Finding minimal P/T-invariants as a CSP

Sylvain Soliman

Sylvain.Soliman@inria.fr
Equipe-Projet Contraintes, INRIA Paris-Rocquencourt,
BP105, 78153 Le Chesnay Cedex, France.

Abstract. We present here a way to compute the minimal semi-positive invariants of a Petri net representing a biological reaction system, as resolution of a CSP. The use of Petri-nets to manipulate those models and make available a variety of tools is quite old, and recently analyses based on invariant computation for biological models have become more and more frequent, especially in the context of module decomposition. In our case, this analysis brings both qualitative and quantitative information on the models, in the form of conservation laws, consistency checking, etc. thanks to finite domain constraint programming. It is noticeable that some of the most recent optimizations of standard invariant computation techniques in Petri-nets correspond to well-known techniques in CSPs, like symmetry-breaking. A simple prototype based on GNU-Prolog's FD solver, and including symmetry detection and breaking, was incorporated into the BIOCHAM modelling environment. Some illustrative examples and a few benchmarks are provided.

1 Introduction

Reaction models like those of `reactome.org`, KEGG pathway database [1] or `biomodels.net` represent a growing part of Systems Biology especially for metabolic or signalling pathways, cell-cycle and more generally post-genomic regulation systems. They build on established standards like BioPAX or SBML [2] to facilitate the exchange and comparison of models and benefit from a large number of available tools, especially ODE integration based simulators.

The use of Petri-nets to represent those models, taking into account the difference between compounds and reactions in the graph, and make available various kinds of analyses is quite old [3], however it remains somehow focused towards mostly qualitative and structural properties. Some have been used for module decomposition, like (I/O) T-invariants [4,5], related to dynamical notions of elementary flux modes [6]. However, there is, to our knowledge, very little use of P-invariant computation, which provides both qualitative information about some notion of module related to the "life cycle" of compounds, and quantitative information related to conservation laws and Jacobian matrix singularity. Conservation law extraction is actually already provided by a few tools, but then using numerical methods, based on the quantitative view of the model, and not integer arithmetic (as in direct P-invariant analysis).

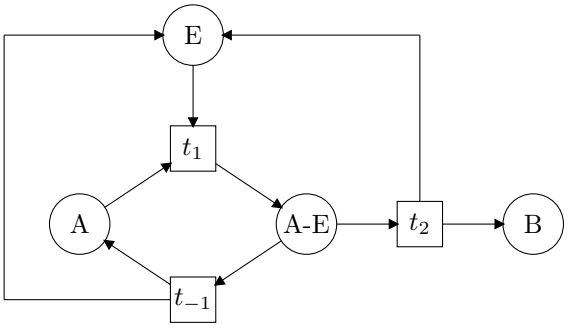
We present here a very simple way to incorporate invariant computation in an existing biological modelling tool, using constraint programming with symmetry detection and breaking. We compare it to other approaches and evaluate it, for the case of P-invariants, on some examples of various sizes, like the MAPK cascade models of [7] and [8]. This experimentation is done through an implementation of the described method in the BIOCHAM modelling environment[1] [9,10].

2 Petri-net view of a reaction model

A Petri-net is a bipartite oriented (weighted) graph of transitions, usually represented as square boxes, and places, usually represented as circles, that defines a (actually not only one) transition relation on *markings* of the net, i.e. multisets of tokens associated to places. The relation is defined by *firings* of transitions, i.e. when there are tokens (as many as the weight of the incoming arc) in all pre-places of a transition, they can be consumed and as many tokens as the weight on the outgoing arc are added to each post-place.

The classical Petri-net view of a reaction model is simply to associate biochemical *species to places* and biochemical *reactions to transitions*.

Example 1. For instance the enzymatic reaction written (in BIOCHAM-like syntax), `A + E <=> A-E => B + E` corresponds to the following Petri-net :



In this Petri-net, starting from a marking with at least one token in A and in E, one can remove one of each to produce one token in A-E (firing of t_1) and then either remove it to add again one token to A and one to E (firing of t_{-1}), or to add one B and one E (firing of t_2).

P (resp. T) invariants are defined, as usual, as vectors V representing a multiset of places (resp. of transitions) such that $V \cdot I = 0$ (resp. $I \cdot V = 0$) where I is the *incidence matrix* of the Petri net, i.e. I_{ij} is the number of arcs

[1] At review time the version containing P-invariant computation might not have been released, but only in beta versions available at `http://www-rocq.inria.fr/~soliman/Biocham.dmg`

from transition i to place j, minus the number of arcs from place j to transition i. Intuitively, a P-invariant is a multiset representing a weighting of the places and such that any such weighted marking remains invariant by any firing; a T-invariant represents a multiset of firings that will leave invariant any marking (see also section 4). As explained in introduction, for reaction models these invariants are used for flux analysis, variable simplification through conservation law extraction, module decomposition, etc.

3 Related work

To compute the invariants of a Petri net, especially if this computation is combined with other Petri-net analyses, like sinks and sources, traps, deadlocks, etc. the most natural solution is to use a Petri-net dedicated tool like INA, PiNA, or Charlie for instance through the interface of Snoopy [11], which will soon allow the import of SBML models as Petri-nets. Standard integer methods like Fourier-Motzkin elimination will then provide an efficient means to compute P or T-invariants. These methods however generate lots of candidates which are afterwards eliminated and also need to incorporate some means (like equality class definition) to avoid combinatorial explosion at least in some simple cases, as explained in section 5.

Another way to extract the minimal semi-positive invariants of a model is to use one of the software tools that provide this computation for biological systems, generally as "conservation law" computation, and based on linear algebra methods like QR factorization [12]. This is the case for instance of the METATOOL [13] and COPASI [14] tools. The idea is to use a linear relaxation of the problem, which suits well very big graphs, but needs again *a posteriori* filtering of the candidate solutions. Moreover, these methods do not incorporate any means of symmetry elimination (see section 5).

4 Finding invariants as a Constraint Solving Problem

We will illustrate our new method for computing the invariants with the case of P-invariants (but T-invariants, being dual, would work in the same fashion). For a Petri net with p places and t transitions ($L_i \rightarrow R_i$), a P-invariant is a vector $V \in \mathbb{N}^p$ s.t. $V \cdot I = 0$, i.e. $\forall 1 \leq i \leq t \; V \cdot L_i = V \cdot R_i$. Since those vectors all live in $\mathbb{N}^p$, it is quite natural to see this as a CSP with t (linear) equality constraints on p Finite Domains variables.

Example 2. Using the Petri-net of example 1 we have:

```
A + E => A-E
  A-E => A + E
  A-E => B + E
```

This results in the following equations:

$$A + E = AE \tag{1}$$
$$AE = A + E \tag{2}$$
$$AE = B + E \tag{3}$$

where obviously equation (2) is redundant.

The task is actually to find invariants with minimal support (a linear combination of invariants belonging to $\mathbb{N}^p$ also being an invariant), i.e. having as few non-zero components as possible, these components being as small as possible, but of course non trivial, we thus add the constraint that $V \cdot \mathbf{1} > 0$.

Example 3. In our running example we thus add $A + E + AE + B > 0$.

Now, to ensure minimality the labelling is invoked from small to big values and a branch and bound procedure is wrapped around it, maintaining a partial base $\mathcal{B}$ of P-invariant vectors and adding the constraint that a new vector V is solution if $\forall B \in \mathcal{B}\ \prod_{B_i \neq 0} V_i = 0$, which means that its support is not bigger than that of any vector of the base.

Unfortunately, even with the last constraint, no search heuristic was found that makes removing subsumed P-invariants unnecessary. Thus, if a new vector is added to $\mathcal{B}$, previously found vectors with a bigger support must be removed.

This algorithm was implemented directly into BIOCHAM [9], which is programmed in GNU-Prolog, and allowed for immediate testing.

Example 4. In our running example we find two minimal semi-positive P-invariants:

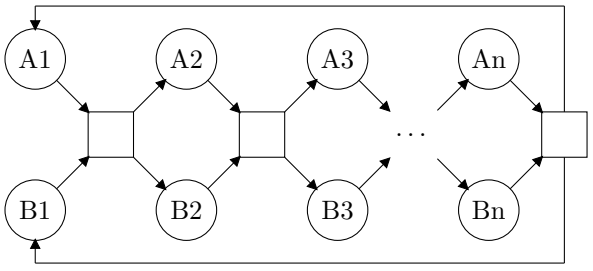
- $E = AE = 1$ and $A = B = 0$
- $A = B = AE = 1$ and $E = 0$

5 Equality classes

The problem of finding minimal semi-positive invariants is clearly EXPSPACE since there can be an exponential number of such invariants. For instance the model given in example 5 has 2^n minimal semi-positive P-invariants (each one with either A_i or B_i equal to 1 and the other equal to 0).

Example 5.

```
A1 + B1 => A2 + B2
A2 + B2 => A3 + B3
...
An + Bn => A1 + B1
```



A first remark is that in this example, there is a variable symmetry between all the pairs (A_i, B_i) of variables corresponding to places. This symmetry is easy to detect (purely syntactical) and can be eliminated through the usual ordering of variables, by adding the constraints $A_i \leq B_i$.

This classical CSP optimization is enough to avoid most of the trivial exponential blow-ups and corresponds to the initial phase of *parallel places* detection and merging of the equality classes optimization for the standard Fourier-Motzkin algorithm [15]. Note however that in that method, classes of equivalent variables are detected and eliminated before and *during* the invariant computation, which would correspond to local symmetry detection and was not implemented in our prototype.

Moreover, in [15], *equality class* elimination is done through replacement of the symmetric places by a representative place. The full method reportedly improves by a factor two the computation speed. Even if in the context of the original article this is done only for ordinary Petri-nets (only one edge from one place to a transition and from one transition to one place), we can see that it can be even more efficient to use this replacement technique in our case:

Example 6.

```
...
A + B => 4*C
...
```

Instead of simply adding $A \leq B$ to our constraints, which will lead to 3 solutions when $C = 1$ before symmetry expansion: $(A, B) \in \{(0, 4), (1, 3), (2, 2)\}$, replacing A and B by D will reduce to a single solution $D = 4$ before expansion of the subproblem $A + B = D$.

This partial detection of independent subproblems, which can be seen as a complex form of symmetry identification, can once again be done syntactically at the initial phase, and can be stated as follows: replace $\sum_i k_i * A_i$ by a single variable A if all the A_i occur only in the context of this sum i.e. in our Petri net all pre-transitions of A_i are connected to A_i with k_i edges and to all other A_j with k_j edges and same for post-transitions. For a better constraint propagation, another intermediate variable can be introduced such that $A = gcd(k_i) \cdot A'$. In our experiments the simple case of *parallel places* (i.e. all k_i equal to 1 in the sum) was however the one encountered most often.

6 Example, the MAPK Cascade

The MAPK signal transduction cascade is a well studied system that appears in lots of organisms and is very important for regulating cell division [16]. It is composed of layers, each one activating the next, and in detailed models shows two intertwined pathways conveying EGF and NGF signals to the nucleus.

A simple MAPK cascade model, that of [17] without scaffold, is used here as an example to show the results of P-invariant computation.

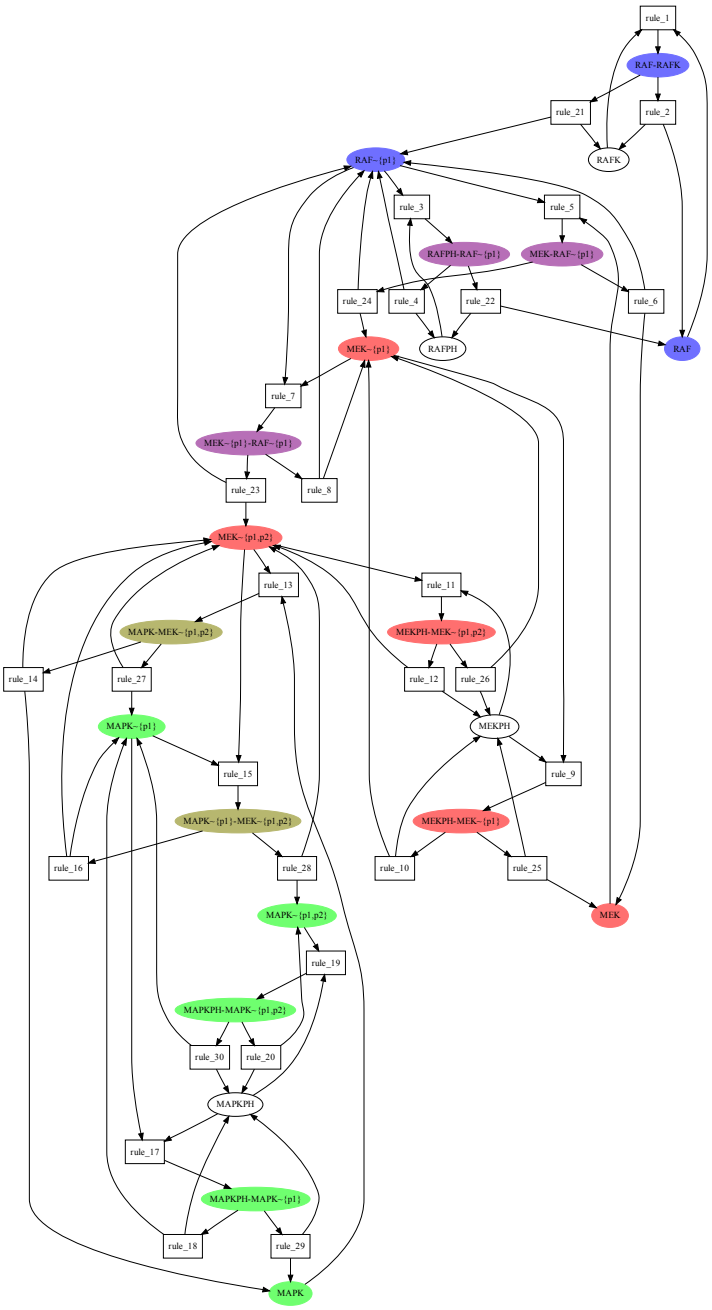


Fig. 1. 3 of the 7 P-invariants found in the MAPK cascade model of [17]. The blue one (RAF), the pink one (MEK) and the green one (MAPK) with intersections in purple (blue+pink) and khaki (pink+green).

Seven minimal semi-positive P-invariants are found almost instantly: RAFK, RAFPH, RAF, MEKPH, MEK, MAPKPH, MAPK. Three of them are depicted in figure 1, the full list is given in table 1.

RAFK, RAF-RAFK
RAFPH, RAFPH-RAF∼{p1}
RAF, MEK-RAF∼{p1}, RAF-RAFK, RAFPH-RAF∼{p1}, MEK∼{p1}-RAF∼{p1}, RAF∼{p1}
MEKPH, MEKPH-MEK∼{p1}, MEKPH-MEK∼{p1, p2}
MEK, MAPK-MEK∼{p1, p2}, MEK-RAF∼{p1}, MEKPH-MEK∼{p1}, MEKPH-MEK∼{p1, p2}, MAPK∼{p1}-MEK∼{p1, p2}, MEK∼{p1}-RAF∼{p1}, MEK∼{p1}, MEK∼{p1, p2}
MAPKPH, MAPKPH-MAPK∼{p1}, MAPKPH-MAPK∼{p1, p2}
MAPK, MAPK-MEK∼{p1, p2}, MAPKPH-MAPK∼{p1}, MAPK∼{p1, p2} MAPK∼{p1}-MEK∼{p1, p2}, MAPK∼{p1}, MAPKPH-MAPK∼{p1, p2},

Table 1. P-invariants of the MAPK cascade model of [17]

Note that these 7 P-invariants define 7 algebraic conservation rules and thus decrease the size of the corresponding ODE model from 22 variables and equations to only 15.

7 Evaluation on other examples

Schoeberl's model is a more detailed version of the MAPK cascade, which is quite comprehensive [8], but too big to be studied by hand. It can however be easily broken down into fourteen more easily understandable units formed by P-invariants, as shown in table 2, along other examples representing amongst the biggest reaction networks publicly available.

Model	transit.	places	P-invar.	time (s)	Invariant size
Schoeberl's MAPK [8]	125	105	14	<1	from 2 to 44
Curie's E2F/Rb [18]	∼500	∼400	79	∼10	from size 1 (EP300) to about 230 (E2F1 box)
Kohn's map [19]	∼800	∼500	65	∼40	from size 1 (Myt1) to about 200 (pRb or cdk2)

Table 2. Minimal semi-positive P-invariant computation on bigger models of biochemical reaction networks

We could not compare our results with those provided in [12] since the models they use, coming from metabolic pathways flux analyses, do not have an integer stoichiometry matrix, however the examples of table 2 show the feasibility of P-invariant computation by constraint programming for quite big networks.

Note that for networks of this size, the upper bound of the domain of variables had to be set manually (to a reasonable value like 8 since actually only 2 or 3 was needed in all the biological models we have encountered up to now). Otherwise, the only over-approximation of the upper bound found was the product of the *l.c.m.* of stoichiometric coefficients of each reaction, which explodes really fast and leads to unnecessarily long computation. We thereby lose completeness, but it is not enforced either by QR-factorization methods, and does not seem to miss anything on real life examples.

8 Conclusion

P-invariants of a biological reaction model are not so difficult to compute in most cases. They carry information about conservation laws that are useful for efficient and precise dynamical simulation of the system, and provide some notion of module, which is related to the life cycle of molecules. T-invariants are already used more commonly, and get more and more focus recently.

We introduced a new method to efficiently compute P and T-invariants of a reaction network, based on FD constraint programming. It includes symmetry detection and breaking and scales up well to the biggest reaction networks found. Completeness is lost on the biggest examples but we still look for a better upper bound on domains to restore it.

The idea of applying constraint based methods to classical problems of the Petri-net community is not new, but seems currently mostly applied to the model-checking. We argue that structural problems (invariants, sinks, attractors, etc.) can also benefit from the know-how developed for finite domain CP solving, like symmetry breaking, search heuristics, etc. and thus intend to generalize our approach to other problems of this category.

References

1. Kanehisa, M., Goto, S.: KEGG: Kyoto encyclopedia of genes and genomes. Nucleic Acids Research **28** (2000) 27–30
2. Hucka, M., et al.: The systems biology markup language (SBML): A medium for representation and exchange of biochemical network models. Bioinformatics **19** (2003) 524–531
3. Reddy, V.N., Mavrovouniotis, M.L., Liebman, M.N.: Petri net representations in metabolic pathways. In Hunter, L., Searls, D.B., Shavlik, J.W., eds.: Proceedings of the 1st International Conference on Intelligent Systems for Molecular Biology (ISMB), AAAI Press (1993) 328–336
4. Gilbert, D., Heiner, M., Lehrack, S.: A unifying framework for modelling and analysing biochemical pathways using petri nets. In: CMSB'07: Proceedings of the fifth international conference on Computational Methods in Systems Biology. Volume 4695 of Lecture Notes in Computer Science., Springer-Verlag (2007)

5. Grafahrend-Belau, E., Schreiber, F., Heiner, M., Sackmann, A., Junker, B.H., Grunwald, S., Speer, A., Winder, K., Koch, I.: Modularization of biochemical networks based on classifica- tion of petri net t-invariants. BMC Bioinformatics **9** (2008)
6. Schuster, S., Fell, D.A., Dandekar, T.: A general definition of metabolic pathways useful for systematic organization and analysis of complex metabolic networks. Nature Biotechnology **18** (2002) 326–332
7. Chickarmane, V., Kholodenkob, B.N., Sauro, H.M.: Oscillatory dynamics arising from competitive inhibition and multisite phosphorylation. Journal of Theoretical Biology **244** (2007) 68–76
8. Schoeberl, B., Eichler-Jonsson, C., Gilles, E., Muller, G.: Computational modeling of the dynamics of the map kinase cascade activated by surface and internalized egf receptors. Nature Biotechnology **20** (2002) 370–375
9. Calzone, L., Fages, F., Soliman, S.: BIOCHAM: An environment for modeling biological systems and formalizing experimental knowledge. BioInformatics **22** (2006) 1805–1807
10. Fages, F., Soliman, S., Chabrier-Rivier, N.: Modelling and querying interaction networks in the biochemical abstract machine BIOCHAM. Journal of Biological Physics and Chemistry **4** (2004) 64–73
11. Heiner, M., Richter, R., Schwarick, M.: Snoopy - a tool to design and animate/simulate graph-based formalisms. In: Procedings of the International Workshop on Petri Nets Tools and APplications (PNTAP 2008), Marseille, ACM Digital Library (2008) to appear.
12. Vallabhajosyulaa, R.R., Chickarmane, V., Sauro, H.M.: Conservation analysis of large biochemical networks. BioInformatics (2005) Advance Access.
13. von Kamp, A., Schuster, S.: Metatool 5.0: fast and flexible elementary modes analysis. Bioinformatics **22** (2006) 1930–1931
14. Hoops, S., Sahle, S., Gauges, R., Lee, C., Pahle, J., Simus, N., Singhal, M., Xu, L., Mendes, P., Kummer, U.: Copasi – a complex pathway simulator. BioInformatics **22** (2006) 3067–3074
15. Law, C.F., Gwee, B.H., Chang, J.: Fast and memory-efficient invariant computation of ordinary petri nets. IEE Proceedings: Computers and Digital Techniques **1** (2007) 612–624
16. Roovers, K., Assoian, R.K.: Integrating the MAP kinase signal into the G1 phase cell cycle machinery. BioEssays **22** (2000) 818–826
17. Levchenko, A., Bruck, J., Sternberg, P.W.: Scaffold proteins may biphasically affect the levels of mitogen-activated protein kinase signaling and reduce its threshold properties. PNAS **97** (2000) 5818–5823
18. Calzone, L., Gelay, A., Zinovyev, A., Radvanyi, F., Barillot, E.: A comprehensive imodular map of molecular interactions in RB/E2F pathway. Molecular Systems Biology **4** (2008)
19. Kohn, K.W.: Molecular interaction map of the mammalian cell cycle control and DNA repair systems. Molecular Biology of the Cell **10** (1999) 2703–2734

A hybrid approach mixing local search and constraint programming applied to the protein structure prediction problem

Raffele Cipriano[1], Alessandro Dal Palù[2], and Agostino Dovier[1]

[1] Univ. di Udine, Dip. di Matematica e Informatica. (cipriano|dovier)@dimi.uniud.it
[2] Univ. di Parma, Dip. di Matematica, alessandro.dalpalu@unipr.it

Abstract. We present a hybrid system that combines local search techniques and constraint solving. We apply it to the ab-initio protein structure prediction problem, modeled in the Face Centered Cubic Lattice with a pairwise contact energy function. In the literature, the problem is successfully solved using constraint programming for proteins with length up to 160 using the HP energy model. In the case of more complex models, w.r.t. energy and structure, current techniques can not be easily extended and the constraint approach is not applicable to proteins of length over 100. The idea described in this paper is based on the alternation of CSP solving phases and local search phases that modify the predicted spatial conformation. The approach is implemented and tested in Gecode and EasyLocal with encouraging results.

1 Introduction

The protein structure prediction problem is recognized to be a challenging problem for computational biology. Even with strong approximations of the spatial model (simple discrete lattices) and of the energy model (simple contact energy function), the problem is proved to be NP-hard. Nevertheless, minimizing simple hydrophobic-polar energy function and using the discrete lattice model FCC (Face Centered Cube), Backofen and Will solve it in seconds for proteins of length 160 and more [1]. Moreover, other researchers (e.g. [9]) approximated the solution to the same problem using local search and refined meta-heuristics.

More complex models have been proposed for the protein structure prediction problem. In [3] the problem have been formalized in the FCC lattice using a 20x20 energy matrix (different contributions for each pair of amino acids) and using information from secondary structure (known and/or predicted presence of α-helices and β-strands). The original implementation in SICStus Prolog CLP(FD) evolved in various directions (e.g., [4]) and an ad-hoc constraint solver on lattices (COLA) has been developed [5]. However, this approach is computationally infeasible when applied to the prediction of protein structures with more than hundred amino acids. Only the presence of other kind of partial information (e.g., known folds for sub-blocks picked from the protein data bank) can speed up significantly the search.

Extended models do not translate into an easy extension of the core computation idea used in [1]. This becomes unapplicable since the presence of different kinds of

contacts generates an explosion of the number of possible cores. Moreover, the definition of optimal core does not account for any complex structural constraints (e.g. secondary structure). Any admissible conformation containing further structural constraints often can only be obtained from a suboptimal core, namely a set of contacts less packed in the space. Since the number of such cores is exponential in the number of cavities in the volume, it is infeasible to precompute them in advance.

In this paper we would like to mix constraint-based and local-search techniques to improve the performance of the above mentioned (FCC 20x20) constraint-based tools. This hybrid system combines local search techniques and constraint solving. During the computation we consider the notion of *conformation*, which is a protein representation mapped to the spatial domain (i.e., FCC). Each conformation represents a possible state of a protein and it is associated to a particular energy, directly derived from the application of the pairwise 20x20 energy function. Each conformation may be constrained to other structural properties (see [3] for a complete list).

The presence of secondary structure information, obtained through neural network prediction, is necessary in order to predict more realistic conformations. In particular, it is shown that the contact energy function is not sufficient to reproduce local arrangements such as helices and/or sheets. The secondary structure information compensates the roughness of the energy model in use. Another advantage from using secondary structure constraints is that the search space reduces, since rigid blocks with no internal degree of freedom are imposed in the conformation.

The idea is to alternate CSP solving phases to local search phases. In the former, given a conformation as input, a CSP is built in order to search a spatially close conformation which respects every structural constraint (e.g., two amino acids may not overlap).

In the latter phase, the conformation is altered by means of a set of moves, which rotate part of the protein using a specific amino acid as pivot of the rotation. The part of the protein rotated is weakly allowed to change shape, in order to satisfy the overall conditions (i.e. the block is not kept fully rigid).

The platform is implemented and tested in Gecode and EasyLocal. Gecode is a recent C++ constraint solving platform with excellent performances [7], while EasyLocal++ is an object oriented, general and configurable, local search tool [6]. We compared the pure constraint programming approach and the one that combines constraint programming and local search on 12 proteins with different length and structure: even without developing particular combination strategies, the conformations found by the hybrid method improve those found with the pure CP approach.

2 Modeling PF in Gecode

As first test, we encoded in GECODE the same model presented in [3], using some enhanced representations for rigid substructures like helices and sheets [4]. We briefly summarize here the essential aspects of the encoding, which is based on the schema presented in [2]. The interested reader can refer to the just cited references.

The *Primary* structure of a protein is a sequence $s = s_1 \dots s_n$, where each s_i is an amino acid identified by a letter of an alphabet $\mathcal{A}$, $|\mathcal{A}| = 20$. The 3D conformation

of the protein is named *Tertiary* structure. *Tertiary* structures often contain *Secondary Structure elements* (e.g., α-helices and β-sheets).

We model the protein on the FCC lattice, namely, each amino acid i occupies a position $\omega(i)$ in the lattice. FCC points are points $\langle x, y, z\rangle \in \mathbb{N}^3$ such that $x + y + z$ is even. Two FCC points $\langle x_1, y_1, z_1\rangle$ and $\langle x_2, y_2, z_2\rangle$ are

- contiguous (or `next`) iff $|x_1 - x_2| \leq 1, |y_1 - y_2| \leq 1, |z_1 - z_2| \leq 1, |x_1 - x_2| + |y_1 - y_2| + |z_1 - z_2| = 2$.
- in `contact` iff they are not contiguous *and* $|x_1 - x_2| + |y_1 - y_2| + |z_1 - z_2| = 2$.

The choice of using the FCC lattice has been often adopted in thermodynamical studies of stability of small proteins (e.g. [10]) and this lattice is able to represent with a certain degree of accuracy the typical backbone angles and the shape of secondary structures. A *folding* of s is a function $\omega : \{1, \ldots, n\} \rightarrow \mathcal{D}$ such that:

1. $\texttt{next}(\omega(i), \omega(i + 1))$ for $i = 1, \ldots, n - 1$, and
2. $\omega(i) \neq \omega(j)$ for $i \neq j$ (namely, ω introduces no loops).

The second property is encoded using the well known `alldifferent` constraint, after a conversion from 3D coordinates to 1D FD variables. The conversion is based on an enumeration of the 3D lattice, using a relation of the kind $V = xM^2 + yM + z$, where M is a sufficiently large number. As shown in [5], using distinct FD variables for each coordinate hampers the propagators effectiveness and thus the `alldifferent` constraint has a limited effect. However, we based this approach on FD variables, instead of 3D box domains an in [5], in order to ease the interaction with GECODE. Let `Pot` be a 20x20 matrix associating an energy contribution measure to each pair of amino acids types s_i and s_j. The contribution is accounted for when $\omega(i)$ and $\omega(j)$ are in contact.

The *protein structure prediction problem* can be modeled as the problem of finding the folding ω of S such that the following energy cost function is minimized:

$$E(\omega, S) = \sum_{1 \leq i < n} \sum_{i+2 \leq j \leq n} \texttt{contact}(\omega(i), \omega(j)) \cdot \texttt{Pot}(s_i, s_j).$$

Let us observe that in the FCC each point is adjacent to 12 neighboring points. However, as explained in [3], we add some extra constraints (e.g. `angles`) that restrict to 90° and 120° the bend angles between three consecutive amino acids.

`secondary_info` constraints encode the Secondary Structure information in the program. The secondary structure is described by a list of elements of the type:

`helix`(i, j)**:** $s_i, s_{i+1}, \ldots, s_j$ form an α-helix. The modeling of α-helices builds on the observation that it is sufficient to constrain the first 4 amino acids of the helix to guarantee its shape—the shape can then be propagated to the rest of the helix via simple vector equalities [4].

`strand`(i, j)**:** $s_i, s_{i+1}, \ldots, s_j$ are in a β-strand. Similar to the case above.

`ssbond`(i, j)**:** presence of a disulfide bridge between s_i and s_j. If $\langle x_1, y_1, z_1\rangle$ and $\langle x_2, y_2, z_2\rangle$ are the variables for the positions of the two amino acids, then we set the constraints $|x_1 - x_2| \leq 4, |y_1 - y_2| \leq 4, |z_1 - z_2| \leq 4$.

We summarized each protein main features in a file format. Every protein is described by: its ID (according to the name used in the protein data bank), the sequence S of amino acids and its secondary structure information (if any).

We run some tests in order to compare the ability of FD solvers to handle the CSP described. We removed on purpose every heuristics and search optimizations described in [4, 5] and we noticed that with the same search parameters Gecode outperforms the running time of the equivalent SICStus Prolog code by a rather constant speedup. This search as well as the hybrid approach produce an output file that can be handled by standard molecular viewers. Complete code is available at www.dimi.uniud.it/dovier/PF/LS.

3 Local Search Moves

In this section we describe the Local Search perturbations that form the second phase in the hybrid technique. The local modifications of a conformation are defined by a set of moves that maps a conformation into another one.

3.1 The pivot move

A convenient move we studied is the *pivot move*, which is proved to be ergodic [8]. The idea of this class of moves is to keep unchanged the first part of the protein (for example the first half) and to rotate the second one. The second part should be rotated in the FCC space as a rigid block while looking for a better associated energy cost. A pivot move is identified by:

- the *pivot* amino acid (s_i), the last amino acid of the part that remains unchanged;
- a *firstfixed* amino acid (s_j), that identifies the rotating part of the protein (thus $i < j$). Below we explain why we require $i + 1 < j$.
- some *rigid block constraints*, that constrain the position of the amino acids of the moving part of the protein (from s_j to s_n).

A good move is influenced by the selection of the *pivot*. In particular is preferable to select an amino acid not involved in α-helices and β-strands: in fact such amino acids are in the middle of a well-structured section of the protein that must not be modified (e.g. it is not possible to break apart a helix).

The firstfixed amino acid identifies a section of the the protein (between s_i and s_j) completely free to move in the FCC lattice (only structural constraints are active, e.g. `next`). A firstfixed amino acid too close to the *pivot* (e.g. $s_j = s_i + 1$) limits the possible rotations of the rigid part of the protein, due to the non overlap constraints and to the poor degree of freedom of the subsequence between s_i and s_j. As this subsequence is enlarged, the possible accommodations of the subsequent rigid block increase exponentially. However, a firstfixed too far from the *pivot* causes the exploration of the huge search space for the subsequence $s_i \dots s_j$.

Lastly, the rigid block constraints must be selected carefully. They are a set of distance constraints between all pairs of amino acids in the rotating block as in the input conformation. The constraints can be relaxed (e.g. distances within a range w.r.t. the original distance, reduced number of pairs) and this case allows multiple solutions

which are spatially close to the original conformation. Once again, the degree of relaxation influences the search space and thus the solution times. On one hand, an exact rigid block set of constraints reproduces exactly the block, but, once rotated, it is not tolerant to local modifications of the block to avoid some overlaps to the first part of the protein. On the other hand the complete absence of rigid block constraints (the second part of the protein is totally free to move in the lattice) causes an inefficient search for the next conformation and every information about the second part of the protein is lost. We have experimentally chosen an intermediate approach, where *some constraints* between the amino-acids in the rigid block are added (obtaining a semi-rigid block). Observe that in this way we naturally mix local search and constraint based search.

We performed various preliminary test, to identify the better combination of these parameters. We decided to select as *pivot* only the amino acids *not* involved in α-helices or β-strands, to select as the firstfixed amino acid the fifth one after the *pivot* (i.e. $s_j = s_i + 5$) and to post distance constraint on the rigid block only between the amino acids of distance six in the primary structure (reduced number of pairs).

3.2 Pivot move implementation

To implement the pivot move we start from a conformation p of the protein encoded into a Gecode object (a Gecode::Space object). We create a new Gecode::Space object representing a protein *nextp*, where we post all the spatial and structural constraint (FCC lattice, `next`, `no loops`, `angles` and `secondary_info` constraints). Then we copy the amino acids $s_1 \dots s_i$ from p to $nextp$; the amino acids from *pivot* + 1 to firstfixed- 1 of *nextp* ($s_{i+1} \dots s_{j-1}$) are only subject to structural constraint; then we post the rigid block constraints on the amino acids of *nextp* from firstfixed to the last one ($s_j \dots s_n$).

Once the new protein object is created and all these constraints are posted, the Gecode search routine is launched for *nextp*. This CP search explores all the possible conformations (with respect to the constraints posted), trying to reach a folding with a better energy cost than the one of *p*. If the search finds such a folding, we iterate the process, starting from the conformation of the protein reached in *nextp*.

3.3 The Local Search algorithm

We inserted the implementation of the pivot move into a basic local search algorithm, using the functionalities provided by the framework EasyLocal++. The main idea of the algorithm is to start from an admissible conformation obtained as the first solution of the constraint programming search, then to randomly select a pivot move and to search a new conformation with better energy, according to the selected move using constraint programming search.

The CSP search (both for the first solution and for the pivot move) invokes a labeling with a *leftmost* variable selection and *median* value selection. We investigated various labeling options and observed experimentally that these ones better fit our problem.

The search on a local move has a timeout, that we call *moveTimeout* (we used a value of 1 minute); if a new conformation with a better energy is discovered before

moveTimeout, it is accepted and it becomes the current one. A *globalTimeout* is selected at the beginning of the execution, so the algorithm iterates until it reaches the *globalTimeout*. At the end, the best solution reached is returned. The iteration of the process depicts a classical hill-climbing algorithm.

When selecting the move, the choice of *pivot* amino acid determines firstfixed amino acid and rigid block constraints). As said above the *pivot* amino acid is randomly selected only among all the amino acids of the protein not involved in α-helices and/or β-strands. During the random moves exploration, we ensure that the same move is not tested many times. We need to keep track of the moves already tested, in order to skip the candidate moves in the history.

Once every move have been tested and no move produces an improvement in the energy cost, the *moveTimeout* is multiplied by a constant *Inc* (we used *Inc*=2) and the process is iterated.

4 Results

After preliminary tests performed with the aim of tuning the various parameters (firstfixed, rigid block constraints, timeouts, search strategies), we tested our hybrid algorithm on 12 proteins of different length and with different secondary structures (the same used in [5]). For each protein we executed 1 run with the pure CP algorithm and 5 runs with the hybrid approach: the selection of a pivot move is randomly guided, so different runs may lead to different solutions and we report results on the best run obtained. In Table 1 we report for each protein the search time in minutes and the energy cost (Ecost) of the best conformation found.

Tests have been performed on a AMD Opteron 280 at 2.2GHz, Linux CentOS machine with a *globalTimeout* of 2 hours. We compare the energy cost of the solution found with the pure constraint programming approach and with the hybrid constraint programming-local search algorithm. The energy costs do not account for the contribution of the internal contacts of the secondary structures, since they are constant during the search process, and thus these results are not comparable to the ones of [3, 4, 5].

We can notice that the energy costs found with the hybrid approach are generally better than the ones found with the only use of constraint programming. This confirms our hypothesis that the hybrid approach leads to better solutions in the same amount of time. With some small proteins the hybrid algorithm stops improving after few minutes: in this case, it falls into a local minimum and neither increasing the *moveTimeout* nor trying different runs can avoid this problem. On the other hand, with some proteins the pure CP search stops finding better solutions in few minutes, because the search space to explore is too big and the search diverges. It must be noticed that the energy values obtained for longer proteins are not yet satisfactory.

Work needs to be done in the local search stage: in fact we noticed that some simple and useful rotations between contiguous secondary structures are not performed; such rotations are probably forbidden by the blocks overlap (the rigidity should be be further relaxed) and by an insufficient number of free amino acids between two contiguous secondary structures.

Protein		CP		CP + LS		Protein		CP		CP + LS	
ID	Length	Time	Ecost	Time	Ecost	ID	Length	Time	Ecost	Time	Ecost
1EDP	17	15	-12279	1	-12140	2IGD	60	70	-7583	87	-16631
1E0N	27	39	-7619	5	-12742	1SN1	62	4	-20764	59	-28853
1VII	36	16	-9194	1	-14402	1L6T	78	107	-23883	52	-7117
2GP8	40	53	-12472	20	-13501	1HS7	96	30	-5797	1	-2067
1ED0	45	7	-10917	8	-14747	1TQG	104	49	-8384	117	-15333
1ENH	54	63	-8928	59	-12386	1SA8	105	67	-14219	120	-26443

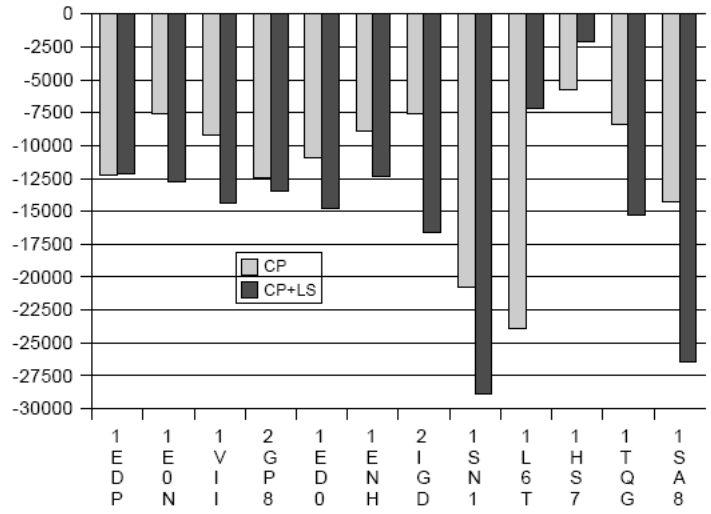


Table 1. Comparison of the solutions obtained by the two approaches on 12 different proteins. Timings are expressed in minutes.

5 Future Work and Conclusions

This is an ongoing work. Our aim was to prove that on the PF problem on FCC lattice with 20x20 energy matrix, the hybrid use of local search and constraint programming outperforms the only use of constraint programming, in terms of quality of solutions and execution time. We first encoded a basic PF model on FCC with 20x20 energy matrix into the constraint programming framework Gecode, without including strong search heuristics (like the ones used in [4, 5]); then we defined a local search move (the pivot move) in the local search framework EasyLocal++, in such a way that the pivot move can interact with the constraint programming model. Our tests confirm that the hybridization of these techniques leads to better solutions with respect to the pure constraint programming model.

Now that we have ensured the feasibility of this idea and the goodness of the results, additional tests and algorithm improvements can be performed. We plan to run the algorithm on other longer and more complex proteins, to refine the parameters with

massive testing, if needed. We can also try to run our algorithm with a longer globalTimeout. Various ideas can be applied to the hybrid algorithm. For example, we plan to embed into the constraint programming model some already tested heuristics (the ones used in [4, 5]): this should improve the performance when searching for a new conformation, and thus speed up the search.

We can refine the local search strategy: the hill-climbing algorithm is very efficient, but it is a local algorithm; the use of more refined strategies (such as tabu search) could avoid falls into local minima. We also think to elaborate more complex local search heuristics and metaheuristics, derivable from the EasyLocal++ framework.

We can speed-up the performance using the COLA solver [5], a constraint solver in C specifically designed for the Protein folding problem: embedding local search routines directly into COLA, instead of using Gecode and EasyLocal++ should outperform the execution time of the present current approach.

Acknowledgements The work is partially supported by MUR FIRB RBNE03B8KK and PRIN projects. We thank Luca Di Gaspero and Andrea Formisano for the useful discussions and the help in installing packages.

References

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

[2] P. Clote and R. Backofen. *Computational Molecular Biology: An Introduction*. John Wiley & Sons, 2001.

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

[4] A. Dal Palù, A. Dovier, and E. Pontelli. Heuristics, optimizations, and parallelism for protein structure prediction in CLP(FD). Proc. of PPDP 2005: 230-241, 2005.

[5] A. Dal Palù, A. Dovier, and E. Pontelli. A constraint solver for discrete lattices, its parallelization, and application to protein structure prediction. *Software Practice and Experience*, DOI: 10.1002/spe.810, 2007.

[6] L. Di Gaspero and A. Schaerf. EasyLocal++: an object-oriented framework for the flexible design of local-search algorithms. *Software Practice and Experience*, 33(8):733–765, 2003.

[7] Gecode Team. Gecode: Generic Constraint Development Environment. *Available from http://www.gecode.org*, 2006.

[8] N. Madras and A. D. Sokal, The Pivot Algorithm: A Highly Efficient Monte Carlo Method for the Self-Avoiding Walk. *Journal of Statistical Physics*, 50:109–186, 1988.

[9] A. Shmygelska and H. H. Hoos. An ant colony optimisation algorithm for the 2D and 3D hydrophobic polar protein folding problem. *BMC Bioinformatics* 6(30), 2005.

[10] J. Skolnick and A. Kolinski. Reduced models of proteins and their applications. *Polymer*, 45:511–524, 2004.

A Pseudo-Boolean Solution to the Maximum Quartet Consistency Problem *

António Morgado and João Marques-Silva

School of Electronics and Computer Science, University of Southampton, UK
ajrm@soton.ac.uk,jpms@ecs.soton.ac.uk

Abstract. Determining the evolutionary history of a given biological data is an important task in biological sciences. Given a set of quartet topologies over a set of taxa, the *Maximum Quartet Consistency* (MQC) problem consists of computing a global phylogeny that satisfies the maximum number of quartets. A number of solutions have been proposed for the MQC problem, including Dynamic Programming, Constraint Programming, and more recently Answer Set Programming (ASP). ASP is currently the most efficient approach for optimally solving the MQC problem. This paper proposes encoding the MQC problem with pseudo-Boolean (PB) constraints. The use of PB allows solving the MQC problem with efficient PB solvers, and also allows considering different modeling approaches for the MQC problem. Initial results are promising, and suggest that PB can be an effective alternative for solving the MQC problem.

1 Introduction

The amount of existing biological data (DNA and protein sequences) has increased the need for larger and faster determination of evolutionary history (or *phylogeny*) given a set of taxa (i.e. a set of related biological species [2]). Moreover, the availability of data is not always the same for different taxa. This is known as the data disparity problem [11,12]. In recent years, quartet based methods have received greater attention from the computational biology community as a way to overcome the data disparity problem. Quartet-based methods are characterized by first inferring a set of evolutionary relationships between four taxa, and then from these relationships assemble a global evolutionary tree. Considering only four taxa in the first step to build the evolutionary relationships, leads to a greater confidence on the relationships produced. Nevertheless, the relationships obtained may be conflicting or even missing. The aim of this work is to obtain the evolutionary tree, under the parsimony assumption, that respects the maximum number of these relationships on four taxa.

Given a set of quartet topologies over a set of taxa, the *Maximum Quartet Consistency* (MQC) problem consists of computing a global phylogeny that satisfies the maximum number of quartets. A number of solutions have been proposed for the MQC problem, including Dynamic Programming, Constraint Programming, and more recently Answer Set Programming (ASP) [11,9,10]. ASP is currently the most efficient approach for optimally solving the MQC problem. This paper develops an encoding for the MQC problem with pseudo-Boolean (PB) constraints. Initial results are promising, and suggest that PB can be an effective alternative for solving the MQC problem.

* This work is partially supported by the European Scholarship Program of Microsoft Research.

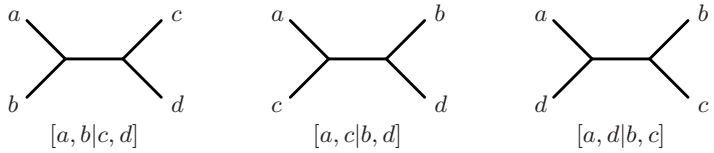


Fig. 1. Graphical representation of the quartet topologies $[a, b|c, d]$, $[a, c|b, d]$ and $[a, d|b, c]$.

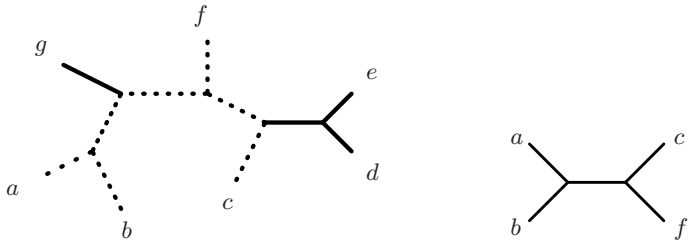


Fig. 2. Graphical representation of a phylogeny and of the quartet topology for the quartet $\{a, b, c, f\}$ derived from the phylogeny.

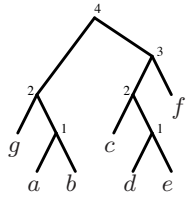
The paper is organized as follows. The first section introduces both the MQC problem and the MQI problem. The following section develops a Pseudo Boolean Optimization (PBO) model for the MQC problem and Section 4 proposes three optimizations to the PBO model. Section 5 shows the experimental results obtained and Section 6 presents some conclusions and points some directions for future research.

2 Preliminaries

A *phylogeny* is an unrooted tree whose leaves are bijectively mapped to a given set of taxa S, where each internal node has degree three. A *quartet* is a size four subset of S. For each quartet there exist three different possible phylogenies, called *quartet topologies*. Consider the quartet $\{a, b, c, d\}$, the three possible quartet topologies will be denoted by $[a, b|c, d]$, $[a, c|b, d]$ and $[a, d|b, c]$. Figure 1 gives a graphical representation of the three possible quartet topologies for the quartet $\{a, b, c, d\}$. For example, quartet topology $[a, b|c, d]$ means that the path that connects a and b does not intersect the path connecting c and d.

Given a phylogeny T on S and a quartet $q = \{a, b, c, d\}$, a quartet topology qt is said to be the quartet topology of q derived from T, if qt is the topology obtained from T, by removing all the edges and nodes not in the paths connecting the leaves that are mapped to taxa in q. Figure 2 represents a phylogeny, and the quartet topology derived from the phylogeny for the quartet $\{a, b, c, f\}$. The dotted branches show the path connecting the taxa in the quartet. Since the path that connects a and b does not intersect the path that connects c and f, then the derived quartet topology is $[a, b|c, f]$.

The set of quartet topologies derived from a phylogeny T is denoted by Q_T. If a quartet topology q is the same as the quartet topology derived from T, then T is said to



	a	b	c	d	e	f	g
a	0	1	4	4	4	4	2
b		0	4	4	4	4	2
c			0	2	2	3	4
d				0	1	3	4
e					0	3	4
f						0	4
g							0

Fig. 3. Graphical representation of a rooted phylogeny and the associated ultrametric matrix.

satisfy q and q is said to be *consistent* with T. In the example of Figure 2, $[a, b|c, f]$ is consistent with the phylogeny shown, but $[a, c|f, g]$ is not.

Given a set of quartet topologies Q on the set of taxa $S = \{s_1, \ldots, s_n\}$, if there exists a phylogeny T that satisfies all the quartet topologies in Q, then Q is said *compatible*. In practice the quartet topologies in Q may be inaccurate or even missing. If the set Q contains a quartet topology for each possible quartet of S, then Q is *complete* otherwise *incomplete*.

The problem of *Maximum Quartet Consistency (MQC)* is the problem where a set of quartet topologies Q on a set of taxa $S = \{s_1, \ldots, s_n\}$ is given, and returns a phylogeny T on S, that satisfies the maximum number of quartet topologies of Q.

The MQC problem is NP-hard [1] and if Q is complete, then MQC admits a polynomial-time approximation scheme [5]. If Q is incomplete, then MQC is MAX SNP-hard [5]. The dual problem to the MQC is the problem of *Minimum Quartet Inconsistency* (MQI). The MQI problem is the problem that given a set of quartet topologies Q (as in the MQC problem), returns a phylogeny that minimizes the number of quartet errors, where the set of quartet errors is the set $Q - Q_T$. The rest of the paper assumes that the set of quartet topologies Q is complete. In the recent past, different approaches have been reviewed in the literature for both the MQC and MQI problems. A detailed review is presented in [10].

3 Pseudo Boolean Model for the MQC Problem

This section develops a Pseudo Boolean Optimization(PBO) model for solving the MQC problem. The idea of the model is to obtain a rooted phylogeny, from which it is possible to construct an unrooted phylogeny [6]. Similarly to the existing ASP solution [10], the PBO model encodes the constraints of representing the rooted phylogeny tree as an ultrametric matrix. Moreover, an ultrametric phylogeny satisfies the maximum number of quartets topologies of a set Q if and only if the corresponding ultrametric matrix M satisfies the maximum number of quartets topologies in Q [10].

Consider the set of taxa $S = \{s_1, \ldots, s_n\}$ and a set of quartets Q. An *ultrametric* matrix M is a symmetric square matrix $n \times n$, where for each i such that $1 \leq i \leq n$ then $M(i, i) = 0$, for each i, j such that $1 \leq i < j \leq n$ then $1 \leq M(i, j) = M(j, i) \leq n$,

and for each triple of indices i, j, k such that $1 \leq i, j, l \leq n$, there is a tie between the maximum value of $M(i, j)$, $M(i, l)$ and $M(j, l)$.

The values in the ultrametric matrix M, represent the lowest common ancestor in the rooted phylogeny, that is the value of $M(i, j)$ corresponds to the internal node of the phylogeny that is the lowest common ancestor between taxa i and j. Figure 3 presents a rooted phylogeny, where the internal nodes have been labeled. The labels correspond to integers in decreasing order from the root to the leaves. On the right side of the figure is represented half of the associated ultrametric matrix. In [4] it is explored the relationship between rooted phylogenies and ultrametric matrixes and presents an algorithm to obtain a rooted phylogeny from the associated ultrametric matrix in polynomial time.

It was proven in [10] that in order to obtain an optimal phylogeny, the values of the entries of M can be restricted to $1 \leq M(i, j) \leq \lceil \frac{n}{2} \rceil$. To encode the values of $M(i, j)$ the PBO model introduces a set of Boolean variables $M_{i,j,k}$ where $1 \leq i < j \leq n$ and $1 \leq k \leq \lceil \frac{n}{2} \rceil$. $M_{i,j,k}$ has value 1 iff $M(i, j) = k$, otherwise $M_{i,j,k}$ is 0. To ensure that, for each pair (i, j), one and only one of the variables $M_{i,j,k}$ is selected to be true, the model introduces the following constraint:

$$\sum_{k=1}^{\lceil \frac{n}{2} \rceil} M_{i,j,k} = 1 \tag{1}$$

The value of each $M(i, j)$ variable is given by $M(i, j) = \sum_{k=1}^{\lceil \frac{n}{2} \rceil} k \times M_{i,j,k}$.

To ensure that the resulting matrix M is ultrametric, one of the following three conditions must be satisfied, for each $1 \leq i < j < l \leq n$:

$$M(i, j) = M(i, l) \land M(i, l) > M(j, l), \text{or} \tag{2}$$

$$M(i, j) = M(j, l) \land M(j, l) > M(i, l), \text{or} \tag{3}$$

$$M(j, l) = M(i, l) \land M(i, l) > M(i, j) \tag{4}$$

The PBO model associates three new Boolean variables $c1_{i,j,l}$, $c2_{i,j,l}$, $c3_{i,j,l}$ with constraints (2), (3) and (4), respectively. Each of the variables $cx_{i,j,l}$ is true iff the associated constraint is satisfied.

Constraint (2) is the logical AND of an equality constraint and a greater than constraint. In the PBO model each of these constraints is associated with additional Boolean variables, respectively, $c1^1_{i,j,l}$ and $c1^2_{i,j,l}$. $c1^1_{i,j,l} = 1$ iff $M(i, j) = M(i, l)$, and can be implemented with a comparator circuit on the unary representation of $M(i, j)$ and $M(i, l)$, using variables $M_{i,j,k}$ and $M_{i,l,k}$. $c1^2_{i,j,l} = 1$ iff $M(i, l) = M(j, l)$, and can also be implemented with a comparator circuit on the unary representation of $M(i, l)$ and $M(j, l)$, using variables $M_{i,l,k}$ and $M_{j,l,k}$. As a result, $c1_{i,j,l}$ is defined as:

$$c1_{i,j,l} = AND(c1^1_{i,j,l}, c1^2_{i,j,l}) \tag{5}$$

Variables $c2_{i,j,l}$ and $c3_{i,j,l}$ are encoded similarly. Finally to guarantee that one of the conditions (2), (3) or (4) is satisfied, the PBO model uses the following constraint:

$$c1_{i,j} + c2_{i,j} + c3_{i,j} \geq 1 \tag{6}$$

As the objective is to compute the phylogeny that maximizes the number of quartets that can be satisfied, then with each quartet is associated with a Boolean variable q_t,

where $1 \leq t \leq |Q|$. q_t will be true if quartet number t is consistent, otherwise q_t is false. A quartet $[i, j|l, m]$ is consistent if and only if one of the following conditions is satisfied [10]:

$$M(i,l) > M(i,j) \wedge M(j,m) > M(i,j), \text{or} \tag{7}$$

$$M(i,l) > M(l,m) \wedge M(j,m) > M(l,m) \tag{8}$$

Suppose that quartet number t is the quartet $[i, j|l, m]$. The model associates two new variables to each of the conditions (7) and (8). Let $d1_{i,j,l,m}$ be associated with condition (7) and $d2_{i,j,l,m}$ be associated with condition (8). The associated variable q_t is encoded as a gate OR:

$$q_t = OR(d1_{i,j,l,m}, d2_{i,j,l,m}) \tag{9}$$

Both the conditions (7), (8) consist of logical ANDs of two greater than conditions. Thus variable $d1_{i,j,l,m}$ and $d2_{i,j,l,m}$ are encoded as gates AND in a analogous way to variables $c1_{i,j,l}$.

The cost function of the PBO model is then to maximize the number of quartets that are consistent, that is:

$$\max : \sum_{t=1}^{|Q|} q_t \tag{10}$$

4 Optimizations to the PBO Model

This section describes three optimizations to the basic PBO model. The first optimization aims reusing auxiliary variables that serve for encoding of some of the circuits associated with the PBO model. The second optimization is related with the Boolean variables used for representing the value of each entry in the ultrametric matrix. The third optimization sets the values for some of $M(i, j)$ variables when it is known that s_i and s_j are siblings.

4.1 First Optimization

The objective of the first optimization is to reduce the number of variables used in the encoding. The reduction is achieved by exploiting the information provided by the auxiliary variables used for encoding cardinality constraints. In order to implement this optimization, sequential counters [8] are used. The uniqueness constraint (1) of the PBO model in Section 3 is split into two constraints. The first constraint deals with the need to have one at least one variable selected by adding the constraint:

$$\sum_{k=1}^{\lceil \frac{n}{2} \rceil} M_{i,j,k} \geq 1 \tag{11}$$

The second constraint is:

$$\sum_{k=1}^{\lceil \frac{n}{2} \rceil} M_{i,j,k} \leq 1 \tag{12}$$

and is encoded in CNF with a sequencial counter [8]. This sequential counter introduces variables $s_{k,1}$. These variables have the property that if $M_{i,j,a} = 1$ then for $1 \leq k < a$ all variables have $s_{k,1} = 0$ and for $a \leq k \leq \lceil \frac{n}{2} \rceil$ then $s_{k,1} = 1$. The property enables the encoding of $M(i,j) < M(l,m)$ by considering the associated variables $s_{k,1}$ of $M(i,j)$ and of $M(l,m)$. In order to better understand, let the variables $s_{k,1}$ associated to the sequential counter of $M(i,j)$ be denoted by $s_k^{i,j}$. The objective is to encode that $M(i,j) < M(l,m)$ by re-using the variables $s_k^{i,j}$ and $s_k^{l,m}$. Using the above property, this can be done by searching for the k where $s_k^{i,j} = 1$ and $s_k^{l,m} = 0$, which can be encoded in a variable $e_k^{(i,j)(l,m)}$ as a gate AND:

$$e_k^{(i,j)(l,m)} = AND(s_k^{i,j}, NOT(s_k^{l,m})) \tag{13}$$

Then variable $LT_{i,j,l,m}$ encodes that $M(i,j) < M(l,m)$ by a gate OR:

$$LT_{i,j,l,m} = OR(e_k^{(i,j)(l,m)} : 1 \leq k \leq \lceil \frac{n}{2} \rceil) \tag{14}$$

For this optimization, all the other constraints of the PBO model of Section 3 are maintained, but making use of the variables $LT_{i,j,l,m}$ as appropriate.

4.2 Second Optimization

For the PBO model described in Section 3, for each pair of taxa (i,j), the values of the variables $M(i,j)$ are encoded through selection variables $M_{i,j,k}$ where $1 \leq k \leq \lceil \frac{n}{2} \rceil$.

The first optimization described here replaces the encoding of the selection variables. Variables $M_{i,j,k}$ are still going to be used to encode $M(i,j)$, but here $M_{i,j,k}$ represents the $k-$th bit of the binary representation of $M(i,j)$. Now k is limited by $0 \leq k \leq \lfloor \log_2(\lceil \frac{n}{2} \rceil) \rfloor$. With this encoding $M(i,j)$ can be obtained by $M(i,j) = \sum_{k=0}^{\lfloor \log_2(\lceil \frac{n}{2} \rceil) \rfloor} 2^k \times M_{i,j,k}$. Moreover, the constraints used in the encoding need to be modified. The constraints in Equation (1) that encode the uniqueness of the selection variables are no longer used. All the other constraints are maintained, but with the new limit for variable k. Instead of the uniqueness constraints, this optimization requires that the encoded variables $M(i,j)$ are restricted to $\{1, \ldots, \lceil \frac{n}{2} \rceil\}$, that is $1 \leq M(i,j)$ and $M(i,j) \leq \lceil \frac{n}{2} \rceil$. The first part is obtained by adding the constraint:

$$\sum_{k=0}^{\lfloor \log_2(\lceil \frac{n}{2} \rceil) \rfloor} M_{i,j,k} \geq 1 \tag{15}$$

For the second part, a new Boolean variable $ltb_{i,j}$ is introduced, that captures the condition that $M(i,j)$ is not larger than $\lceil \frac{n}{2} \rceil$. The variables $M_{i,j,k}$ are used to representing this constraint as a comparator circuit.

In order to ensure that $ltb_{i,j}$ is true, the following constraint is added to the model:

$$ltb_{i,j} \geq 1 \tag{16}$$

4.3 Third Optimization

The optimization described in this section follows [11,9,10]. The objective of this optimization is to previously determine the value of some variables, namely when a pair of taxa is know to be siblings. The optimization can be used independently of the model (or optimization) used.

Let $S = \{s_1, \ldots, s_n\}$ be a set of taxa and Q be a complete set of quartets . A *Bipartition* of S is a pair (X, Y) of nonempty subsets of S, such that $S = X \cup Y$ and $X \cap Y = \emptyset$. Consider a bipartition (X, Y) of S, such that $|X| \geq 2$ and $|Y| \geq 2$, let $Q_{(X,Y)}$ be defined as $Q_{(X,Y)} = \{[x_1, x_2|y_1, y_2] : x_i \in X \wedge y_i \in Y \text{ for } i \in \{1, 2\}\}$. Suppose that three taxa from Y are fixed and also that $|X| = l$. An l-*subset* with respect to (X, Y) is the set of l quartets from Q that contain the three fixed taxa from Y and one taxa from X. There are a total of $\binom{n-l}{3}$ of l-subsets.

An l-subset is said to be *exchangeable* on X, if by ignoring the difference of the taxa from X on the quartets in the l-subset, it produces a unique quartet topology, otherwise the l-subset is said to be *nonexchangeable*. In the case where $l = 2$, then both taxa in X are said to be siblings and the following corollary holds:

Proposition 1 (Corollary 2.5 from [10]). *Let $S = \{s_1, \ldots, s_n\}$ be a set of taxa, Q be a complete set of quartets on taxa S. For the pair of taxa (s_i, s_j) from S, let $p_1 = |Q_{(\{s_i,s_j\},Y)} - Q|$, p_2 be the number of nonexchangeable pairs on $\{s_i, s_j\}$. If $2p_1 + p_2 \leq n - 3$ then s_i, s_J are siblings in an optimal phylogeny.*

In the optimization described in this section, for every pair of taxa, the condition of the corollary is tested. When the condition is true, for example for taxa i and j, then the PBO model is augmented with the following constraints:

$$M_{i,j,1} \geq 1 \tag{17}$$

$$-1 \times M_{i,j,k} \geq 0 \quad , k \in \{2, \ldots, upperLimit\} \tag{18}$$

The $upperLimit$ in Equation (18) is dependent on the encoding of variable $M_{i,j,k}$ (either as described in Section 3 or as described in Section 4.2).

5 Experimental Results

This section presents experimental results comparing the PBO model proposed in Section 3 and the ASP model described in [10]. The instances considered were obtained from [10]. These instances correspond to quartet topologies derived from random generated trees with a percentage of quartet topologies randomly altered. The percentage of altered quartet topologies introduces errors in the quartet topologies. Higher percentage of altered quartet topologies means a higher possibility of errors in the quartet topologies of the instance.

In the experiments four models were considered, three obtained from the PBO formulation and one from the ASP formulation. The first PBO model considers the first optimization described in Section 4.1 and will be referred as *PBO+fst*. The second PBO model includes both the optimizations of Section 4.2 and Section 4.3. This second model will be referred as *PBO+(scd+trd)*. The last PBO model, called *PBO+trd*,

	N. Variables			N. Constraints		
% Altered	PBO+fst	PBO+(scd+trd)	PBO+trd	PBO+fst	PBO+(scd+trd)	PBO+trd
01	5760	4514.4	6276.6	19890	16238.8	24464
05	5760	4537.2	6310.8	19890	16301.5	24568.5
10	5760	4566.4	6354.4	19890	16385.2	24708
15	5760	4587.6	6386.4	19890	16448.8	24814
20	5760	4611.2	6421.8	19890	16519.6	24932
25	5760	4628.4	6447.6	19890	16571.2	25018
30	5760	4648.4	6477.6	19890	16631.2	25118

Table 1. Average number of variables and number of constraints for instances with 10 taxa.

	CPU Time			
% Altered	phy+SModels	PBO+fst	PBO+(scd+trd)	PBO+trd
01	0.0464	0.7696	0.4704	0.7316
05	0.3048	2.2673	1.686	7.0885
10	1.3264	5.7819	5.8872	28.8291
15	2.4324	12.7119	11.78235	52.6487
20	9.0915	32.2536	17.78277	68.77968
25	28.4901	60.7041	28.0254	117.6832
30	65.4176	121.3564	52.75086	239.2057

Table 2. Average CPU time in seconds for instances with 10 taxa.

includes only the third proposed optimization (Section 4.3). In all the PBO models an encoder was implemented that receives as input the quartet topologies and returns as output a file in PB format. The generated file was then given as input to the PBO solver. For all experiments the PBO solver used was *minisat+* [3].

The fourth model is the ASP model described in [10]. The `phy` program, that encodes the quartet topologies into answer set programming, was obtained from [10]. The instances were given to `phy`, and for each, the parameters given were the number of taxa involved and the maximum number of quartet errors known in the instance. This last parameter was set as the number of quartet topologies in the instance. After obtaining the encoded instance, the encoded file was given to the ASP-solver SModels [7] SModels was configured to obtain all the stable models in order to maximize the number of quartets satisfied.

The results were obtained on an Intel Xeon 5160, 3GHz server, with 4 GB of RAM. The results comparing the average number of variables and number of constraints between the three PBO models is shown in Table 1. As can be seen from the table the model that requires more variables and more constraints is the PBO+trd model, whereas, the model that requires less variables and less constraints is the PBO+(scd+trd).

Table 2 compares the average CPU times on the instances considered for all the PBO models and the phy+Smodes model.

A few conclusions can be drawn from the results. First comparing the *PBO+fst* and the basic *PBO+trd* model. The sharing of auxiliary variables introduced by the first optimization is an important aspect in this problem. This optimization reduces the number of variables used by the encoding as well as the number of constraints. This reduction

leads to lower CPU time spent by the PBO-solver. Nevertheless, model *PBO+(scd+trd)* reduces even further the model by considering the selection variables as bits of the binary representation of values in M. Again, it can be seen from Table 2, that the reduction on the number of variables and constraints used by the encoding resulted in lower CPU times spent by the PBO-solver, where the model *PBO+(scd+trd)* is on average approximately 4 times faster than the *PBO+trd* and 1.6 times faster than *PBO+fst*.

Comparing the best of our PBO models (*PBO+(scd+trd)*) with the ASP model, the ASP model is more effective when the percentage of modified quartets is small, but the *PBO+(scd+trd)* model becomes more when the percentage of modified quartets increases.

6 Conclusions

This paper proposes a first attempt at solving the MQC problem with PBO. The new PBO model is compared with a recent solution based on ASP [10], which is currently the most efficient for the MQC problem. Despite the number of the taxa considered being modest, the results show that the PBO model can be beneficial when the number of expected quartet errors is high. The PBO model is still recent, and additional modeling insights and corresponding performance improvements are to be expected in the near future.

Future research will involve developing optimizations to the PBO model. For example, by encoding with PB constraints some of the optimizations proposed in the literature for the MQC problem. Furthermore, experiments will consider larger sets of taxa as well as real world data.

References

1. V. Berry, T. Jiang, P. Kearney, M. Li, and T. Wareham. Quartet cleaning: Improved algorithms and simulations. *Proceedings of the Seventh European Symposium on Algorithms (ESA99), Lecture Notes in Computer Science*, 1643:313–324, 1999.
2. B. Chor. From quartets to phylogenetic trees. In *Conference on Current Trends in Theory and Practice of Informatics*, pages 36–53, 1998.
3. N. Een and N. Sorensson. Translating pseudo-boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, 2:1–26, 2006.
4. D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, January 1997.
5. T. Jiang, P. Kearney, and M. Li. Orchestrating quartets: approximation and data correction. *Proceedings of the 39th IEEE Symposium on Foundations of Computer Science*, pages 416–425, 1998.
6. D. Pelleg. *Algorithms for Constructing Phylogenies from Quartets*. PhD thesis, Masters thesis, Israel Institute of Technology, 1998.
7. P. Simons. Computing the stable model semantics.
8. C. Sinz. Towards an optimal cnf encoding of boolean cardinality constraints. In *Principles and Practice of Constraint Programming - CP 2005, 11th International Conference*, volume 3709, pages 827–831, 2005.
9. G. Wu, G. Lin, J.-H. You, and X. Wu. Faster solution to the maximum quartet consistency problem with constraint programming. *Proceedings of 3rd Asia-Pacific Bioinformatics Conference*, pages 329–338, 2005.

10. G. Wu, J. You, and G. Lin. Quartet-based phylogeny reconstruction with answer set programming. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 4(1):139–152, 2007.
11. G. Wu, J.-H. You, and G. Lin. A lookahead branch-and-bound algorithm for the maximum quartet consistency problem. *Algorithms in Bioinformatics: 5th International Workshop, WABI 2005, Mallorca, Spain, October 3-6, 2005: Proceedings*, 2005.
12. G. Wu, J.-H. You, and G. Lin. A polynomial time algorithm for the minimum quartet inconsistency problem with O(n) quartet errors. *Information Processing Letters*, 100(Issue 4):167–171, 2006.

Generic ILP *vs* Specialized 0-1 ILP for Haplotype Inference

Ana Graça[1], Inês Lynce[1], João Marques-Silva[2], and Arlindo L. Oliveira[1]

[1] IST/INESC-ID, Technical University of Lisbon, Portugal
{assg,ines}@sat.inesc-id.pt,aml@inesc-id.pt
[2] School of Electronics and Computer Science, University of Southampton, UK
jpms@ecs.soton.ac.uk

Abstract. Haplotype inference is an important and computationally challenging problem in genetics. A well-known approach to haplotype inference is pure parsimony (HIPP). Despite being based on a simple optimization criterion, HIPP is a computationally hard problem. Recent work has shown that approaches based on Boolean satisfiability namely pseudo-Boolean optimization (PBO), are very effective at tackling the HIPP problem. Extensive work on PBO-based HIPP approaches has been recently developed. Considering that the PBO problem, also known as 0-1 ILP problem, is a particular case of the integer linear programming (ILP) problem, generic ILP solvers can be considered. This paper compares the performance of PBO and ILP solvers on a variety of HIPP models. We conclude that specialized PBO solvers are more suitable than generic ILP solvers.

1 Introduction

Understanding genetic differences between human beings is a crucial step towards the diagnosis and prevention of genetic diseases. Haplotype inference is a key problem to solve for achieving this goal, since haplotypes include most of the information about human genetic variations. A well-known haplotype inference approach is the pure parsimony (HIPP) which, among the possible solutions, chooses the one with the smallest number of distinct haplotypes [7].

Former work on the HIPP problem was mainly based on integer linear programming (ILP) [7,2,3]. Afterwards, Boolean satisfiability (SAT) [9] and pseudo-Boolean optimization (PBO) [5] have been used to solve the problem. Recently, PBO HIPP-based approaches have been improved, generating further reduced models [6]. Considering that PBO is a particular case of ILP, existing PBO models can also be solved by generic ILP solvers.

This work compares the performance of different HIPP models described in the literature [2,5,6], using different PBO solvers [11,10,4,1] and the generic ILP solver CPLEX. To the best of our knowledge, such a comparison has never been made in the past. This paper aims at performing a comprehensible evaluation of different models and solvers. The analysis of the experimental results gives insights to select the most appropriate modelling techniques depending on the kind of solver being used.

The paper is organized as follows. The next section describes the HIPP problem. Section 3 details recent PBO HIPP models, namely RPoly [5] and the recent improvements to the model [6]. Afterwards, on section 4, experimental results comparing ILP

and PBO approaches to the HIPP problem are presented. Finally, the paper concludes in section 5.

2 Haplotype Inference by Pure Parsimony

Single nucleotide polymorphisms (SNPs) correspond to sites in the DNA sequence where mutations have occurred and which represent a significant variability on the population. Haplotypes can be seen as a sequence of SNPs highly correlated, within a single chromosome. It is technically difficult to obtain haplotypes directly. Instead, genotypes, which correspond to the mixed data of two haplotypes on homologous chromosomes, are experimentally obtained. The haplotype inference problem consists in finding the set of haplotypes which originated a given set of genotypes.

Considering that mutations are rare, we may assume that each SNP can only have two values. Each haplotype is therefore represented by a binary string with size $m \in \mathbb{N}$, where 0 represents the wild type and 1 represents the mutant type. Each site of the haplotype h_i is represented by h_{ij} ($1 \leq j \leq m$). Each genotype is represented by a string, with size m, over the alphabet $\{0, 1, 2\}$, and each site of the genotype g_i is represented by g_{ij}. Each genotype is explained by two haplotypes. A genotype $g_i \in \mathcal{G}$ is explained by a pair of haplotypes (h_i^a,h_i^b) such that

$$g_{ij} = \begin{cases} h_{ij}^a \text{ if } h_{ij}^a = h_{ij}^b \\ 2 \quad \text{if } h_{ij}^a \neq h_{ij}^b \end{cases}, \tag{1}$$

with $1 \leq j \leq m$. A genotype site g_{ij} with either value 0 or 1 is a homozygous site, whereas a site with value 2 is a heterozygous site.

Definition 1. *Given a set $\mathcal{G}$ of n genotypes each with size m, the haplotype inference problem consists in finding a set of haplotypes $\mathcal{H}$, such that each genotype $g_i \in \mathcal{G}$ is explained by two haplotypes $h_i^a, h_i^b \in \mathcal{H}$.*

For each genotype g with k heterozygous sites, there are 2^{k-1} pairs of haplotypes that can explain g. For example, genotype $g_i = 202$ can be explained either by haplotypes (000,101) or by haplotypes (001,100). Several approaches to the haplotype inference problem have been suggested. Given that individuals from the same population share many haplotypes, the pure parsimony approach searches for a solution with the smallest number of distinct haplotypes.

Definition 2. *The haplotype inference by pure parsimony (HIPP) problem consists in finding a solution to the haplotype inference problem which minimizes the number of distinct haplotypes [7].*

Example 1. Consider the set of genotypes $\mathcal{G}$: $g_1 = 022$, $g_2 = 221$ and $g_3 = 222$. There are solutions using 6 different haplotypes $\mathcal{H}_1$: $h_1^a = 001, h_1^b = 010, h_2^a = 011$, $h_2^b = 101, h_3^a = 000$ and $h_3^b = 111$. However the HIPP solution only requires 4 distinct haplotypes $\mathcal{H}_2$: $h_1^a = 011, h_1^b = 000, h_2^a = 011, h_2^b = 101, h_3^a = 011$ and $h_3^b = 100$.

It has been shown that the HIPP problem is NP-hard [8].

3 ILP/PBO-based HIPP Models

With a few notable exceptions [12], early work on the HIPP problem used models based on integer linear programming [7,2,3]. The original ILP model, *RTIP* [7], has exponential space complexity on the number of heterozygous sites because, in the worst case, it requires the enumeration of all possible pairs of haplotypes that can explain each genotype. Afterwards, two polynomial ILP models, *PolyIP* [2] and *HybridIP* [3], were proposed [1].

Recently, a very competitive SAT-based approach, *SHIPs* [9], suggested an incremental algorithm that, starting from a clique-based lower bound on the number of required haplotypes, models the problem into SAT and searches for a HIPP solution. If no solution is found, the lower bound is incremented by one and a new SAT instance is generated. When a solution is found, the minimum number of haplotypes is given by the value of the lower bound. More recent approaches use pseudo-Boolean optimization models [5,6]. These models represent an improvement in terms of the efficiency of HIPP solvers.

The *Reduced Poly* model (RPoly) [5] proposed a number of simplifications to the Poly model. The RPoly model associates two haplotypes (h_i^a, h_i^b) with each genotype g_i, for $1 \leq i \leq n$. A variable t_{ij} is associated with each heterozygous site g_{ij}, such that $t_{ij} = 1$ if $h_{ij}^a = 1$ and $h_{ij}^b = 0$, whereas $t_{ij} = 0$ if $h_{ij}^a = 0$ and $h_{ij}^b = 1$.

Another key issue in the RPoly's formulation is the notion of incompatibility. Two genotypes are incompatible if they cannot be explained by a common haplotype, or equivalently, genotypes g_i and g_k, are incompatible if there exists j ($1 \leq j \leq m$) such that $g_{ij} + g_{kj} = 1$. Otherwise, they are said to be compatible. For candidate haplotypes h_i^p and h_k^q, with $p, q \in \{a, b\}$ and $1 \leq k < i \leq n$, a variable $x_{i\,k}^{p\,q}$ is defined, such that, $x_{i\,k}^{p\,q} = 1$ if haplotype h_i^p of genotype g_i and haplotype h_k^q of genotype g_k are different. If two genotypes are incompatible, then they cannot share an explaining haplotype, and consequently, for the four possible combinations of p and q, $x_{i\,k}^{p\,q} = 1$.

Finally, in order to count the number of distinct haplotypes used, variables u_i^p are defined such that $u_i^p = 1$ if haplotype h_i^p, which explains genotype g_i, is different from all the haplotypes which explain genotypes g_k, with $k < i$. The conditions on variables u_i^p are

$$\sum_{1 \leq k < i \,;\, q \in \{a,b\}} x_{i\,k}^{p\,q} - u_i^p \leq 2i - 3, \tag{2}$$

with $p \in \{a, b\}$ and $1 \leq i \leq n$. The objective function minimizes the sum of variables u_i^p.

An improved RPoly model was recently proposed [6]. This new model, *NRPoly*, integrates the SHIPs clique-based lower bound in the RPoly model and extends the model with additional constraints. The components of the SHIPs lower bound allow both fixing the value of some of the u_i^p variables and also avoiding generating the constraints involving fixed u_i^p variables [9]. Moreover, the order in which the genotypes are considered must reflect the order in which the genotypes are used in the lower bound [6]. In

[1] Throughout the paper we will remove the suffix *IP* and use only *Poly* and *Hybrid*.

practice, the integration of SHIP's lower bound allows fixing the value of many u_i^p variables and, as several constraints need not be generated, allows significantly reducing the size of the model.

The second optimization is related with a key simplification of the RTIP model, which consists in not considering pairs of haplotypes when both of them do not explain more than one genotype. Actually, if a genotype g_i is not incompatible with all other genotypes, then at least one of the haplotypes that explain g_i must explain other genotype. For each genotype $g_i \in \mathcal{G}$ compatible with at least one more genotype in $\mathcal{G}$, the following constraint is generated,

$$\sum_{k>i\,;\,p,q\in\{a,b\}\,;\,\kappa(k,i)} x_{k\,i}^{p\,q} + u_i^a + u_i^b \leq 4\mathcal{K} + 1, \tag{3}$$

where predicate $\kappa(k,i)$ is defined true if g_k and g_i are compatible and $\mathcal{K}$ is the cardinality of the set $\{g_k \in \mathcal{G} : k > i \wedge \kappa(k,i)\}$.

The last optimization consists in adding cardinality constraints on the values of variables x. For many combinatorial problems, adding cardinality constraints to the model can prune the search space, helping the solver to find a solution. Clearly, two different genotypes g_i and g_k cannot be explained by the same pair of haplotypes, and then g_i and g_k can be at most explained by one haplotype in common. Therefore, for each pair of distinct genotypes g_i and g_k ($k < i$), if g_i and g_k are compatible and non-homozygous, then

$$\sum_{p,q\in\{a,b\}} x_{i\,k}^{p\,q} \geq 3. \tag{4}$$

4 Generic ILP *vs* Specialized 0-1 ILP for the HIPP problem

In this section we compare the relative performance of discrete optimization HIPP models using different 0-1 ILP solvers and a generic ILP solver. A considerable number of HIPP models and solvers are evaluated. To the best of our knowledge, such comparation has never been performed so far.

4.1 Experimental Setup

An extensive evaluation, using 1183 problem instances [5] including real and synthetic data, has been performed. The solvers used were MiniSat+ [4], Pueblo [11] version 1.5, the latest version of BSOLO [10], PBS4 [1], glpPB release 0.2 and CPLEX version 11.0 (www.ilog.com/products/cplex/). The results were obtained on an Intel Xeon 5160 server (3.0GHz, 4MB RAM) running Red Hat Enterprise Linux WS 4. The timeout for each instance was set to 1000 seconds.

4.2 Results

The Poly, RPoly and NRPoly models were adapted to be run by the five different 0-1 ILP solvers (Minisat+, Pueblo, BSOLO, PBS4 and glpPB) and the generic ILP solver

Table 1. Number of instances aborted (out of 1183) for each model and solver (timeout 1000s)

	PBO solver					**ILP Solver**
Model	MiniSat+	Pueblo	BSOLO	PBS4	glpPB	CPLEX
Poly	82	251	486	605	1091	705
RPoly	36	127	290	326	723	234
NRPoly	18	55	120	108	611	249

Table 2. Number of instances aborted (out of 1183) using CPLEX, on each ILP model (timeout 1000s)

Model	RTIP	Poly	Hybrid	RPoly	NRPoly
# aborted	378	707	717	234	249

CPLEX. All solvers are then being evaluated on exactly the same models. Table 1 provides a summary of the results obtained, with the number of instances aborted (out of 1183) for each model.

Clearly, the best performing solver for each of the three models is MiniSat+. In general, MiniSat+, Pueblo, BSOLO and PBS4 solvers outperform CPLEX. The only exception is with the RPoly model, where CPLEX solves more instances than both BSOLO and PBS4.

For the results shown, the Poly model used is a re-implementation of the model described in [2]. The original Poly model gives similar results, aborting 707 instances instead of the 705 shown, using CPLEX. Even though the original Poly model was developed to be solved using CPLEX, the results suggest that most of the specialized 0-1 ILP solvers perform better for this model.

The glpPB solver is the worst performing solver for each of the three models (Poly, RPoly, NRPoly). glpPB is a ILP-based pseudo-Boolean solver, that uses the GNU linear programming kit (GLPK, www.gnu.org/software/glpk/). Hence, the glpPB ILP-based solver implements some of the techniques also used by CPLEX, but glpPB is not as optimized as CPLEX.

For all PBO solvers, NRPoly is shown to be more robust than the previous RPoly model. Solving the NRPoly model using MiniSat+, Pueblo, BSOLO or PBS4, reduces at least by 50% the number of instances not solved within 1000 seconds. Using the glpPB solver , the number of aborted instances is reduced in 15%. However, the generic ILP solver, CPLEX, does not benefit from the techniques introduced in the new model. Indeed, the NRPoly model aborts 15 instances more than the RPoly model, using CPLEX.

Table 2 summarizes the number of aborted instances for each model using CPLEX. For RTIP, Poly and Hybrid the same code used in [3], developed to be used with CPLEX, has been run. The RPoly and NRPoly models were adapted to be run by CPLEX. The number of instances aborted by the most recent models, RPoly and NRPoly, is much smaller than the number of instances aborted by the previous models, confirming that the new models are more robust. However, as already mentioned be-

Table 3. Number of instances aborted (out of 1183) by each version of NRPoly model using CPLEX

Model	RPoly	NRPoly *v1*	NRPoly *v2*	NRPoly
# aborted	234	258	257	249

fore, the most recent model, NRPoly, does not perform as well as the RPoly model, in contrast with PBO solvers.

In order to understand whether NRPoly performs worse than RPoly due to a particular feature of the NRPoly model, we analyzed the performance of NRPoly for each additional new technique included in this model. Table 3 presents the number of aborted instances for each NRPoly version. We call *NRPoly v1* to the version that only integrates the lower bound of SHIPs. *NRPoly v2* corresponds to the version with the lower bound of SHIPs and cardinality constraints on the x variables. The final version, that includes also the RTIP pruning, is simply *NRPoly*. As can be concluded, the integration of the lower bound of SHIPs is the reason why NRPoly performs worse than RPoly (24 more instances are aborted) when using CPLEX. In fact, both the integration of cardinality constraints and the RTIP pruning have been shown to help the CPLEX solver.

Finally, figure 1 provides a plot comparing RPoly using either MiniSat+ or CPLEX, and NRPoly using either MiniSat+ or CPLEX [2]. RPoly with MiniSat+ is more efficient than RPoly with CPLEX (36 vs. 234 aborted instances) and NRPoly with MiniSat+ is more efficient than NRPoly with CPLEX (18 vs. 249 aborted instances). The set of instances aborted using MiniSat+ is a subset of instances aborted by CPLEX. This result is not surprising given that Poly with MiniSat+ has been shown in the past to be more efficient than Poly with CPLEX [5]. However, taking into account that the two versions of Poly were not implemented by the same authors, this new comparison was deemed necessary.

5 Conclusions

This paper analyzes the performance of different generic and specialized ILP solvers on recently proposed HIPP models. Our experiments show that the SAT-based PBO solvers are, in general, more suitable than the state of the art generic ILP solver CPLEX. The experimental results confirm that the poor performance of CPLEX is a consequence of the ILP techniques used. Similar conclusions can be drawn for the ILP-based glpPB solver, which is the worst-performing ILP solver for the HIPP problem. glpPB uses some of the techniques used by CPLEX, but is significantly less optimized. Moreover, the results for CPLEX and glpPB suggest that similar results would be obtained in case a different ILP solver was considered.

Our conclusion is that, for the HIPP case and probably for other problems which can be naturally formulated as a 0-1 ILP problem, specific PBO solvers should be con-

[2] Each point in the plot corresponds to a problem instance, where the x-axis corresponds to the CPU time required by MiniSat+ and the y-axis corresponds to the CPU time required by CPLEX.

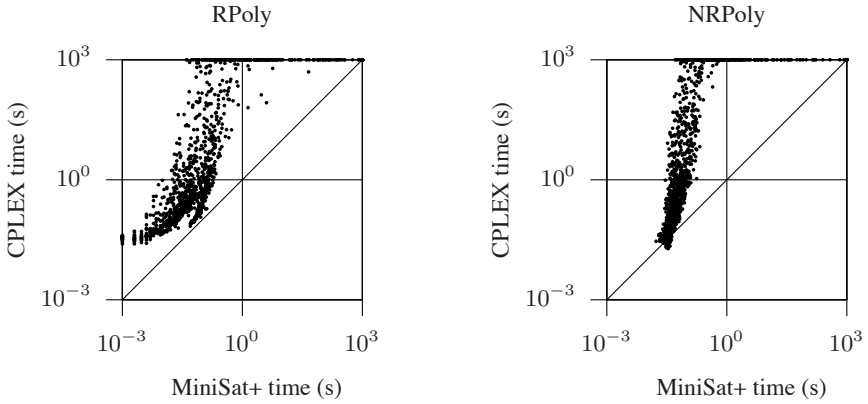


Fig. 1. CPU time results for RPoly and NRPoly

sidered. Furthermore, we observe that some modeling techniques used to optimize the PBO approaches do not produce improvements when the ILP solver CPLEX is used.

Acknowledgments

This work is partially supported by Fundação para a Ciência e Tecnologia under research projects SATPot (POSC/EIA/61852/2004) and SHIPs (PTDC/EIA/64164/2006) and PhD grant SFRH/BD/28599/2006, and by Microsoft under contract 2007-017 of the Microsoft Research PhD Scholarship Programme.

References

1. F. Aloul, A. Ramadi, I. Markov, and K. Sakallah. Generic ILP versus specialized 0-1 ILP: an update. In *Proc. IEEE/ACM International Conference on Computer-Aided Design*, pages 450–457, 2002.
2. D. Brown and I. Harrower. A new integer programming formulation for the pure parsimony problem in haplotype analysis. In *Workshop on Algorithms in Bioinformatics (WABI'04)*, pages 254–265, 2004.
3. D. Brown and I. Harrower. Integer programming approaches to haplotype inference by pure parsimony. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(2):141–154, 2006.
4. N. Eén and N. Sörensson. Translating pseudo-Boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, 2:1–26, 2006.
5. A. Graça, J. Marques-Silva, I. Lynce, and A. Oliveira. Efficient haplotype inference with pseudo-Boolean optimization. In *Algebraic Biology 2007 (AB'07)*, pages 125–139, 2007.
6. A. Graça, J. Marques-Silva, I. Lynce, and A. Oliveira. Efficient haplotype inference with combined CP and OR techniques (short paper). In *5th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Problems (CPAIOR'08)*, 2008. Accepted for publication.

7. D. Gusfield. Haplotype inference by pure parsimony. In *14th Annual Symposium on Combinatorial Pattern Matching (CPM'03)*, pages 144–155, 2003.
8. G. Lancia, C. M. Pinotti, and R. Rizzi. Haplotyping populations by pure parsimony: complexity of exact and approximation algorithms. *INFORMS Journal on Computing*, 16(4):348–359, 2004.
9. I. Lynce and J. Marques-Silva. Efficient haplotype inference with Boolean satisfiability. In *National Conference on Artificial Intelligence (AAAI)*, 2006.
10. V. Manquinho and J. Marques-Silva. Effective lower bounding techniques for pseudo-Boolean optimization. In *Design, Automation and Test in Europe Conference and Exhibition (DATE'05)*, pages 660–665, 2005.
11. H. M. Sheini and K. A. Sakallah. Pueblo: A hybrid pseudo-Boolean SAT solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 2:165–189, 2006.
12. L. Wang and Y. Xu. Haplotype inference by maximum parsimony. *Bioinformatics*, 19(14):1773–1780, 2003.

Constraints Programming for Unifying Gene Regulatory Networks Modeling Approaches

Damien Eveillard[1], Jonathan Fromentin[2] & Olivier Roux[2]

[1] Computational Biology (ComBi) group, LINA UMR 6241, CNRS & Université de Nantes
2, rue de la Houssinière - BP 92 208 - 44322 Nantes CEDEX 03
damien.eveillard@univ-nantes.fr
[2] IRCCyN UMR 6597, CNRS & École Centrale de Nantes
1, rue de la Noë - BP 92 101 - 44321 Nantes CEDEX 03
{jonathan.fromentin,olivier.roux}@irccyn.ec-nantes.fr

Abstract. Qualitative approaches, like Piecewise-Affine Differential Equations (PADEs) or those inspired from the R. Thomas formalism, represent one of the major recent improvements in biological modeling. We show herein that these approaches might be naturally represented within a unified theoretical framework using constraints. This result allows us to reason about biological models which is helpful for (i) passing from one qualitative formalism to another one and as well for (ii) building a constraints-based protocol that opens perspectives on modeling large genetic regulatory systems.

1 Introduction

Experimental approaches that study living system behaviors, focus on various and complementary aspects: (i) a set of genes that composes gene regulatory networks and (ii) a set of proteins that shapes metabolic networks. However, despite their clear experimental distinction, both components belong to the same system and interact between them for producing specific dynamical biological behaviors (see Fig. 1 for illustration). It hence remains interesting to mix up this distinct information through a unique modeling approach. It is achieved by various recent modeling techniques that focus on the dynamical biological behavior (see [1] for review) with a special emphasis on their qualitative behaviors. These approaches consider the gene interaction as the corner stone of an accurate macromolecular system modeling. Like this, each gene regulatory reaction summarizes a protein production that activates/represses the target gene. Among these modeling techniques, the approaches based on Piecewise-Affine Differential Equations (PADEs) [2, 3] and the R. Thomas formalism [4] showed astonishing achievements at investigating gene regulatory network properties and share as well common biological assumptions (i.e. discretizing the gene interaction impact). However, although these modeling techniques show at similar biological results, they focus on distinct theoretical features. We propose to present herein theoretical investigations that show that two modeling approaches might be

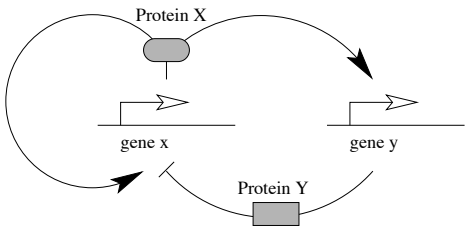


Fig. 1. Description of a two genes interaction network that resumes a system composed of genes x and y. The gene x produces the protein X that activates the transcription of genes x (i.e. auto-activation) and y. It implies a production of the protein Y that represses the transcription of the gene x.

unified within a unique framework using the constraints programming. It emphasizes a description of a novel biological modeling protocol that deals with assumptions used in either formalisms. It allows as well further investigations like a fine reasoning on constraints related to specific biological behaviors.

This paper will introduce first in Sec. 2 the unified constraints based framework. After a brief overview of the modeling approaches of interest (Sec. 2.1), we are going to show how to transform a PADEs model and a Thomas's model into a set of constraints (Sec. 2.2). This set of constraints describes the discrete dynamics that might be investigated for a better understanding of the biological behaviors. As a guideline, such a protocol will be illustrated on a simplistic system shown in Fig. 1. Second, Sec. 3 will present two kinds of analysis based on the previous constraints. In particular, for investigating large gene regulatory network, Sec. 3.1 will show a constraints based trimming approach that restricts the study of the model on the behaviors of interest (i.e. experimentally investigated genes). Such a refinement will allow a more precise reasoning using a symbolic model-checking (Sec. 3.2) that focuses on interesting behaviors for an experimental validation.

2 Constraints for Modeling Genetic Regulatory Systems

2.1 Qualitative Approaches

The regulation of genetic system is achieved via macromolecular interactions that describe positive and negative feedback loops. Qualitative approach appeared quickly as an appropriate way for investigating such a complexity. We mention herein the formalisms that have been successful during the last decade and that might be expressed in a natural manner by a set of constraints.

The Biological Regulatory Graph (BRG) is widely applied for a discrete modeling of gene regulatory networks like in Fig. 1. A BRG is a labelled directed graph $G = (V, E)$ where V is the set of vertices and E is the set of edges (see Fig. 2(a)). Each edge $(i \to j) \in E$ is labelled with a couple $(\alpha_{ij}, \theta_{ij})$ where $\alpha \in \{+, -\}$ is

the sign of interactions (respectively activation and repression) and θ_{ij} is the concentration threshold beyond which the regulation is effective.

Notation 1 *We note L_i, the set of labels related to the regulatory functions of the gene i that we call the resources of i.*

The System of Piecewise-Affine Differential Equations (PADEs) represents as well the dynamic of a genetic regulatory network [5, 6]. The system follows the form:

$$\dot{x}_i = f_i(x) - \gamma_i x_i \quad with \quad 0 \leq x_i \quad and \quad 1 \leq i \leq n \tag{1}$$

where $x = (x_1, \ldots, x_n)$ is a vector of protein concentrations called the quantitative state of the system. (1) describes the variation of the concentration x_i as the difference between the rate of synthesis $f_i(x)$ and the rate of degradation $\gamma_i x_i$. Note that $f_i(x)$ expresses the dependency between the synthesis rate of i and its regulator concentrations. It can be defined as:

$$f_i(x) = k_i + \sum_{j \in L_i} k_{ij} b_{ij}(x) \tag{2}$$

where $k_i, k_{ij} \in \mathbb{R}^{+*}$ are the kinetic parameters and, b_{ij} is a sigmoidal function approximated by a combination of step functions s^+ and s^- such as for a regulator gene i' of i, we have:

$$s^+(x_{i'}, \theta_{i'}) = \begin{cases} 1, & x_{i'} > \theta_{i'} \\ 0, & x_{i'} < \theta_{i'} \end{cases} \quad \text{et} \quad s^-(x_{i'}, \theta_{i'}) = 1 - s^+(x_{i'}, \theta_{i'}) \tag{3}$$

where θ_i is a concentration threshold. For illustration, Fig. 2(b) shows the PADE system that models the biological behavior of the system in Fig. 1.

The Discrete Modeling Formalism of R. Thomas (Fig. 2(c) for illustration) is a natural discrete description of the BRG shown above and represents as well a discretization of a PADE system. The Thomas's formalism have to take into account two kinds of parameters.

Numbering thresholds and discretization state. The thresholds numbering keeps the order between the qualitative thresholds mentioned above. Therefore, for the thresholds of i like $\theta_i^1 < \theta_i^2 < \cdots < \theta_i^n$ then its qualitative thresholds are $t_i^1 < t_i^2 < \cdots < t_i^n$ with $\forall j \in [1, n]$, $t_i^j = j$. The states of the system are thus discretized into domains by the function $\mathcal{D}$ defines as:

$$\mathcal{D}_i(x_i) = \begin{cases} t_i^j, & \theta_i^j < x_i < \theta_i^{j+1} \\ 0, & x_i < \theta_i^1 \end{cases} \tag{4}$$

Parameters in discrete modeling. To each qualitative domain s is associated a qualitative focal point standing for the tendency of evolution in s. For each qualitative domain s within the discrete abstraction, the vector of

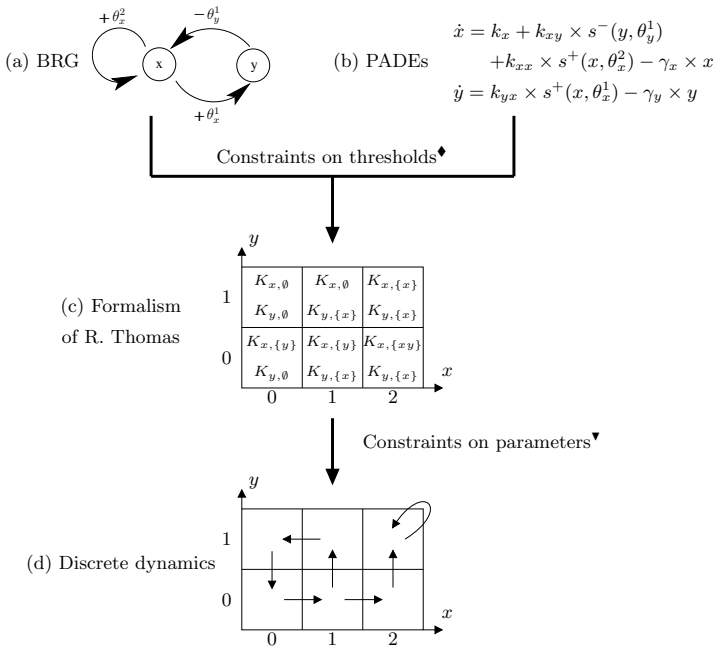


Fig. 2. Constraints-based protocol applied on the two genes system, where ♦ are $0 < \theta_x^1 < \theta_x^2 < max_x$ and $0 < \theta_y^1 < max_y$; ▼ are $0 < \frac{k_x}{\gamma_x} < \theta_x^1$ and $\theta_y^1 < \frac{k_{yx}}{\gamma_y} < max_y$ and $\theta_x^2 < \frac{k_x + k_{xy}}{\gamma_x}, \frac{k_x + k_{xx}}{\gamma_x}, \frac{k_x + k_{xy} + k_{xx}}{\gamma_x} < max_x$.

discrete parameters $(K_{1,\omega_1(s)}, \ldots, K_{n,\omega_n(s)})$ gives the position of the qualitative focal point. The focal point is the abstract region containing the steady state for the PADEs in each domain. The concentration evolution is continuous in a domain and the system state tends toward the focal point of the domain. The discrete parameter of the gene i in the domain s is obtained with $\dot{x}_i = 0$ in (1) and $b_{ij} = 1$ (due to the presence of the resource j) in (2):

$$K_{i,\omega_i(s)} = \mathcal{D}_i \left(\frac{k_i + \sum_{j \in \omega_i(s)} k_{ij}}{\gamma_i} \right) \tag{5}$$

where $\omega_i(s) \subseteq L_i$ represents the resources of i in the domain s. The valuation of these discrete parameters gives a discrete dynamics where each transition between two contiguous domains is asynchronous. For illustration, it gives the discrete dynamics shown in Fig. 2(d).

2.2 Using Constraints for Building a Discrete Dynamics Model

Based on previous descriptions, we propose to transform one formalism into another using an automatic approach that integrates the qualitative formalisms (see Fig. 2).

Transforming a BRG or a PADEs system into the Thomas's formalism (Fig. 2(a) and Fig. 2(b) to Fig. 2(c)) is achieved by reasoning on the knowledge associated with the thresholds. There can be simple equality or inequality constraints that allow the numbering of thresholds. These constraints are of the form $\theta_i = \theta'_i$ or $\theta_i < \theta'_i$ where i is a gene.

Transforming the R. Thomas formalism into discrete dynamics (Fig. 2(c) to Fig. 2(d)) is achieved by two distinct approaches.

Using the inequality constraints on the kinetic parameters like

$$\theta_i < \frac{k_i + \sum_{j\in\omega} k_{ij}}{\gamma_i} \quad \text{or} \quad \theta_i > \frac{k_i + \sum_{j\in\omega} k_{ij}}{\gamma_i} \tag{6}$$

where $\omega \subseteq L_i$, θ_i is a threshold of i and, where $\left(k_i + \sum_{j\in\omega} k_{ij}\right)/\gamma_i$ is a component of a focal point that gives the tendency of the evolution of i with the ressources ω. Both constraints are directly extracted from the PADEs formalism [7] and provide the discrete parameters values. The number of these constraints is proportional to the number of kinetic parameters. Both inequality constraints indicate the localization of the focal point within a domain. Therefore the number of these constraints is twice the number of components of focal points.

Using temporal qualitative specification when inequality constraints are difficult to obtain. Among studies that propose such an approach, two use the constraints programming. Both approaches chosen the use of *reified constraints*[1], because the formalism of R. Thomas produces graphs that might contain domains with multiple successors. The set of constraints are based on simple inequality constraints on the discrete parameters, which give the notion of successors. These constraints are usually not sufficient for depicting a unique discrete dynamic but give a set of possible discrete dynamics. In this purpose, F. Corblin *et al.* [8] uses constraint logic programming for analysis of GRN by knowing qualitative pathways or stable qualitative domains (with no out-going transitions). The number of constraints is a linear function of the number of qualitative domains in the pathway. And, the number of equations expressing a transition between qualitative domains is also a linear function of the number of component of the qualitative focal

[1] by adding boolean parameters such that the parameter is true iff the linked constraints are true

points. On the other hand, J. Fromentin *et al.* [9] uses constraint programming and the CTL language to find the discrete parameters. For example, we consider the CTL formula $x = 0 \Rightarrow EF(x = 1)$ in Fig. 2(c) to force the discrete dynamics in Fig. 2(d) in order to have a pathway from $x = 0$ to $x = 1$. For any operator $\diamond$, we associate a Constraint $C^{s_i}_{\diamond}$ at each discrete domain s_i. In addition for a CTL operator $\triangle$, we associate a boolean variable $B^{s_i}_{\triangle}$ at each discrete domain s_i. This boolean variable indicates if the related discrete domain validates or not the operator constraints. Therefore, the principle is to propagate the information given by the possible successors of the discrete domains *via* their reified constraints and their boolean variables. In this case, the reified constraints are more complex than those explained in [8] because they must be equivalent to those applied within the CTL formulae. Nevertheless, the basic constraints for the transitions are the same: similar equality or inequality constraints on the discrete parameters. For illustration, we consider the sub-graph of Fig. 2 (c) that includes two domains $s_1 = (0, 0)$ and $s_2 = (1, 0)$. The application of $x = 0 \Rightarrow EF(x = 1)$ on this sub-graph implies this following decomposition for s_1:

- $x = 0$ implies the constraint $C^{s_1}_{x=0} \equiv true$
- $x = 1$ implies $C^{s_1}_{x=1} \equiv false$
- $EF(x = 1)$ implies $C^{s_1}_{EF(x=1)} \equiv B^{s_1}_{EF(x=1)}$ for which
  $B^{s_1}_{EF(x=1)} \Leftrightarrow C^{s_1}_{x=1} \vee \left(B^{s_2}_{EF(x=1)} \wedge K_{x,\{y\}} > 0\right)$
- $x = 0 \Rightarrow EF(x = 1)$ implies $C^{s_1}_{x=0 \Rightarrow EF(x=1)} \equiv C^{s_1}_{x=0} \Rightarrow C^{s_1}_{EF(x=1)}$

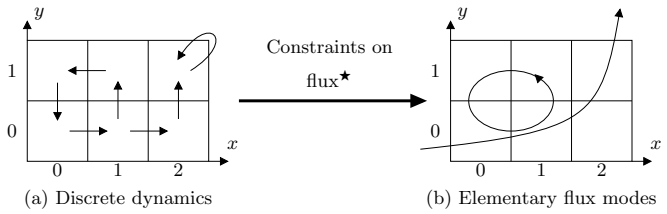
and this decomposition for the domain s_2:

- $x = 0$ implies the constraint $C^{s_2}_{x=0} \equiv false$
- $x = 1$ implies $C^{s_2}_{x=1} \equiv true$
- $EF(x = 1)$ implies $C^{s_2}_{EF(x=1)} \equiv B^{s_2}_{EF(x=1)}$ for which
  $B^{s_2}_{EF(x=1)} \Leftrightarrow C^{s_2}_{x=1} \vee \left(B^{s_1}_{EF(x=1)} \wedge K_{x,\{y\}} < 1\right)$
- $x = 0 \Rightarrow EF(x = 1)$ implies $C^{s_2}_{x=0 \Rightarrow EF(x=1)} \equiv C^{s_2}_{x=0} \Rightarrow C^{s_2}_{EF(x=1)}$

Note herein that the constraint that satisfies $x = 0 \Rightarrow EF(x = 1)$ in s_2 is a tautology whereras $K_{x,\{y\}} > 0$ (i.e. the transition $s_1 \rightarrow s_2$) have to be true for satisfying $C^{s_1}_{x=0 \Rightarrow EF(x=1)}$. Thus, and according to [9], the number of constraints is related to the number of domains and CTL operators.

3 Reasoning on the Biological Constraints Based Model

Biological knowledge is obviously – by nature – incomplete. Only specific behaviors related to genes of interest are experimentally investigated. On the other hand, biological models are often too large and/or complex for using standard constraints reasoning approaches. Among several solving or analysis techniques, we propose to use the constraints based framework for refining the model on distinct biological components, i.e. genes or gene products, in order (i) to validate a complex model based on specific behaviors, (ii) to emphasize behaviors that might be experimentally studied.



(a) Discrete dynamics (b) Elementary flux modes

Fig. 3. Reasoning on the discrete dynamics given by a result of the constraints-based protocol where ★ allow the flux balance analysis.

3.1 Qualitative Behaviors Trimming

Previously shown constraints mainly describe qualitative behaviors. However analyzing such behaviors remains difficult for large gene regulatory networks. The use of our theoretical framework overcomes in a natural manner this weakness by allowing us to trim qualitative behaviors using an additional constraints-based approach: the flux balance analysis using the Minimal Metabolic Behavior (MMB) technique [10] that is an elegant extension of the Elementary Flux Modes (EFM) approach (see [11] for flux balance analysis overview). This technique is already well-known for analyzing the metabolic flux of a balanced (steady-state) system. It decomposes the flux constraints into minimal elementary pathways. Combinations of these pathways describe multiple paths that material can follow through the system. For applying these techniques on discrete dynamics graphs, we assume (i) a specific qualitative behavior as a combination of qualitative pathways and (ii) the discrete abstraction depicts transient behaviors between stable domains which implies flux between initial and stable domains (or set of domains that produce stable dynamics). Mathematically, the constraints that describe the discrete dynamics graph have the form:

$$Sv = 0, v_i \geq 0, \text{ for } i \in Irr \tag{7}$$

where Irr is the set of the transitions (i.e. irreversible transitions), S is the $s \times m$ stoichiometric matrix of the discrete dynamical network, with s domains (rows) and m transitions (columns), and $v \in \mathbb{R}^m$ is the flux vector. As explained in [10], the set of all possible flux distribution through the discrete dynamics graph at steady state (i.e. all possible solutions of the constraints system described in (7)), defines a polyhedral cone, named the steady state flux cone.

$$C = \{v \in \mathbb{R}^m \mid Sv = 0, v_i \geq 0, i \in Irr\} \tag{8}$$

As illustration, we depict this flux analysis applied on the discrete dynamics model in Fig. 3(a) that represents the behaviors of the simplistic system in Fig. 1. It is obtained by adding an input transition to the domain $(0, 0)$: the initial domain. We consider as well the stable domain $(2, 1)$ as a natural output of the system that finally consists of nine domains and eight transitions (six

regular plus two added transitions). The steady state cone can be represented by two minimal proper faces (Fig. 3(b)) named MMB:

$$\text{MMB}_1 : \rightarrow (0,0) \rightarrow (1,0) \rightarrow (2,0) \rightarrow (2,1) \rightarrow$$
$$\text{MMB}_2 : (0,0) \rightarrow (1,0) \rightarrow (1,1) \rightarrow (0,1) \rightarrow (0,0)$$

where MMB_1 and MMB_2 show respectively a linear and a circular qualitative pathway that passes through the qualitative domains. Interestingly, these two pathways represent the two characteristic behaviors of the system. Note that the lineality space $\text{lin.space}(C) = \{v \in C \mid v_i = 0, i \in Irr\}$ has dimension 0 due to the absence of reversible transitions in the discrete dynamics graph. Such an approach is particularly helpful for trimming a large biological model. Indeed, focusing on a specific gene, we consider only the pathways that possess domains and transitions related to the gene investigated. A linear combination of these MMBs hence produces a subgraph that describes all qualitative behaviors of interest.

3.2 Symbolic Model-Checking

The constraint-based protocol shown above is a natural unified theoretical framework for the qualitative modeling approaches. Furthermore, it achieves to combine additional constraints-based techniques usually applied for analyzing metabolic networks. Beyond these qualitative applications, our framework provides the opportunity to extend the modeling towards quantitative aspects by adding delays on the discrete transitions, hence producing a hybrid model. Several studies were done for analyzing hybrid models of genetic regulatory networks: [3, 12–15]. The common assumption is to partition the qualitative domains. This partition provides a finer transition system such that the sign patterns of the derivatives of concentrations levels are preserved. The methods for partitioning differ according to the different works, and the aim for each one is to give raise to executions that have to be compared with the experimental knowledge. For this purpose, different kinds of *symbolic model checking* techniques are applied, i.e. verify biological temporal properties (e.g. CTL formulae, reachability). It is either classical model checking ([3, 13]) or timed model checking ([14]) or hybrid (parametric) model checking ([12, 15]), and properties are either chronological ([3, 13]) or chronometrical ([14, 15]). Our constraint-based protocol integrates a rather different approach [9] since it implements CTL-model checking algorithms by the mean of constraints programming, which reinforces our unified approach on biological system modeling.

4 Discussion

This study shows that PADEs and Thomas based approaches are convergent. Indeed, we emphasize that both formalisms might be expressed using constraints. In practice, our modeling approach allows to choose between both formalizations according to the experimental knowledge at disposal, i.e. known constraints on

kinetic or discrete parameters. Moreover, our unified framework achieves a novel hybrid description of biological systems that exploits the advantages of both formalisms.

As a natural extension of the biological problem formalization, our unified framework allows several constraints based analyses that focus on distinct goals. Comparing our formalization with different solving frameworks (CP, CSP or SAT solvers [16]) represents by itself an interesting investigation area. However, we consider that one of the advantages of our approach is to produce a well-formalized and/or biologically certified problem that might be suitable for further constraints based investigations.

Our protocol aims at reasoning on more realistic biological networks. As illustration, we applied it on the gene regulatory network of the carbon starvation response in E. coli formalized using a PADEs system (following the description given in [17]). Six genes compose the gene regulatory network that might be represented using 37 constraints (constraints on inequalities and thresholds used in the PADEs system). They produce a discrete dynamics graph with 912 qualitative domains. This problem is hence formalized with simple constraints. However, although the constraints formalization is a relatively easy task, the problem remains difficult to analyze with standard techniques due to the complexity of the discrete dynamics graphs. It confirms the interest of dedicated constraints based techniques for investigating the biological properties of the complete genes interaction networks.

Acknowledgements D.E. thanks Olivier Bernard for long-term discussions on the qualitative modeling approaches. The authors alsos thanks Jamil Ahmad for fruitful discussions during this work.

References

1. de Jong, H.: Modeling and simulation of genetic regulatory systems: a literature review. J Comput Biol **9**(1) (Jan 2002) 67–103
2. de Jong, H., Gouzé, J.L., Hernandez, C., Page, M., Sari, T., Geiselmann, J.: Qualitative simulation of genetic regulatory networks using piecewise-linear models. Bull Math Biol **66**(2) (Mar 2004) 301–40
3. Batt, G., Ropers, D., de Jong, H., Geiselmann, J., Mateescu, R., Page, M., Schneider, D.: Validation of qualitative models of genetic regulatory networks by model checking: analysis of the nutritional stress response in escherichia coli. Bioinformatics **21 Suppl 1** (Jun 2005) i19–28
4. Thomas, R., Thieffry, D., Kaufman, M.: Dynamical behaviour of biological regulatory networks–i. biological role of feedback loops and practical use of the concept of the loop-characteristic state. Bull Math Biol **57**(2) (Mar 1995) 247–76
5. Glass, L., Kauffman, S.: The logical analysis of continuous non linear biochemical control networks. J. Theor. Biol. **39**(1) (1973) 103–129

6. Snoussi, E.: Qualitative dynamics of a piecewise-linear differential equations : a discrete mapping approach. Dynamics and stability of Systems **4** (1989) 189–207
7. de Jong, H., Page, M., Hernandez, C., Geiselmann, J.: Qualitative simulation of genetic regulatory networks: Method and application. In: IJCAI. (2001) 67–73
8. Corblin, F., Fanchon, E., Trilling, L.: Modélisation de réseaux biologiques discrets en programmation logique par contraintes. Technique et science informatiques **26**(numéro spécial "Modélisation et simulation pour la post-génomique") (2007) 73
9. Fromentin, J., Comet, J., Gall, P.L., Roux, O.: Analysing gene regulatory networks by both constraint programming and model-checking. In: EMBC07, 29th IEEE EMBS Annual Int. Conf. (2007) 4595–4598
10. Larhlimi, A., Bockmayr, A.: A new approach to flux coupling analysis of metabolic networks. In Berthold, M.R., Glen, R., Fischer, I., eds.: CompLife. Volume LNBI 4216., Berlin Heidelberg, Springer-Verlag (2006) 205–215
11. Gagneur, J., Klamt, S.: Computation of elementary modes: a unifying framework and the new binary approach. BMC Bioinformatics **5** (Nov 2004) 175
12. Adélaïde, M., Sutre, G.: Parametric analysis and abstraction of genetic regulatory networks. In: Proc. 2nd Workshop on Concurrent Models in Molecular Biol. (BioCONCUR'04), London, UK, Aug. 2004. Electronic Notes in Theor. Comp. Sci., Elsevier (2004)
13. Bernot, G., Richard, A., Comet, J.P., Guespin-Michel, J.: Application of formal methods to biol. regulatory networks: Extending thomas' asynchronous logical approach with temporal logic. J. Theor. Biol. **229**(3) (2004) 339–347
14. Siebert, H., Bockmayr, A.: Incorporating time delays into the logical analysis of gene regulatory networks. In: Int. Conf. on Comput. Methods in Systems Biol. (CMSB'06), Trento, Italy. LNBI 4210, Springer (2006) 169–183
15. Ahmad, J., Bernot, G., Comet, J.P., Lime, D., Roux, O.: Hybrid modelling and dynamical analysis of gene regulatory networks with delays. ComPlexUs **3**(4) (2007) 231–251
16. Tamura, N., Taga, A., Kitagawa, S., Banbara, M.: Compiling finite linear csp into sat. In: Constraints Programming (CP). (2006)
17. Ropers, D., de Jong, H., Page, M., Schneider, D., Geiselmann, J.: Qualitative simulation of the carbon starvation response in escherichia coli. BioSystems **84**(2) (May 2006) 124–52

A Constraint-Based Approach to the Phase Problem in X-Ray Crystallography

Corinna Brinkmann and Alexander Bockmayr

Freie Universität Berlin, FB Mathematik und Informatik, Arnimallee 6, 14195 Berlin, Germany
brinkmann@mi.fu-berlin.de, bockmayr@mi.fu-berlin.de

Abstract. X-ray crystallography is one of the main methods to establish the three-dimensional structure of biological macromolecules. In an X-ray experiment, one can measure only the magnitudes of the complex Fourier coefficients of the electron density distribution under study, but not their phases. The problem of recovering the lost phases is called the phase problem. Building on earlier work by Lunin/Urzhumtsev/Bockmayr we describe a constraint-based approach to the phase problem. We introduce the mathematical foundations, derive a basic integer programming formulation, and discuss possible refinements by including additional constraints.

1 Introduction

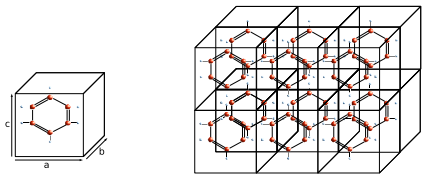
Knowledge about the three-dimensional structure of biological macromolecules is an essential foundation of structural biology and biotechnology. In X-ray crystallography the arrangement of atoms within a crystal is determined from a three-dimensional representation of the electron density. From X-ray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

Lunin, Urzhumtsev and Bockmayr [2] proposed a 0-1 integer programming approach to direct phasing. As a research contribution to a crystallographic journal, this article is not easily accessible. In the present paper, we introduce this work to the constraint programming community. We describe the mathematical foundations, derive step by

step the basic integer programming formulation, and discuss possible refinements by including additional constraints.

2 Basic terminology

Vectors, matrices as well as higher-dimensional arrays will be noted with bold letters, $\mathbf{x} \cdot \mathbf{y}$ denotes the scalar product of two vectors $\mathbf{x}$ and $\mathbf{y}$.
Every crystal consists of identical molecules, resp. complexes of molecules strictly ordered in all three dimensions. This means that we can find a parallelepiped containing such a complex of molecules which builds up the whole crystal if it is repeatedly stacked together in all three dimensions. This parallelepiped, in general, is not unique. It is defined by the length of its edges as well as the angles between them and is called *unit cell*. These base units, translated in three dimensions, build up a *crystal lattice*.
We will denote the unit cell's volume with V_{cell}. Let $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3 \in \mathbb{R}^3$ span the unit cell. Then we can write every vector $\mathbf{r} \in \mathbb{R}^3$ in this basis, i.e., $\mathbf{r} = x_1\mathbf{b}_1 + x_2\mathbf{b}_2 + x_3\mathbf{b}_3$, where $\mathbf{x} = (x_1, x_2, x_3)^T \in \mathbb{R}^3$ is the vector of coordinates of $\mathbf{r}$ with respect to the basis $\{\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3\}$.
The real function $\rho(\mathbf{r}),\ \mathbf{r} \in \mathbb{R}^3$, describing the electron density distribution in the crystal, has three linearly independent periods corresponding to the length of the unit cell's edges, i.e., $\rho(\mathbf{r}) = \rho(\mathbf{r} + k_1 \cdot \mathbf{b}_1 + k_2 \cdot \mathbf{b}_2 + k_3 \cdot \mathbf{b}_3), k_1, k_2, k_3 \in \mathbb{Z}$. This means, if we know the electron density distribution's values in the unit cell, we know its values in the whole crystal, due to periodicity.
For the vector of coordinates $\mathbf{x} = (x_1, x_2, x_3)$ the electron density function has integer periods in all three directions, i.e. $\rho(\mathbf{x}) = \rho(\mathbf{x}+\mathbf{k}), \forall \mathbf{x} \in V, \forall \mathbf{k} \in \mathbb{Z}^3$. A vector $\mathbf{r} \in \mathbb{R}^3$ is inside the unit cell iff $\mathbf{x} \in V = [0, 1)^3$.



(a) Unit cell (b) Crystal

Fig. 1: Unit cells building a crystal: a) a unit cell, b) crystal built of unit cells

3 The phase problem

We are searching for the electron density distribution $\rho(\mathbf{x})$ over the crystal. Due to the crystal structure, ρ is a periodic function and therefore can be developed into a *Fourier series* [1,4]

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

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

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

Since the structure factors are complex numbers, they can be written as

$$\mathbf{F}(\mathbf{h}) = F(\mathbf{h}) \exp(i\varphi(\mathbf{h})), \tag{3}$$

where $F(\mathbf{h}) = |\mathbf{F}(\mathbf{h})|$ is the *magnitude* and $\varphi(\mathbf{h}) \in [0, 2\pi[$ the *phase*.
The only experimental data we get in X-ray-crystallography are the reflection intensities. The intensity $I(\mathbf{h})$ of a reflection is proportional to the magnitude of the squared structure factors, with a known constant of proportionality, i.e., $C \cdot I(\mathbf{h}) = |\mathbf{F}(\mathbf{h})|^2, C \in \mathbb{R}$. Thus, all we can calculate from our experimental data are the structure factor magnitudes. The phase information is lost and must be restored by other means. This is called the *phase problem*.

4 The electron density distribution on a grid

4.1 Grid electron density

Instead of calculating the electron density distribution in the whole unit cell, we will work on a grid. Using discrete Fourier transforms we will then calculate electron densities at the grid points. The chosen division numbers along the unit-cell axes represent the resolution of the electron density map we are searching for.
Consider a grid $\mathit{\Pi} = [0, M_1 - 1] \times [0, M_2 - 1] \times [0, M_3 - 1] \subseteq \mathbb{Z}^3$, where $M = M_1 M_2 M_3$ is the total number of grid points. Let $\mathbf{M}$ be the diagonal matrix $\mathbf{M} = \text{diag}(M_1, M_2, M_3), M_1, M_2, M_3 \in \mathbb{N}$. Given the values $\mathbf{y_j}$ of a periodic function f on the grid points $\mathbf{j}$, i.e.,

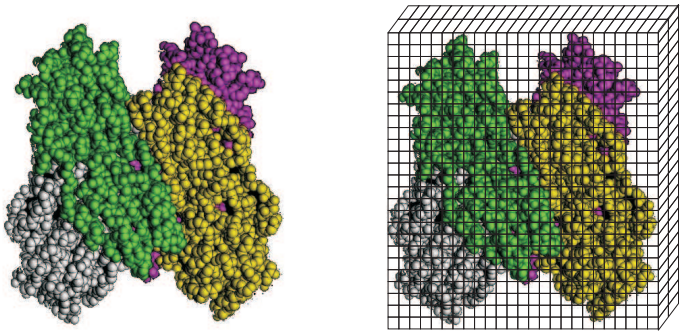
$$\mathbf{y_j} = f(\frac{j_1}{M_1}, \frac{j_2}{M_2}, \frac{j_3}{M_3}), \quad \mathbf{j} = (j_1, j_2, j_3) \in \mathit{\Pi} \tag{4}$$

the *three dimensional discrete Fourier transform* $\mathcal{F}$ calculates the Fourier coefficients of a trigonometric polynomial interpolating f in these grid points [6]:

$$\mathcal{F}(\mathbf{h}) = \frac{1}{M} \sum_{\mathbf{j} \in \mathit{\Pi}} \mathbf{y_j} \exp(2\pi i(\mathbf{h} \cdot \mathbf{M}^{-1}\mathbf{j})), \ \forall \mathbf{h} \in \mathit{\Pi}. \tag{5}$$

The values $\mathbf{y_j}, \mathbf{j} \in \mathit{\Pi}$, can be recovered from the Fourier coefficients $\mathcal{F}(\mathbf{h}), \mathbf{h} \in \mathit{\Pi}$, by the *inverse discrete Fourier transform*:

$$\mathbf{y_j} = \sum_{\mathbf{h} \in \mathit{\Pi}} \mathcal{F}(\mathbf{h}) \exp(-2\pi i(\mathbf{h} \cdot \mathbf{M}^{-1}\mathbf{j})), \ \forall \mathbf{j} \in \mathit{\Pi}. \tag{6}$$



(a) Protein (b) Discretisation

Fig. 2: Discretisation of a protein: a) Protein, b) Protein discretisation

The values of the electron density function $\rho(\mathbf{x}),\ \mathbf{x} \in V$ at the grid points are described by the *grid electron density function* $\rho_g(\mathbf{j}) = \rho(\mathbf{M}^{-1}\mathbf{j})$. We define the *grid structure factor* $\mathbf{F}_g(\mathbf{h})$ by the discrete Fourier transform

$$\mathbf{F}_g(\mathbf{h}) = \frac{1}{M} \sum_{\mathbf{j} \in \mathit{\Pi}} \rho_g(\mathbf{j}) \exp(2\pi i(\mathbf{h} \cdot \mathbf{M}^{-1}\mathbf{j})),\ \forall \mathbf{h} \in \mathit{\Pi}. \tag{7}$$

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

$$\rho_g(\mathbf{j}) = \sum_{\mathbf{h} \in \mathit{\Pi}} \mathbf{F}_g(\mathbf{h}) \exp(-2\pi i(\mathbf{h} \cdot \mathbf{M}^{-1}\mathbf{j})),\ \forall \mathbf{j} \in \mathit{\Pi}, \tag{8}$$

using the inverse discrete Fourier transform.

4.2 Structure factors vs. grid structure factors

In order to clarify the relation between the structure factors and the grid structure factors, we start with equation (7) and use (1), see also [5]:

$$\begin{aligned} V_{cell}\mathbf{F}_g(\mathbf{h}) &= \frac{V_{cell}}{M} \sum_{\mathbf{j} \in \mathit{\Pi}} \rho(\mathbf{M}^{-1}\mathbf{j}) \exp(2\pi i(\mathbf{h} \cdot \mathbf{M}^{-1}\mathbf{j})) \\ &= \frac{1}{M} \sum_{\mathbf{j} \in \mathit{\Pi}} \Big(\sum_{\mathbf{p} \in \mathbb{Z}^3} \mathbf{F}(\mathbf{p}) \exp(-2\pi i(\mathbf{p} \cdot (\mathbf{M}^{-1}\mathbf{j})))) \cdot \exp(2\pi i(\mathbf{h} \cdot \mathbf{M}^{-1}\mathbf{j})) \\ &= \frac{1}{M} \sum_{\mathbf{p} \in \mathbb{Z}^3} \mathbf{F}(\mathbf{p}) \sum_{\mathbf{j} \in \mathit{\Pi}} \exp(2\pi i((\mathbf{h} - \mathbf{p}) \cdot \mathbf{M}^{-1}\mathbf{j})) = \sum_{\mathbf{k} \in \mathbb{Z}^3} \mathbf{F}(\mathbf{h} + \mathbf{M}\mathbf{k}). \end{aligned}$$

The last equation holds due to

$$\sum_{\mathbf{j} \in \mathit{\Pi}} \exp\left(2\pi i\left((\mathbf{h} - \mathbf{p}) \cdot \mathbf{M}^{-1}\mathbf{j}\right)\right) = \begin{cases} \mathbf{M}, & \text{if } \mathbf{h} - \mathbf{p} = \mathbf{M}\mathbf{k},\ \text{for } \mathbf{k} \in \mathbb{Z}^3 \\ 0, & \text{otherwise.} \end{cases}$$

Introducing $R(\mathbf{h}) = \dfrac{M}{V_{cell}} \sum_{\mathbf{k}\in\mathbb{Z}^3\setminus\{0\}} \mathbf{F}(\mathbf{h}+\mathbf{M}\mathbf{k})$ we can write

$$\mathbf{F}_g(\mathbf{h}) = \frac{\mathbf{F}(\mathbf{h})}{V_{cell}} + \frac{R(\mathbf{h})}{M}. \tag{9}$$

The value of $R(\mathbf{h})$ depends on the magnitudes and phases of all structure factors and is generally unknown. But, it may be negligibly small if the grid is fine enough and if the indexes $\mathbf{h}$ are relatively small in comparison with the grid dimensions. Still, it may be significant if one of the indices is close to $\frac{M_1}{2}$, $\frac{M_2}{2}$ or $\frac{M_3}{2}$, cf. [2].

4.3 Inequalities for the grid electron density values

Using (7) we can obtain grid density values $\rho_g(\mathbf{j})$ from the grid structure factors $\mathbf{F}_g(\mathbf{h})$ by solving the system of equations

$$\mathbf{F}_g(\mathbf{h}) = \frac{1}{M} \sum_{\mathbf{j}\in\mathit{\Pi}} \rho_g(\mathbf{j}) \exp(2\pi i(\mathbf{h}\cdot\mathbf{M}^{-1}\mathbf{j})),\ \forall \mathbf{h} \in \mathit{\Pi}. \tag{10}$$

This equation system would be linear in $\rho_g(\mathbf{j})$ if the grid structure factors were known. Next we use (9) to relate the unknown grid structure factors $\mathbf{F}_g(\mathbf{h})$ to the true structure factors $\mathbf{F}(\mathbf{h})$, whose magnitude can be observed in the X-ray experiment. Given an upper bound $|R(\mathbf{h})| \le \varepsilon_1(\mathbf{h}),\ \forall \mathbf{h} \in \mathit{\Pi}$, we get the following system of inequalities for the grid density function:

$$\Big| \sum_{\mathbf{j}\in\mathit{\Pi}} \rho_g(\mathbf{j}) \exp(2\pi i(\mathbf{h}\cdot\mathbf{M}^{-1}\mathbf{j})) - \frac{M}{V_{cell}}\mathbf{F}(\mathbf{h}) \Big| \le \varepsilon_1(\mathbf{h}),\ \forall \mathbf{h} \in \mathit{\Pi} \tag{11}$$

5 Recovering the phases

We will now deduce further constraints restricting the possible phases of the structure factors $\mathbf{F}(\mathbf{h})$. Here, we have to distinguish between centric and acentric reflections.

5.1 Symmetries

We say that the density distribution $\rho(\mathbf{x})$ displays the symmetries of a space group $\mathit{\Gamma} = \{(\mathbf{R}_\nu, \mathbf{t}_\nu)\}_{\nu=1}^{n_{sym}}$, $n_{sym} \in \mathbb{N}$, with $\mathbf{R}_\nu$ being a rotation matrix and $\mathbf{t}_\nu$ a translation vector if the following holds [9]:

$$\rho(\mathbf{R}_\nu\mathbf{x} + \mathbf{t}_\nu) = \rho(\mathbf{x}), \forall \mathbf{x} \in \mathbb{R}^3, \forall \nu \in \{1,\dots,n_{sym}\}. \tag{12}$$

From (12) and (2) we can derive the following symmetries for the structure factors [8]:

$$\mathbf{F}(\mathbf{h}) = \exp(2\pi i(\mathbf{h}\cdot\mathbf{t}_\nu))\ \mathbf{F}(\mathbf{R}_\nu^T\mathbf{h}),\ \forall \mathbf{h} \in \mathit{\Pi}, \forall \nu \in \{1,\dots,n_{sym}\}. \tag{13}$$

If $\mathbf{R}_\nu^T\mathbf{h} = -\mathbf{h}$ for some ν, $\mathbf{h}$ is called *centric reflection*, otherwise it is called *acentric*.

5.2 Centric reflections

Using (13) and the Hermitian symmetry $\mathbf{F}(-\mathbf{h}) = \overline{\mathbf{F}(\mathbf{h})}$ of the structure factors, we obtain the following phase restrictions for centric reflections:

$$\text{if } \mathbf{R}_\nu^T\mathbf{h} = -\mathbf{h} \text{ then } \varphi(\mathbf{h}) = \psi(\mathbf{h}) \text{ or } \varphi(\mathbf{h}) = \psi(\mathbf{h}) + \pi \text{ with } \psi(\mathbf{h}) = \pi(\mathbf{h}\cdot\mathbf{t}_\nu). \quad (14)$$

So, if the reflection is centric, only two values of the phase, $\psi(\mathbf{h})$ or $\psi(\mathbf{h}) + \pi$, with $\psi(\mathbf{h})$ being known, are possible. Thus, we can introduce a new variable $\alpha(\mathbf{h}) \in \{0, 1\}$, representing the phase ambiguity. In our inequality system (11), we can replace $\mathbf{F}(\mathbf{h})$:

$$\mathbf{F}(\mathbf{h}) = F(\mathbf{h})\exp(i\varphi(\mathbf{h})) = F(\mathbf{h})(2\alpha(\mathbf{h}) - 1)\exp(i\psi(\mathbf{h})). \quad (15)$$

Taking real and imaginary parts, this results in the following inequalities for centric reflections $h \in \mathit{\Pi}$:

$$\Big|\sum_{\mathbf{j}\in\mathit{\Pi}} \cos(2\pi(\mathbf{h}\cdot\mathbf{M}^{-1}\mathbf{j}))\rho_g(\mathbf{j}) - (2\alpha(\mathbf{h}) - 1)\frac{M}{V_{cell}}F(\mathbf{h})\cos\psi(\mathbf{h})\Big| \le \varepsilon_1(\mathbf{h}), \quad (16)$$

$$\Big|\sum_{\mathbf{j}\in\mathit{\Pi}} \sin(2\pi(\mathbf{h}\cdot\mathbf{M}^{-1}\mathbf{j}))\rho_g(\mathbf{j}) - (2\alpha(\mathbf{h}) - 1)\frac{M}{V_{cell}}F(\mathbf{h})\sin\psi(\mathbf{h})\Big| \le \varepsilon_1(\mathbf{h}). \quad (17)$$

Thus the inequalities become linear in $\rho_g(\mathbf{j})$ and $\alpha(\mathbf{h})$ if the structure factor magnitudes $F(\mathbf{h})$ are known.

5.3 Acentric reflections

For the acentric reflections, the phase can take any value from 0 to 2π. [2] suggests for this case to restrict the phase of the structure factor to one of four possible values $\pm\frac{\pi}{4}$, $\pm\frac{3\pi}{4}$. Introducing two new variables $\alpha(\mathbf{h}), \beta(\mathbf{h}) \in \{0, 1\}$ and taking the real and imaginary part leads to the following inequalities for acentric reflections $\mathbf{h} \in \mathit{\Pi}$:

$$\Big|\sum_{\mathbf{j}\in\mathit{\Pi}} \cos(2\pi(\mathbf{h}\cdot\mathbf{M}^{-1}\mathbf{j}))\rho_g(\mathbf{j}) - (2\alpha(\mathbf{h}) - 1)\frac{M}{V_{cell}}F(\mathbf{h})\frac{1}{\sqrt{2}}\Big| \le \varepsilon(\mathbf{h}), \quad (18)$$

$$\Big|\sum_{\mathbf{j}\in\mathit{\Pi}} \sin(2\pi(\mathbf{h}\cdot\mathbf{M}^{-1}\mathbf{j}))\rho_g(\mathbf{j}) - (2\beta(\mathbf{h}) - 1)\frac{M}{V_{cell}}F(\mathbf{h})\frac{1}{\sqrt{2}}\Big| \le \varepsilon(\mathbf{h}). \quad (19)$$

The error $\varepsilon(\mathbf{h})$ is given by $\varepsilon(\mathbf{h}) = \varepsilon_1(\mathbf{h}) + \varepsilon_2(\mathbf{h})$. Here $\varepsilon_2(\mathbf{h})$, introduced by the sampling of the phase value can be estimated by $\varepsilon_2(\mathbf{h}) \le \frac{1}{\sqrt{2}}\frac{M}{V_{cell}}F(h)$.

6 Constraint-based modeling of the phase problem

6.1 Constraint system

In the context of direct phasing, it may be sufficient to find a binary *envelope* of the regarded molecules, i.e., a binary function representing areas where the electron density

is above a certain level [2]. Using this idea, we may replace the unknowns $\rho_g(\mathbf{j})$ by binary variables $z_\mathbf{j} \in \{0,1\}$, for each grid point $\mathbf{j} \in \mathit{\Pi}$. The value of $z_\mathbf{j}$ should be 1 if the electron density $\rho_g(\mathbf{j})$ is above a certain level and 0 otherwise, so the solution of the problem provides a binary envelope of the regarded molecules.
We end up with a system of linear inequalities in 0-1 variables for representing the electron density values at grid points. We use the following notations (the superscripts R and I stand for the real and imaginary part resp.)

$$a^R_\mathbf{j}(\mathbf{h}) = \cos(2\pi(\mathbf{h}\cdot\mathbf{M}^{-1}\mathbf{j})),\ a^I_\mathbf{j}(\mathbf{h}) = \sin(2\pi(\mathbf{h}\cdot\mathbf{M}^{-1}\mathbf{j})), \tag{20}$$

For centric reflections, we set

$$y^R_\mathbf{h} = y^I_\mathbf{h} = \alpha(\mathbf{h}), \tag{21}$$
$$b^R_\mathbf{h} = 2\kappa F(\mathbf{h})\cos\psi(\mathbf{h}),\ \ b^I_\mathbf{h} = 2\kappa F(\mathbf{h})\sin\psi(\mathbf{h}), \tag{22}$$
$$c^R_\mathbf{h} = \kappa F(\mathbf{h})\cos\psi(\mathbf{h}),\ \ c^I_\mathbf{h} = \kappa F(\mathbf{h})\sin\psi(\mathbf{h}), \tag{23}$$

and for acentric reflections

$$y^R_\mathbf{h} = \alpha(\mathbf{h}),\ \ y^I_\mathbf{h} = \beta(\mathbf{h}), \tag{24}$$
$$b^R_\mathbf{h} = 2\kappa F(\mathbf{h})2^{-1/2},\ \ b^I_\mathbf{h} = 2\kappa F(\mathbf{h})2^{-1/2}, \tag{25}$$
$$c^R_\mathbf{h} = \kappa F(\mathbf{h})2^{-1/2},\ \ c^I_\mathbf{h} = \kappa F(\mathbf{h})2^{-1/2}. \tag{26}$$

Here $\kappa \geq 0$ is a scaling factor reflecting that the magnitudes $F^{obs}(\mathbf{h})$ we get from the analysis of the diffraction pattern correspond to a real electron density distribution, and not to a binary one [2].
To further simplify, we will write

$$A^R(\mathbf{h},\mathbf{z},\mathbf{y}(\mathbf{h})) = \sum_{\mathbf{j}\in\mathit{\Pi}} a^R_\mathbf{j}(\mathbf{h})z_\mathbf{j} - \left(b^R_\mathbf{h}y^R_\mathbf{h} - c^R_\mathbf{h}\right), \tag{27}$$
$$A^I(\mathbf{h},\mathbf{z},\mathbf{y}(\mathbf{h})) = \sum_{\mathbf{j}\in\mathit{\Pi}} a^I_\mathbf{j}(\mathbf{h})z_\mathbf{j} - \left(b^I_\mathbf{h}y^I_\mathbf{h} - c^I_\mathbf{h}\right). \tag{28}$$

Then our binary variables $z_\mathbf{j}, y^R_\mathbf{h},\ y^I_\mathbf{h}$, with $\mathbf{j},\mathbf{h} \in \mathit{\Pi}$ have to satisfy

$$|A^R(\mathbf{h},\mathbf{z},\mathbf{y}(\mathbf{h}))| \leq \varepsilon_\mathbf{h} \text{ and } |A^I(\mathbf{h},\mathbf{z},\mathbf{y}(\mathbf{h}))| \leq \varepsilon_\mathbf{h},\ \forall\mathbf{h} \in \mathit{\Pi}, \tag{29}$$

where $\varepsilon_\mathbf{h} = \varepsilon_1(\mathbf{h})$ for centric and $\varepsilon_\mathbf{h} = \varepsilon(\mathbf{h})$ for acentric reflections.

6.2 Objective function

One possibility to work with the inequality system (29) is to apply a penalty method. Whenever $|A^R(\mathbf{h},\mathbf{z},\mathbf{y}(\mathbf{h}))| > \varepsilon_\mathbf{h}$, we include $|A^R(\mathbf{h},\mathbf{z},\mathbf{y}(\mathbf{h}))|$ as a penalty term, similarly for $|A^I(\mathbf{h},\mathbf{z},\mathbf{y}(\mathbf{h}))|$. This can be modelled as a mixed-integer optimisation problem with the help of additional variables $r^R_\mathbf{h}, r^I_\mathbf{h}, \mathbf{h} \in \mathit{\Pi}$ representing the penalties:

$$\min \quad \sum_{\mathbf{h}\in\Pi} (r^R_{\mathbf{h}} + r^I_{\mathbf{h}}) \tag{30}$$

$$\text{subject to} \quad 0 \le r^R_{\mathbf{h}},\ 0 \le r^I_{\mathbf{h}}, \quad \forall \mathbf{h} \in \Pi \tag{31}$$

$$-\varepsilon_{\mathbf{h}} - r^R_{\mathbf{h}} \le A^R(\mathbf{h}, \mathbf{z}, \mathbf{y}(\mathbf{h})) \le \varepsilon_{\mathbf{h}} + r^R_{\mathbf{h}}, \quad \forall \mathbf{h} \in \Pi, \tag{32}$$

$$-\varepsilon_{\mathbf{h}} - r^I_{\mathbf{h}} \le A^I(\mathbf{h}, \mathbf{z}, \mathbf{y}(\mathbf{h})) \le \varepsilon_{\mathbf{h}} + r^I_{\mathbf{h}}, \quad \forall \mathbf{h} \in \Pi \tag{33}$$

$$z_{\mathbf{j}}, y^R_{\mathbf{h}},\ y^I_{\mathbf{h}} \in \{0,1\}, \quad \forall \mathbf{h}, \mathbf{j} \in \Pi \tag{34}$$

7 Ongoing and further work

In this paper, we have described for a constraint programming audience the basics of our constraint-based approach to the phase problem in X-ray crystallography. Preliminary computational experiments and a crystallographic discussion of the results can be found in [2]. In this earlier work, we used the local search pseudo-Boolean solver WSATOIP [7], which was efficient only for a very small grid size ($6\times6\times6$ or $8\times8\times8$). To increase performance, we are currently experimenting with state-of-the-art integer programming and pseudo-Boolean solvers that have been developed in recent years.
Another line of research consists in modeling different geometric properties of crystals. This results in new constraints which can be added to the model, in order to increase the quality of the solutions. One such constraint is the connectivity constraint stating that the number of connected components in the binary envelope has to be less or equal to the number of molecules in the unit cell [3]. A corresponding integer programming formulation has been developed and is currently being tested.

Acknowledgment

We would like to thank Alexandre Urzhumtsev for his continuing support and many fruitful discussions on the topics of this paper.

References

1. J. Drenth. *Principles of protein X-ray crystallography*. Springer-Verlag, 1994.
2. V. Y. Lunin, A. Urzhumtsev, and A. Bockmayr. Direct phasing by binary integer programming. *Acta Cryst. A*, 58(Pt 3):283–291, May 2002.
3. N. Lunina, V. Y. Lunin, and A. Urzhumtsev. Connectivity-based ab initio phasing: from low resolution to a secondary structure. *Acta Cryst. D*, 59(Pt 10):1702–1715, Oct 2003.
4. D. Sherwood. *Crystals, X-rays and Proteins*. Longman, 1976.
5. L. F. TenEyck. Efficient structure-factor calculation for large molecules by the fast Fourier transform. *Acta Cryst.*, A33:486–492, 1977.
6. J. S. Walker. *Fast Fourier Transforms*. CRC Press, 1991.
7. J. P. Walser. *Integer optimization by local search*. Springer, LNAI 1637, 1999.
8. J. Waser. Symmetry relations between structure factors. *Acta Cryst.*, 8:595, 1955.
9. W. H. Zachariasen. *Theory of X-ray Diffraction in Crystals*. Wiley, New York, 1945.

Combinatorial Optimisation to Design Gene Regulatory Networks

Guillermo Rodrigo[1], Javier Carrera[1] and Alfonso Jaramillo[2]

[1] Inst. Biologia Molecular y Celular de Plantas, CSIC-UPV, 46022 Valencia, Spain
[2] Lab. Biochimie, Ecole Polytechnique, 91128 Palaiseau, France
http://synth-bio.org

Abstract. We propose a methodology to design gene regulatory networks with targeted dynamics based on combinatorial optimisation that poses new challenges for constraints programming. We use genetic programming techniques to evolve from scratch a transcriptional circuit with unconstrained number of genes that works as a logic gate or as an oscillator. Our circuits are defined by a set of non-linear differential equations describing the protein concentrations. At each evolutive step we could add or remove a concentration and its corresponding differential equation. This corresponds to adding or removing a gene. We can also modify the functional form of a differential equation or modify some kinetic parameter. This corresponds to mutation events in the regulatory or coding sequence. We define as the scoring function the distance between the circuit dynamics and the targeted behaviour We explore the space of all possible transcriptional regulation networks, where at each step we would add/subtract new interactions or modify kinetic parameters, to find the optimal circuit with specified system behaviour. We apply our methodology to the design of genetic devices having a desired switching or oscillatory behaviour. Our circuits could be constructed experimentally by assembling biological parts with appropriate kinetic properties. This is not possible in general and the designer, who will only have a small set of available biological parts, will be forced to evolve some of its parts. This introduces a parameter range for each part that it will propagate into an evolvability range for each designed circuit. Those ranges are best described by using constraints and the evolution process could implement model-checking prior to the evaluation of the dynamics.

Keywords: Biological Systems, Regulatory Networks, Genetic Programming, Combinatorial Optimisation.

1 Introduction

One of the most intriguing aspects of networks of complex systems is their temporal dynamics. Very often in complex systems the dynamics does not follow from the network topology. Among the chief examples of complex networks are the genetic transcription networks. The study of those networks has important applications in understanding the circuitry of living systems. There has been a tremendous work on elucidating the network topology of transcription networks [1]. Studies of recurrent network motifs showed that their dynamics could provide useful functions [2,3]. These reverse-engineering studies are very useful to plan the forward-engineering of synthetic circuits. The new development of standardized genetic parts [4] will allow designing much complex networks in a modular way according to some specifications by the assembly of those parts. Usually genetic parts are taken from wild-type organisms. Nevertheless, some experimental work has been performed on building synthetic parts such as promoters with altered operator sites [5,6,7], modified ribosome binding sites [8], or codon-optimized coding regions [9]. The *de novo* design of protein has engineered new coding regions with specified functions and sometimes they have no similarity with any natural sequence [10,11,12]. In addition, most synthetic promoters are regulated by a single transcription factor, but there has also been some work on the design of promoters regulated by two transcription factors [13,14].

The design of artificial genetic networks [15,16,17] has boosted the emerging field of synthetic biology [18]. Still most of the work has been done using rational design techniques, limiting the computational facilities to the solving of dynamical equations. It would be extremely useful to be able to use computational methods to aid in the optimization and design of new circuits. For instance, we could use a catalogue of genetic circuits with optimized transfer functions as educated guesses to aid in the design of a given genetic circuit. Previous work has already used evolutionary methods to design circuits able to oscillate, although they were composed of electronic components [19]. Another work [20] did focus on biological networks, using protein species and a post-translational regulation to design several types of circuits, although this type of regulation is difficult to implement experimentally. Here, we propose to address transcriptional regulatory interactions, neglecting post-translation regulations, to implement genetic networks that could eventually be synthesized.

Our computational algorithm (Genetdes) searches the space of artificial genetic networks to find the optimal circuits with a targeted temporal behaviour [21]. During our simulation, we add or subtract genes, change kinetic constants or the operator-binding logic function at promoters. Each generated circuit is evolved in time and we use the average deviation to an expected temporal function as scoring function. We use Monte Carlo Simulated Annealing [22] method to do the optimization in the space of all possible genetic circuits. Our circuits could be constructed experimentally by assembling biological parts with appropriate kinetic properties. This is not possible in general and the designer, who will only have a small set of available biological parts, will be forced to evolve some of her parts. This introduces a parameter range for each part that it will propagate into an evolvability range for each designed circuit. Those

ranges are best described by using constraints and the evolution process could implement model-checking prior to the evaluation of the dynamics.

2 Methods

2.1. Mathematical model

The dynamics of transcriptional regulatory networks can be depicted by systems of nonlinear first-order ordinary differential equations. We have considered an effective model of protein concentrations for the transcriptional regulations. The dynamics of a transcription factor concentration (Yi) follows the differential equation

$$d[Y_i]/dt = \alpha_i R_i - \beta_i [Y_i] + \gamma_i , \quad (\mathbf{1})$$

where α_i is the transcription-translation rate of gene i, β_i the corresponding degradation rate, and γ_i the basal rate. The function R_i defines the regulatory factor for the promoter of gene i, defined by

$$R_i = 1/(1+([Y_j]/K_{ij})^{nij}) , \quad (\mathbf{2})$$

where K_{ij} is the regulatory coefficient and n_{ij} is the Hill coefficient (chosen positive for repressions and negative for activations) for the transcription factor j.

2.2. Fitness function

We compute the fitness function as the deviation of the circuit dynamics (y) respect to the targeted dynamics (z) as

$$J = \int |y-z| \chi \, dt , \quad (\mathbf{3})$$

where χ is a weighting factor used to only compute a region of interest (e.g., to avoid transients or to impose an oscillatory dynamics). In that way, we construct a minimization problem, where we evolve networks to behave close to the specified dynamics (z).

We use several transfer functions to specify the target behaviour. Each transfer function gives the behaviour of the system for a given input state. In that way, four transfer functions are required to design a circuit working as a logic gate of two inputs, as there are four entries in the corresponding truth table. On the other hand, to design an autonomous oscillator we need just one transfer function. Therefore, for

each transfer function we compute the fitness of the system, and the global fitness function is the sum of all them.

However, the landscape proposed by that fitness function has not large biological referents as these systems have to be robust as well as functional. Thus, we extend the fitness function to

$$F = (1-r)\, J + r\, J' , \qquad (4)$$

where J is the fitness function given by the equation 3, J' is a new term to count the robustness of the system and r is the degree of robustness for our design. In this work we just study the robustness under parameter perturbations. However, further works will study the robustness under topological perturbations, which will give important issues for understanding the evolution of biological systems. Therefore, we compute J' as the average value of all fitness functions (here we compute 10) after perturbing randomly all the model parameters.

2.3. Optimization procedure

We use Monte Carlo Simulated Annealing [22] to optimize transcription circuits in the space of topologies and parameters. We define a mutation operator to evolve the circuit. This operator consists of two moves, both with a probability of occurrence. The first move is in the parameter space. We select randomly a parameter of the model and we perturb it within the corresponding range of values. The second move is in the topology space. There are five possibilities: (i) change the logic function of a binary promoter, (ii) add a new regulation, (iii) remove a regulation, (iv) add a new gene in the circuit or (v) remove a gene from the circuit. We can specify the probabilities to do these moves according to our design purposes. In addition, for convergence purposes, the probability to do a parameter move is taken much higher than the one to do a topology move (e.g., 0.99). In that way, for each evolved topology we explore the parameter space.

In case of no initial network specification, we start from a disconnected circuit where the number of genes of the circuit is equal to the number of inputs plus the number of outputs. We take a Metropolis criterion to accept a mutation, using an exponential cooling scheme. As each mutation only involves a small change in the network it could be possible to obtain an analytical approximation to the dynamics. This would speed up our methodology in at least one order of magnitude.

3 Design of small networks

We have applied our methodology to design genetic devices implementing a given behaviour. We focus in designing small functional modules, which could later be assembled arriving to large and sophisticated networks. We have targeted digital behaviours. Our devices consist on genetic circuits having the concentration of two and one transcription factors as input and output respectively. We have targeted AND,

OR, NAND and NOR gates, and in Fig. 1 we show the designed circuit with AND behaviour. u1 and u2 are the input transcription factors and y is the output corresponding to the concentration of a gene product. To compute the objective function we have averaged the score obtained with each transfer function corresponding to every entry of the truth table. We have evaluated the score by computing (3) during 100 minutes, which provides one order of magnitude more time that the transient needed to attain the steady state. We have computed a score for transfer function and we have averaged it. However, for visualization purposes, we have plotted a temporal dynamics where the input transcription factors concentrations u1 and u2 take all possible Boolean values of a two-input truth table. Inputs can be activators or repressors according to the chosen promoter during the simulation.

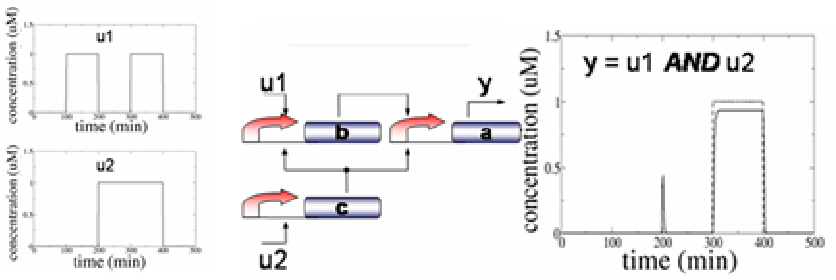


Fig. 1. Transcriptional network composed of three genes (a, b and c) designed to behave as an AND gate. On the left, time evolution of transcription factor inductors u1 and u2 corresponding to (u1, u2)=(0, 0), (0, 1), (1, 0) and (1, 1) for times 0-100, 100-200, 200-300 and 300-400 minutes respectively. On the centre, the circuit obtained with our methodology. On the right, the network dynamics (solid line) superimposed to the targeted behaviour (dashed line).

We have also designed circuits showing an oscillatory behaviour. Towards this end, we have targeted a sinusoidal function. We have considered a weighting factor to compute the score such that it was 1 only in the neighbourhood of a maximum or minimum of the targeted sinusoidal function. This was done to improve the convergence. Fig. 2 shows the optimal genetic network and its time behaviour. We plot as a dashed line the targeted sinusoidal function and as a solid line the corresponding time evolution of the output gene expression. This forward engineering approach has allowed us to design a large set of oscillators and to study evolutionary principles on natural occurring circadian clocks [23]. In addition, we have studied the behaviour of such networks when forcing with external cyclic stimuli at different periods [24].

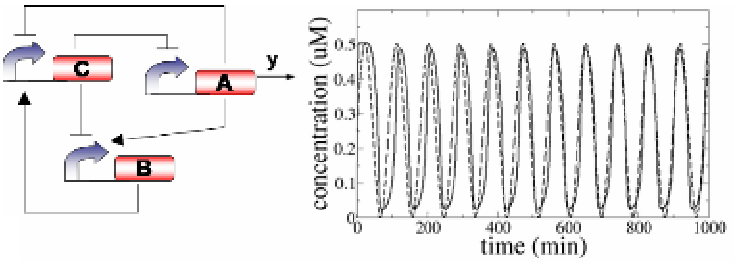


Fig. 2. Transcriptional network designed to show an oscillatory behaviour. The dashed line on the right plot denotes the targeted dynamics.

It is not possible to estimate the complexity of our evolution because it depends on a heuristic optimization process, which will change for each system analyzed. the search throughout the space of genetic networks. Our algorithm writes and reads in SBML level 2 [25], which allows to interface it with a large number of other software. In the species definition, if the specie is an input then its boundary condition will be set to true (false otherwise). Each reaction (transcription-translation) has 1 product and at most 2 reactants. To describe this we need the corresponding kinetic parameters and two additional variables specifying the logical function at the promoter and whether a gene is considered a reporter.

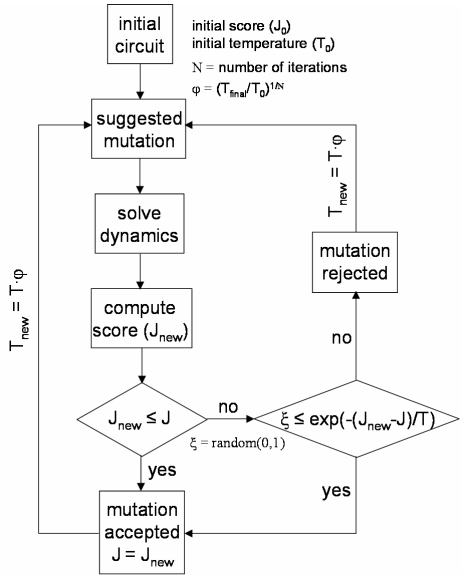


Fig. 3. Scheme showing the flux of Genetdes.

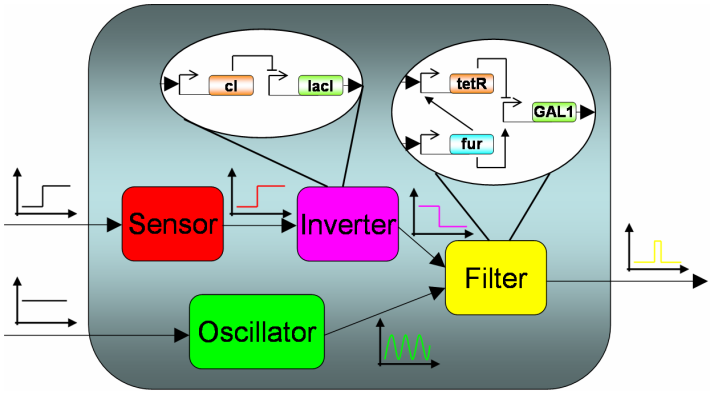


Fig. 4. Example of a design of a complex system using simple functional devices.

4 Discussion

One question to address, from a systems/synthetic biology point of view, is whether natural genetic networks are understandable as systems of devices. Have natural circuits a selective pressure for a given network motif or for a given function? If there could exist a selective pressure for given network modules behaviour, then some circuits within a module could get rewired by evolution while maintaining their functionality. For instance, it could be that some AND circuits would occasionally appear in evolution substituted by another AND circuit. On the other hand, natural gene circuits may not rely on functional modules, but on a complex intertwined network of interactions, as it usually happens with evolutionary design. In this later case, maybe the only way to design a system of devices would be by using an evolutionary design procedure. We could then use directed evolution of gene circuits or in a combination with a computational procedure. In that way, in further work we will expand our methodology to design systems with complex behaviours from a library of functional devices (see Fig. 4).

The implementation of a circuit in a given cellular context usually requires a constant fine-tuning of the model to obtain a successful prototype. In that way, the fact of having a repository of already characterized parts is very useful when implementing a circuit. Therefore, we have developed a software (Asmparts) to assemble part models to construct large systems [26]. We have combined Asmparts with Genetdes to construct and optimize genetic networks.

Acknowledgments. GR acknowledges a graduate fellowship from the Conselleria d'Educacio de la Generalitat Valenciana (BFPI 2007/160). This work was supported by the EU grant BioModularH2 (FP6-NEST 043340), the Spanish Ministry of

Education and Science (TIN 2006-12860) and the Structural Funds of the European Regional Development Fund (ERDF).

References

1. Babu, M. M. & Teichmann, S. A. (2003) Evolution of transcription factors and the gene regulatory network in Escherichia coli. Nucleic Acids Res. 31, 1234-1244.
2. Mangan, S. & Alon, U. (2003) Structure and function of the feed-forward loop network motif. Proc. Natl. Acad. Sci. USA 100, 11980-11985.
3. Brandman, O., Ferrell Jr., J. E., Li, R. & Meyer, T. (2005) Interlinked fast and slow positive feedback loops drive reliable cell decisions. Science 310, 496-498.
4. Endy, D. (2005) Foundations for engineering biology. Nature 438, 449-453.
5. Buchler, N. E., Gerland, U. & Hwa, T. (2003) On schemes of combinatorial transcription logic. Proc. Natl. Acad. Sci. USA 100, 5136-5141.
6. Bintu, L., Buchler, N. E., Garcia, H. G., Gerland, U., Hwa, T., Kondev, J. & Phillips, R. (2005) Transcriptional regulation by the numbers: models. Current Opinion in Genetics & Development 15, 116-124.
7. Bintu, L., Buchler, N. E., Garcia, H. G., Gerland, U., Hwa, T., Kondev, J., Kulhman, T. & Phillips, R. (2005) Transcriptional regulation by the numbers: applications. Current Opinion in Genetics & Development 15, 125-135.
8. Basu, S., Mehreja, R., Thiberge, S., Cheng, M. T. & Weiss, R. (2004) Spatiotemporal control of gene expression with pulse-generating networks. Proc. Natl. Acad. Sci. USA 101, 6355-6360.
9. Basu, S., Gerchman, Y., Collins, C. H., Arnold, F. H. & Weiss, R. (2005) A synthetic multicellular system for programmed pattern formation. Nature 434, 1130-1134.
10. Kuhlman, B., Dantas, G., Ireton, G. C., Varani, G., Stoddard, B. L., Baker, D. (2003) Design of a novel globular protein fold with atomic-level accuracy. Science 302, 1364-1368.
11. Looger, L. L., Dwyer, M. A., Smith, J. J. & Hellinga, H. W. (2003) Computational design of receptor and sensor proteins with novel functions. Nature 423, 185-190.
12. Jaramillo, A., Wernisch, L., Hery, S. & Wodak, S. J. (2002) Folding free energy function selects native-like protein sequences in the core but not on the surface. Proc. Natl. Acad. Sci. USA 99, 13554-13559.
13. Joung, J. K., Koepp, D. M. & Hochschild, A. (1994) Synergistic activation of transcription by bacteriophage l-cI protein and E. coli cAMP receptor protein. Science 295, 1863-1866.
14. Mayo, A. E., Setty, Y., Shavit, S., Zaslaver, A. & Alon, U. (2006) Plasticity of the cis-regulatory input function of a gene. Plos Biol. 4, e45.
15. Elowitz, M. B. & Leibler, S. (2000) A synthetic oscillatory network of transcriptional regulators. Nature 403, 335-338.
16. Gardner, T. S., Cantor, C. R. & Collins, J. J. (2000) Construction of a genetic toggle switch in E. coli. Nature 403, 339-342.
17. Atkinson, M. R., Savageau, M. A., Myers, J. T. & Ninfa, A. J. (2003) Development of genetic circuit exhibiting toggle switch or oscillatory behavior in Escherichia Coli. Cell 113, 597-607.
18. Andrianantoandro, E., Basu, S., Karig, D. K. & Weiss, R. (2006) Synthetic biology: new engineering rules for an emerging discipline. Mol. Syst. Biol. doi:10.1038/msb4100073.
19. Mason, J., Linsay, P. S., Collins, J. J. & Glass, L. (2004) Evolving complex dynamics in electronic models of genetic networks. Chaos 14, 707-715.
20. Francois, P. & Hakim, V. (2004) Design of genetic networks with specified functions by evolution in silico. Proc. Natl. Acad. Sci. USA 101, 580-585.

21. Rodrigo, G., Carrera, J. & Jaramillo, A. (2007) Genetdes: automatic design of transcriptional networks. Bioinformatics 23, 1857-1858.
22. Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. (1983) Optimization by simulated annealing. Science 220, 671-680.
23. Rodrigo, G., Carrera, J. & Jaramillo, A. (2007) Evolutionary mechanisms of circadian clocks. Cent. Eur. J. Biol. 2, 233-253.
24. Rodrigo, G., Carrera, J. & Jaramillo, A. (2007) In silico evolution of the oscillatory response under light-dark cycles. Biochimie, in press.
25. Hucka, M., et al. (2003) The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. Bioinformatics 19, 524-531.
26. Rodrigo, G., Carrera, J. & Jaramillo, A. (2008) Asmparts: assembly of biological model parts. Syst. Synth. Biol., in press.

Detecting Inconsistencies in Large Influence Networks with Answer Set Programming

Martin Gebser[1], Torsten Schaub[1], Sven Thiele[1], Björn Usadel[2], and Philippe Veber[1]

[1] Universität Potsdam, Institut für Informatik, August-Bebel-Str. 89, D-14482 Potsdam
[2] Max-Planck-Institut für Molekulare Pflanzenphysiologie, Am Mühlenberg 1, D-14476 Golm

Abstract. Siegel and colleagues recently proposed a principled definition of consistency between biochemical/genetic reactions and high-throughput profiles of cell activity. Following this work, we present an approach based on Answer Set Programming to check the consistency of large-scale datasets. Furthermore, we extend this approach to provide explanations for inconsistencies in the data, by determining minimal representations of conflicts. In practice, this can be used to identify unreliable data or missing reactions.
Correspondence to: philippe.veber@googlemail.com

1 Introduction

This paper deals with the analysis of high-throughput measurements in molecular biology, like microarray data or metabolic profiles [1]. Up to now, it is still a common practice to use expression profiles merely for detecting over- or under-expressed genes in a given condition, leaving to human experts the task of making biological sense out of tens of gene identifiers. However, many efforts have also been made these years to make a better use of high-throughput data, in particular, by integrating them into large-scale models of transcriptional regulation or metabolic processes [2,3].

One possible approach consists in investigating the compatibility between the experimental measurements and the knowledge that is available in reaction databases. This can be done by using formal frameworks, for instance, those developed in [4] and [5]. A crucial feature of this methodology is its ability to cope with qualitative knowledge (for instance, reactions lacking kinetic details) and noisy data. In this work, we rely on the model by Siegel and colleagues [4], later on referred to as the *Sign Consistency Model* (SCM for short), for developing declarative techniques based on *Answer Set Programming* (ASP for short) [6] to detect and explain inconsistencies in large datasets.

The SCM imposes constraints between experimental measurements and a graph representation of cellular interactions, named an *influence* (or interaction) *graph* [7]. These constraints, later on referred to as *sign consistency constraints*, are described in Section 2. Section 3 provides an intuitive introduction to ASP, a logic-programming paradigm popular due to its declarativeness. In Section 4, we develop an ASP formulation of checking the consistency between experimental profiles and influence graphs. We further extend this approach in Section 5 to identifying minimal representations of conflicts if the experimental data is inconsistent with an influence graph. For both problems, we report preliminary experimental results on randomly generated instances.

Section 6 concludes this paper with a brief discussion and an outlook on future progression.

2 Influence Graphs and Sign Consistency Constraints

Influence graphs [7] (also called interaction graphs in the literature) are a common representation for a wide class of dynamical systems. Basically, an *influence graph* is a directed graph whose vertices are the input and state variables of a system and whose edges express the effect of variables on each other. Informally, an edge $j \to i$ means that the rate of variation of j in time influences the level of i. Each edge $j \to i$ of an influence graph is labeled with a sign, either **+** or **–**, denoted by $\sigma(j, i)$. Sign **+** (resp., **–**) indicates that j tends to increase (resp., decrease) i. An example of influence graph is given in Fig. 1; it represents a simplified model for the operon lactose in *E. coli*.

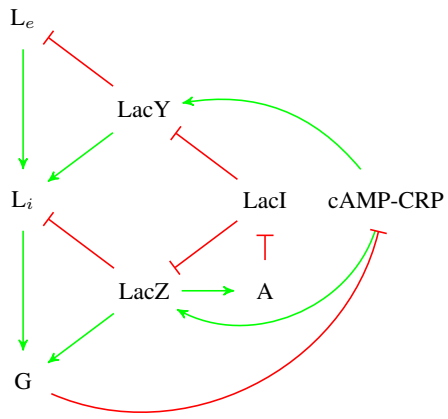


Fig. 1. Simplified model of operon lactose in *E. coli*, represented as an influence graph. The vertices represent either genes, metabolites or proteins, while the edges indicate the regulations among them. Green arrows with normal head stand for positive regulations (activations) while red arrows with tee heads stand for negative regulations (inhibitions). Vertices G and L_e are considered to be inputs of the system, that is they are unconstrained.

In the field of genetic networks, influence graphs have been investigated under various classes of dynamical systems – from ordinary differential equations [8], to synchronous [9] and asynchronous [10] Boolean networks. Influence graph have also been introduced in the field of qualitative reasoning [11], to describe physical systems where a detailed quantitative description is not available. This has been the main motivation for using influence graphs for knowledge representation in the context of biological systems.

In the SCM, *experimental profiles* are supposed to come from steady state shift experiments where, initially, the system is at steady state, then perturbed using control pa-

rameters, and eventually, it settles into another steady state (after a while). It is assumed that the data measures the differences between the initial and the final state. Thus, for each gene, protein, or metabolite, we know whether its concentration has increased or decreased, while quantitative values are unavailable, unessential, or unreliable. By $\mu(i)$, we denote the sign, again + or –, of the variation of a species i between the initial and the final condition. One can easily enhance this setting by also considering null (or more precisely, non-significant) variations, by exploiting the concept of sign algebra [11].

We above introduced influence graphs (as a representation of cellular interactions) and labelings of their vertices with signs (as a representation of experimental profiles). We now describe the constraints that relate both. Informally, for any vertex i, the observed variation $\mu(i)$ should be explained by the influence of at least one predecessor j of i in the influence graph. Thereby, the *influence* of j on i is given by the sign $\mu(j)\sigma(j,i) \in \{+,-\}$, where the multiplication of signs is derived from the multiplication on real numbers. Sign consistency constraints can then be formalized as follows.

Definition 1 (sign consistency constraints). *Let (V, E, σ) be an influence graph, where V is the set of vertices, E the set of edges, and $\sigma : E \to \{+,-\}$ a labeling of the edges. Furthermore, let $\mu : V \to \{+,-\}$ be a vertex labeling. Then, for any $i \in V$, the sign $\mu(i)$ of i is consistent, if there is some edge $j \to i$ in E such that $\mu(i) = \mu(j)\sigma(j,i)$.*

Table 1 shows four different vertex labelings of the influence graph given in Fig. 1. The labeling μ_1 is consistent with the influence graph: the variation of each vertex (apart from the inputs G, and L_e, see Fig. 1) can be explained by the effect of one of its regulators. For instance LacY receives one positive influence from cAMP-CRP, and one negative influence from LacI, which accounts for the variation of LacY. The second labeling, μ_2 is not consistent: this time LacY receives only negative influences from cAMP-CRP and LacI and its variation cannot be explained.

Species	L_e	L_i	G	LacY	LacZ	LacI	A	cAMP-CRP
μ_1	–	–	–	–	–	+	–	+
μ_2	+	+	–	+	–	+	–	–
μ_3	+	?	–	?	?	+	?	?
μ_4	?	?	?	–	+	?	?	+

Table 1. Some labelings for the influence graph depicted in Fig. 1.

The notion of (sign) consistency is extended to whole influence graphs in the natural way, requiring the sign of each vertex to be consistent. Furthermore, in practice, influence graphs and experimental profiles are likely to be partial. Thus, we say that a partial labeling of the vertices is consistent with a partially labeled influence graph, if there is some consistent extension of vertex and edge labelings to all vertices and edges. For instance, vertex labeling μ_3 is consistent with the influence graph given in Fig. 1, as setting the signs +, –, –, –, + to L_i, LacY, LacZ, A and cAMP-CRP respectively extends μ_3 in a consistent labeling. On the other hand, μ_4 cannot be consistently extended.

3 Answer Set Programming

This section provides a brief, informal introduction to ASP (see [6] for formal details). ASP is an attractive declarative paradigm for knowledge representation and reasoning, offering a rich modeling language [12,13] along with highly efficient inference engines based on Boolean constraint solving technology [14,15,16]. The basic idea of ASP is to encode a problem as a logic program such that its answer sets represent solutions.

In view of our application, we take advantage of the elevated expressiveness of disjunctive programs, being able to capture problems at the second level of the polynomial hierarchy [17]. A *disjunctive logic program* is a finite set of rules of the form

$$a_1; \ldots; a_l \leftarrow b_{l+1}, \ldots, b_m, \mathit{not}\ c_{m+1}, \ldots, \mathit{not}\ c_n\ , \tag{1}$$

where a_i, b_j, c_k are *atoms* for $0 < i \leq l < j \leq m < k \leq n$. A rule r as in (1) is called a *fact*, if $l = n = 1$, and an *integrity constraint*, if $l = 0$. Intuitively, a rule amounts to an implication, with ',' standing for '$\wedge$' and ';' for '$\vee$'; however, standard transformations valid in classical logic, e.g., contraposition, are not valid under the answer set semantics. In fact, the answer sets of a program are particular classical models of the program satisfying a certain stability criterion (cf. [6]). Roughly, a set X of atoms is an *answer set* of a program, if for each program rule of form (1), X contains a minimum number of atoms among $a_1, \ldots, a_l$ when $b_{l+1}, \ldots, b_m$ belong to X and no $c_{m+1}, \ldots, c_n$ belongs to X. However, note that the disjunction in heads of rules, in general, is not exclusive.

Though answer sets are usually defined on ground (i.e., variable-free) programs, the elegance of ASP comes from the possibility to provide non-ground *problem encodings*, where schematic rules amount to their ground instantiations. Grounders, like *lparse* [13], are capable of combining a problem encoding and an instance (typically a set of ground facts) into an equivalent ground program, which is then processed by some ASP solver. We follow this methodology and provide encodings for the problems considered below.

4 Checking Consistency

We now come to the first main question addressed in this paper, namely, how to check whether an experimental profile is consistent with a given influence graph. Note that, if the profile provides us with a sign for each vertex of the influence graph, the task can be accomplished in polynomial time. However, as soon as the experimental profile has missing values (which is very likely in practice), the problem becomes NP-hard [18].

Here, we present an encoding of the problem as a logic program such that each of its answer sets matches a consistent extension of vertex and edge labelings. Our program is composed of three parts, which we describe next.

Problem Instance The influence graph as well as the profile are given by ground facts. For each species i, we introduce a fact *vertex*(i), and for each edge $j \rightarrow i$, a fact *edge*(j,i). Furthermore, if the variation s (either **+** or **–**) of a species i is given in the profile, it is modeled by a fact *obs_vlabel*(i,t), where $t = p$ if $s =$ **+** and $t = n$ if $s =$ **–**, and if the sign s of an edge $j \rightarrow i$ is known, it is expressed by a fact *obs_elabel*(j,i,t).

Generating Solution Candidates As stated above, our goal is to check whether an experimental profile is consistent with an influence graph. If so, it is witnessed by total labelings of the vertices and edges, which are generated via the following rules:

$$\begin{array}{c} \mathit{vlabel}(V,p)\,;\,\mathit{vlabel}(V,n) \leftarrow \mathit{vertex}(V)\ , \\ \mathit{elabel}(U,V,p)\,;\,\mathit{elabel}(U,V,n) \leftarrow \mathit{edge}(U,V)\ . \end{array} \tag{2}$$

Moreover, the following rules ensure that known labels are respected by total labelings:

$$\begin{array}{c} \mathit{vlabel}(V,S) \leftarrow \mathit{obs\_vlabel}(V,S)\ , \\ \mathit{elabel}(U,V,S) \leftarrow \mathit{obs\_elabel}(U,V,S)\ . \end{array} \tag{3}$$

Note that the stability criterion for answer sets implies that a known label derived via rules in (3) is also derived via rules in (2), thus, excluding the opposite label.

Testing Solution Candidates Finally, we check whether the generated total labelings satisfy the sign consistency constraints stated in Definition 1, stipulating an influence of sign s for each vertex i with variation s. We thus define *infl*(i,s) to indicate that i receives an influence of sign s, where the encoding contains facts *sign*(p) and *sign*(n):

$$\begin{array}{l} \mathit{infl}(V,p) \leftarrow \mathit{edge}(U,V), \mathit{elabel}(U,V,S), \mathit{vlabel}(U,S), \mathit{sign}(S)\ , \\ \mathit{infl}(V,n) \leftarrow \mathit{edge}(U,V), \mathit{elabel}(U,V,S), \mathit{vlabel}(U,T), \mathit{sign}(S), \mathit{sign}(T), S \neq T\ . \end{array} \tag{4}$$

Inconsistent labelings are then ruled out by integrity constraints of the following form:

$$\leftarrow \mathit{vertex}(V), \mathit{vlabel}(V,S), \mathit{sign}(S), \mathit{not}\ \mathit{infl}(V,S)\ . \tag{5}$$

Benchmarks We assessed the efficiency of our approach on artificially generated instances. Each instance is composed of a graph, a complete labeling of its edges with signs, and a partial labeling of its vertices. Our random generator of instances has three parameters: the number of vertices in the influence graph n, the average degree in the graph β and the proportion of observed nodes γ. To generate one instance, we compute a random graph under the model by Erdős-Rényi [19], where each pair of vertices has equal probability to be an edge. The parameter β is fixed to 2.5, and varying between 2 and 3 (which are usual values in biological networks [20]) does not change the results significantly (data not shown). The labels on edges are chosen independently with probability $\frac{1}{2}$ for each sign. Then $\lfloor \gamma n \rfloor$ vertices are chosen with uniform probability, and assigned a label with probability $\frac{1}{2}$ for each sign.

The instances were solved using the grounder *lparse* [13] and the solver *cmodels* [21]. These programs were run on an Intel Core 2 Duo 2.4 GHz processor, with 4GB of memory under Linux. All reported times correspond to Unix user time.

The results are presented in Fig. 2: we separated execution time into grounding time and solving time to show their relative contribution. The grounding stage transforms the original logic program into an equivalent variable-free program. This step is required because only ground programs can be solved efficiently in practice. Both graphics display the time needed for the corresponding phase as a function of the number of vertices in the instance. For each size, we generate 50 instances distributed in 5 groups having a different γ value (here: $\frac{1}{100}$, $\frac{1}{50}$, $\frac{1}{30}$, $\frac{1}{20}$, $\frac{1}{10}$).

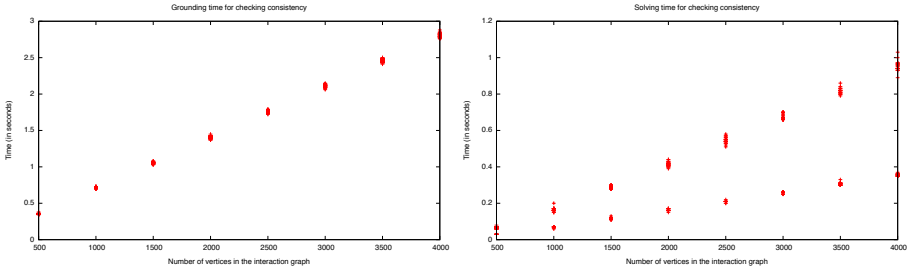


Fig. 2. Execution time for checking the consistency of an influence graph with an expression profile. The execution time is separated into its two contributions, namely grounding (on the left hand) and actual solving (on the right hand).

Grounding time increases linearly with the size of the instance, and does not vary significantly for instances of equal size. Solving time also increases linearly with the size of the instance, though in a more sophisticated way: for a given size of the instances, one can distinguish two clusters of instances having well-separated solving time. Interestingly, all "easy" instances are inconsistent (or equivalently, all instances which are consistent are in the "hard" cluster). Now for each one of these two types of instances, the solving time grows linearly with the size of the instance.

From these results, we can see that checking consistency on data of realistic size (*i.e.* influence graphs of several thousands of vertices) takes no more than a couple of seconds on a standard PC.

5 Identifying Minimal Inconsistent Cores

Once it is proved that an experimental profile is inconsistent with a given influence graph (i.e., if the logic program given in the previous section has no answer set), due to the amount of data, it is crucial to provide explanations that are as concise as possible. Here, we adopt a strategy that was successfully applied on real biological data in [22], where the basic idea is to isolate minimal subgraphs of an influence graph such that the vertices and edges cannot be labeled in a consistent way. This task is closely related to extracting Minimal Unsatisfiable Cores (MUCs) [23] in the context of Boolean Satisfiability (SAT) [14]. In allusion, we call a minimal subgraph of the influence graph whose vertices and edges cannot be labeled consistently a *Minimal Inconsistent Core* (MIC).

As in the previous section, we present a disjunctive program such that its answer sets match MICs. We assume that a problem instance, that is, an influence graph along with an experimental profile, is represented by facts as specified in Section 4. The remainder of the logic program is the problem encoding, consisting of three parts: the first generating MIC candidates, the second asserting inconsistency, and the third verifying minimality. Thereby, the generating part comprises the rules in (2) and (3) as well as:

$$\mathit{active}(V)\,;\,\mathit{inactive}(V) \leftarrow \mathit{vertex}(V)\,.$$

The purpose of this additional rule is to permit guessing a set of vertices to be marked as active. The subgraph of the influence graph induced by the active vertices forms a MIC candidate, which is tested via the two encoding parts described next.

Testing for Inconsistency By adapting a methodology used in [24], the following subprogram makes sure that the active vertices belong to a subgraph that cannot be labeled consistently, taking into account all labelings of the residual vertices and edges:

$$
\begin{aligned}
\mathit{op}(U,V) &\leftarrow \mathit{active}(V), \mathit{vlabel}(V,n), \mathit{edge}(U,V), \mathit{elabel}(U,V,S), \mathit{vlabel}(U,S), \mathit{sign}(S)\ ,\\
\mathit{op}(U,V) &\leftarrow \mathit{active}(V), \mathit{vlabel}(V,p), \mathit{edge}(U,V), \mathit{elabel}(U,V,S), \mathit{vlabel}(U,T), \mathit{sign}(S),\\
&\qquad \mathit{sign}(T), S \neq T\ ,\\
\mathit{bottom} &\leftarrow \mathit{active}(V), \mathit{vertex}(V), \mathit{op}(U,V) : \mathit{edge}(U,V)\ ,^{3}\\
&\leftarrow \mathit{not}\ \mathit{bottom}\ ,\\
\mathit{vlabel}(V,S) &\leftarrow \mathit{bottom}, \mathit{vertex}(V), \mathit{sign}(S)\ ,\\
\mathit{elabel}(U,V,S) &\leftarrow \mathit{bottom}, \mathit{edge}(U,V), \mathit{sign}(S)\ .
\end{aligned}
$$

In this (part of the) encoding, $\mathit{op}(U,V)$ indicates that the influence of vertex U on active vertex V is opposite to the variation of V. If all regulators of V have such an opposite influence, the sign consistency constraint for V is violated. In this case, atom *bottom* is produced, along with all labels for vertices and edges. Here, the stability criterion for an answer set X imposes that *bottom* and all labels can only belong to X if the given problem instance does not permit consistent labelings. Finally, integrity constraint $\leftarrow \mathit{not}\ \mathit{bottom}$ necessitates the inclusion of *bottom* in any answer set, thus, stipulating an inevitable violation of the sign consistency constraint for some vertex that is active.

Testing for Minimality The second test is based on the idea that, picking any active vertex, the sign consistency constraints for all other active vertices should be satisfied by appropriate labelings. This conception is implemented in the following subprogram:

$$
\begin{aligned}
\mathit{vlabel'}(W,V,p)\,;\mathit{vlabel'}(W,V,n) &\leftarrow \mathit{active}(W), \mathit{vertex}(W), \mathit{vertex}(V)\ ,\\
\mathit{elabel'}(W,U,V,p)\,;\mathit{elabel'}(W,U,V,n) &\leftarrow \mathit{active}(W), \mathit{vertex}(W), \mathit{edge}(U,V)\ ,\\
\mathit{vlabel'}(W,V,S) &\leftarrow \mathit{active}(W), \mathit{vertex}(W), \mathit{obs\_vlabel}(V,S)\ ,\\
\mathit{elabel'}(W,U,V,S) &\leftarrow \mathit{active}(W), \mathit{vertex}(W), \mathit{obs\_elabel}(U,V,S)\ ,\\
\mathit{infl'}(W,V,p) &\leftarrow \mathit{active}(W), \mathit{vertex}(W), \mathit{active}(V), V \neq W, \mathit{edge}(U,V),\\
&\qquad \mathit{elabel'}(W,U,V,S), \mathit{vlabel'}(W,U,S), \mathit{sign}(S)\ ,\\
\mathit{infl'}(W,V,n) &\leftarrow \mathit{active}(W), \mathit{vertex}(W), \mathit{active}(V), V \neq W, \mathit{edge}(U,V),\\
&\qquad \mathit{elabel'}(W,U,V,S), \mathit{vlabel'}(W,U,T), \mathit{sign}(S), \mathit{sign}(T), S \neq T\ ,\\
&\leftarrow \mathit{active}(W), \mathit{vertex}(W), \mathit{active}(V), V \neq W, \mathit{vertex}(V),\\
&\qquad \mathit{vlabel'}(W,V,S), \mathit{sign}(S), \mathit{not}\ \mathit{infl'}(W,V,S)\ .
\end{aligned}
$$

This subprogram is similar to the consistency check encoded via the rules in (2–5). However, sign consistency constraints are here only checked for active vertices, and they must be satisfiable for all but one arbitrary active vertex W. Since W ranges over all vertices of the given influence graph, each active vertex is taken into consideration.

[3] In the language of *lparse* [13], $\mathit{op}(U,V) : \mathit{edge}(U,V)$ is used to refer to all ground atoms $\mathit{op}(j,i)$ for which $\mathit{edge}(j,i)$ holds, with the respective ground atoms connected by ','.

Benchmarks We assess the scalability of this approach within the setting given in the previous section. There again, we distinguish between grounding time and solving time. The results are presented in Fig. 3 (note that X and Y axis are in log-scale). For grounding, we found a nearly perfect linear relationship (in log scale) between the size of the instance and the grounding time, and the slope of the line is 2. In other words, the grounding stage is here in $O(n^2)$, which is absolutely consistent with the fact that our encoding for finding MIC grows also quadratically with the number of vertices (see for instance the predicate *vlabel'*).

Concerning the solving time, we also obtain linear relationships, in the following sense. For each size, the 50 instances are distributed into two groups. Most of them are easily solvable, that is in less than a minute. However, a couple of instances are strikingly more difficult, and may be between 100 and 1000 times longer to solve. Nevertheless, we observe that the time needed to solve the easiest (resp., the hardest) instances grows linearly (in log-scale) with the size of the instances. This time, the slope of the line is slightly greater than 2, that is 2.3 and 3.2 for the easiest and the hardest instances respectively. Unfortunately, we could not characterize the hardest instances further.

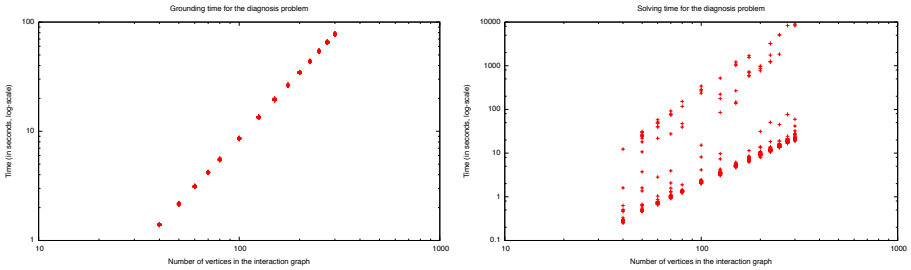


Fig. 3. Execution time for finding one MIC in an inconsistent instance. The execution time is separated into its two contributions, namely grounding (on the left hand) and actual solving (on the right hand). Note that on both cases, X and Y-axis are in log-scale.

These results suggest that finding a MIC in an inconsistent instance is computationally harder than checking consistency. This would not be surprising as extracting MUCs from unsatisfiable set of clauses is provably complete for the second level of the polynomial hierarchy [23]. However it should be noted that our instances are most often very easily solved, and it is still an open question whether instances coming real data fall into this category.

6 Discussion

We have provided an approach based on ASP to check the consistency between experimental profiles and influence graphs. In case of inconsistency, the concept of a MIC can be exploited for identifying concise explanations, pointing to unreliable data or missing

reactions. Such MICs can also be determined by means of ASP, and we have provided an encoding for this purpose. The problem of finding MICs is closely related to the extraction of MUCs in the context of SAT. From a knowledge representation point of view, we however argue for our technique based on ASP, as it allows for an elegant way to describe problems in terms of a uniform encoding and specific instances.

By now, a variety of efficient ASP tools are available, both for grounding and for solving logic programs. An empirical assessment of them (on random as well as real data), along with a comparison to existent methods not based on ASP, is defered to an extended version of this paper. If the ASP approach computationally scales well, its elegance and flexibility in problem modeling might make it attractive for biological questions beyond the ones addressed here. It will also be interesting to investigate how far the performance of ASP tools can be tuned by varying and optimizing encodings.

Acknowledgments Ph. Veber is supported by a grant from the Deutscher Akademischer Austausch Dienst (DAAD). This work was partially funded by the Federal Ministry of Education and Research within the GoFORSYS project (http://www.goforsys.org/; grant 0313924).

References

1. Joyce, A., Palsson, B.: The model organism as a system: Integrating 'omics' data sets. Nature Reviews Molecular Cell Biology **7**(3) (2006) 198–210
2. Klamt, S., Stelling, J.: Stoichiometric and constraint-based modelling. In: System Modeling in Cellular Biology: From Concepts to Nuts and Bolts. MIT Press (2006) 73–96
3. Friedman, N., Linial, M., Nachman, I., Pe'er, D.: Using Bayesian networks to analyze expression data. Journal of Computational Biology **7**(3-4) (2000) 601–620
4. Siegel, A., Radulescu, O., Le Borgne, M., Veber, P., Ouy, J., Lagarrigue, S.: Qualitative analysis of the relation between DNA microarray data and behavioral models of regulation networks. Biosystems **84**(2) (2006) 153–174
5. Gutierrez-Rios, R., Rosenblueth, D., Loza, J., Huerta, A., Glasner, J., Blattner, F., Collado-Vides, J.: Regulatory network of Escherichia coli: Consistency between literature knowledge and microarray profiles. Genome Research **13**(11) (2003) 2435–2443
6. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press (2003)
7. Soulé, Christophe: Graphic Requirements for Multistationarity, Complexus **1**(3) (2003) 123–133
8. Soulé, Christophe: Mathematical approaches to differentiation and gene regulation. Comptes Rendus Biologies **329** (2006) 13–20
9. Remy, Élisabeth, Ruet, Paul, Thieffry, Denis: Graphic requirements for multistability and attractive cycles in a Boolean dynamical framework To appear in Advances in Applied Mathematics (2008)
10. Richard, Adrien, Comet, Jean-Paul: Necessary conditions for multistationarity in discrete dynamical systems Discrete Applied Mathematics (2007)
11. Kuipers, B.: Qualitative reasoning: Modeling and simulation with incomplete knowledge. MIT Press (1994)
12. Simons, P., Niemelä, I., Soininen, T.: Extending and implementing the stable model semantics. Artificial Intelligence **138**(1-2) (2002) 181–234
13. Syrjänen, T.: Lparse 1.0 user's manual. http://www.tcs.hut.fi/Software/smodels/lparse.ps.gz

14. Mitchell, D.: A SAT solver primer. Bulletin of the European Association for Theoretical Computer Science **85** (2005) 112–133
15. Giunchiglia, E., Lierler, Y., Maratea, M.: Answer set programming based on propositional satisfiability. Journal of Automated Reasoning **36**(4) (2006) 345–377
16. Gebser, M., Kaufmann, B., Neumann, A., Schaub, T.: clasp: A conflict-driven answer set solver. In: Proceedings of the Ninth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'07). Springer (2007) 260–265
17. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. Freeman and Co. (1979)
18. Veber, P., Le Borgne, M., Siegel, A., Lagarrigue, S., Radulescu, O.: Complex qualitative models in biology: A new approach. Complexus **2**(3-4) (2004) 140–151
19. Erdős, P., Rényi, A.: On Random Graphs. I. Publicationes Mathematicae **6** (1959) 290–297
20. Jeong, H., Tombor, B., Albert, R., Oltvai, Z.N., Barabási, A.L.: The large-scale organization of metabolic networks Nature **407** (2000) 651–654
21. Lierler, Yu.: Cmodels for Tight Disjunctive Logic Programs Proceedings of the Workshop on Constraint Logic Programming (2005)
22. Guziolowski, C., Veber, P., Le Borgne, M., Radulescu, O., Siegel, A.: Checking consistency between expression data and large scale regulatory networks: A case study. Journal of Biological Physics and Chemistry **7**(2) (2007) 37–43
23. Dershowitz, N., Hanna, Z., Nadel, A.: A scalable algorithm for minimal unsatisfiable core extraction. In: Proceedings of the Ninth International Conference on Theory and Applications of Satisfiability Testing (SAT'06). Springer (2006) 36–41
24. Eiter, T., Gottlob, G.: On the computational cost of disjunctive logic programming: Propositional case. Annals of Mathematics and Artificial Intelligence **15**(3-4) (1995) 289–323

Stochastic local search for large-scale instances of the Haplotype Inference Problem by Parsimony

Luca Di Gaspero[1] and Andrea Roli[2]

[1] DIEGM, University of Udine, via delle Scienze 208, I-33100, Udine, Italy
l.digaspero@uniud.it
[2] DEIS, University of Bologna, via Venezia 52, I-47023 Cesena, Italy
andrea.roli@unibo.it

Abstract. Haplotype Inference is a challenging problem in bioinformatics that consists in inferring the basic genetic constitution of diploid organisms on the basis of their genotype. This information allows researchers to perform association studies for the genetic variants involved in diseases and the individual responses to therapeutic agents.
A notable approach to the problem is to encode it as a combinatorial problem (under certain hypotheses, such as the *pure parsimony* criterion) and to solve it using off-the-shelf combinatorial optimization techniques. The main methods applied to Haplotype Inference are either simple greedy heuristic or exact methods that, at present, are adequate only for moderate size instances.
In this paper, we present an approach based on the combination of local search metaheuristics and a reduction procedure based on an analysis of the problem structure. Results on a set of Haplotype Inference benchmarks show that this approach achieves a good trade-off between solution quality and execution time.

1 Introduction

A fundamental tool of analysis to investigate the genetic variations in a population is based on *haplotype* data. A haplotype is a copy of a chromosome of a diploid organism (i.e., an organism that has two copies of each chromosome, one inherited from the father and one from the mother).

The collection of haplotypes from the genetic material is not an easy task: in fact, due to technological limitations it is currently infeasible to directly collect haplotypes in an experimental way, but rather it is possible to collect *genotypes*, i.e., the conflation of a pair of haplotypes. Therefore, haplotypes have to be inferred from genotypes in order to reconstruct the detailed information and trace the precise structure of human populations. This process is called *Haplotype Inference* and the goal is to find a set of haplotype pairs so that all the genotypes are *resolved*.

Current approaches for solving the problem include simple greedy heuristics [1] and exact methods such as Integer Linear Programming [2,3], Semidefinite Programming [4,5], SAT models [6] and Pseudo-Boolean Optimization algorithms [7]. These approaches, however, at present seem not to be particularly

adequate for very-large size instances. Conversely, we believe that metaheuristic (and hybrid) approaches could provide better scalability than exact algorithms. To the best of our knowledge, the only attempt to employ metaheuristic techniques for the problem is a recently proposed Genetic Algorithm [8]. However, the cited paper does not report results on real size instances.

In this work we present a metaheuristic approach to tackle the Haplotype Inference problem by pure parsimony. We introduce the problem in Section 2 and we sketch an analysis of the problem structure. The outcome of the analysis is a reduction procedure that can be combined with the metaheuristic approach developed in Section 3 in order to improve the performance of local search. Experimental results concerning a comparison of our technique against the state-of-the-art for Haplotype Inference by parsimony are discusses in Section 4.

2 The Haplotype Inference problem

In the Haplotype Inference problem we deal with *genotypes*, that is, strings of length m that corresponds to a chromosome with m sites. Each value in the string belongs to the alphabet $\{0, 1, 2\}$. A position in the genotype is associated with a site of interest on the chromosome (called a SNP: single nucleotide polymorphism) and it has value 0 (wild type) or 1 (mutant) if the corresponding chromosome site is a homozygous site (i.e., it has that state on both copies) or the value 2 if the chromosome site is heterozygous. A *haplotype* is a string of length m that corresponds to only one copy of the chromosome (in diploid organisms) and whose positions can assume the symbols 0 or 1 according to the following rules:

$$g[j] = 0 \Rightarrow h[j] = 0 \wedge k[j] = 0 \tag{1}$$
$$g[j] = 1 \Rightarrow h[j] = 1 \wedge k[j] = 1 \tag{2}$$
$$g[j] = 2 \Rightarrow (h[j] = 0 \wedge k[j] = 1) \vee (h[j] = 1 \wedge k[j] = 0) \tag{3}$$

We say that h is a *resolvent* of g, and we write $h \trianglelefteq g$, if there exists a companion haplotype k such that $\langle h, k\rangle \triangleright g$. This notation can be extend to sets of haplotypes, and we write $H = \{h_1, \ldots, h_l\} \trianglelefteq g$, meaning that $h_i \trianglelefteq g$ for all $i = 1, \ldots, l$, or to sets of genotypes, in this case we write $h \trianglelefteq A$ if $h \trianglelefteq g$ for all $g \in A$.

Conditions (1) and (2) require that both haplotypes must have the same value in all homozygous sites, while condition (3) states that in heterozygous sites the haplotypes must have different values.

Observe that, according to the definition, for a single genotype string the haplotype values at a given site are predetermined in the case of homozygous sites, whereas there is a freedom to choose between two possibilities at heterozygous places. This means that for a genotype string with l heterozygous sites there are 2^{l-1} possible pairs of haplotypes that resolve it.

As an example, consider the genotype $g = (0212)$, then the possible pairs of haplotypes that resolve it are $\langle(0110), (0011)\rangle$ and $\langle(0010), (0111)\rangle$.

The *Haplotype Inference* problem under the *pure parsimony* hypothesis is the problem of finding a set R of n pairs of (not necessarily distinct) haplotypes $R = \{\langle h_1, k_1\rangle, \ldots, \langle h_n, k_n\rangle\}$, so that $\langle h_i, k_i\rangle \triangleright g_i, i = 1, \ldots, n$. We call H the set of haplotypes used in the construction of R, i.e., $H = \{h_1, \ldots, h_n, k_1, \ldots, k_n\}$ and our goal is to minimize the cardinality of H. It has been shown that this problem is APX-hard [9] and therefore NP-hard.

It is possible to define a graph that expresses the compatibility between genotypes, so as to avoid unnecessary checks in the determination of the resolvents. Let us build the graph $\mathcal{G} = (G, E)$, in which the set of vertices coincides with the set of the genotypes; in the graph, a pair of genotypes g_1, g_2 are connected by an edge if they are *compatible*, i.e., one or more common haplotypes can resolve both of them. The same concept can be expressed also between a genotype and a haplotype.

On the basis of the compatibility graph it is possible to devise a reduction procedure whose goal is to try to decrease the number of distinct haplotypes while satisfying the resolution constraint. The intuition behind the procedure is that a possible way of reducing the haplotype number is to resolve a genotype by a haplotype that is compatible, but not currently resolving it. A step of the reduction procedure is described by the following proposition.

Proposition 1 (Haplotype local reduction). *Given n genotypes $G = \{g_1, \ldots, g_n\}$ and the resolvent set $R = \{\langle h_1, k_1\rangle, \ldots, \langle h_n, k_n\rangle\}$, so that $\langle h_i, k_i\rangle \triangleright g_i$. Suppose there exist two genotypes $g, g' \in G$ such that $g \triangleleft \langle h, k\rangle$, $g' \triangleleft \langle h', k'\rangle$, h is compatible also with g' and $h \neq h'$, $h \neq k'$, $h' \trianglelefteq A$, $k' \trianglelefteq B$.*

The replacement of $\langle h', k'\rangle$ with $\langle h, g' \ominus h\rangle$[3] in the resolution of g' is a correct resolution that employs a number of distinct haplotypes according to the following criteria:

- *if $|A| = 1$ and $|B| = 1$, the new resolution uses* at most *one less distinct haplotype;*
- *if $|A| > 1$ and $|B| = 1$ (or symmetrically, $|A| = 1$ and $|B| > 1$), the new resolution uses* at most *the same number of distinct haplotypes;*
- *in the remaining case the new resolution uses* at most *one more distinct haplotype.*

Proof. The proof of the proposition is straightforward. The resolution is obviously correct because h is compatible with g' and $g' \ominus h$ is the complement of h with respect to g'.

Concerning the validity of the conditions on the cardinality, let us proceed by cases and first consider the situation in which $g' \ominus h$ does not resolve any other genotype but g'.

If $|A| = |B| = 1$, then h' and k' are not shared with other genotype resolutions so they will not appear in the set H after the replacement, therefore since in the new resolution h is shared between g and g' the cardinality of H is decreased by one.

[3] With $g' \ominus h$ we denote the complementary haplotype of h w.r.t. g. It is straightforward to prove that such a haplotype exists and is unique.

Conversely, if one of the sets $|A|$ or $|B|$ consists of more than a genotype and the other set of just one genotype, there is no guarantee of obtaining an improvement from the replacement. Indeed, since one of the two haplotypes is already shared with another genotype there is just a replacement of the shared haplotype with another one in the set H.

Finally, when $|A| > 1$ and $|B| > 1$ both h' and k' are shared with other genotypes therefore the replacement introduces the new haplotype $g' \ominus h$ in the set H.

Moving to the situation in which $g' \ominus h$ resolves also other genotypes, the same considerations apply; additionally, given that $g' \ominus h$ is already present in H, the number of distinct haplotypes employed in the resolution is decreased by one. For this reason the estimation of the changes of $|H|$ is conservative. □

Even though in principle the reduction procedure can be employed with any selective solution method (such as Local Search or Genetic Algorithms), in this paper we decided to focus on a tabu search algorithm which seemed to be very promising.

3 Local Search techniques for Haplotype Inference

As the search space for this problem we adopt a *complete* representation of the genotype resolution. That is, we consider, for each genotype g, the pair of haplotypes $\langle h, k \rangle$ that resolves it. In this representation all the genotypes are fully resolved at each state by construction. The search space is therefore the collection of sets R defined as in the problem statement. The complete representation has the advantage of allowing to design anytime algorithms, since the search can be interrupted any moment and return a feasible solution, i.e., a set (not necessarily minimal) of haplotypes that resolve the given genotypes.

For the cost function, we identify different components related either to optimality or to heuristic measures. A natural component is the objective function of the original problem, that is the cardinality $|H|$ of the set of haplotypes employed in the resolution. Moreover, we also include some heuristic related to the potential quality of the solution, namely the number of incompatible sites between each genotype/haplotype pair. The cost function F is then the weighted sum of the two components.

We designed a family of local search strategies, namely *Best improvement*, *Stochastic first improvement*, *Simulated annealing*, and *Tabu search*. The techniques are instances of the general strategies described in [10]. All of them start with a set of haplotypes of cardinality $2n$, where n is the number of genotypes, and they explore the search space by iteratively modifying pairs of resolving haplotypes trying to reduce the number of distinct ones. Best improvement and Stochastic first improvement traverse the search space by moving from a state to a neighboring one with a lower cost function value, by choosing the best and first neighbor respectively. Simulated annealing moves also to worse states than the current one, on the basis of a probabilistic choice function. Finally, Tabu search

behaves in principle like Best improvement but restricts the neighborhood by forbidding recently performed moves.

Local search moves are defined upon a Hamming neighborhood function. A good trade-off between exploration and execution time is the 1-Hamming distance neighborhood w.r.t. each haplotype in the current solution. This kind of move can be thought as a *flip*, performed at a given position in a pair of haplotypes resolving a given genotype. The complete exploration of such a neighborhood has a time complexity bounded from above by $O(nk)$, where k is the number of haplotypes and n the number of sites per haplotype. In practice, the time complexity can be further reduced by restricting the number of neighbors to heterozygous sites and haplotypes resolving non isolated genotypes.

4 Experimental results

We developed a set of local search solvers (Tabu Search, Hill Climbing and Simulated Annealing) using EasyLocal++ [11], a framework for the development of local search algorithms. The algorithms have been implemented in C++ and compiled with *gcc 3.2.2* and run on a Intel Xeon CPU 2.80GHz machine with SUSE Linux 2.4.21-278-smp. Each algorithm was run on every instance one time and we allotted 300 seconds for each execution of the algorithms. Since Tabu search (TS) showed superior performance over the other local search algorithms, we only discuss results of this technique.

Our Tabu search implementation considers as tabu all the moves that insist on a pair of haplotypes that recently changed. The tabu list scheme adopted is a *dynamic* one, that is for each move performed we consider it as prohibited for a number of iterations that randomly varies between two values k_{min} and k_{max}. The values of these parameters were chosen according to the results of an exploratory analysis based on the F-Race method [12], and were set to $k_{min} = 10$, $k_{max} = 20$. These settings have shown to be quite robust across the variety of instances tested. Moreover, since the algorithm that incorporates the initial graph reduction sharply outperforms the one without graph reduction, we report only the results of the former one.

The benchmark instances are composed of two parts. The first one, composed of the sets Harrower uniform, Harrower non-uniform and Harrower hapmap, is the benchmark used in [3]. The second part of the instances, namely Marchini SU1, Marchini SU2, Marchini SU3 and Marchini SU-100kb, were taken from the website `http://www.stats.ox.ac.uk/~marchini/phaseoff.html`.

The main characteristics of the instance sets are summarized in Table 1.

In order to estimate the quality of solutions produced by TS, we need to compute the optimal solution of the benchmark instances. We tackled the instances with *rpoly* [7], a state-of-the-art exact solver for the Haplotype Inference. The solver is run on the same benchmark instances and on the same machine. We allotted *rpoly* 24 hours of computation for each instance. The instances of the set Harrower uniform, Harrower non-uniform, Harrower hapmap, Marchini SU1 and Marchini SU2 were completely solved. From Marchini SU3 and Mar-

Table 1: A summary of the main characteristics of the benchmarks.

Benchmark set	**N. of instances**	**N. of genotypes**	**N. of sites**
Harrower uniform	200	10÷100	30÷50
Harrower non-uniform	90	10÷100	30÷50
Harrower hapmap	24	5÷68	30÷75
Marchini SU1	100	90	179
Marchini SU2	100	90	171
Marchini SU3	100	90	187
Marchini SU-100kb	29	90	18

Table 2: Fraction of instances solved by *rpoly* from each benchmark.

Benchmark set	Fraction of solved instances	**Benchmark set**	Fraction of solved instances
Harrower uniform	200/200	Marchini SU1	100/100
Harrower non-uniform	90/90	Marchini SU2	100/100
Harrower hapmap	24/24	Marchini SU3	89/100
		Marchini SU-100kb	23/29

chini SU-100kb only a portion of the instances were solved. Overall, most of the instances could be solved with a runtime higher than 12 hours per instance. A summary of the fraction of solved instances is reported in Table 2.

The plots in Figure 1 report the comparison between the TS and *rpoly*; a point (x, y) in the plot represents the number of haplotypes in the best solution returned by TS and *rpoly*, respectively. A point below the line means that the solution returned by the algorithm corresponding to the y-axis is better than the one returned by the algorithm associated to the x-axis.

Notice that the solution quality achieved by TS approximates the optimal one returned by *rpoly* on some benchmarks, namely Harrower sets and Marchini SU-100kb, whilst the performance on Marchini SU2 is considerably inferior. The performance on benchmarks Marchini SU1 and Marchini SU3 is inferior, but it has to be taken into account that TS returned a feasible solution to all the instances of the sets, whilst *rpoly* solved only a fraction of the instances of Marchini SU3. We also observe that our approach scales very smoothly.

These results enlighten the complementarity of the two approaches: the algorithm that also returns the proof of optimality is definitely preferable over the incomplete one when the execution time allotted can be large, while we can resort to the approximate algorithm to have a feasible and (hopefully) near-optimal solution in very short time.

References

1. A. G. Clark, Inference of haplotypes from PCR-amplified samples of diploid populations, Molecular Biology and Evolution 7 (1990) 111–122.

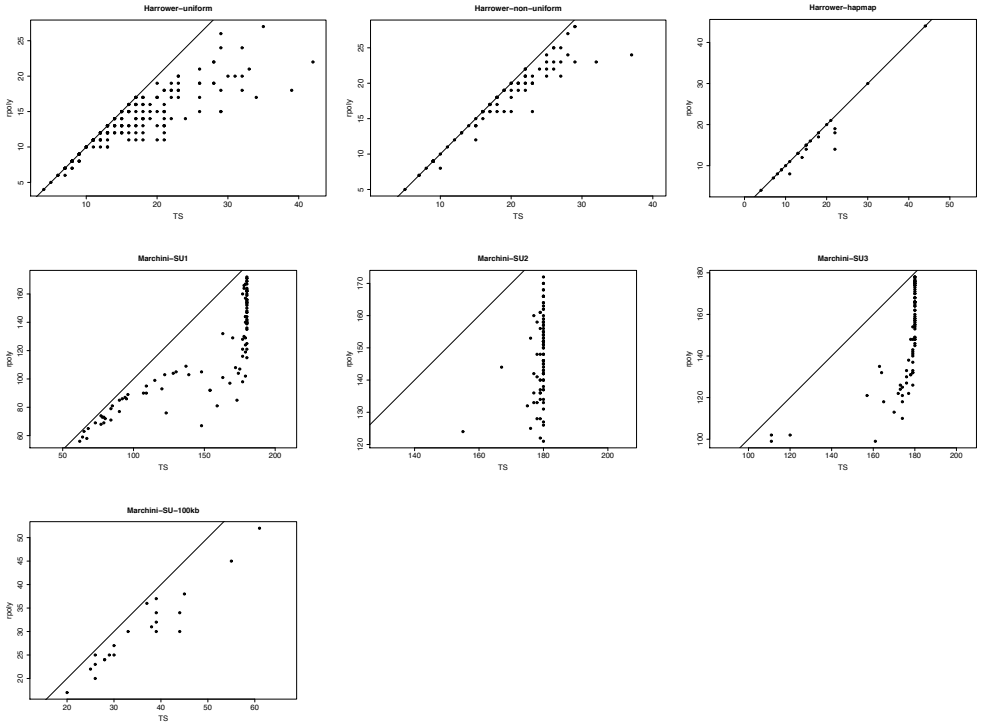


Fig. 1: Comparison between TS and rpoly in terms of number of haplotypes.

2. D. Gusfield, Haplotype inference by pure parsimony., in: R. A. Baeza-Yates, E. Chávez, M. Crochemore (Eds.), Combinatorial Pattern Matching (CPM 2003), Proceedings of the 14th Annual Symposium, Vol. 2676 of Lecture Notes in Computer Science, Springer-Verlag, Berlin-Heidelberg, Germany, 2003, pp. 144–155.
3. D. G. Brown, I. M. Harrower, Integer programming approaches to haplotype inference by pure parsimony., IEEE/ACM Transactions on Computational Biology and Bioinformatics 3 (2) (2006) 141–154.
4. K. Kalpakis, P. Namjoshi, Haplotype phasing using semidefinite programming., in: BIBE, IEEE Computer Society, 2005, pp. 145–152.
5. Y.-T. Huang, K.-M. Chao, T. Chen, An approximation algorithm for haplotype inference by maximum parsimony., in: H. Haddad, L. M. Liebrock, A. Omicini, R. L. Wainwright (Eds.), Proceedings of the 2005 ACM Symposium on Applied Computing (SAC 2005), ACM, 2005, pp. 146–150.
6. I. Lynce, J. Marques-Silva, Efficient haplotype inference with boolean satisfiability., in: Proceedings of the 21st National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, AAAI Press, Menlo Park, CA, USA, 2006.
7. A. Graça, J. Marques-Silva, I. Lynce, A. L. Oliveira, Efficient haplotype inference with pseudo-boolean optimization, in: H. Anai, K. Horimoto, T. Kutsia (Eds.),

Algebraic Biology, Second International Conference, AB 2007, Castle of Hagenberg, Austria, July 2-4, 2007, Proceedings, Vol. 4545 of Lecture Notes in Computer Science, Springer-Verlag, Berlin-Heidelberg, Germany, 2007, pp. 125–139.
8. R.-S. Wang, X.-S. Zhang, L. Sheng, Haplotype inference by pure parsimony via genetic algorithm, in: X.-S. Zhang, D.-G. Liu, L.-Y. Wu (Eds.), Operations Research and Its Applications: the Fifth International Symposium (ISORA'05), Tibet, China, August 8–13, Vol. 5 of Lecture Notes in Operations Research, Beijing World Publishing Corporation, Beijing, People Republic of China, 2005, pp. 308–318.
9. G. Lancia, M. C. Pinotti, R. Rizzi, Haplotyping populations by pure parsimony: Complexity of exact and approximation algorithms., INFORMS Journal on Computing 16 (4) (2004) 348–359.
10. C. Blum, A. Roli, Metaheuristics in combinatorial optimization: Overview and conceptual comparison, ACM Computing Surveys 35 (3) (2003) 268–308.
11. L. Di Gaspero, A. Schaerf, EasyLocal++: An object-oriented framework for flexible design of local search algorithms, Software—Practice and Experience 33 (8) (2003) 733–765.
12. M. Birattari, T. Stützle, L. Paquete, K. Varrentrapp, A racing algorithm for configuring metaheuristics, in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002), Morgan Kaufmann Publishers, New York (NY), USA, 2002, pp. 11–18.