. 42a, 04277 Leipzig, Germany
[2] Leipzig University, Computer Science Department
Härtelstrasse 16-18, 04107 Leipzig, Germany

Abstract. We solve the RNA secondary structure design problem applying a generic system for program inversion; literally inverting RNA folding. The system enables formulating the problem declaratively in a high-level language, while taking advantage of the recent tremendous progress in SAT solving. For this purpose, we formulate the problem of RNA secondary structure prediction in Haskell, apply our constraint compiler CO4 to translate it to propositional logic, and apply the SAT solver MiniSat to obtain a solution efficiently. Untainted by problem specific optimizations, our encouraging preliminary results shed further light on the current state of fully declarative methods in bioinformatics.

1 Introduction

RNAs perform various regulatory and catalytic functions, going far beyond the traditional picture of RNAs as messengers for coding proteins. Regularly, these functions do not directly depend on the RNAs' sequences, but are mediated by their distinct spatial structures. RNA molecules are uniquely described as chains of organic bases. While these linear molecules fold into three-dimensional tertiary structures, many aspects of RNA structures are commonly studied at the level of RNA secondary structure, i.e. the set of pairs of bases that are brought into close spatial contacts by hydrogen bonds. Thus for an RNA molecule of length n, its *sequence* $p = p_1 \dots p_n$ (aka. its *primary structure*) is a string over the alphabet $\{A, C, G, U\}$. Its *secondary structure* s is a matching of the sequence positions; its elements (i, j) represent *base pairs*. For simplicity, one commonly assumes that base pairs are *canonical*, i.e. $p_i p_j \in \{AU, CG, GC, GU, UA, UG\}$. Furthermore, we require s to be *non-crossing*, i.e. s does not contain *crossing* base pairs a and b, where $a_1 < b_1 < a_2 < b_2$ (or symmetrically).

The folding of an RNA with sequence p into a structure s is associated with a *free energy* $E(p, s)$. There are different energy models for computing such free energies. The most accurate models sum over energy terms of loops (single structural elements that are limited by one or several base pairs), where the energy terms are derived from experimental (or statistical) parameters [12,6,2].

For outlining novel algorithmic ideas, commonly simpler energy models are utilized, where energy terms are assigned to single base pairs. Applying some care, results do regularly generalize well to more complex energy models. Consequently, we study a model that assigns finite energy terms to each *canonical* base pair (i, j) (e.g. distinguishing GC, AU, and GU pairs); we forbid non-canonical pairs by assigning infinite energy. Notably similar models are applied to the fast comparison of RNAs based on RNA ensembles [5,16].

An elementary problem of RNA bioinformatics is *RNA secondary structure prediction*. Since RNA folding, like every physical process, strives to minimize the free energy, this problem asks for the structure s of p with minimum $E(p, s)$. Commonly RNA structure analysis is limited to non-crossing structures, since this allows to predict structures efficiently by dynamic programming; this holds for base-pair based energy models [13] as well as for more realistic loop-based models [17].

A similarly fundamental problem is known as RNA design, which asks for a sequence of an RNA that folds "optimally" into a given structure. The *RNA secondary structure design problem* is most naturally phrased as the exact inverse of structure prediction (therefore, its alias name "inverse folding"). Its input is a secondary structure s, while its output is a primary structure p such that s is the solution of the structure prediction problem for p. Formally, it asks for p such that

$$E(p, s) = \min\{E(p, s') \mid s' \in \text{secondary structures}\}, \tag{1}$$

which we call the *design constraint*.

The design problem has important applications ranging from the study of RNA evolution to drug design. Several heuristic approaches have been proposed, which commonly use some kind of stochastic search procedure to find solutions. Examples are RNAinverse [10], RNA-SSD [1], InfoRNA [4], MODENA [15], and RNAifold [8]. The latter stands out by combining a constraint approach with large neighborhood search. While

it is already based on constraint-programming, which implies various potential advantages like provable optimality (for small instances), we aim at an even higher level of declarativity.

We propose to make use of the inverse relation between folding and design in the most immediate declarative way due to general program inversion (without any problem-specific optimizations.) Essentially, we employ a fully automatized system to invert a program for structure prediction. By and large, we formulate the conditions on the solution as a purely functional program, written in a Haskell-like language. Then, applying our constraint compiler CO4 [3], we generate a program that, given input s, builds a specific formula F_s of propositional logic. This formula is designed together with a mapping sol such that we can derive a solution $\mathrm{sol}(\sigma)$ from any satisfying assignment σ of F_s. In our system, F_s is solved using MiniSat [7]. Resorting to a modern SAT solver grants us immediate access to the tremendous progress in the SAT solving area over the last years. In this work, we rely on the impressive capabilities of current solvers when coping with large instances even without any domain-specific tweaking.

Due to its fully declarative nature, our RNA design approach can be extended flexibly. As immediate benefit, the input may contain additional constraints on p, e.g. bases at certain sequence positions can be fixed or restricted. Moreover, we solve a more realistic design variant that asks for a sequence p that has s as its unique and stable optimal structure. Formally,

$$\text{for each } s' \neq s : E(p, s') > E(p, s) + \Delta, \tag{2}$$

which we call the *stability constraint*.

We shortly review program inversion built around the constraint compiler CO4. Consequently, we describe our constraint programs for the standard design problem and the design with additional stability requirements. The latter demonstrates the extensibility of our general approach. Finally, we discuss the novel approach in the light of our preliminary results and look out to future developments.

2 The CO4 Language and Compiler

CO4 is a high-level declarative language for describing constraint systems. The language is a subset of the purely functional programming language Haskell [11] that includes user-defined algebraic data types and recursive

functions defined by pattern matching, as well as higher-order polymorphic types. This language comes with a compiler that transforms a high-level constraint system into a satisfiability problem in propositional logic. The motivation of this transformation are tremendous developments in the area of constraint solvers for propositional satisfiability (SAT solvers) in the recent years. Modern SAT solvers like MiniSat [7] are able to find satisfying assignments for conjunctive normal forms with 10^6 and more clauses within minutes (and sometimes, even seconds). With the availability of powerful SAT solvers, *propositional encoding* is a promising method to solve constraint systems that originate in different domains. CO4 makes available the power of SAT solvers (the back-end) via a high-level programming language (the front-end), allowing the programmer to write clean and concise code, without worrying about details of its translation.

CO4 programs are handled and executed in two stages: The input program contains a *top-level constraint* that defines a relation between a parameter domain K and a domain U for unknowns. In our RNA design application, K consists of secondary RNA structures, while U consists of primary RNA structures. The top-level constraint is represented as a Haskell function of type $f : K \times U \to \{\textsc{False}, \textsc{True}\}$. In the first processing stage (at *compile-time*), the program for f is translated into a program $g : K \to F \times (\Sigma \to U)$ with F being the set of formulas of propositional logic, and Σ being the set of assignments from variables of F to truth values. In the second stage (at *run-time*), a parameter value $p \in K$ is given, and $g\ p$ is evaluated to produce a pair $(v, d) \in F \times (\Sigma \to U)$. An external SAT solver then tries to determine a satisfying assignment $\sigma \in \Sigma$ of v. On success, $d(\sigma)$ is evaluated to a *solution* value $s \in U$. Proper compilation ensures that $f\ p\ s = \textsc{True}$.

CO4 features built-in natural numbers, with primitives for arithmetics and comparisons. Numbers use a fixed-width binary representation, and arithmetical operations use a ripple-carry adder, and Wallace tree multiplier. These operations fail on overflow, i.e., the outgoing carry bit of addition is implicitly constrained to be 0. Practically, one avoids overflow by choosing the required bit width for individual problem instances; for many problems, including RNA design, this width can be estimated.

In principle, the CO4 language could also be compiled to other constraint systems, e.g. finite domain constraints or SMT, such that arithmetical constraints are handled in the back-end, instead of encoding them to SAT. Comparing different compilation targets would be very interesting, but is beyond the current scope of the CO4 project.

In [3] we specify the compilation, elaborate our implementation of the given specification, and prove its correctness. A prototype implementation of CO4 is available at `https://github.com/apunktbau/co4`. The source code the RNA design constraint system is at `CO4/Test/WCB_Matrix.hs`.

3 RNA Design Constraints in CO4

For a constraint system that expresses the design constraint (Eq. 1), we describe the value of the left-hand side by the computation of a stack automaton, and we describe the value of the right-hand side by an algebraic dynamic programming (ADP) matrix computation.

We use this data type to represent energy values:

```
data Energy = MinusInfinity | Finite Nat
```

Recall that an infinite energy represents a non-standard base pair. `Energy` is a semiring, where addition is the "max" operation, used for selecting the best option when several are available; and multiplication is the "plus" operation, used for combining energies from sub-solutions.

```
plus :: Energy -> Energy -> Energy
plus x y = case x of
    MinusInfinity -> y ; Finite f -> case y of
        MinusInfinity -> x ; Finite g -> Finite (maxNat8 f g)
times :: Energy -> Energy -> Energy
times x y = case x of
    MinusInfinity -> x ; Finite f -> case y of
        MinusInfinity -> y ; Finite g -> Finite (plusNat8 f g)
```

CO4 represents an unknown energy value by one propositional variable for the choice between `MinusInfinity` and `Finite`, and a sequence of propositional variables to represent the argument of `Finite` (a binary number), and it compiles the `case` expression accordingly.

Our energy model is a function `cost :: Base -> Base -> Energy` that contains canonical base pairs with different energies, `cost C G = cost G C = Finite 3`, `cost A U = cost U A = Finite 2`, and a wobbly base pair, `cost G U = cost U G = Finite 1`, while all other costs are `MinusInfinity`.

We can refine the energy model by using larger numbers in the cost function. We choose a bit width such that the maximal energy can be represented. E.g., if the maximal cost of a pair is 10, and the the input sequence length is 40, then there are at most 20 pairs, and the energy is bounded by 200, so we need 8 bit for numbers.

The energy of p when folded along s is computed by a stack automaton:

```
data Base = A | C | G | U ; data Primary = [ Base ]
data Paren = Open | Blank | Close ; data Secondary = [ Paren ]
type Stack = [ Base ]
parse :: Stack -> Primary -> Secondary -> Energy
parse stack p s = case s of
    [] -> Finite 0
    y : s' -> case y of
      Blank -> parse stack (tail p) s'
      Open ->  parse (head p : stack) (tail p) s'
      Close -> times (cost (head stack) (head p))
                     (parse (tail stack) (tail p) s')
```

This is a Haskell program (`case s of ...` is a case distiction on the top constructor of the list `s`, and `y:s'` is a pattern for a non-empty list). We will use this program to generate SAT constraints for the unknown argument `p` of type `Primary`. We emphasize that the program can be used literally to compute the energy in case the primary and secondary structure are given, and we indeed do this, for double-checking the solution that we obtained from the solver.

The optimal energy for the (unknown) primary structure is computed by the ADP framework [9]. For a primary structure $w = [w_1, \ldots, w_n]$, we describe the energy matrix E of dimension $(n+1) \times (n+1)$ that contains in $E(i,j)$ the maximal energy for the substring $w[i+1, \ldots, j]$. This E is the (pointwise) least solution of the equation (derived from the grammar)

$$E = \mathrm{Item} + E \cdot E + \sum_{x,y \in \mathrm{Base}} \mathrm{cost}(x,y) \cdot \mathrm{Item}_x \cdot \mathrm{gap}_3(E) \cdot \mathrm{Item}_y \,. \qquad (3)$$

Here, "+" and "·" mean matrix addition and multiplication over the energy semiring; Item_x is the matrix with $\mathrm{Item}_x(i,j) := \text{if } (i+1) = j \wedge w_j = x$ then 0 else $-\infty$ (this is used to check that a certain base is present); $\mathrm{Item} := \sum_{x \in \mathrm{Base}} \mathrm{Item}_x$; and $\mathrm{gap}_d(M)(i,j) := \text{if } i + d \leq j$ then $M(i,j)$ else $-\infty$ (this is applied to require a minimal hairpin length). Also, $\mathrm{cost}(\cdot,\cdot)$ is a scalar (or a scaled identity matrix).

We can replace the equation by

$$E = \mathrm{Item} + E \cdot E + C \odot \lfloor \mathrm{gap}_3(E) \rceil \qquad (4)$$

which requires fewer operations, where C is the cost matrix (with $C(i,j) :=$ if $i+1 < j$ then $\mathrm{cost}(w_{i+1}, w_j)$ else $-\infty$); and we use an index shift operation $\lfloor M \rceil(i,j) :=$ if $i < n \wedge j > 0$ then $M(i+1, j-1)$ else $-\infty$; and point-wise multiplication $(A \odot B)(i,j) := A(i,j) \cdot B(i,j)$.

The top-level constraint is of type

```
design :: Secondary -> (Primary, Matrix Energy) -> Bool
```

and `design s (p,m)` is true iff `m` is the ADP matrix for `p`, and the design constraint (Eq. 1) for `(s,p)` holds.

The size of the generated SAT formula is bound by the runtime of the compiled program, which itself is bound by the runtime of the original program. The dominant operation in (3) is the multiplication of energy matrices E. So the size of the SAT formula is cubic in the length of the given primary structure.

4 RNA Design Stability Constraints in CO4

To formulate the stability constraint (Eq. 2), we describe in the ADP framework the computation of energy pairs,

```
data Energy2 = Energy2 Energy Energy
```

where the first component is the maximal energy, and the other component is the second best energy. In particular, if the there is only one secondary structure that realizes the maximal energy, the second component is `MinusInfinity`; and if there is more that one secondary structure that realizes the maximal energy, then both components are identical. We obtain the matrix of energy pairs from a version of Equation 4 corresponding to a non-ambiguous grammar: $E = (\mathrm{Item} + C \odot \lfloor \mathrm{gap}_3(E) \rceil) \cdot (E + \mathrm{Id})$

ADP matrix computations from the previous section can be re-used since we use higher order function that takes semiring operations as arguments:

```
plus2 :: Energy2 -> Energy2 -> Energy2
plus2  (Energy2 x1 y1) (Energy2 x2 y2) =
    Energy2 (plus x1 x2) (plus (min x1 x2) (plus y1 y2))
times2 :: Energy2 -> Energy2 -> Energy2
times2 (Energy2 x1 y1) (Energy2 x2 y2) =
    Energy2 (times x1 x2) (plus (times x1 y2) (times x2 y1))
lift2 :: Energy -> Energy2
lift2 e = Energy2 e MinusInfinity
```

We apply `lift2` to all weights, to obtain a constraint system for the ADP matrix with entries of type `Energy2`.

Its top right entry is a pair (e, f) for which we require $e > f + \Delta$, cf. Eq. 2, by adding this to the top-level constraint.

5 Results and Discussion

The objective of this paper has been to study a radical approach to the RNA design problem. While previous methods have introduced specialized heuristics, which are carefully tuned to the specific problem, we have formulated the problem in the most natural way as inverse of the folding problem. This purely declarative approach is enabled by a very recent generic constraint programming system for program inversion built around the compiler CO4. Building on powerful SAT solving, the approach does furthermore guarantee the optimality of designed RNA sequences and even proves non-existence of solutions in over-constrained instances.

Table 1. Test results on exemplary RNA structures.

structure	len	#variables	#clauses	#literals	runtime overall	runtime solver
((((...))..((...))))	20	77951	368036	1166086	5s	2s
((((((....))))))..((((((....))))))	30	235714	1164984	3712214	16s	4s
((((((....)))..(((....)))..(((....))))))	40	526111	2666878	8525686	7m22s	7m
..((((((((((((((((((((....))))))))))..))))))))))	50	989133	5096071	16324618	1m35s	36s

According to our preliminary results (cf. Table 1) on design constraints without the stability condition (cf. Section 3), this system is currently capable of designing RNA sequences of length 40–50 within minutes (on Intel® Core™ 2 Duo CPU with 2.20GHz and 4GB RAM). Design constraints including the stability condition lead to larger formulas, which result in longer solver runs (cf. Table 2).

Table 2. Test results on exemplary RNA structures including the stability condition.

structure	len	#variables	#clauses	#literals	runtime overall	runtime solver
.(((...)))	10	80465	303893	917948	4s	1s
((((...))..((...))))	20	615934	2482954	7500992	1m9s	52s
((((((....))))))..((((((....))))))	30	2042077	8434257	25514060	2m47s	1m58s

These experiments verify the feasibility of our approach. We anticipate that ongoing development of CO4 will allow to produce smaller propositional encodings, and this will speed up solving; as will further advances

in SAT solvers. Whereas several existing approaches support stability design by optimizing the probability of the target structure, we have proposed the alternative design of an energy gap, which is elegantly expressed adopting an algebraic DP perspective. Conceptually straightforward, our approach can be extended to richer energy models by exchanging the grammar in Eq. 3 (and adapting the energy evaluation by `parse`.) Similarly, one could support pseudoknots (from restricted structure classes) by plugging in appropriate grammars (e.g. [14].) However, the current limitation to simple energy models prevents a final evaluation by direct empirical comparison to existing design approaches, which generally support richer energy models. Nevertheless, already the current implementation demonstrates the feasibility of the proposed RNA design approach; its declarativity supports easy extension to many interesting, even novel, variants of RNA design, most significantly design under complex design constraints like the stability constraint.

Acknowledgements

Alexander Bau is supported by the European Science Foundation (ESF grant 100088525).

References

1. Rosalia Aguirre-Hernandez, Holger H. Hoos, and Anne Condon. Computational RNA secondary structure design: empirical complexity and improved methods. *BMC Bioinformatics*, 8:34, 2007.
2. Mirela Andronescu, Anne Condon, Holger H. Hoos, David H. Mathews, and Kevin P. Murphy. Efficient parameter estimation for RNA secondary structure prediction. *Bioinformatics*, 23(13):i19–28, 2007.
3. A. Bau and J. Waldmann. Propositional Encoding of Constraints over Tree-Shaped Data. Technical report, HTWK Leipzig, 2013. ArXiv e-prints. http://arxiv.org/abs/1305.4957.
4. Anke Busch and Rolf Backofen. INFO-RNA–a server for fast inverse RNA folding satisfying sequence constraints. *Nucleic Acids Res*, 35(Web Server issue):W310–3, 2007.
5. Chuong B. Do, Chuan-Sheng Foo, and Serafim Batzoglou. A max-margin model for efficient simultaneous alignment and folding of RNA sequences. *Bioinformatics*, 24(13):i68–76, 2008.
6. Chuong B. Do, Daniel A. Woods, and Serafim Batzoglou. CONTRAfold: RNA secondary structure prediction without physics-based models. *Bioinformatics*, 22(14):e90–8, 2006.
7. Niklas Eén and Niklas Sörensson. An extensible sat-solver. In *SAT*, pages 502–518, 2003.

8. Juan Antonio Garcia-Martin, Peter Clote, and Ivan Dotu. RNAiFOLD: a constraint programming algorithm for rna inverse folding and molecular design. *J Bioinform Comput Biol*, 11(2):1350001, 2013.
9. Robert Giegerich. A systematic approach to dynamic programming in bioinformatics. *Bioinformatics*, 16(8):665–677, 2000.
10. Ivo L. Hofacker, Walter Fontana, Peter F. Stadler, Sebastian Bonhoeffer, Manfred Tacker, and Peter Schuster. Fast folding and comparison of RNA secondary structures. *Monatshefte Chemie*, 125:167–188, 1994.
11. Simon Peyton Jones, editor. *Haskell 98 Language and Libraries, The Revised Report*. Cambridge University Press, 2003.
12. DH Mathews, J Sabina, M Zuker, and DH Turner. Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *J Mol Biol*, 288(5):911–40, 1999.
13. Ruth Nussinov, George Pieczenik, Jerrold R. Griggs, and Daniel J. Kleitman. Algorithms for loop matchings. *SIAM J Appl Math*, 35(1):68–82, July 1978.
14. Jens Reeder and Robert Giegerich. Design, implementation and evaluation of a practical pseudoknot folding algorithm based on thermodynamics. *BMC Bioinformatics*, 5:104, 2004.
15. Akito Taneda. MODENA: a multi-objective RNA inverse folding. *Adv Appl Bioinform Chem*, 4:1–12, 2011.
16. Sebastian Will, Christina Schmiedl, Milad Miladi, Mathias Möhl, and Rolf Backofen. SPARSE: Quadratic time simultaneous alignment and folding of RNAs without sequence-based heuristics. In Minghua Deng, Rui Jiang, Fengzhu Sun, and Xuegong Zhang, editors, *Proceedings of the 17th International Conference on Research in Computational Molecular Biology (RECOMB 2013)*, volume 7821 of *LNCS*, pages 289–290. Springer Berlin Heidelberg, 2013.
17. M. Zuker and P. Stiegler. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Res*, 9(1):133–48, 1981.

Index