Proceedings of WCB09
Workshop on
Constraint Based Methods for Bioinformatics

Alessandro Dal Palù and Sebastian Will

September 20th, 2009, Lisbon (Portugal)

Program Committee

Rolf Backofen, Freiburg Univ., Germany
Pedro Barahona, Univ. Nova de Lisbona, Portugal
Alexander Bockmayr, Freie Univ. Berlin, Germany
Mats Carlsson SICS, Uppsala Sweden
Alessandro Dal Palù (co-chair), Parma Univ., Italy
Agostino Dovier, Udine Univ., Italy
Esra Erdem, Sabanci University, Turkey
François Fages, INRIA Rocquencourt, France
Inês Lynce, INESC-ID Lisboa, Portugal
Neil Moore, Univ. of St Andrews, UK
Enrico Pontelli, NMSU, USA
Sebastian Will (co-chair), Freiburg Univ., Germany

External referees

Magnus Ågren
Luca Bortolussi
Ana Graca

Preface

Modern Life Sciences strongly rely on computational methods. In the last decades we witnessed an exponential growth of data available and to be handled by scientists. These ever increasing demands power the research in bioinformatics, which is consequently a rapidly growing discipline. Many of the computational problems faced by the bio-sciences can be naturally formalized in constraint-systems over finite domains or reals. The contributions to the WCB series of workshop of the last years, starting from 2005, provide an excellent overview over recent approaches for tackling bioinformatics problems using constraint methodology. Constraint techniques proved to be successful for a variety of problems. Some of the discussed topics were sequence analysis, biological systems simulations, protein structure prediction and docking, pedigree analysis, haplotype inference, and many others.

This year, we could select 7 strong workshop contributions out of 8 submissions. The discussed topics will comprise the application of constraint-based methods to SNP selection, biosystems simulation, hidden markov models, haplotype inference, and protein structure prediction. The variety of topics and the continuing high interest in the workshop emphasize once more that constraint-based methods are a very valuable tool for bioinformatics and promise significant advance in a broad application range.

Alessandro Dal Palù
Sebastian Will

Contents

TagSNP selection using Weighted CSP and Russian Doll Search with Tree Decomposition

D. Allouche, S. de Givry, M. Sanchez, T. Schiex

UBIA, UR 875, INRA, F-31320 Castanet Tolosan, France.
{allouche,degivry,msanchez,tschiex}@toulouse.inra.fr

Abstract. The TagSNP problem is a specific form of compression problem arising in genetics. Given a very large set of SNP (genomic positions where polymorphism is observed in a given population), the aim is to select a smallest subset of SNPs which represents the complete set of tagSNP reliably. This is possible because strong correlations existing between neighboring SNPs. Typically, besides minimizing the tagSNP set size (mostly for economical reasons), one also seek a maximally informative subset for the given size, generating different secondary criteria.
This problem, which is also closely related to a set covering problem, can be simply described as a weighted CSP. We report here our experiments with human tag SNP data using a recently designed WCSP algorithm combining the "Russian Doll Search" algorithm with local consistency for cost functions and an active exploitation of the problem structure, through a tree decomposition of the problem.

Introduction

In bioinformatics, uncertainty and variability are usually ubiquitous and combinatorial problems modelling biological objects or phenomenon usually involve a combination of a large number of local uncertainties or correlations which must be incorporated in a global criteria. It is therefore not surprising to see that graphical models (HMM, Bayesian nets, (conditional) Markov random fields...) are massively used in bioinformatics.

The constraint satisfaction problem and its implementation as constraint programming languages represent a deterministic variant of graphical models which allows to represent combinatorial problems on discrete variables. Optimization (if any) is often considered as a second order target that can be reached by explicitly modeling a cost variable which is then iteratively bounded until an optimum is reached. This however requires the use of global constraints that cover all variables, thereby hindering all the problem structure.

In this paper, we consider the TagSNP problem, a specific form of lossy compression problem arising in genetics. Given a very large set of SNP (genomic positions where polymorphism is observed in a given population), the aim is to select a smallest subset of SNPs which represents the complete set of tagSNP reliably. This problem can be naturally defined as a variant of the set covering problem. By modeling the problem as a weighted CSP (or cost function network), we get a model which still offers opportunities to exploit the problem structure described through a tree-decomposition of its graph.

Specifically, this enables the fruitful application of a recent algorithm [12] called "Russian Doll Search with Tree Decomposition" that exploits problem structure for solving cost function networks. As the BTD algorithm [13], this algorithm is based on the identification

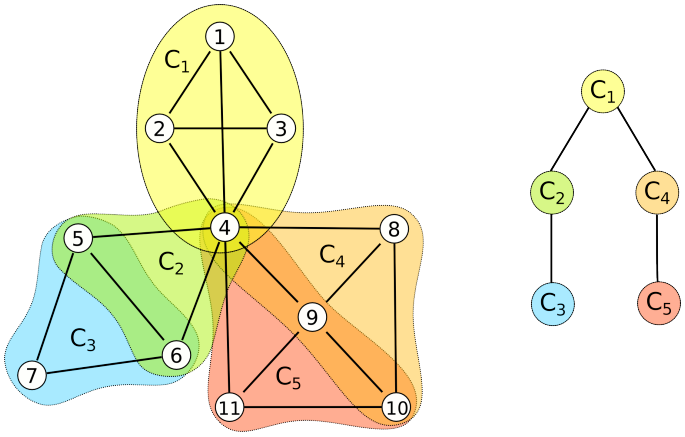


Fig. 1. The graph of a WCSP and its tree decomposition.

of conditionally independent subproblems, which are solved independently, using a lower bound based on local consistency, and whose optimum is cached during Branch and Bound search. The time complexity of the algorithm obtained is only exponential in the largest subproblem size instead of the global problem size. These conditionally independent subproblems are called clusters ad form a tree. The intersection between two clusters, if not empty, is called the separator between these two clusters (see Fig. 1 for an illustration). Such a decomposition can be obtained by computing a so-called tree-decomposition of the problem. The main novelty of our algorithm lies in the incorporation of a "Russian Doll Search" like approach enabling the computation of stronger initial local bounds by inductively solving a relaxation of each subproblem identified in the tree-decomposition.

The algorithm obtained, BTD-RDS, generalizes both RDS and other tree-decomposition based algorithms such as BTD or AND-OR Branch and Bound. As BTD, it uses a restricted dynamic variable ordering which must be compatible with the tree decomposition exploited : a variable from a cluster cannot be instantiated before variables from its parent cluster.

This algorithm has been applied to radio link frequency assignment problem instances defined in the CELAR benchmark [2], closing a very hard frequency assignment instance which has been open for more than 10 years. It has also been used to tackle a problem from bioinformatics, defining a new benchmark for constraint-based approaches : the tagSNP selection problem.

In this paper, we will present this problem with associated instances defining benchmarks and the main results obtained in our experiments.

1 Modeling the TagSNP problem

TagSNP selection occurs in genetics and polymorphism analysis. Single nucleotide polymorphisms, or SNPs, are DNA sequence variations that occur when a single nucleotide (`A`,`T`,`C`,or `G`) in the genome sequence of an individual is altered. For example a SNP might change the DNA sequence `AAGGCTAA` to `AT`**`T`**`GGCTAA`. For a variation to be considered as a

SNP, it must occur in at least 1% of the population. There are several millions SNPs in the 3 billions nucleotides long human genome, explaining up to 90% of all human genetic variation. SNPs may explain a portion of the heritable risk of common diseases and can affect response to pathogens, chemicals, drugs, vaccines, and other agents. The TagSNP problem is a sort of lossy compression problem which consists in selecting a small subset of SNPs such that the selected SNPs, called tag SNPs, will capture most of the genetic information. The goal is to capture a maximally informative subset of SNPs to make screening of large populations feasible [7]. From the combinatorial point of view the TagSNP problem is equivalent to a set covering problem (NP-hard) with additional quadratic criteria.

By sampling a first relatively small population, it is possible to compute a link (correlation) measure r^2 between each pair of SNPs. A tag SNP is considered as representative of another SNP if the two SNPs are sufficiently linked. The simplest TagSNP problem is to select a minimum number of SNPs (primary criteria) such that all SNPs are represented. This is captured by the fact that the r^2 measure between the two SNPs is larger than a threshold θ (often set to $\theta = 0.8$ [3]). We therefore consider a graph where each vertex is a SNP and where edges are labelled by the r^2 measure between pairs of nodes. Edges are filtered if their label is lower than the threshold θ. The graph obtained may have different connected components. The TagSNP problem then reduces to a set covering problem on these components. This simplest variant has been studied in [1] where it is solved using the d-DNNF compiler `c2d` with good results. A polynomial approach exists for simpler problems [6]

In practice the number of optimal solutions may still be extremely large and secondary criteria are considered by state-of-the-art tools such as FESTA [10]. Between tag SNPs, a low r^2 is preferred, to maximize tag SNP dispersion. Between a non tag SNP and its representative tag SNP, a high r^2 is preferred to maximize the representativity. To optimize these criteria, FESTA uses two incomplete algorithms, the simplest is called FESTA-greedy, and it uses a simple reedy approach. The second, called FESTA-hybrid combined the greedy approach with a limited exhaustive approach.

For a given connected graph $G = (V, E)$, we build a binary weighted CSP with integer costs capturing the TagSNP problem with the above secondary criteria. For each SNP i, two variables i_s and i_r are used. i_s is a boolean variable that indicates if the SNP is selected as a tag SNP or not. The domain of i_r is the set of neighbors of i together with i itself. It indicates the representative tag SNP which covers i. For a SNP i, hard binary cost functions (with 0 or infinite costs) enforces the fact that $i_s \Rightarrow (i_r = i)$. Similar hard cost functions enforce $(i_r = j) \Rightarrow j_s$ with neighbor SNPs j in G. A unary cost function on every variable i_s generates an elementary cost U if the variable is *true*. The resulting weighted CSP captures the set covering problem defined by TagSNP.

To account for the representativity, a unary cost function is associated with every variable i_r that generates cost when $i_r \neq i$. In this case, the cost generated is $\lfloor 100.\frac{1-r^2_{i,i_r}}{1-\theta} \rfloor$. For dispersion between SNPs i and j, a binary cost function between the boolean i_s and j_s is created which generates a cost of $\lfloor 100.\frac{r^2_{ij}-\theta}{1-\theta} \rfloor$ when $i_s = j_s =$ *true* The resulting WCSP captures both dispersion and representativity. In order to keep these criteria as secondary, we just use a large enough value for U (the elementary cost used for tag selection).

This problem is similar to a set covering problem with additional binary costs. Such secondary criteria are ignored by [1]. Here, `c2d` yields a compact compiled representation of the set of solutions of the pure set covering problem, but the number of solutions is so

huge (typically more than billions) that applying the second criteria on solutions generated by `c2d` would be too expensive. A direct compilation of the criteria in the d-DNNF does not seem straightforward and would probably necessitate a Max-SAT formulation as the authors acknowledge in their conclusion.

2 Experiments

The algorithms tested (BTD and RDS-BTD) have been implemented in `toulbar2` C++ solver[1]. Note than when the tree decomposition used reduced to a single cluster, BTD is equivalent to a Depth First Branch and Bound algorithm.

The *min domain / max degree* dynamic variable ordering, breaking ties with maximum unary cost, is used inside clusters (BTD and RDS-BTD) and by DFBB. The dynamic variable ordering heuristic is modified by a conflict back-jumping heuristic as suggested in [9]. EDAC local consistency is enforced [5] during search. Tree decompositions are built using the Maximum Cardinality Search (MCS [11]) heuristic, with the largest cluster used as root. A variable ordering compatible with the rooted tree decomposition used is used for DAC enforcing [4].

All the solving methods exploit a binary branching scheme. If $d > 10$, the *ordered* domain is split in two parts (around the middle value), else the variable is assigned to its EDAC fully supported value or this value is removed from the domain. In both cases, it selects the branch which contains the fully supported value first, except if a previous solution is available (the corresponding value is used in this case). Reported CPU times correspond to finding the optimum and proving its optimality.

The instances we considered have been derived from human chromosome 1 data provided by courtesy of Steve Qin [10]. Two values, $\theta = 0.8$ and 0.5 have been tried. For $\theta = 0.8$, a usual value in tag SNP selection, $43,251$ connected components are identified among which we selected the 82 largest ones. These problems, with 33 to 464 SNPs, define WCSP with domain sizes ranging from 15 to 224 and are relatively easy. Solving them to optimality selects 359 tag SNPs in 2h37' instead of 487 in 3' for FESTA-greedy (21% improvement) or 370 in 39h17' for FESTA-hybrid (3% improvement, 15-fold speedup).

To get more challenging problems, we lowered θ to 0.5. This defined $19,750$ connected components, among which 516 are not solved to optimality by FESTA. We selected the 25 largest one. These problems, with 171 to 777 SNPs have graph densities between 6% and 37%. They define WCSP with max domain size ranging from 30 to 266 and include between 8000 to $250,000$ cost functions. The decomposability of these problems, estimated by the ratio between the treewidth of the original MCS tree-decomposition (without any cluster merging, i.e. with $s_{max} = +\infty$) and the number of variables varies from 14% to 23%.

In theory, algorithms exploiting tree-decompositions are only exponential in the maximum cluster size and the ideal tree-decomposition that should be used is therefore a tree decomposition minimizing the size of the largest cluster (also called the treewidth or induced width of the problem). In practice however, when a Branch and Bound like approach such as BTD is used, small separators are also attractive because the worst-case space complexity of the algorithm is exponential in the separator size. Even if this space complexity is usually far from being reached because of the pruning induced by lower bounds, small

[1] `http://mulcyber.toulouse.inra.fr/projects/toulbar2`, version 0.8.

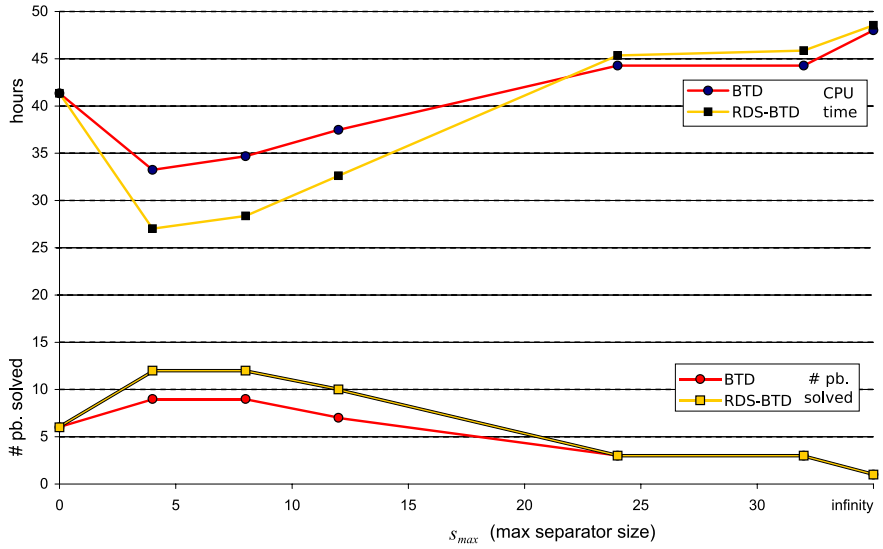


Fig. 2. CPU-time and number of solved instances for different values of the maximum separator threshold (s_{max}).

separators are also attractive because the removal of large separators creates larger clusters that gives more freedom to dynamic variable ordering. There is therefore a compromise to reach : small clusters are desirable but large separators should be avoided. Starting from an original tree-decomposition, it is always possible to reach a decomposition with a maximum separator size below a given threshold $s_{\max}$ by just merging any pair of clusters which has a separator of size above $s_{\max}$ [8].

All the problems have been tackled with an initial upper bound found by FESTA-greedy on 2.8 Hz CPU with 32 GB RAM. To better show the importance of bounded separator size (using s_{max}), we considered values ranging from 0 (DFBB), 4, 8, 12, 24, 32 to $+\infty$ for both BTD and RDS-BTD. We report both the number of problems solved within a 2-hour limit per instance and the total amount of CPU time used (an unsolved instance contributes for 2 hours).

Using $s_{max} = 4$, our implementation improves the compression ratio of FESTA-greedy by 15% (selecting 2952 tag SNPs instead of 3477 for the $516 - 13$ solved instances). Note that the differences in CPU time between BTD and RDS-BTD would increase if a larger time limit had been used. From a practical viewpoint, the criterion of the TagSNP problem could be further refined to include : sequence annotation information (*e.g.* preferring tag SNPs occurring in genes), and measures between triplets of markers as proposed in [1] (SNPs covered by a pair of tag SNPs). The good performances of RDS-BTD may allow to tackle this more complex problem with realistic $\theta = 0.8$.

Figure 3 presents the evolution of the ratio between the tree width and the problem size for different values of s_{max}. All instances ($\theta = 0.5$) solved at least once during the experimentation are shown in this graph.

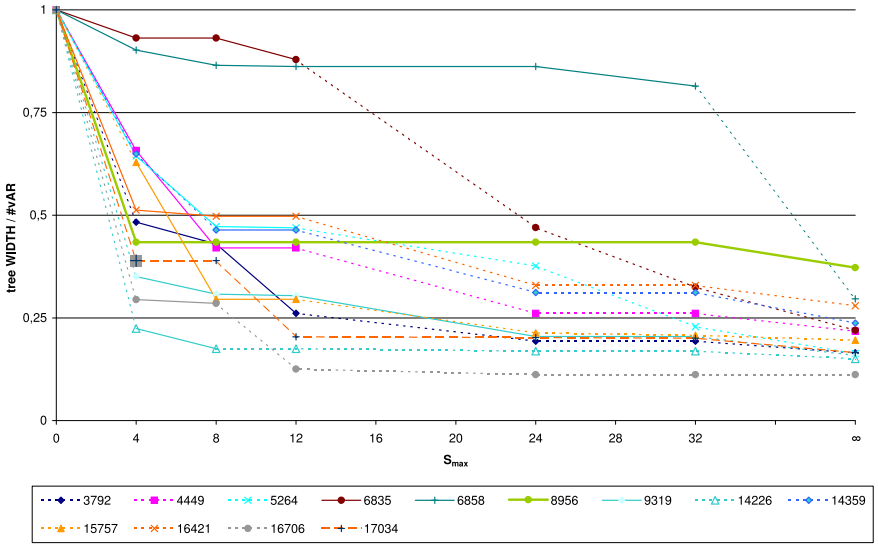


Fig. 3. Evolution of treewidth normalized by variable number for different maximum separator size threshold s_{max}.

Problems which are successfully solved (with an optimality proof) are materialized as continuous line,while dashed lines represent failures. The instance 17034, which could be solved only for $s_{\max} = 4$ is represented by a gray square.

The Figure 3 allows to materialize the 12 instances which could be solved for $s_{max} = \{4, 8\}$. This underlines the fact that the nature of each solved instance may be different depending of the s_{max} value. For example, instance 17034 is only solved for $s_{max} = 4$, which is balanced by the resolution of instance 14359, solved only on the interval 8 - 12 of $s_{\max}$.. Overall, on the whole range of variation of $s_{\max}$, 13 instances are solved but only 12 can be solved for an optimal choice of $s_{max} = 4, 8$. This may plead for an adaptive choice of the parameter $s_{max} = 4, 8$. Moreover, a significant decrease of the normalized treewidth is often correlated with the occurrence of a successful resolution.

Indeed, for most of the tested instances, reflecting the structure of the problems, the curves show an important decrease for $s_{max} \in \{0, 8\}$, which is very likely one of the significant events explaining the high number of successful resolution in this range. The decrease of the treewidth is then lower. It gradually reaches ($s_{max} > 12$), a plateau with a gentle slope, asymptotically reaching the ratio corresponding to a full decomposition of the problem (without cluster merging). In the experimentation, this area is often associated to failure.

This behavior is also reflected in the resolution speedup. Considering for example instance 15757, an optimal solution is found for $s_{max} = \{4, 8, 12, 24\}$ with respective cpu-time of of 514, 7, 299 and 3810 seconds. Simultaneously, the tree width ratio varies from 0.63 for $s_{max} = 4$ to 0.30 for $s_{max} = \{8, 12\}$ and then to about 0.2 for $s_{max} = 24$. The optimal resolution time occurs for $s_{max} = 8$. Several instances follow a similar behavior. Most of solved instances are optimally resolved for a specific s_{max} value . This behavior

follows very likely from the compromise between the gains provided by the decomposition and the losses related to a decrease in the freedom of choice during search in dynamic variables ordering in smaller clusters.

3 Conclusion

In this paper , we specifically addressed the tagSNP selection problem, a variant of the covering problem coming from bioinformatics. Our WCSP model allows to take into account additional criteria such as dispersion and representativity. The data used in our experimentation is likely the largest data set available in the public domain. With a usual threshold ($\theta = 0.8$), our model provides good results. But the situation quickly become more challenging if less stringent (more artificial) filtering conditions such as $\theta = 0.5$ are used. In the near future, we intend to extend our study to include an integer linear programming model solved using CPLEX.

Beside this, from the methodological point of view, this work underlines the fact that the practical exploitation of tree decompositions to solve structured combinatorial optimization problems is not straightforward. Our experiments show that, even on problems that have a nice visible structure, it is often very profitable and sometimes crucial to restrict the maximum size of the separators of the decomposition used. Theory says that separator size influences the space complexity of structure-based algorithms such as BTD and RDS-BTD but in practice, the improvement in efficiency is mostly explainable by the added freedom in variable ordering allowed by cluster merging, an observation consistent with JÃ©gou et al. conclusions [8].

Our current algorithm still leaves areas for improvements. A attractive direction would be to try to design a dedicated heuristic allowing for dynamic adaptation of the maximum separator size of the decomposition. This could be based on the specific structure of the input instance and could be achieved, for a given decomposition, by analyzing treewidth variations as a function of s_{max}. Another more challenging direction would be to provide decomposition algorithms aimed at producing small separators in the graph decomposition.

Acknowledgments This research has been partly funded by the French *Agence Nationale de la Recherche* (`STALDECOPT` project).

References

1. A. Choi, N. Zaitlen, B. Han, K. Pipatsrisawat, A. Darwiche, and E. Eskin. Efficient Genome Wide Tagging by Reduction to SAT. In *Proc. of WABI-08*, volume 5251 of *LNCS*, pages 135–147, 2008.
2. B. Cabon, S. de Givry, L. Lobjois, T. Schiex, and J.P. Warners. Radio Link Frequency Assignment. *Constraints*, 4(1):79–89, 1999.
3. C. S. Carlson, M. A. Eberle, M. J. Rieder, Q. Yi, L. Kruglyak, and D. A. Nickerson. Selecting a maximally informative set of single-nucleotide polymorphisms for association analyses using linkage disequilibrium. *Am. J. Hum. Genet.*, 74(1):106–120, 2004.
4. M. Cooper and T. Schiex. Arc consistency for soft constraints. *Artificial Intelligence*, 154:199–227, 2004.

5. S. de Givry, M. Zytnicki, F. Heras, and J. Larrosa. Existential arc consistency: Getting closer to full arc consistency in weighted CSPs. In *Proc. of IJCAI-05*, pages 84–89, Edinburgh, Scotland, 2005.
6. E. Halperin, G. Kimmel, and R. Shamir. Tag SNP selection in genotype data for maximizing SNP prediction accuracy. *Bioinformatics*, 21 Suppl 1:i195–203, 2005.
7. J.N. Hirschhorn and M.J. Daly. Genome-wide association studies for common diseases and complex traits. *Nature Reviews Genetics*, 6(2):95–108, 2005.
8. P. Jégou, S. N. Ndiaye, and C. Terrioux. Dynamic management of heuristics for solving structured CSPs. In *Proc. of CP-07*, pages 364–378, Providence, USA, 2007.
9. C. Lecoutre, L. Sais, S. Tabary, and V. Vidal. Last conflict based reasoning. In *Proc. of ECAI-2006*, pages 133–137, Trento, Italy, 2006.
10. Z. S. Qin, S. Gopalakrishnan, and G. R. Abecasis. An efficient comprehensive search algorithm for tagsnp selection using linkage disequilibrium criteria. *Bioinformatics*, 22(2):220–225, 2006.
11. D.J. Rose. Tringulated graphs and the elimination process. *Journal of Mathematical Analysis and its Applications*, 32, 1970.
12. M. Sanchez, D. Allouche, S. de Givry, and T. Schiex. Russian doll search with tree decomposition. In *In Proc. of IJCAI'09*, Pasadena, USA, July 11-17 2009.
13. C. Terrioux and P. Jégou. Bounded backtracking for the valued constraint satisfaction problems. In *Proc. of CP-2003*, pages 709–723, Kinsale, Ireland, 2003.

Dynamical Compartments in stochastic Concurrent Constraint Programming

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

Abstract. We deal with the problem of modeling in **sCCP**, a stochastic extension of concurrent constraint programming, biological systems with a dynamical compartmentalization. The basic idea is to exploit the constraint store to program a library of functions implementing such functionalities. In order to simplify the code, we extend the language with a primitive to probabilistically sample from a given finite distribution.

1 Introduction

Systems biology is changing the approach of biology, towards an integration of wet lab experiments with mathematical modeling and computational analysis. This discipline, as a byproduct, is giving new impulses to the development of mathematics of complex systems and of computational analysis techniques. One promising field, genuinely coming from computer science, is the use of high level, concurrent languages (i.e. Process Algebras [6,7]) to describe such systems and to generate automatically mathematical models. Analysis techniques can be applied both at the syntactic and at the semantic level, and range from static analysis to simulation and model checking.

Among process algebras, we have focussed on stochastic Concurrent Constraint Programming (**sCCP** [2]), a stochastic extension of CCP [18], and studied its application as a modeling tool for biological systems [4]. Our claim is that **sCCP** offers a powerful and flexible framework for this task. Differently from classical process algebras used in biology, like stochastic π-calculus, **sCCP** requires a modeling style in which the state of the system is described as a set of constraints in the constraint store, while agents correspond to the different interaction capabilities which can alter the system's state.

The key ingredient is the constraint store itself, which can be manipulated in sophisticated manners, by exploiting the simple fact that it can be programmed in a Prolog-like style. As a consequence, by suitably defining predicates and constraints, we can tailor **sCCP** to specific domains without changing the language itself. One example in this direction is the modeling of complex biochemical networks described graphically by Molecular Interaction Maps [1].

An important issue in biology is that systems are highly compartmentalized and organized in a hierarchical manner: cells contain a wide variety of organelles and other substructures which are used to separate and spatially organize the activities. Such organization is not static: there is a constant flow of material among those ambients, and the compartments themselves move and reorganize dynamically. In order to deal with such crucial problems, several extensions of process algebras have been proposed, like Bioambients [16], Brane Calculi [5], and others [21]. The problem in extending process algebras is that different biological domains require different extensions, making difficult their integration into a unified framework.
We take here a different direction, exploiting extensibility of **sCCP**: we will model dynamic compartmentalization by describing the compartment structure in the constraint store and by defining different agents corresponding to different actions: from local reactions, to flow of material between compartments, to dynamic rearrangement of compartments. All this can be done quite straightforwardly, although we will extend the language with another primitive, allowing to perform instantaneous probabilistic choices. This simplifies quite a lot the definition of the agents, especially the protocols to update the store.

The paper is organized as follows: in Section 2, we introduce **sCCP** and its extension. In Section 3, instead, we show how to encode dynamical compartmentalization, discussing a simple example. As we are in a preliminary stage, we still do not have a working implementation, hence we cannot show simulation's results. Finally, we draw conclusions on Section 4.

2 Stochastic Concurrent Constraint Programming

Concurrent Constraint Programming (CCP [17]) is a process algebra which can be seen as a concurrent extension of Prolog. Its definition is organized around two concepts: the constraint store and the agents. The constraint store $\mathcal{C}$ is the repository of information and is is able to perform computations, too. Informational units are the constraints, which are interpreted first-order logical formulae, stating relationships among variables (e.g. $X = 10$ or $X + Y < 7$). Moreover, constraints of the store can be defined using the computational machinery of Prolog [19,17].
Agents, instead, are the basic actors that modify the state of the constraint store by adding new information ($\mathsf{tell}()$) and by checking entailment of relationships ($\mathsf{ask}()$). The communication mechanism among agents is therefore asynchronous, as information is exchanged through global variables. Other basic operators are at disposal to define agents: non-deterministic choice, parallel composition, procedure call, plus the declaration of local variables.

Using a CCP-like language to model the dynamics of biological systems requires to store time-varying quantities [4], which can be modeled as *stream variables*, i.e. growing lists with an unbounded tail.

The stochastic extension of CCP we are considering (sCCP [2,4])[1] is obtained by adding a stochastic duration to all instructions interacting with the constraint store $\mathcal{C}$, i.e. $\mathsf{ask}()$ and $\mathsf{tell}()$. Each instruction has an associated random variable, exponentially distributed with rate given by a *function* associating a real number to each configuration of the constraint store: $\lambda : \mathcal{C} \to \mathbb{R}^+$. More precisely:

[1] There are other proposed probabilistic extensions [10,15], which are however not suited to model biological systems, as they do not have a notion of continuous time.

Definition 1. *An **sCCP** program is a tuple $(A, \mathcal{D}, \mathcal{C}, init)$, where*

1. *A is the initial agent and $\mathcal{D}$ is the set of definitions, defined according to the following grammar:*

$$\mathcal{D} = \varepsilon \mid \mathcal{D}.\mathcal{D} \mid p(\mathbf{X}) : -A$$
$$A = \mathbf{0} \mid M \mid \mathsf{local}\ X.A \mid A \parallel A$$
$$M = \pi.G \mid M + M$$
$$\pi = \mathsf{tell}_\lambda(c) \mid \mathsf{ask}_\lambda(c)$$
$$G = \mathbf{0} \mid \mathsf{tell}_\infty(c).G \mid \mathsf{choose}(X, D, P).G \mid p(\mathbf{Y}) \mid M \mid \mathbf{local}\ X.G \mid G \parallel G$$

2. *$\mathcal{C}$ is the constraint store, which is closed by conjunction and provided also with a computable entailment relation $\vdash$.*
3. *$init \in \mathcal{C}$ is the initial state of the constraint store.*

In the previous definition, $\mathsf{tell}_\lambda(c)$ is the addition of constraint c to the store at store-dependent rate λ, while $\mathsf{tell}_\infty(c)$ is the addition of c at infinite rate. Note that recursive call is always guarded by a stochastic action, hence we cannot have an infinite unfolding in a single time instant. The version of **sCCP** considered here differs from the standard one [2,4] by the addition of a new basic construct, namely $\mathsf{choose}(X, D, P)$. Essentially, it is a primitive instruction which allows to choose an element from the finite set D (represented as a list) with probability given by P (which is a list of rational numbers of the same size of D, adding up to one). Moreover, $\mathsf{choose}(X, D, P)$ is an instantaneous action. We pinpoint that this instruction is similar to the one used in [10] to introduce probabilities in CCP.

The underlying semantic model of the language (defined via structural operational semantic, cf. [2]) is a Continuous Time Markov Chain [13] (CTMC), i.e. a stochastic process whose temporal evolution is a sequence of discrete jumps among states in continuous time. States of the CTMC correspond to configurations of the sCCP-system, consisting in the current set of processes and in the current configuration of the constraint store. The next state is determined by a race condition between all active instructions such that the fastest one is executed, like in stochastic π-calculus [14] or PEPA [12]. More specifically, there are two transition relations, one dealing with instantaneous actions and one dealing with stochastic transitions.

In [3,4] we argued that sCCP can be conveniently used for modeling a wide range of biological systems, like biochemical reactions, genetic regulatory networks, the formation of protein complexes, and the process of folding of a protein. In fact, while maintaining the compositionality of process algebras, the presence of a customizable constraint store and of variable rates gives a great flexibility to the modeler. Such features have been exploited in [1] to provide a simple encoding of Molecular Interaction Maps (a graphic formalism to describe complex combinatorial biochemical networks). In the next section we will show how to describe a dynamical compartment structure.

3 Modeling Compartments in sCCP

In this Section we will discuss how to build **sCCP** models of biological system in presence of compartments (called throughout also ambients). More specifically, we are interested in describing dynamic compartments, which can be created, destroyed, and moved at run time.

We will adopt here the "classical" reaction-centric modeling style [4], in which the physical system (objects, ambient structure, etc) is described in the constraint store, while agents model the different interactions that can modify the system's state. As far as we are interested in dynamics, all the quantities of the store that are subject to evolution in time are modeled by stream variables. The description of each compartment requires some information, concerned with its location, its shape, and its content.

- The *location* of an ambient is specified in terms of an inclusion relationship, hence we have a hierarchy of ambients where ambients contain other ambients. Formally, compartments are organized by inclusion in a tree structure. Additional information can be added as well, like spatial coordinates or a neighborhood structure, but we do not pursue this generalization at this stage. Neglecting spatial information essentially means that we are assuming compartments to be homogeneously distributed in space.
- The *shape* of an ambient can be described at several levels of detail. Here we will consider only the *volume*, which is needed in the modeling of biochemical reactions.
- An ambient can contain other ambients and several molecular species, which are described and counted by (stream) variables.

Compartment description. The compartmentalization structure is described by suitable predicates and stream variables of the constraint store. First of all, we assume a finite set of different compartment types, like cells, organelles, vesicles, and so on, and a finite set of molecular species.

We assume to have the following predicates in the store:

- compartment_list(Type,Instances). This predicate contains the list of all compartments (Instances) for each type (Type). We assume that each compartment is identified by a unique name. This predicate is required to allow a rapid access to all compartments of a given type.
- compartment(Name,Parent,Volume,Children). For each compartment (Name), we store three stream variables: Parent contains the name of the parent ambient (which may be NULL, for the root ambient), Volume is the compartment's volume, and Children is the list of children ambients (which can be empty). We describe this structural information by stream variables, because the hierarchical structure can change at runtime. Also the volume may be modified at run-time.
- molecule(Species,Compartment,N). For each molecular species (Species), we store the number of molecules N contained in each compartment (Compartment).

Intra-compartment reactions. In previous **sCCP** models, reactions are described by an **sCCP**-agent, which fires at a rate proportional to the rate of the reaction and updates the store as prescribed. In the setting of a multi-compartment model, there is a problem: a reaction can fire in all compartments having enough reactants. The naive solution would be that of having a reaction agent in each compartment, but this creates problems in keeping track of the compartment structure's changes. A more efficient solution can be obtained by exploiting the new choice operator introduced in the language. To fix the ideas, consider the second order reaction $\rho : a + b \rightarrow_\lambda z$, with rate given by the real function $\lambda = \lambda(A, B, V)$, depending on the the numerousness A, B of molecules a, b and on the volume V of the compartment in which the reaction takes place [9]. The reaction is modeled by the following agent:

$$\begin{aligned}
\mathsf{reaction}([a,b],[z],\lambda) \text{ :- } & \mathsf{ask}_{\lambda^g}(\text{enabled}).\\
& \mathsf{local}\ D.\mathsf{local}\ P.\\
& \mathsf{tell}_\infty(\text{active_compartments_reaction}([a,b],D,P)).\\
& \mathsf{choose}(X,D,P).\\
& \mathsf{tell}_\infty(\text{update_reaction}(X,[a,b],[z])).\\
& \mathsf{reaction}([a,b],[z],k)
\end{aligned}$$

We explain in more detail the mechanism:

- The first instruction of the reaction is a (stochastically timed) ask. The agent checks if the reaction is enabled, by seeing if in the store there is a compartment containing at least one molecule of species a and one of species b. The rate of such action, indicated by λ^g, is computed by adding up the rates of ρ for each compartment in which the reaction is active, hence it is the global rate of seeing a ρ-type reaction somewhere in the system. If $c_1, \ldots, c_k$ are all the compartments of the system, A_{c_i}, B_{c_i} represent the quantity of species a and b in compartment c_i, and V_i is the volume of c_i, then $\lambda^g = \sum_{c_i} \lambda(A_{c_i}, B_{c_i}, V_i)$.
- If the agent wins the race condition and executes, then it will compute the list of compartments in which reaction ρ is active, by calling the constraint $\text{active\_compartments\_reaction}([a,b],D,P)$. Such list is stored in the local variable D, while P is the list of the rates of ρ within each such compartment, normalized by λ^g (hence it is a probability distribution on active compartments). The call of choose determines in which compartment X ρ actually fires.
- The store is updated consistently, by reducing the number of a and b molecules in the compartment X, and increasing the number of the product z of the reaction in X. Finally, the agent recursively calls itself.

It can be proven that this agent is equivalent to a model in which there is a ρ-reaction agent in each compartment c, executing the reaction only within c.

Flow among compartments. The flow of substances in and out cellular compartments is a crucial biological process. For instance, the flow of ions in neurons is used to create and propagate electrical signals. Modeling flow of molecules in **sCCP** is done similarly to intra-compartment reactions. As an example, consider a compartment type c, for instance a cell, and a molecule a which can enter into c at a certain rate λ_{in}. Similarly to reactions, we define an agent that can execute at the global rate of inflow λ^g_{in} of a into any compartment of type c, then selecting the actual compartment using a $\mathsf{choose}(\cdot,\cdot,\cdot)$ agent and performing the required store updates (decreasing the amount of a outside and increasing a inside).[2]

$$\begin{aligned}
\mathsf{inflow}(a,c,\lambda_{in}) \text{ :- } & \mathsf{ask}_{\lambda^g_{in}}(\text{enabled}).\\
& \mathsf{local}\ D.\mathsf{local}\ P.\\
& \mathsf{tell}_\infty(\text{active_compartments_inflow}(c,a,D,P)).\\
& \mathsf{choose}(X,D,P).\\
& \mathsf{tell}_\infty(\text{update_inflow}(X,a)).\\
& \mathsf{inflow}(a,c)
\end{aligned}$$

[2] Note that, in order to identify the parent ambient, we simply have to look at the information stored in the predicate compartment(Name,Parent,Volume,Children).

Outflow and inter-compartment reactions (like the binding of a molecule to a receptor membrane protein) can be modeled similarly.[3]

Compartment dynamics. Modeling dynamics at the compartment level, like moving compartments in and out, merging them, destroying them or creating new ones, requires suitable bookkeeping of the constraint store. For instance, moving an ambient inside another ambient requires to modify the (stream) variable storing the parent of the moved ambient and to modify the lists of children of the new and previous parent ambients. As an example, consider the entrance of an ambient of type $\mathbf{c_{enter}}$ inside a sibling ambient of type $\mathbf{c_{accept}}$.

$$
\begin{aligned}
&\mathsf{enter}(\mathbf{c_{enter}},\mathbf{c_{accept}},\lambda_{enter}) \text{ :- } \mathsf{ask}_{\lambda^g_{enter}}(\mathrm{enabled}).\\
&\qquad \mathsf{local}\ D.\mathsf{local}\ P.\\
&\qquad \mathsf{tell}_\infty(\mathrm{active\_compartment\_pairs\_enter}(\mathbf{c_1},\mathbf{c_2},D,P)).\\
&\qquad \mathsf{choose}(X,D,P).\\
&\qquad \mathsf{tell}_\infty(\mathrm{update\_enter}(X)).\\
&\qquad \mathsf{enter}(\mathbf{c_{enter}},\mathbf{c_{accept}})
\end{aligned}
$$

In this case, the agent executes with a rate λ^g_{enter}, which is the global entrance rate, computed by adding the entrance rate for each pair of $\mathbf{c_{enter}} - \mathbf{c_{accept}}$ ambients which are siblings (having the same parent), and for which the entrance is enabled (model dependent conditions must be satisfied). Then, as usual, the set of all active pairs of ambients in computed, and one pair is chosen proportionally to their entrance rate. Finally, the store is updated consistently, as explained above.

Exit and merging can be modeled analogously, say by $\mathsf{expel}(\mathbf{c_{exit}},\mathbf{c_{expel}},\lambda_{expel})$ and $\mathsf{merge}(\mathbf{c_1},\mathbf{c_2},\mathbf{c_{merged}},\lambda_{merge})$, the main difference being the way the store is modified to account for the modified inclusion relationships. In particular, in case of a merge, we must ensure that the content of the two merged ambients is added. Destroying a compartment ($\mathsf{destroy}(\mathbf{c},\lambda_{destroy})$) is also quite straightforward, just some care is needed to add the substances contained in the dissolved ambient to its parent compartment.

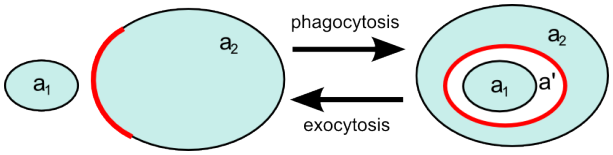


Fig. 1. Schematic representation of phagocytosis and exocytosis. The portion of the membrane in red envelops the phagocytosed ambient a_1, and it merges again to the cell membrane on exocytosis.

Indeed, more complex biologically activities involving compartments can also be modeled. For instance, consider phagocytosis (cf. Figure 1), the process with which a cell captures an external entity by enveloping it into the cell's membrane. This operation can be

[3] The rate function can be quite complex. For instance, in modeling osmosis in plants, i.e. the flow of water in and out the cell, the rate depends on the concentration inside and outside the cell and on volume-dependent pressure effects, cf. [20].

modeled by the agent phagocytosis($\mathbf{c_{eaten}}, \mathbf{c_{eater}}, \mathbf{c_{wrapper}}, \lambda_{phago}$), which works similarly to agent enter: a $\mathbf{c_{eaten}}$-ambient is phagocytosed by a $\mathbf{c_{eater}}$-ambient and enveloped by a newly created $\mathbf{c_{wrapper}}$-ambient. In case we are modeling also some membrane proteins, like receptors or channels,[4] part of them may be moved to the newly created compartment. Exocytosis, the reverse operation of phagocytosis (cf. Figure 1), can be modeled in a similar, but opposite, manner, by the agent exocytosis($\mathbf{c_{expelled}}, \mathbf{c_{expeller}}, \lambda_{exo}$). Other operations of biological membranes, like pinocytosis, budding, and so on, see [5] can be easily described, too.

3.1 An example: a virus entering a cell

We discuss here a very simple example, taken from [5], in which we model a virus attacking a cell and injecting inside it its viral RNA. Schematically, the process starts with the cell phagocytosing the virus, whose membrane resembles the cellular one (in fact, it is created from it in the later phase of infection). Once the virus is inside, it is targeted by an endosome, a small organelle that usually is involved in capturing the material which enters the cell. The endosome merges with the virus (which is encapsulated in a piece of the cell membrane), but is then deceived by special viral proteins on the surface of the virus, which trigger an exocytosis step. Hence, the virus nucleo-capside is released into the cell cytosol and then dismantled, freeing viral RNA in the cell.

The virus is thus modeled as a compartment, **virus**, containing another compartment, **nucleo-capside**, which contains viral RNA inside (molecular species: **vRNA**). Each virus contains also some of the membrane proteins involved in the process (molecular species, **vProt**). The other compartments types required for the model are the **cell**, the **endosome**, the vesicle containing the virus after phagocytosis (**virus-vesicle**), and the virus-endosome complex (**virus-endosome**).

In addition to the definition of suitable constraints, storing information about compartments and substances involved, the previous operations can be specified by following parallel composition of agents:

$$\begin{gathered}\text{phagocytosis}(\textbf{virus},\textbf{cell},\textbf{virus-vesicle},\lambda_p) \parallel \\ \text{merge}(\textbf{virus-vesicle},\textbf{endosome},\textbf{virus-endosome},\lambda_m) \parallel \\ \text{exocytosis}(\textbf{nucleo-capside},\textbf{virus-endosome},\lambda_e) \parallel \text{destroy}(\textbf{nucleo-capside},\lambda_d).\end{gathered}$$

4 Conclusions

Compartmentalization is one of the most relevant features of biological systems, yet one of the most difficult to express, due to the dynamical evolution of the hierarchical structure. This problem has been tackled by defining dedicated languages or by extending existing ones.

In this paper, we showed how to deal with dynamic ambients in a simple manner in **sCCP**, a stochastic extension of concurrent constraint programming. What is required is

[4] Membrane proteins, in the simplest case, are modeled as molecular species belonging to the compartment enclosed by the membrane. A more complex and faithful description can be achieved by modeling the membrane as a dedicated compartment, containing membrane proteins.

a simple programming of the constraint store, by defining suitable constraints describing the hierarchical structure and encoding other structural and spatial information. Most importantly, such activity can be done just once, defining a *library of constraints* and agents that can be used to describe a wide variety of systems, and could be extended with minor efforts. Hence, our approach provides an *easy-to-use framework*, as all technical details are naturally hidden from the standard user, which takes the constraint library as a black box.

Another point in favor of **sCCP** is that constraints can be used to describe different features of biological systems, like combinatorial biochemical networks, in the same framework, making straightforward the problem of *integrating different modeling domains*.

Finally, **sCCP** has at disposal different semantics apart from the standard discrete and stochastic one, based on ordinary differential equations and on (stochastic) hybrid systems. This allows to use, at least in principle, a wide variety of analysis techniques, ranging from numerical simulation to model checking.

In this direction, however, there are some open theoretical issues. In fact, all the approximate semantics are defined on a subset of **sCCP** in which the store is flat, as it contains only a set of integer-valued stream variables. Hence, to use them in a more general setting, we should set up a method to "flatten" the description of the constraint store, characterizing the cases in which this could be done.

Another fundamental issue which needs to be faced is the implementation of an efficient general purpose simulator, integrating efficient stochastic simulation and efficient management of the constraint store and of stream variables. The current simulator, implemented in Prolog, is not sufficiently efficient to deal with the massive computations required in simulation and analysis of biochemical systems.

References

1. L. Bortolussi, S. Fonda, and A. Policriti. Constraint-based simulation of biological systems described by molecular interaction maps. In *Proceedings of IEEE conference on Bioinformatics and Biomedicine, BIBM 2007*, 2007.
2. L. Bortolussi. Stochastic concurrent constraint programming. In *Proceedings of 4th International Workshop on Quantitative Aspects of Programming Languages (QAPL 2006)*, volume 164 of *ENTCS*, pages 65–80, 2006.
3. L. Bortolussi. *Constraint-based approaches to stochastic dynamics of biological systems*. PhD thesis, PhD in Computer Science, University of Udine, 2007. Available at `http://www.dmi.units.it/~bortolu/files/reps/Bortolussi-PhDThesis.pdf`.
4. L. Bortolussi and A. Policriti. Modeling biological systems in concurrent constraint programming. *Constraints*, 13(1), 2008.
5. L. Cardelli. Brane calculi. In *Proceeding of CMSB 2004*, 2004.
6. L. Cardelli. Abstract machines of systems biology. *Transactions on Computational Systems Biology*, III, LNBI 3737:145–168, 2005.
7. M. Calder, S. Gilmore, and J. Hillston. Modelling the influence of rkip on the erk signalling pathway using the stochastic process algebra PEPA. *Transactions on Computational Systems Biology*, 4230:1–23, 2006.
8. D.T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J. of Computational Physics*, 22, 1976.
9. D.T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. of Physical Chemistry*, 81(25), 1977.

10. V. Gupta, R. Jagadeesan, and P. Panangaden. Stochastic processes as concurrent constraint programs. In *Proceedings of POPL'99*, 1999.
11. D. Gillespie and L. Petzold. *System Modelling in Cellular Biology*, chapter Numerical Simulation for Biochemical Kinetics. MIT Press, 2006.
12. J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.
13. J. R. Norris. *Markov Chains*. Cambridge University Press, 1997.
14. C. Priami. Stochastic π-calculus. *The Computer Journal*, 38(6):578–589, 1995.
15. A. Di Pierro and H. Wiklicky. An operational semantics for probabilistic concurrent constraint programming. In *Proceedings of IEEE Computer Society International Conference on Computer Languages*, 1998.
16. A. Regev, E.M. Panina, W. Silverman, L. Cardelli, and E.Y. Shapiro. Bioambients: an abstraction for biological compartments. *Theoretical Computer Science*, 325(1):141–167, 2004.
17. V. A. Saraswat. *Concurrent Constraint Programming*. MIT press, 1993.
18. V. Saraswat and M. Rinard. Concurrent constraint programming. In *Proceedings of 18th Symposium on Principles Of Programming Languages (POPL)*, 1990.
19. L. Shapiro and E. Y. Sterling. *The Art of PROLOG: Advanced Programming Techniques*. The MIT Press, 1994.
20. L. Tainz and E. Zeiger. *Plant Physiology*. Sinauer Associated Inc., 2006.
21. C. Versari and N. Busi. Efficient stochastic simulation of biological systems with multiple variable volumes. In *Proceeding of FBTC 2007*, 2007.

A Constraint Model for Constrained Hidden Markov Models: a First Biological Application*

Henning Christiansen, Christian Theil Have,
Ole Torp Lassen, and Matthieu Petit

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

Abstract. A Hidden Markov Model (HMM) is a common statistical model which is widely used for analysis of biological sequence data and other sequential phenomena. In the present paper we extend HMMs with constraints and show how the familiar Viterbi algorithm can be generalized, based on constraint solving methods. HMMs with constraints have advantages over traditional ones in terms of more compact expressions as well as opportunities for pruning during Viterbi computations. We exemplify this by an enhancement of a simple prokaryote gene finder given by an HMM.

1 Introduction

Hidden Markov Models (HMMs) are one of the most popular models for analysis of sequential processes taking place in a random way, where "randomness" may also be an abstraction covering the fact that a detailed analytical model for the internal matters are unavailable. Such a sequential process can be observed from outside by its emission sequence (letters, sounds, measures of features, all kinds of signals) produced over time, and a HMM postulates a hypothesis about the internal machinery in terms of a finite state automaton equipped with probabilities for the different state transitions and single emissions. Decoding or prediction for a given observed sequence means to compute the most probably state transitions that the HMM can go through to produce the sequence, and thus this represents a best hypothesis for the internal structure or "content" of the sequence. HMMs are widely used in speech recognition and biological sequence analysis [9,2].

Gene prediction aims at algorithmically identifying stretches of a DNA sequence that are biologically functional, in particular protein-coding genes but also other functional elements such as RNA genes [17]. Several HMM based gene finders have been proposed for gene prediction, including [4,7,6]. Decoding an "observed" DNA sequence using an HMM produces a state sequence, that appears as an annotation that identifies regions of genes and non-genes. The automaton defines the regular language for these annotations [13].

With the usual decoding algorithms, such as the Viterbi algorithm [15], it is difficult to add prior knowledge to an HMM about, say, verified coding regions in a specific sequence, or other side-constraints (e.g., this-and-this subsequence cannot occur in a coding region).

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

For instance, fixing a known coding region at a given position n would require to modify the HMM so it is guaranteed to be in this state after n iterations. This HMM transformation may require exponentially many new states.

In this paper, we focus on an extension of HMMs, called Constrained HMMs (CHMMs). The concept of CHMMs is introduced by Sato et al. in [12], although earlier and unrelated systems have used the same or similar names (commented on below). CHMMs restrict the set of allowed state and emission sequences (runs) by the addition of constraints to a standard HMM. The contribution of this paper is to introduce CHMM into Constraint Programming. A constraint model is proposed to represent the allowed "runs". With this model, decoding essentially becomes a constraint optimization problem. We adapt the Viterbi algorithm to take into account such constraints using constraint solving techniques, and we exemplify it for enhancing an HMM based prokaryote gene finder by constraints that state existence of already known genes.

The paper is organized as follows: section 1 describes the background on HMM required to understand the rest of the paper and exemplifies it with a simple gene finder. In section 3, the constraint model for CHMM is described. Finally, section 4 presents related works and our plans for further work.

2 Background

Here we present Hidden Markov Models (HMMs) and the Viterbi algorithm so we can adapt them later with constraints.

2.1 Hidden Markov Model

For simplicity of the technical definitions, we limit ourselves to discrete first order HMMs with a distinguished initial state and no explicit final state (i.e., any state is final); the generalization to more initial states is straightforward.

Definition 1. *A* Hidden Markov Model *(HMM) is a 4-tuple* $\langle S, A, T, E \rangle$*, where*

- $S = \{s_0, s_1, \ldots, s_m\}$ *is a set of* states *which includes an* initial *state referred to as* s_0*;*
- A *is a finite set of emission symbols, individually denoted* e_i*;*
- T *is a set of* transition probabilities $\{p(s_i; s_j)\}_{s_i \in S}$ *representing the probability to transit from one state to another. For each such* s_i*,* $\sum_{s_j \in S \setminus \{s_0\}} p(s_i; s_j) = 1$.
- E *is a set of* emission probabilities $\{p(s_i; e_j)\}_{s_i \in S \setminus \{s_0\}}$ *representing the probability of emitting symbol from a state. For each such* s_i*,* $\sum_{e_j \in E} p(s_i; e_j) = 1$.

A run *of a HMM is defined as a pair consisting of a sequence of states* $s^{(0)} s^{(1)} \ldots s^{(n)}$*, called a* path *and a corresponding sequence of emissions* $e^{(1)} \ldots e^{(n)}$*, called an* observation*, such that*

- $s^{(0)} = s_0$*;*
- $\forall i, 0 \leq i \leq n-1, p(s^{(i)}; s^{(i+1)}) > 0$ *(probability to transit from* $s^{(i)}$ *to* $s^{(i+1)}$*);*
- $\forall i, 0 < i \leq n, p(s^{(i)}; e^{(i)}) > 0$ *(probability to emit* $e^{(i)}$ *from* $s^{(i)}$*).*

The probability *of such a run is defined as* $\prod_{i=1..n} p(s^{(i-1)}; s^{(i)}) \cdot p(s^{(i)}; e^{(i)})$.

HMMs are commonly used to find the probability of a given observation, decoding the path corresponding to an observation and finally finding the model probabilities that maximizes the likelihood of generating a given set of observations. In this paper, we will only explain the decoding computation using the Viterbi algorithm [15].

2.2 The Viterbi Algorithm

The Viterbi algorithm [15] is a dynamic programming algorithm for finding a most probable path corresponding to a given observation. The algorithm keeps track of, for each prefix of an observed emission sequence, the most probable (partial) path leading to each possible state, and extends those step by step into longer paths, eventually covering the entire emission sequence.

We present the algorithm here as a rewriting system on a set of 4-tuples Σ, each representing a (potentially most probable) path for such prefixes; a fixed emission sequence $e^{(1)} \cdots e^{(n)}$ is assumed to be given. Each such 4-tuple is of form $\langle s, i, p, \pi \rangle$ where π is a partial path ending in state s and representing a path corresponding to the emission sequence prefix $e^{(1)} \cdots e^{(i)}$; p is the collected probability for the emissions and transitions applied in the construction of π.

The algorithm can be described by the two rewriting rules given by **Fig.** 1. The $trans$

$trans$: $\Sigma := \Sigma \cup \{\langle s', i{+}1, p \cdot p(s; s') \cdot p(s'; e^{(i+1)}), \pi\, s' \rangle\}$
whenever $\langle s, i, p, \pi \rangle \in \Sigma$, $p(s; s'), p(s'; e^{(i+1)}) > 0$
and $prune$ does not apply.

$prune$: $\Sigma := \Sigma \setminus \{\langle s, i{+}1, p', \pi' \rangle\}$
whenever $\langle s, i{+}1, p, \pi \rangle, \langle s, i{+}1, p', \pi' \rangle \in \Sigma$ and
$p \geq p'$.

Fig. 1. Rewriting rules for the Viterbi algorithm for traditional HMMs

rule expands an existing partial path one step in each possible direction whereas $prune$ removes those that are not optimal up to a given state; the condition that $trans$ cannot apply in case $prune$ is possible, ensures that no non-optimal partial path is expanded. The rules are expected to execute as long as possible, except that $trans$ is only applied when it adds a 4-tuple to Σ that has not been added before. We take the following correctness property for granted.

Proposition 1. *Assume a HMM H with the notation as above and an observation $Obs = e^{(1)} \cdots e^{(n)}$. When the Viterbi algorithm* FIG. 1 *is executed from an initial set of 4-tuples $\{\langle s_0, 0, 1, \epsilon \rangle\}$, ϵ being the empty path and s_0 the initial state of H, it terminates with a set of 4-tuples Σ_{final}. It holds that*

- *For any $\langle s, n, p, \pi \rangle \in \Sigma_{final}$, π is a most probable path for Obs ending in s and with probability p.*

– *Whenever there exists a path for Obs ending in s, Σ_{final} includes a 4-tuple of the form $\langle s, n, p, \pi \rangle$.*

The algorithm can run in time linear in the length of the given emission sequence times a quadratic factor of the number of states in the HMM; the latter is thus constant for a specific HMM.

2.3 An example HMM: a simple gene finder

As an example of an HMM that we later extend with constraints, we consider the problem of identifying protein coding genes in prokaryotes. A DNA sequence is composed of molecules, called *nucleotides*, represented by the four letters a, c, t and g. Some parts of a DNA sequence code for genes, called *coding regions*, while other parts do not and are called *non coding regions*. Coding regions contain a number of *codons*, triplets of nucleotides, each coding for an amino acid in a protein (to be produced by the gene). For prokaryotes, a coding region is contiguous, and it begins with a specific *start codon*, which is often atg, and ends with a *stop codon*, which is one of taa, tga or tag.

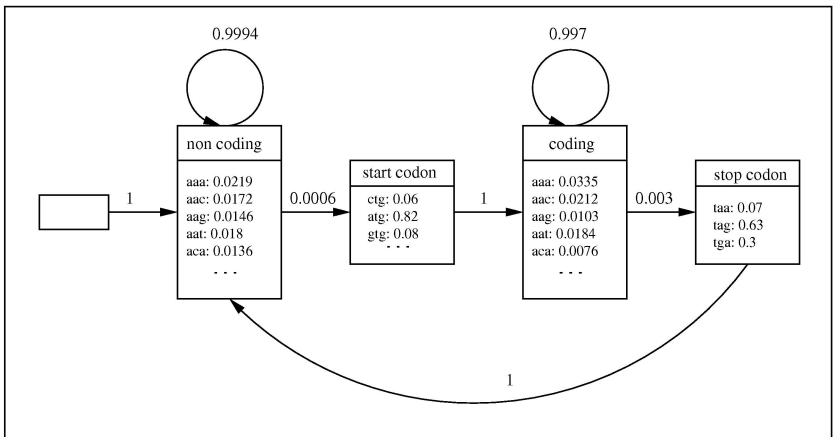


Fig. 2. A simple HMM for Prokaryote genes prediction

Fig. 2 shows a simple HMM for prediction of genes; more advances HMMs are used in successful gene finders that have been reported in the literature, e.g., Genemark.HMM [7] and EasyGene [6], and they can also be handled by our approach. The emission symbols of this HMM are codons, thus three letters form one symbol. It has four states: **start codon, non coding, coding, stop codon**. From **non coding**, any codon can be emitted. From state **start codon**, only start codons ata, atg, att, ctg, gtg and ctg can be emitted. From **coding**, any codon can be emitted except stop codons, taa, tga and tag. From state **stop codon**, only stop codons taa, tga or tag can be emitted. A consequence of the simplification of only emitting entire codons and not individual letters in this HMM is that we restrict to

non coding regions whose length measured in codons is divisible, which is not the case in reality. Transition probabilities have been computed from an already annotated genome, Escherichia coli, K-12 substr. MG1655 (Genbank access NC_000913).

We can illustrate the annotation process as follows; we consider a small piece of E.coli from position 115 to 255.

ctt agg tca cta aat act tta acc aat ata ggc ata gcg cac aga cag ata aaa att aca gag tac aca aca tcc atg aaa cgc att agc acc acc att acc acc acc atc acc att acc aca ggt aac ggt gcg ggc tga,

The Viterbi algorithm computes the most probable path which is indicated as follows:

ctt	$\cdots$	tcc	atg	aaa	$\cdots$	ggc	tga
non coding	$\cdots$	**non coding**	**start codon**	**coding**	$\cdots$	**coding**	**stop codon**

From the indicated path, we can extract an annotation that states a non coding region from position 115 to 189 and a coding region from position 190 to 255.

3 CHMMs and Constraint Models for gene prediction

In this section, we give a formal definition of CHMMs and propose a constraint model for CHMM runs which is employed in an extended Viterbi algorithm.

3.1 Constrained Hidden Markov Model

A CHMM restricts the behavior of an HMM by constraints that must hold on paths that are considered.

Definition 2. *A* constrained HMM (CHMM) *is defined by a 5-tuple* $\langle S, A, T, E, C\rangle$ *where* $\langle S, A, T, E\rangle$ *is an HMM and* C *is a set of constraints, each of which is a mapping from HMM runs into* $\{true, false\}$.

A run *of a such a CHMM,* $\langle path, obs\rangle$ *is a run of the corresponding HMM for which* $C(path, observation)$ *is true (understood as the conjunction of the individual constraints in* C*).*

Notice that we defined constraints in a highly abstract way, independently of any specific constraint language. However, the Markov processes considered in this paper are discrete, and in the following we will apply constraints over finite domains [14]. In [12], constraints were expressed using in Prolog using tests and failure, in a way that do not invite to using constraint solving techniques.

3.2 A constraint model for runs of a CHMM

The constraint model represents runs of a CHMM. A run corresponds to a solution of the constraint model.

Let $\langle S, A, T, E, C\rangle$ be a CHMM and n an integer value that represents the run length. The constraint model is described by the following syntax:

$$run([s^{(0)}, S_1, \ldots, S_n], [E_1, \ldots, E_n])$$

where each variable S_i and E_i represents the state and respectively the emission at the step i. The domain of S_i and E_i, noted dom(S_i) and respectively dom(E_i), is $S \setminus \{s_0\}$ and respectively E.

$run([s^{(0)}, S_1, \ldots, S_n], [E_1, \ldots, E_n])$ is *true* iff

$$\begin{aligned}\exists s^{(1)} \in \text{dom}(S_1), \ldots, \exists s^{(n)} \in \text{dom}(S_n) \text{ and} \\ \exists e^{(1)} \in \text{dom}(E_1), \ldots, \exists e^{(n)} \in \text{dom}(E_n), \\ C(s^{(0)} s^{(1)} \ldots s^{(n)}, e^{(1)} \ldots e^{(n)}) \text{ is true, } s^{(0)} = s_0 \text{ and} \\ p(s^{(0)}; s^{(1)}) \cdot p(s^{(1)}; e^{(1)}) \ldots p(s^{(n-1)}; s^{(n)}) \cdot p(s^{(n)}; e^{(n)}) > 0 \quad (1).\end{aligned}$$

By definition of a CHMM, variables S_i and E_i are constrained to satisfy by C. Formula (1) states that $s^{(0)} s^{(1)} \ldots s^{(n)}$ and $e^{(1)} \ldots e^{(n)}$ is a run the HMM. From this formula, restrictions on the domain of the variables S_i and E_i can be added.

The formula (1) is composed of a product of probabilities. Then, its value is positive iff the value of all the transition or emission probabilities are positive. We used this property to establish a (local) relationship between S_i and S_{i+1} and S_i and E_i. Indeed, valuation of S_i to $s^{(i)}$ and S_{i+1} to $s^{(i+1)}$ can be part of a solution of the constraint model whenever $p(s^{(i)}; s^{(i+1)}) > 0$. These relationships between variables of $run/2$ are modeled by the following constraints added on them:

$$trans(S_i, S_{i+1}) \text{ and } emit(S_i, E_i)$$

where S_i, S_{i+1} and E_i are variables of *run/2*.

$trans(S_i, S_{i+1})$ is *true* iff

$$\exists s^{(i)} \in \text{dom}(S_i) \text{ and } s^{(i+1)} \in \text{dom}(S_{i+1}), p(s^{(i)}; s^{(i+1)}) \neq 0.$$

$emit(S_i, E_i)$ is *true* iff

$$\exists s^{(i)} \in \text{dom}(S_i) \text{ and } e^{(i)} \in \text{dom}(E_i), p(s^{(i)}; e^{(i)}) \neq 0.$$

These constraints are used for domain pruning during constraint propagation. For example, let us suppose that emission taa is observed at the step i of a run of the simple gene finder. During constraint propagation, $emit(S_i, E_i)$ prunes the domain of S_i to $\{\mathbf{non\ coding}, \mathbf{stop\ codon}\}$.

Constraints C of a CHMM are simply added on the variables of *run/2*. In the following, an example of C is defined to include prior knowledge on the simple gene finder.

3.3 A constrained gene finder

We illustrate the constraint model on the simple gene finder presented subsection 2.3. The HMM associated with the simple gene finder is constrained to be in certain states at given positions. For instance, this CHMM allows the inclusion of information about known coding regions during the Viterbi computation.

Consider

$$run([s^{(0)}, S_1, \ldots, S_n], [e^{(1)}, \ldots, e^{(n)}])$$

the constraint model associated with the simple gene finder where $e^{(1)}, \ldots, e^{(n)}$ is a sequence of n codons. A set of variables S_i is constrained to be equal to $State$ with the following constraint:

$$fix(State, Position_1, Position_2)$$

where $State \in S \backslash \{s^{(0)}\}$, $Position_1 \in \{1, \ldots, n\}$, $Position_2 \in \{1, \ldots, n\}$ and $Position_1 \leq Position_2$.

$fix(State, Position_1, Position_2)$ is ***true*** iff

$$\exists k \in \text{dom}(Position_1) \text{ and } \exists l \in \text{dom}(Position_2), \forall i, k \leq i \leq l, S_i = State.$$

For example, fix a position of a coding region can be expressed as the conjunction of

$$fix(\textbf{start codon}, P_1, P_1) \wedge P_1 + 1 = P_2 \wedge \\ fix(\textbf{coding}, P_2, P_3) \wedge P_3 + 1 = P_4 \wedge fix(\textbf{stop codon}, P_4, P_4).$$

These constraints on the simple gene finder oblige runs to be in a coding region between the position P_1 and P_4.

3.4 Viterbi Computation for a CHMM

Consider a CHMM $\langle S, A, T, E, C \rangle$, an observation $e^{(1)} \ldots e^{(n)}$ and a constraint model

$$run([s^{(0)}, S_1, \ldots, S_n], [e^{(1)}, \ldots, e^{(n)}]).$$

The most probable path is computed by finding the solution $s^{(1)}, \ldots, s^{(n)}$ of the constraint model that maximizes the objective function: run probability.

Viterbi computation for CHMM is expressed as a rewriting system on a set of 5-tuples Σ. Each such 5-tuple is of form $\langle s, i, p, \pi, \sigma \rangle$ where π is a partial path ending in state s and representing a path corresponding to the emission sequence prefix $e^{(1)} \cdots e^{(i)}$; p is the collected probability for the emissions and transitions applied in the construction of π and σ is the current constraint store, a conjunction of constraints. Solutions of the constraint store are denoted by sol(σ).

The two rules of the classical Viterbi algorithm are adapted for CHMM (see **Fig.** 3). Viterbi computation is executed from an initial set of 5-tuples

$$\{\langle s^{(0)}, 0, 1, \epsilon, C \wedge trans(s^{(0)}, S_1) \wedge \\ \bigwedge_{0 < i \leq n-1} trans(S_i, S_{i+1}) \wedge \bigwedge_{0 < i \leq n} emit(S_i, E_i) \rangle\}.$$

The $trans\_ctr$ rule expands an existing partial path one step in a restricted number of directions that satisfy the constraint store. This satisfiability checking of the constraint store is denoted by check_sat. $prune\_ctr$ removes partial paths that are non optimal solutions. This is the case when two conditions are satisfied: the probability to reach s is not optimum and solutions of σ' are also solutions of σ. The second condition avoids removing partial paths that could be part of an optimal solution. If sol(σ') and sol(σ) can not be compared, we can not conclude that the partial path π' is not part the optimal solution.

$trans\_ctr$: $\Sigma := \Sigma \cup \{\langle s', i{+}1, p \cdot p(s; s') \cdot p(s'; e^{(i+1)}), \pi\, s', \sigma \wedge S_{i+1} = s' \rangle\}$
whenever $\langle s, i, p, \pi, \sigma \rangle \in \Sigma$, $p(s; s'), p(s'; e^{(i+1)}) > 0$
check_sat($\sigma \wedge S_{i+1} = s'$) and $prune\_ctr$ does not apply.

$prune\_ctr$: $\Sigma := \Sigma \setminus \{\langle s, i{+}1, p', \pi', \sigma' \rangle\}$
whenever $\langle s, i{+}1, p, \pi, \sigma \rangle, \langle s, i{+}1, p', \pi', \sigma' \rangle \in \Sigma$,
$p \geq p'$ and sol(σ') $\subseteq$ sol(σ).

Fig. 3. Rewriting rules for the Viterbi algorithm for CHMM

Correctness property is argued in the previous paragraph. $prune\_ctr$ rule allows us to remove only partial paths detected as part of a non optimal solution. Unlike the classical Viterbi algorithm, this algorithm can run in time exponential in the length of the given emission sequence. Indeed, potentially all the extended partial paths need to be kept in Σ. However, with an efficient $check\_sat$, partial paths that can not lead to the solution of the constraint model will be discarded as soon as possible. Size reductions of Σ due to $prune\_ctr$ is also important. The size for Σ will stay reasonable if the two constraint stores can easily be compared. That is the case for the fix constraint. Indeed, this constraint constrains only a restricted set of variables of the path. Outside this set, the remaining variables are not constrained. Then, the comparison is straightforward.

4 Discussion and future work

The term "Constrained HMM" is used in [10,5] refers to restrictions on the finite automaton associated with a HMM but not as constraint on HMM runs. In [12], CHMMs were introduced to exemplify an EM algorithm for models with possible derivation failures. Our approach differs since we take care about constraints on the HMM during Viterbi computation whereas they do that during the learning process.

In constraint modeling, we notice two recent works with a similar approach of ours [16,18]. In [16], Will et al. propose a constraint model to represent pairwise alignment problem which integrates constraints related to non-coding RNA. In [18], Zytnicki et al. describe a weighted CSP model for locating motifs of non-coding RNA.

In this paper, a constraint model that represents runs of constrained HMMs is defined. In this framework, a Viterbi computation is expressed as an optimization problem and conditions for an efficient computation are presented. Finally, a gene finder that includes prior knowledge is presented along the paper to exemplify a usage of CHMM.

A first implementation based on PRISM [11] allows us to perform a first experiment on fixing a coding region for the simple gene finder. PRISM build-ins give us for free a Viterbi computation for CHMM. To improve the efficiency of the *pruning_ctr* rule to reduce the search, we work on an implementation of the proposed algorithm in CHR [3], a multiset based rewriting system, that closely follows rewriting systems described in this paper.

We also work on an automatic generation of the constraint model given a HMM. This implementation is based on our Probabilistic Choice Constraints library [8]. This constraints

library implemented with clp(fd) library of SICStus Prolog [1] allow the simulation of partially defined probabilistic choices in Constraint Programming. In the constraint model, probabilistic choice is partially known when origin state of an transition or emission is not known. This framework and clp(fd) library will facilitate the definition of other kinds of constraints which can be combined with existing gene finding methods, improving flexibility and the quality of the results.

References

1. M. Carlsson, G. Ottoson, and B. Carlson. An open–ended finite domain constraint solver. In *Proc. of the International Symposium on Programming Languages: Implementations Logics and Programs*, LNCS, pages 191–206, Soupthampton, UK, September 1997. Springer.
2. R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis*. Cambrigde University Press, 1998.
3. T. Fruhwirth. *Constraint Handling Rules*. Cambridge University Press, 2009.
4. A. Krogh, I.S. Mian, and D. Haussler. A hidden markov model that finds genes in e.coli dna. *Nucleic Acids Research*, 22(22):4768–4778, 1994.
5. N. Landwehr, T. Mielikäinen, L. Eronen, H. Toivonen, and H. Mannila. Constrained hidden markov models for population-based haplotyping. *BMC Bioinformatics*, 8(S-2), 2007.
6. T.S. Larsen and A. Krogh. Easygene - a prokaryotic gene finder that ranks orfs by statistical significance. *BMC Bioinformatics*, 4(21):21, 2003.
7. A.V. Lukashin and M. Bordovsky. GeneMark.hmm: new solutions for gene finding. *Nucleic Acids Research*, 26(4):1107–1115, February 1998.
8. M. Petit and A. Gotlieb. Boosting probabilistic choice operators. In *Proceedings of the International Conference on Principles and Pratice of Constraint Programming*, LNCS, pages 559–573, Providence, USA, September 2007.
9. L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognitation. *IEEE*, 77(2):257–286, February 1989.
10. S.T. Roweis. Constraint hidden markov models. In *Proc. of the International Conference of Advances in Neural Information Processing System*, pages 782–788, Denver, USA, Dec. 1999.
11. T. Sato and Y. Kameya. PRISM: a language for symbolic-statistical modeling. In *Proc. of the International Joint Conference of on Artificial Intellingence*, pages 1330–1335, Nagoya, Japan, August 1997.
12. T. Sato and Y. Kameya. New advances in logic-based probabilistic by PRISM. In *Probabilistic Inductive Logic Programming*, LNCS, pages 118–155. Springer, 2008.
13. D.B Searls. *The computational linguistics of biological sequences, chapter of Artificial intelligence and molecular biology*. AAAI, 1993.
14. P. Van Hentenryck, V.A. Saraswat, and Y. Deville. Design, implementation, and evaluation of the constraint language cc(fd). *Constraint Programming*, 910:293–316, May 1995.
15. Andrew J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–269, April 1967.
16. S. Will, A. Bush, and R. Backofen. Efficient sequence alignment with side-constraints by cluster tree elimination. *Constraints*, 13(1–2):110–129, 2008.
17. J. Xiong. *Essential Bioinformatics*. Cambridge Unversity Press, 2006.
18. M. Zytnicki, C. Gaspin, and T. Schiex. DARN! A weighted constraint solver for RNA motif localization. *Constraints*, 13(1–2):91–109, 2008.

Haplotype Inference Combining Pedigrees and Unrelated Individuals

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 Computer Science and Informatics, University College Dublin, Ireland
jpms@ucd.ie

Abstract. Haplotype inference is a crucial topic in genetic studies and also represents a challenging computational problem. A significant number of combinatorial approaches tackle the haplotype inference problem either for pedigrees or for unrelated individuals. This work integrates two relevant and well-known constraint based haplotyping approaches. The Minimum Recombinant Haplotyping Configuration (MRHC) problem targets the haplotyping solution which minimizes the number of recombinant events within a pedigree. MRHC only takes into consideration the family information. In contrast, the Haplotype Inference by Pure Parsimony (HIPP) problem aims at finding a solution which minimizes the number of distinct haplotypes. The HIPP approach is adequate for phasing unrelated individuals from the same population. This paper proposes a method for inferring haplotypes for individuals of the same population, although organized in different families, thus combining both MRHC and HIPP approaches. This new method can take into account family information and population information, both important in haplotype inference. Experimental results show that the proposed approach is more accurate, both in terms of switch error rate and missing error rate, than the MRHC approach (performed by the PedPhase tool), on sets of families from the same population.

1 Introduction

Genetic association studies with phenotypic variations are only possible with a deep knowledge of the genetic differences between individuals. A very important and challenging task to understand genetic variations consists of inferring haplotypes from genotypes.

Constraint based methods for haplotype inference have been shown to be a practical and relevant alternative to statistical approaches, either for phasing pedigrees [9,11] or unrelated individuals [4]. Nonetheless, a study comparing the haplotype inference methods using pedigrees and unrelated individuals [12] points out that a new method which takes into consideration both pedigree and population information is necessary. Indeed, existing haplotyping methods for pedigrees ignore the population information, while haplotyping methods for unrelated individuals do not take into account the pedigree information.

The work presented in this paper was motivated by the comparison study described in [12]. A constraint based model to deal with families and unrelated individuals is proposed. The new method is based on two well-known combinatorial approaches: MRHC and HIPP. The Minimum Recombinant Haplotype Configuration (MRHC) approach is used to phase individuals organized in pedigrees, by minimizing the number of recombination

events within each pedigree. In general, a significant number of solutions can be obtained using only the minimum recombinant paradigm, especially when several families are considered. Thus, the Haplotype Inference by Pure Parsimony (HIPP) approach is considered to choose a solution that uses the minimum number of distinct haplotypes, among all the minimum recombinant solutions. The new method for haplotype inference, named PedRPoly, is shown to be more accurate than the method traditionally used for inferring haplotypes on pedigrees using the minimum recombinant approach, as performed by the PedPhase tool [11]. PedRPoly is also shown to be more accurate than the pure parsimony approach, performed by the RPoly tool [4], when pedigrees are considered. In addition, this paper suggests some reductions on the size of the original integer programming MRHC model.

The paper is organized as follows. The next section describes the haplotype inference problem and overviews the MRHC and HIPP approaches. Section 3 details the new proposed model, PedRPoly, which combines MRHC and HIPP formulations. Afterwards, experimental results comparing the accuracy of PedPhase and PedRPoly are presented and discussed. Finally, the conclusions are presented in section 5.

2 Haplotype Inference

Single Nucleotide Polymorphisms (SNPs) are the most common variations between human beings, which occur when a nucleotide is mutated into another nucleotide at a single DNA position. Haplotypes correspond to the set of closely linked SNPs, within a single chromosome, which tends to be inherited together. However, it is very expensive and time consuming to determine experimentally the haplotypes. Instead, only genotypes, which correspond to the conflated data of two haplotypes on homologous chromosomes are obtained. The haplotype inference problem consists in determining the haplotypes which originate a given genotype.

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 nucleotide and 1 represents the mutant type nucleotide. Each site of the haplotype h_i is represented by $h_{i\,j}$ ($1 \leq j \leq m$). 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 $g_{i\,j}$. Each genotype is explained by two haplotypes. A genotype g_i is explained by a pair of haplotypes (h_i^a,h_i^b) such that

$$g_{i\,j} = \begin{cases} h^a_{i\,j} \text{ if } h^a_{i\,j} = h^b_{i\,j} \\ 2 \quad \text{if } h^a_{i\,j} \neq h^b_{i\,j} \end{cases} . \tag{1}$$

A genotype site $g_{i\,j}$ with either value 0 or 1 is a homozygous site (the same allele is inherited from both parents), whereas a site with value 2 is a heterozygous site (different alleles are inherited from each parent).

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).

Most often genotyping procedures leave a percentage of missing data. To represent missing sites, the alphabet of the genotypes is extended to $\{0, 1, 2, ?\}$.

When the considered individuals are organized in pedigrees, additional information may be associated with the haplotype inference problem. Considering the Mendelian law of inheritance, every site in a single haplotype is inherited from a single parent, assuming no mutations within a pedigree. In a pedigree, an individual is a *founder* if he does not have parents on the pedigree (and a *non-founder* if he has both parents on the pedigree). We assume that haplotype h^a is inherited from the father and h^b is inherited from the mother, thus breaking a symmetry on the pairs of haplotypes for non-founder individuals. However, a recombination may occur, where the two haplotypes of a parent get shuffled and the shuffled haplotype is passed on to the child. For example, suppose a father has the haplotype pair $(011, 100)$ and the haplotype that he passed on to his child is 111. Hence one recombination event must have occurred: haplotypes 011 and 100 have mixed together and originated a new haplotype $h = 111$. Although every site of the child's haplotype h was inherited from the father, the first site came from the paternal grandmother, while the second and third sites came from the paternal grandfather.

2.1 Minimum Recombinant Haplotype Configuration

Recombination events are rare in DNA regions with high linkage disequilibrium. Therefore, most rule-based haplotype inference methods for pedigrees assume no recombination among SNPs within each pedigree [19,20,13]. Although the assumption of no recombination is valid in many cases, this assumption can be violated even for some dense markers [9]. Therefore, the problem of minimizing the number of recombinations was suggested [6,18]. The Minimum Recombinant Haplotype Configuration (MRHC) problem is a well-known approach to solve the haplotype inference problem in pedigrees. The MRHC problem is NP-hard [9,10,15].

Definition 2. *The Minimum Recombinant Haplotype Configuration (MRHC) problem aims at finding a haplotype inference solution for a pedigree which minimizes the number of required recombination events [6,18].*

The PedPhase tool [9] implements an Integer Linear Programming (ILP) model for MRHC with missing alleles.

2.2 Haplotype Inference by Pure Parsimony

The Haplotype Inference by Pure Parsimony (HIPP) approach aims at finding a minimum-cardinality set of haplotypes $\mathcal{H}$ that can explain a given set of genotypes $\mathcal{G}$. The idea of searching for the solution with the smallest number of haplotypes is biologically motivated by the fact that individuals from the same population have the same ancestors and mutations do not occur often. Moreover, it is also well-known that the number of haplotypes in a population is much smaller than the number of genotypes. It has been shown that the HIPP problem is NP-hard [7].

Definition 3. *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 [5].*

RPoly [4] is a state-of-the-art solver implementing a 0-1 ILP model for solving the HIPP problem.

3 PedRPoly: Minimum Recombinant Maximum Parsimony

This section describes the PedRPoly model which aims at finding a haplotype inference solution for sets of pedigrees from the same population. The PedRPoly model is a combination of the MRHC PedPhase model [10] and the HIPP RPoly model [4].

Definition 4. *The Minimum Recombinant Maximum Parsimony model aims at finding a haplotype inference solution which minimizes the number of recombination events within pedigrees and the number of distinct haplotypes used.*

Example 1. Consider two trios (father, mother and child), from two families A and B, with the following genotypes: $g_A^{father} = 102$, $g_A^{mother} = 222$, $g_A^{child} = 202$, $g_B^{father} = 211$, $g_B^{mother} = 202$ and $g_B^{child} = 222$. Consider the following haplotype inference solutions.

Solution 1: $h_A^{father} = (100, 101)$, $h_A^{mother} = (001, 110)$, $h_A^{child} = (100, 001)$, $h_B^{father} = (011, 111)$, $h_B^{mother} = (000, 101)$ and $h_B^{child} = (111, 000)$. Solution 1 is a 0-recombinant solution with 7 distinct haplotypes $(100, 101, 000, 111, 011, 001, 110)$.

Solution 2: $h_A^{father} = (101, 100)$, $h_A^{mother} = (000, 111)$ and $h_A^{child} = (101, 000)$, $h_B^{father} = (011, 111)$, $h_B^{mother} = (000, 101)$ and $h_B^{child} = (011, 100)$. Solution 2 is a 1-recombinant solution (there is one recombination event in family B) and uses 5 distinct haplotypes $(100, 101, 000, 111, 011)$.

Solution 3: $h_A^{father} = (101, 100)$, $h_A^{mother} = (000, 111)$ and $h_A^{child} = (101, 000)$, $h_B^{father} = (011, 111)$, $h_B^{mother} = (000, 101)$ and $h_B^{child} = (111, 000)$. Solution 3 is a 0-recombinant solution using 5 distinct haplotypes $(100, 101, 000, 111, 011)$.

Clearly, solution 3 is preferred to the other solutions. Solution 3 is both a MRHC and a HIPP solution. Consequently, solution 3 is a Minimum Recombinant Maximum Parsimony solution. If there exists no solution that minimizes both criteria, then preference is given to the MRHC criterion and hence, the MRHC solution which uses the smallest number of distinct haplotypes would be chosen.

PedRPoly is a 0-1 ILP model which combines the ILP PedPhase and the RPoly models. The constraints of the model are detailed in Table 1. Following the RPoly model, PedRPoly associates two haplotypes, h_i^a and h_i^b, with each genotype g_i, and these haplotypes are required to explain g_i. Moreover, PedRPoly associates a variable $t_{i\,j}$ with each heterozygous site $g_{i\,j}$, such that $t_{i\,j} = 1$ indicates that the mutant value was inherited from the father ($h_{i\,j}^a = 1$) and the wild value was inherited from the mother ($h_{i\,j}^b = 0$) whereas $t_{i\,j} = 0$ indicates that the wild value was inherited from the father ($h_{i\,j}^a = 0$) and the mutant value was inherited from the mother ($h_{i\,j}^b = 1$). In addition, PedRPoly associates two variables with each missing site. Variable $t_{i\,j}^a$ is associated with the paternal haplotype site $h_{i\,j}^a$, whereas variable $t_{i\,j}^b$ is associated with the maternal haplotype site $h_{i\,j}^b$. The values of h_i^a and h_i^b at homozygous sites are implicitly assumed.

To analyze the recombination events within pedigrees, not only the paternal and maternal haplotypes are considered, but also the grand-paternal and grand-maternal origin of each allele in the haplotypes. Following the PedPhase MRHC model, for each non-founder

Table 1. The PedRPoly Model: Minimum Recombinant Maximum Parsimony.

minimize:	$(2n+1) \times \sum_{non-founder\ i} \sum_{j=1}^{m-1} (r^1_{i\,j} + r^2_{i\,j}) + \sum_{i=1}^{n} (u^a_i + u^b_i)$	
subject to:		
Equation	Constraint	Indexes
	Mendelian Law of Inheritance rules (Table 2)	
(2)	$-r^l_{i\,j} + g^l_{i\,j} - g^l_{i\,j+1} \leq 0$ $-r^l_{i\,j} - g^l_{i\,j} + g^l_{i\,j+1} \leq 0$	$l = 1,\ 2$ $1 \leq i \leq n$, i non-founder $1 \leq j \leq m$
(3)	$\neg(R \Leftrightarrow S) \Rightarrow x^{p\,q}_{i\,k}$ (Table 3)	$p, q \in \{a, b\}$ $1 \leq k < i \leq n$
(4)	$\sum_{k<i\,;\,q\in\{a,b\}} x^{p\,q}_{i\,k} - u^p_i \leq 2i - 3$	$1 < i \leq n$ $p \in \{a, b\}$

individual i and site j, two variables are defined: $g^1_{i\,j}$ and $g^2_{i\,j}$. The assignment $g^1_{i\,j} = 0$ ($g^1_{i\,j} = 1$) represents that the paternal allele of individual i at site j comes from the paternal grandfather (grandmother). In a similar way, $g^2_{i\,j} = 0$ ($g^2_{i\,j} = 1$) represents that the maternal allele of individual i at site j comes from the maternal grandfather (grandmother). Constraints to ensure that the Mendelian law of inheritance is satisfied are defined in Table 2. Note that PedRPoly only associates variables with heterozygous and missing sites (inspired by RPoly), while PedPhase associates variables with both homozygous and heterozygous sites. The new definition of variables associated with sites requires the redefinition of the constraints related with Mendelian laws. For instance, consider the first constraint of Table 2, $t_{f(i)\,j} \Leftrightarrow g^1_{i\,j}$, for the case $g_{i\,j} = 0$ and $g_{f(i)\,j} = 2$. Clearly, if $t_{f(i)\,j} = 1$ (which represents that individual $f(i)$ has inherited value 1 from his father and value 0 from his mother) then $g^1_{i\,j} = 1$ (which represents that individual i must have inherited the value 0 from his paternal grandmother) and vice-versa.

Furthermore, variables are defined to count the number of recombinations. For each non-founder individual i, variable $r^1_{i\,j}$ ($r^2_{i\,j}$) is assigned value 1 if a recombination took place at site j, to create the paternal (maternal) haplotype of individual i. Thus, $r^l_{i\,j} = 1$ if $g^l_{i\,j} \neq g^l_{i\,j+1}$, for $l = 1, 2$ and $1 \leq j \leq m-1$, which is ensured by constraints (2). Here, a simplification to the original MRHC is considered. Actually, in the original model, $r^l_{i\,j} = 1$ *if and only if* $g^l_{i\,j} \neq g^l_{i\,j+1}$. Observe that an implication, instead of an equivalence, is sufficient for correctness and reduces in half the number of these constraints.

Furthermore, the model should be able to determine the number of distinct haplotypes used. Once more following the RPoly model, let $x^{p\,q}_{i\,k}$, with $p, q \in \{a, b\}$ and $1 \leq k < i \leq n$, be 1 if haplotype p of genotype g_i and haplotype q of genotype g_k are different. The conditions on the $x^{p\,q}_{i\,k}$ variables are based on the values of variables $t_{i\,j}$ and $t_{k\,j}$ for heterozygous sites and of variables $t^a_{i\,j}$, $t^b_{i\,j}$, $t^a_{k\,j}$ and $t^b_{k\,j}$ for missing sites, and are described by equations (3).

In addition, the model uses variables u to denote when one of the haplotypes, associated with a given genotype, is different from all previous haplotypes. Hence, u^p_i, with $p \in \{a, b\}$

Table 2. Mendelian Law of Inheritance Rules (regarding variables $g^1_{i\,j}$). The constraints involving variables $g^2_{i\,j}$ are defined similarly ($1 \leq i \leq n$, i non-founder, $1 \leq j \leq m$).$f(i)$ corresponds to the father of i.

Condition	Constraint
$g_{i\,j} = 0 \wedge g_{f(i)\,j} = 2$	$t_{f(i)\,j} \Leftrightarrow g^1_{i\,j}$
$g_{i\,j} = 0 \wedge g_{f(i)\,j} = ?$	$(g^1_{i\,j} \vee \neg t^a_{f(i)\,j}) \wedge (\neg g^1_{i\,j} \vee \neg t^b_{f(i)\,j})$
$g_{i\,j} = 1 \wedge g_{f(i)\,j} = 2$	$t_{f(i)\,j} \Leftrightarrow \neg g^1_{i\,j}$
$g_{i\,j} = 1 \wedge g_{f(i)\,j} = ?$	$(g^1_{i\,j} \vee t^a_{f(i)\,j}) \wedge (\neg g^1_{i\,j} \vee t^b_{f(i)\,j})$
$g_{i\,j} = 2 \wedge g_{f(i)\,j} = 0$	$\neg t_{i\,j}$
$g_{i\,j} = 2 \wedge g_{f(i)\,j} = 1$	$t_{i\,j}$
$g_{i\,j} = 2 \wedge g_{f(i)\,j} = 2$	$(g^1_{i\,j} \vee t_{i\,j} \vee \neg t_{f(i)\,j}) \wedge (g^1_{i\,j} \vee \neg t_{i\,j} \vee t_{f(i)\,j}) \wedge$ $(\neg g^1_{i\,j} \vee t_{i\,j} \vee t_{f(i)\,j}) \wedge (\neg g^1_{i\,j} \vee \neg t_{i\,j} \vee \neg t_{f(i)\,j})$
$g_{i\,j} = 2 \wedge g_{f(i)\,j} = ?$	$(g^1_{i\,j} \vee t_{i\,j} \vee \neg t^a_{f(i)\,j}) \wedge (g^1_{i\,j} \vee \neg t_{i\,j} \vee t^a_{f(i)\,j}) \wedge$ $(\neg g^1_{i\,j} \vee t_{i\,j} \vee \neg t^b_{f(i)\,j}) \wedge (\neg g^1_{i\,j} \vee \neg t_{i\,j} \vee t^b_{f(i)\,j})$
$g_{i\,j} = ? \wedge g_{f(i)\,j} = 0$	$\neg t^a_{i\,j}$
$g_{i\,j} = ? \wedge g_{f(i)\,j} = 1$	$t^a_{i\,j}$
$g_{i\,j} = ? \wedge g_{f(i)\,j} = 2$	$(g^1_{i\,j} \vee t^a_{i\,j} \vee \neg t_{f(i)\,j}) \wedge (g^1_{i\,j} \vee \neg t^a_{i\,j} \vee t_{f(i)\,j}) \wedge$ $(\neg g^1_{i\,j} \vee t^a_{i\,j} \vee t_{f(i)\,j}) \wedge (\neg g^1_{i\,j} \vee \neg t^a_{i\,j} \vee \neg t_{f(i)\,j})$
$g_{i\,j} = ? \wedge g_{f(i)\,j} = ?$	$(g^1_{i\,j} \vee t^a_{i\,j} \vee \neg t^a_{f(i)\,j}) \wedge (g^1_{i\,j} \vee \neg t^a_{i\,j} \vee t^a_{f(i)\,j}) \wedge$ $(\neg g^1_{i\,j} \vee t^a_{i\,j} \vee \neg t^b_{f(i)\,j}) \wedge (\neg g^1_{i\,j} \vee \neg t^a_{i\,j} \vee \neg t^b_{f(i)\,j})$

Table 3. Definition of predicates R and S, accordingly to index values.

Condition	Constraint
$g_{i\,j} \neq 2 \wedge g_{k\,j} = 2$	$R = (g_{i\,j} \Leftrightarrow (q \Leftrightarrow a))$ and $S = t_{k\,j}$
$g_{k\,j} \neq 2 \wedge g_{i\,j} = 2$	$R = (g_{k\,j} \Leftrightarrow (p \Leftrightarrow a))$ and $S = t_{i\,j}$
$g_{i\,j} = 2 \wedge g_{k\,j} = 2$	$R = (p \Leftrightarrow q)$ and $S = (t_{i\,j} \Leftrightarrow t_{k\,j})$
$g_{i\,j} = ? \wedge g_{k\,j} \notin \{2, ?\}$	$R = t^p_{i\,j}$ and $S = g_{k\,j}$
$g_{k\,j} = ? \wedge g_{i\,j} \notin \{2, ?\}$	$R = t^q_{k\,j}$ and $S = g_{i\,j}$
$g_{i\,j} = ? \wedge g_{k\,j} = 2$	$R = (q \Leftrightarrow a)$ and $S = (t^p_{i\,j} \Leftrightarrow t_{k\,j})$
$g_{k\,j} = ? \wedge g_{i\,j} = 2$	$R = (p \Leftrightarrow a)$ and $S = (t^q_{k\,j} \Leftrightarrow t_{i\,j})$
$g_{i\,j} = ? \wedge g_{k\,j} = ?$	$R = t^p_{i\,j}$ and $S = t^q_{k\,j}$

and $1 \leq i \leq n$, is 1 if haplotype p of genotype g_i is different from all previous haplotypes. Then, the conditions on the u^p_i variables are based on the conditions for the $x^{p\,q}_{i\,k}$ variables, with $1 \leq k < i$ and $q \in \{a, b\}$ and correspond to equation (4).

Finally, the cost function minimizes the number of recombination events, which is given by the sum of variables r, and the number of distinct haplotypes used in the solution, which is given by the sum of variables u,

$$\textit{minimize}\ ((2n+1) \times \textstyle\sum_{non-founder\ i} \sum_{j=1}^{m-1} (r^1_{i\,j} + r^2_{i\,j})) + \sum_{i=1}^{n} (u^a_i + u^b_i).$$

A larger weight is given to the number of recombinations, because we want to guarantee that a minimum recombinant solution is chosen. Note that $2n$ is a trivial upper bound on the number of haplotypes in the solution, and therefore giving weight $2n + 1$ to the number

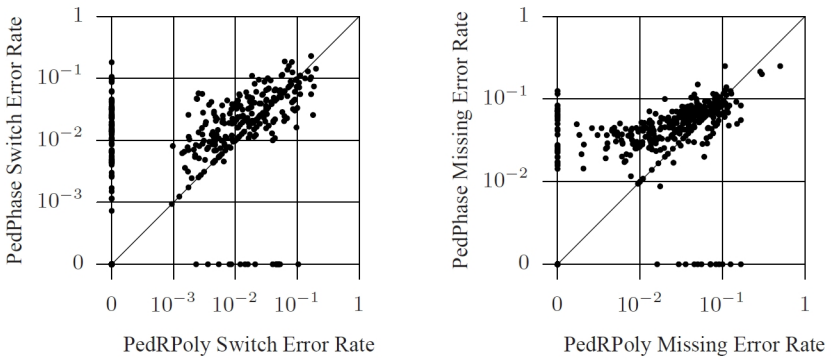


Fig. 1. Comparison of PedRPoly and PedPhase error rates

of recombinations implies that a minimum recombinant solution is always preferred. The idea of giving more weight to the number of recombinations is biological motivated by the fact that recombination events within haplotypes are rare. Moreover, the number of recombination events is related with the number of distinct haplotypes. In fact, a larger number of recombination events suggests a larger number of haplotypes. In general, a recombination event generates a new haplotype, whereas if no recombination occurs, then the haplotypes of the child are exact copies of the parents haplotypes.

4 Experimental Results

The experimental data was simulated using the SimPed program [8]. SimPed generates haplotypes for families, given the pedigree structure, as well as the haplotypes and their frequencies for founders. Three different sets of haplotypes were considered. These sets of haplotypes are real data for which haplotypes have been experimentally identified [1,17], and correspond to the A, B and C data sets used in [2]. Haplotypes in set A have 9 SNPs, haplotypes in set B have 5 SNPs and haplotypes in set C have 17 SNPs. Three different pedigree structures, taken from [10], were considered: pedigree 1 with 15 individuals, pedigree 2 with 29 individuals and pedigree 3 with 17 individuals (with a mating loop). Each simulated instance consists of 10 replicates of the given pedigree, simulating 10 different families from the same population. Recombination events between alleles were considered with probabilities 0.1%, 0.5% and 1%. Three variations on missing rates were considered: 1%, 10% and 20%. For each combination of parameters, 5 independent replicates were selected, resulting in a total of 405 ($= 3^4 \times 5$) input trials. MiniSat+ [3] was used as a 0-1 ILP tool to solve the PedRPoly model.

In order to analyze the accuracy of the methods, two different errors were considered. The *switch error rate* measures the percentage of possible switches in haplotype orientation, used to recover the correct phase in an individual [14]. Missing alleles are not considered for computing the switch error. The *missing error rate* is the percentage of incorrectly inferred missing data [16].

Figure 1 presents two scatter plots comparing the switch error and missing error rates for PedRPoly and PedPhase. Each problem instance corresponds to a point in the plot, where

Table 4. Switch Error Rate and Missing Error Rate for PedRPoly and PedPhase in sets of instances with different parameters (n is the number of genotypes of the instance, with $n = 10 \cdot f$ where f is the size of each pedigree, and m is the number of sites of each genotype).

		Error Rate			
Set		**Switch Error Rate**		**Missing Error Rate**	
		PedRPoly	**PedPhase**	**PedRPoly**	**PedPhase**
Missing Rate	1%	0.0115	0.0143	0.0361	0.0411
	10%	0.0190	0.0232	0.0382	0.0488
	20%	0.0304	0.0437	0.0456	0.0625
Recombination Rate	0.1%	0.0157	0.0221	0.0393	0.0472
	0.5%	0.0212	0.0267	0.0398	0.0506
	1%	0.0240	0.0324	0.0407	0.0546
Pedigree	Ped1 (n=150)	0.0194	0.0245	0.0428	0.0469
	Ped2 (n=290)	0.0199	0.0284	0.0355	0.0471
	Ped3 (n=170)	0.0217	0.0283	0.0415	0.0584
Population	A (m=9)	0.0116	0.0210	0.0428	0.0469
	B (m=5)	0.0450	0.0482	0.0355	0.0471
	C (m=17)	0.0044	0.0120	0.0415	0.0584

the x-axis represents the error rate of the PedRPoly approach and the y-axis represents the error rate of the ILP PedPhase approach.

The switch error rate of the methods is compared in the left plot of Figure 1. The switch error of PedRPoly is smaller than the switch error of PedPhase for 55.3% of the problem instances. The switch error of PedRPoly is larger than the switch error of PedPhase for 16.8% of the problem instances, and for the remaining 27.9% the error is the same for both methods.

With respect to the missing error rate (right plot of Figure 1), it is clear that PedRPoly is more accurate than PedPhase. Indeed, the missing error of PedRPoly is smaller than the missing error of PedPhase for 64.7% of the problem instances. The missing error of PedRPoly is larger than the missing error of PedPhase for 18% of the problem instances, and for the remaining 17.3% instances the error is the same for both methods.

Table 4 presents the accuracy results organized by parameter value. Each value is the average of the error rate for the 135 instances generated with the corresponding parameter value. We conclude that PedRPoly has a smaller (switch and missing) error rate on every class of instances.

In addition, the HIPP solver, RPoly, has also been tested. RPoly does not take into consideration the pedigree information, and therefore, has in general higher error rates which can go up to 50% in some cases. Moreover, RPoly is not able to solve around 25% of the instances within a time out of 10000 seconds (in particular the instances with higher missing rates).

Finally, we have compared the number of distinct haplotypes in the PedRPoly solution and the PedPhase solution with the number of haplotypes in the real solution. PedRPoly has the same number of haplotypes as the real solution for 64.7% of the instances and for 96.8% of the instances the number of haplotypes in the PedRPoly solution differs by less than 3 haplotypes from the number in the real solution. PedPhase solutions are less similar to the real solutions with respect to the number of distinct haplotypes. For only 23.7% of

the instances, the number of haplotypes in the PedPhase solution is equal to the number of haplotypes in the real solution, and for 50% of the instances, the number of haplotypes in the PedRPoly solution differs by more than 2 haplotypes from the real solution.

Regarding the efficiency of PedRPoly and PedPhase methods, while PedPhase is able to solve each instance in a few seconds, PedRPoly can take a few hours. Improving the efficiency of PedRPoly is the main short term goal.

5 Conclusions and Future Work

This paper presents a new method for inferring haplotypes from genotype data of families from the same population. The proposed method (called PedRPoly) integrates the minimum recombinant and the pure parsimony principles, two relevant constraint based haplotyping approaches. Thus, PedRPoly can take into account both the family information and the population information. Experimental results show that PedRPoly is actually more accurate than PedPhase which only uses the minimum recombinant principle.

Future work directions include improving the efficiency of the PedRPoly method and testing the method in larger and real data sets.

Acknowledgments

This work is partially supported by Fundação para a Ciência e Tecnologia under research project SHIPs (PTDC/EIA/64164/2006) and PhD grant SFRH/BD/28599/2006.

References

1. A. M. Andrés, A. G. Clark, L. Shimmin, E. Boerwinkle, C. F. Sing, and J. E. Hixson. Understanding the accuracy of statistical haplotype inference with sequence data of known phase. *Genetic Epidemiology*, 31(7):659–671, 2007.
2. S. Climer, G. Jäger, A. Templeton, and W. Zhang. How frugal is mother nature with haplotypes? *Bioinformatics*, 25(1):68–74, 2009.
3. N. Eén and N. Sörensson. Translating pseudo-Boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, 2:1–26, 2006.
4. A. Graça, J. Marques-Silva, I. Lynce, and A. Oliveira. Efficient haplotype inference with pseudo-Boolean optimization. In *Algebraic Biology (AB'07)*, pages 125–139, 2007.
5. D. Gusfield. Haplotype inference by pure parsimony. In *Symposium on Combinatorial Pattern Matching (CPM'03)*, pages 144–155, 2003.
6. J. L. Haines. Chromlook: an interactive program for error detection and mapping in reference linkage data. *Genomics*, 14(2):517–519, 1992.
7. 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.
8. S. M. Leal, K. Yan, and B. MÃ$\frac{1}{4}$ller-Myhsok. SimPed: A simulation program to generate haplotype and genotype data for pedigree structures. *Human Heredity*, 60(2):119–122, 2005.
9. J. Li and T. Jiang. Efficient inference of haplotypes from genotypes on a pedigree. *Journal of Bioinformatics and Computational Biology*, 1(1):41–69, 2003.
10. J. Li and T. Jiang. Efficient rule-based haplotyping algorithms for pedigree data. In *International Conference on Research in Computational Molecular Biology (RECOMB'03)*, pages 197–206, 2003.

11. J. Li and T. Jiang. Computing the minimum recombinant haplotype configuration from incomplete genotype data on a pedigree by integer linear programming. *Journal of Computational Biology*, 12(6):719–739, 2005.
12. X. Li and J. Li. Comparison of haplotyping methods using families and unrelated individuals on simulated rheumatoid arthritis data. In *BMC Proceedings*, pages S1–S55, 2006.
13. X. Li and J. Li. Efficient haplotype inference from pedigree with missing data using linear systems with disjoint-set data structures. In *International Conference on Computational Systems Bioinformatics (CSB'08)*, pages 297–307, 2008.
14. S. Lin, A. Chakravarti, and D. J. Cutler. Haplotype and missing data inference in nuclear families. *Genome Research*, 14(8):1624–1632, 2004.
15. L. Liu, C. Xi, J. Xiao, and T. Jiang. Complexity and approximation of the minimum recombinant haplotype configuration problem. *Theoretical Computer Science*, 378(3):316–330, 2007.
16. J. Marchini, D. Cutler, N. Patterson, M. Stephens, E. Eskin, E. Halperin, S. Lin, Z. Qin, H. Munro, G. Abecassis, P. Donnelly, and International HapMap Consortium. A comparison of phasing algorithms for trios and unrelated individuals. *American Journal of Human Genetics*, 78(3):437–450, 2006.
17. S. H. Orzack, D. Gusfield, J. Olson, S. Nesbitt, L. Subrahmanyan, and J. V. P. Stanton. Analysis and exploration of the use of rule-based algorithms and consensus methods for the inferral of haplotypes. *Genetics*, 165(2):915–928, 2003.
18. D. Qian and L. Beckmann. Minimum-recombinant haplotyping in pedigrees. *American Journal of Human Genetics*, 70(6):1434–1445, 2002.
19. E. M. Wijsman. A deductive method of haplotype analysis in pedigrees. *American Journal of Human Genetics*, 41(3):356–373, 1987.
20. K. Zhang, F. Sun, and H. Zhao. HAPLORE: a program for haplotype reconstruction in general pedigrees without recombination. *Bioinformatics*, 21(1):90–103, 2005.

Statement of Ongoing Work: Extending Boolean Satisfiability Techniques for Haplotype Inference by Pure Parsimony

Eric I. Hsu and Sheila A. McIlraith

Department of Computer Science
University of Toronto
{eihsu,sheila}@cs.toronto.edu

Abstract. Here the authors overview an ongoing effort to extend satisfiability-based methods for haplotype inference by pure parsimony (HIPP). This genome analysis task, first formulated as a boolean satisfiability problem by Lynce and Marques-Silva [12], has been performed successfully by modern SAT-solvers. But, it is not as widely used as some better-publicized statistical tools, such as PHASE [19]. This paper presents the authors' assessment of the current situation, and a preliminary statement of intention concerning their aims in this area. Namely, the situation suggests three categories of improvements for making HIPP more widely-used within the biological community: 1) the ability to handle larger problems; 2) more detailed empirical understanding of the accuracy of the "pure parsimony" criterion; and 3) additional criteria and methods for improving on this level of accuracy.

1 Background

As detailed in a recent overview paper [11], the haplotype inference problem is defined over a population of individuals represented by their respective genotypes. Each genotype can be viewed as a sequence of nucleotide pairs, where the two values of each pair are split across the individual's two chromosomes as inherited from their two parents. Much of the genetic variation between individuals consists of point mutations at known sites within this sequence, known as single nucleotide polymorphisms (SNP's). Thus, a genotype can be represented (with loss of information) as a sequence of nucleotide pairs at successive SNP sites.

In particular, at each such site, an individual might have two copies of the "minor allele"– in this case, the presumably mutated or at least rarer of the two possible nucleotide values. At this site the individual is then *homozygous (minor)*. Similarly, a SNP site is realized on each chromosome by the nucleotide value that is most common for the species then the site is *homozygous (major)* for this individual. The third possibility is that one of the individual's chromosomes has the major allele at a given site, while the other chromosome has the minor allele at that site–then the genotype is *heterozygous* at the site in question. Notationally, the first case can be represented by the character '0', the second by '1', and the third, heterozygous case, by '2'. So, the sequence "0102" indicates an individual with two minor alleles at both the first and the third SNP sites measured by a particular study, and two major alleles at the second site. At the fourth site, we know that one chromosome exhibits the major allele and the other exhibits the minor. Thus, an individual's genotype is

well-defined by the sequences of its two constituent chromosomes; these two individually inherited sequences are called haplotypes. If we overload '0' and '1' to indicate a single minor or single major allele in a particular haplotype, then we can denote that the genotype "0102" arises from the two haplotypes "0100" and "0101".

Conversely, though, a genotype with more than one heterozygous site can be explained by multiple pairs of haplotypes, as the major and minor alleles at all heterozygous sites in the genotype can be permuted across corresponding sites in the two haplotypes. More concretely, the genotype "02122" could be realized by the pairs "00100"/"01111", "00101"/"01110", "00110"/"01101", or "01100"/"00111". (Naturally, the term "pair" is used colloquially here, and does not signify any sense of ordering within the haplotype sets of size two that arise in this domain.) Accordingly, there are 2^{k-1} candidate haplotype pairs for explaining a genotype with k heterozygous sites.

In practice, this distinction is made relevant by the lack of any practical experimental method to measure an individual's two haplotypes instead of its genotype; in other words, the machinery can identify sites at which the two haplotypes have opposing values, but cannot tell which values are grouped together on which chromosome. The goal of haplotype inference is to guess the most likely haplotypes that generated a given set of genotypes.

2 Underlying Biological Principles

How can one answer to the haplotype inference problem be preferred to any other? Because the ultimate goal is to accurately predict haplotypes appearing in a particular subject (*i.e.* human) population, haplotype inference frameworks must apply standards that seek to model the types of phenomena that actually drove the true state of affairs in the evolution of the subject species' genome. In other words, human haplotypes are not drawn uniformly from the space of all possible pairs that could explain human genotypes. Rather, under the coalescent model of evolution there should be only a small number of human haplotypes that were recombined and mutated to produce any of the genotypes within a given haplotype inference problem instance. Thus, early researchers used greedy methods to try to minimize the set of answers to a haplotype inference problem [3], while later systems integrated models based on "perfect phylogeny" [9] or other hierarchical organizations of answer haplotypes [22]. The most widely-used techniques at this point in time integrate empirical statistical measures of likely haplotypes [14,2,16,19,17], by analyzing the population in question, or consulting outside sources [20]. Such approaches may not require that the inferred set of answer haplotypes can be arranged into a particular evolutionary structure, but they all require the haplotype set to be maximally likely according to a particular statistical model that has been fit to the problem and/or outside frequency data.

On the other hand, the "pure parsimony" methodology seeks to capture such phenomena implicitly by asking directly for the *smallest* possible set of haplotypes that as a whole can explain a given population of genotypes [8]. Methods that are based on this criterion thus perform "HIPP", indicating haplotype inference by pure parsimony. This pure parsimony principle has been achieved optimally and efficiently by employing a variety of satisfiability-based techniques on small to moderately-sized data sets of about 200 sites and 100 individuals [12,7,5,13]. However, aside from a preliminary and limited evaluation of pure parsimony, which did not include such satisfiability-based techniques [21], the overall accuracy and general feasibility HIPP was for a long time somewhat unclear within the

biological community. In contrast, however, recent work based on satisfiability has made it possible to generate all HIPP models, allowing the observation of (presumably) unexpected phenomena: empirical data sets have a large number of HIPP solutions, of which possibly none match the actual known experimental solution [4]. (More informatively, the sizes of true haplotype sets that were found in the studied experiments were close to the the minimum found by HIPP, just not always exactly minimum.)

In short, satisfiability-based HIPP methods must demonstrate two forms of feasibility in order to achieve wider adoption within the biological community: the empirical accuracy of the pure parsimony principle itself, and the efficiency and scalability of SAT methods for achieving this parsimony criterion. While SAT-based techniques can achieve optimal parsimony, on reasonably large data sets, they still cannot be applied to the massive collections characteristic of more popular biological applications that may require on the order of half a million sites. Addressing the issue of scalability will not only enable the assessment of model accuracy and solver efficiency, but can additionally lead to more accurate results. This is because multiple minimum haplotype sets can explain the same population of genotypes [11], but some of them can be safely considered more likely *a priori*. For instance, solution sets whose members are more similar to each other are more realistic with respect to evolutionary theory [19,17], and certainly *one* must certainly be preferred *a posteriori* with respect to the truth–in the real world there was a single haplotype set that produced a set of genotypes (and this set may or not be minimum.) So while the initial goal is evaluation of the HIPP principle and HIPP solvers, and the prerequisite goal is improved solver scalability, in pursuing these we would like to simultaneously attain the overarching goal of making the most accurate predictions possible, as opposed to merely minimal ones.

We propose to do this by making finer-grained use of biological principles in designing SAT-based methodologies for HIPP. One prominent example of such phenomena would be "linkage disequilibrium", or correlation between sites within a sequence [18]. The sequential and mostly non-random nature of genotypes and haplotypes are at core of many of the optimizations and models that underly statistical alternatives to HIPP; finding a way to exploit them within a discrete reasoning framework would make HIPP competitive in efficiency and accuracy. Three proposals for doing so are outlined in the next section.

3 Proposals for Extending the Satisfiability-Based HIPP Framework

In this section we propose three general types of improvements to the HIPP framework that can make it competitive to the current methods of choice within the biological community.

– **Exploiting sequential structure to improve scalability.** The best-performing (and most widely-used) statistical systems [14,2,19] for haplotype inference utilize explicit or implicit variants of the "partition ligation" scheme of Niu *et al.* [15]. The basic idea is to escape the combinatorial explosion of considering the entire space of possible haplotypes for a given sequence, and instead break the sequence into blocks. Each such block is small enough to be solved efficiently and to high accuracy; they are then recombined heuristically through a polynomial-time merging scheme. With the merging scheme comes a loss of optimality; in the case of HIPP we may not get the smallest possible explaining haplotype set by means of this scheme. But, the insight of the

partition ligation scheme is that evolution does not create haplotypes uniformly at random over all possible explanations for the human genotype, and in practice the loss in optimality has proved negligible in comparison to the gains in accuracy for statistical methods [15,14,19]. For beyond being biologically justifiable, block partitioning can actually produce more accurate overall results because each block can be solved to higher standards of likelihood using more computationally expensive methods. For instance, a statistical approach based on MCMC sampling can choose to perform vastly more iterations on a more complex model within the confines of a small block, while realistically hoping to retain much of the benefit during the merging process [19]. Applying this framework to SAT-based approaches can provide similar gains, whether using parsimony alone or integrating statistical information in solving blocks. At this point, we are able to solve individual blocks using the same sorts of methods as Lynce *et al.*, and are experimenting with both a direct translation of Niu's partition ligation scheme, and also with alternative approaches that seek to optimize specific objective functions when performing the merging.

– **Assess the accuracy of HIPP on large-scale data.** Because it is experimentally difficult to determine ground truth haplotype phase from real data [1], most empirical studies have been evaluated the accuracy of various inference methods using data sets that cover up to around 100 SNP sites [14,4]. However, one of the original, highest-level applications for haplotype inference is as a first step genome-wide association studies; HIPP-based approaches are not alone in facing a new hurdle in handling problems with hundreds of thousands of sites. When HIPP is able to handle larger data sets, it will be possible to directly compare its accuracy with other, more-popular approaches. This will allow an assessment of the model's strengths and weaknesses and inform any attempts to improve this accuracy. At this point, SAT methods have been highly successful at solving HIPP, but it remains to be seen whether the resulting answers are themselves successful at modeling the human genome–as mentioned previously here and elsewhere [11,4], there can be many minimum sets that all qualify as HIPP solutions, while some of these are much more empirically likely than others. In empirical studies with a limited number of sites, the true set of haplotypes has not tended to be absolutely minimum in size [4]. Characterizing which types of these solutions usually turn out to be more accurate will go a long way to improving the parsimony model's fit to real populations.

– **Exploiting linkage disequilibrium to improve accuracy.** In the same spirit as the first two points, there are important correlations between various regions of the vast majority of a species' haplotypes, due to the recombination and mutation processes that drive evolution. To compete with statistically-oriented tools, HIPP can be extended to encompass the same sort of empirical information concerning such correlations [6,10]. This may entail a weighted SAT or a MAXSAT formulation that favors solutions that adhere to stronger correlations that have been observed from the same sources of data as used by the competitor techniques. This has motivated to types of developments within the system's SAT-solving framework. The first is to express statistical information as weights that direct the solver to prefer one model over another; the second is to integrate such information into heuristics that guide the search to efficiently optimize over such preferences.

4 Conclusion

The authors have begun to implement block decomposition within the SAT-based HIPP framework, but this description of research is decidedly preliminary and strictly for expository purposes. Once the system is able to handle large problems, the next step will be to study its accuracy, especially in terms of adding additional solution criteria to pure parsimony. The concluding step will be to use information derived from individual problem-instance and/or reference data to actually achieve such criteria. It would be a great opportunity to be able to discuss such plans with others who are working on this problem and related areas[1]!

References

1. Aida M. Andrés, Andrew G. Clark, Lawrence Shimmin, Eric Boerwinkle, Charles F. Sing, and James E. Hixson. Understanding the accuracy of statistical haplotype inference with sequence data of known phase. *Genetic Epidimiology*, 31(7):659–671, 2007.
2. Sharon R. Browning. Missing data imputation and haplotype phase inference for genome-wide association studies. *Human Genetics*, 124:439–450, 2008.
3. Andrew G. Clark. Inference of haplotypes from PCR-amplified samples of diploid populations. *Molecular Biology and Evolution*, 7(2):111–122, 1990.
4. Sharlee Climer, Gerold Jäger, Alan R. Templeton, and Weixiong Zhang. How frugal is mother nature with haplotypes? *Bioinformatics*, 25(1):68–74, 2009.
5. Esra Erdem and Ferhan Türe. Efficient haplotype inference with answer set programming. In *Proc. of 23rd National Conference on A.I. (AAAI '08), Chicago, IL*, pages 436–441, 2008.
6. Laurent Excoffier and Montgomery Slatkin. Maximum-Likelihood estimation of molecular haplotype frequencies in a diploid population. *Molecular Biology and Evolution*, 12(5):921–927, 1995.
7. Ana Graça, João Marques-Silva, Inês Lynce, and Arlindo L. Oliveira. Efficient haplotype inference with pseudo-boolean optimization. In *Proc. of 2nd Int'l Conf. on Algebraic Biology (AB '07), Linz, Austria*, pages 125–139, 2007.
8. Dan Gusfield. Haplotype inference by pure parsimony. In *Proc. of 14th Symp. on Combinatorial Pattern Matching (CPM '03), Morelia, Mexico*, pages 144–155, 2003.
9. Gad Kimmel and Ron Shamir. The incomplete perfect phylogeny problem. *Bioinformatics and Computational Biology*, 3(2):359–384, 2005.
10. Mary K. Kuhner. LAMARC 2.0: Maximum likelihood and Bayesian estimation of molecular haplotype frequencies in a diploid population. *Bioinformatics*, 22(6):768–770, 2006.
11. Inês Lynce, Ana Graça, João Marques-Silva, and Arlindo L. Oliveira. Haplotype inference with boolean constraint solving: An overview. In *Proc. of 20th IEEE Int'l Conf. on Tools with Artificial Intelligence (ICTAI '08), Dayton, OH*, 2008.
12. Inês Lynce and João Marques-Silva. Efficient haplotype inference with boolean satisfiability. In *Proc. of 21st National Conference on Artificial Intelligence (AAAI '06), Boston, MA*, 2006.
13. Inês Lynce, João Marques-Silva, and Steven Prestwich. Boosting haplotype inference with local search. *Constraints*, 13(1-2):155–179, 2008.
14. Jonathan Marchini, David Cutler, Nick Patterson, Matthew Stephens, Eleazar Eskin, Eran Halperin, Shin Lin, Zhaohui S. Qin, Heather M. Munro, Gonçalo R. Abecasis, and Peter Donnelly. A comparison of phasing algorithms for trios and unrelated individuals. *American Journal of Human Genetics*, 78:437–450, 2006.

[1] The authors would like to thank the anonymous reviewers for their considered and useful comments.

15. Tianhua Niu, Zhaohui S. Qin, Xiping Xu, and Jun S. Liu. Bayesian haplotype inference for multiple linked single nucleotide polymorphisms. *American Journal of Human Genetics*, 70(1):157–169, 2002.
16. Rany M. Salem, Jennifer Wessel, and Nicholas J. Schork. A comprehensive literature review of haplotyping software and methods for use with unrelated individuals. *Human Genomics*, 2(1):39–66, 2005.
17. Paul Scheet and Matthew Stephens. A fast and flexible statistical model for large-scale population genotype data: Applications to inferring missing genotypes and haplotypic phase. *American Journal of Human Genetics*, 78:629–644, 2006.
18. Montgomery Slatkin. Linkage disequilibrium–understanding the evolutionary past and mapping the medical future. *Nature Reviews Genetics*, 9:477–485, 2008.
19. Matthew Stephens and Paul Scheet. Accounting for decay of linkage disequilibrium in haplotype inference and missing-data imputation. *American Journal of Human Genetics*, 76:449–462, 2005.
20. The International HapMap Consortium. A second generation human haplotype map of over 3.1 million SNPs. *Nature*, 449:851–862, 2007.
21. Lusheng Wang and Ying Xu. Haplotype inference by maximum parsimony. *Bioinformatics*, 19(14):1773–1780, 2003.
22. Eric P. Xing, Kyung-Ah Sohn, Michael I. Jordan, and Yee-Whye Teh. Bayesian multi-population haplotype inference via a hierarchical dirichlet process mixture. In *Proc. of 23rd Int'l Conf. on Machine Learning (ICML '06), Pittsburgh, PA*, 2006.

Equivalence Classes of Optimal Structures in HP Protein Models Including Side Chains

Martin Mann, Rolf Backofen, and Sebastian Will

University of Freiburg, Bioinformatics, 79110 Freiburg, Germany,
{mmann,backofen,will}@informatik.uni-freiburg.de

Abstract. Lattice protein models, as the Hydrophobic-Polar (HP) model, are a common abstraction to enable exhaustive studies on structure, function, or evolution of proteins. A main issue is the high number of optimal structures, resulting from the hydrophobicity-based energy function applied. We introduce an equivalence relation on protein structures that correlates to the energy function. We discuss the efficient enumeration of optimal representatives of the corresponding equivalence classes and the application of the results.

1 Introduction

Proteins are the central players in the game of life. They are involved in almost all processes in cells and organisms, comprising replication, metabolism, and movement. To be able to perform their specific functions, proteins have to adopt a certain fold or structure, which is encoded by the protein's sequence. Thus, knowledge of a protein's structure elucidates the mechanisms it is involved.

Currently, it is not possible to calculate a protein's functional fold from its sequence nor to simulate the whole folding process in detail. Simplified protein models are used to reduce the computational complexity. A common abstraction are lattice proteins [5,7]. Here, the structure space a protein can adopt is discretized and allows for efficient folding simulations [19,13]. Nevertheless, it is difficult to determine minimal energy structures, which represent the functional folds in such models. Even in the most simple Hydrophobic-Polar (HP) model [14], the optimal structure prediction problem stays computationally hard (NP-complete) [4]. Despite this complexity, a fast calculation of non-symmetrical optimal structures in the HP model is possible using constraint programming techniques applied in the *Constraint-based Protein Structure Prediction (CPSP)* approach [3,20,22,15].

Recently, we have introduced a significantly improved local search scheme for lattice protein folding simulations [29] using a full Miyazawa-Jernigan energy potential [11]. We take advantage of the efficient CPSP approach and initialize the folding simulations with optimal structures from the simpler HP model. This incorporates the phenomenon of hydrophobic collapse of protein structures, a driving force at the beginning of the folding process [1]. The already compact structures from the CPSP application form the starting point of the folding driven by more complex interactions. This scheme outperforms folding simulations using a standard initialization with random structures and yields better results within shorter simulation time [29]. To increase efficiency of local search methods, usually many optimization runs from different starting points are done. Since the set of all HP-optimal structures is usually too large as a starting set, we are interested in a smaller subset

that still covers the structural diversity of the whole set as good as possible. We achieve this by enumerating optimal structures that maintain a given minimal distance to each other.

Due to the hydrophobicity-focusing energy function, proteins in HP models show on average a huge number of optimal structures. Since polar residues do not contribute to the energy, optimal structures usually show a much higher variation in the placement of polar than hydrophobic residues.

Here, we introduce an equivalence relation to partition the set of (optimal) structures into according classes. Two structures are defined to be equivalent, iff they do not differ in the placements of their hydrophobic residues. We introduce an extension to the CPSP approach that enables an efficient calculation of the number of equivalence classes of optimal structures via enumerating one representative per class. The approach is presented for backbone-only and side chain incorporating HP models. We show that a sequence's number of representatives (later defined as core-degeneracy) is several magnitudes smaller than the overall number of all optimal structures (degeneracy).

Thus, the set of optimal representatives is well placed to be used within the combined approach of CPSP and local search [29]. Furthermore, we propose another application of the equivalence classes: Since the equivalence relation is highly correlated to the HP energy function, the number of classes might be a better measure of structural stability than a sequences' degeneracy [12].

2 Preliminaries

A lattice protein in the HP model is specified by its sequence $S \in \{H, P\}^n$, where H and P denote hydrophobic and polar monomers, respectively. The structure positions are confined to nodes of a regular lattice $L \subseteq \mathbb{Z}^3$. A valid *backbone-only* structure $C \in L^n$ of length n is a self-avoiding walk (SAW) in the underlying lattice L, i.e. it holds connectivity $\forall_{1 \leq i < n} : (C_i - C_{i+1}) \in N_L$ and self-avoidance $\forall_{1 \leq i < j \leq n} : C_i \neq C_j$, where N_L denotes the set of distance vectors between neighbored points in L. An example is shown in Fig. 1a). The energy of a lattice protein structure is given by non-consecutive HH-contacts:

$$E(S, C) = \sum_{\substack{1 \leq i < j \leq n \\ (i+1) < j}} \begin{cases} -1 & : (C_i - C_j) \in N_L \wedge S_i = S_j = H \\ 0 & : \text{otherwise} \end{cases} \tag{1}$$

An *optimal structure* minimizes the energy function. The number of optimal structures is denoted as *degeneracy* of a sequence and is an important measure of structural stability [12].

The *CPSP-approach* by Backofen and Will [3] enables the calculation of a sequence's degeneracy without full structure space enumeration [15]. It utilizes the observation that optimal structures show a (nearly) optimal packing of H monomers. Thus, the CPSP-approach can be sketched in two major steps:

1. *H-core construction:* Given the number n_H of H monomers from the target sequence S, all optimal packings of n_H monomers are calculated. These *optimal H-cores* show the maximal number of contacts possible. For a fixed sequence S and the corresponding n_H, we denote the set of optimal H-cores with $\mathcal{O}$. The calculation of $\mathcal{O}$ is computationally difficult on its own and was solved by us using constraint programming [2,3].

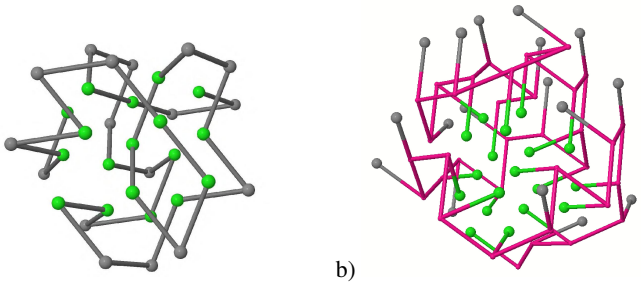


Fig. 1. Optimal structures of `HPPHHPPPHPHHPHHPPHPHPPHHHPHHPPHPHPH` in the face-centered-cubic lattice. (a) backbone-only model with energy -50, (b) side chain model with energy -55. Colors: green - H monomers, gray - P monomers, red - backbone in side chain models. Visualization by `HPview` from CPSP-package [22].

2. *Structure threading:* Given S and $\mathcal{O}$ only structures are enumerated where the H monomers of S are confined to an optimal H-core $O \in \mathcal{O}$, i.e. they are "threaded" through the H-cores. Since all O show the maximally possible number of contacts between H monomers, each resulting structure is optimal according to Eq. 1 as well. The structure threading is done by solving a Constraint Satisfaction Problem (CSP) for each $O \in \mathcal{O}$ as given below.

Since step 1 depends only on the number of H monomers n_H and no further property of any sequence, we can precalculate the H-cores for different n_H and store them in a database. This significantly speeds up the approach and reduces the computation time to step 2, i.e. the structure threading.

It might happen, that we find no appropriate structure threading for a sequence S and the according set of optimal H-cores $\mathcal{O}$. Thus, we revert to the set of the best suboptimal H-cores $\mathcal{O}'$ that show at least one contact less than an optimal H-core $O \in \mathcal{O}$ and iterate the procedure. Still it holds: the first successive structure threading is an optimal structure, since no H monomer packing with more contacts was found before. Further details on the CPSP approach in [3].

The CSPs solved in step 2 are given by $(\mathcal{X}, \mathcal{D}, \mathcal{C})$, where we denote the set of variables $\mathcal{X}$, their domains $\mathcal{D}$, and a set of constraints $\mathcal{C}$. For each monomer $S_i \in S$ a variable $X_i \in \mathcal{X}$ is introduced. The SAW is modeled by a sequence of binary neighboring constraints $\mathrm{neigh}(X_i, X_{i+1})$ and a global $\mathrm{alldiff}(\mathcal{X})$ to enforce the self-avoidingness. The optimal H-core $O \in \mathcal{O}$ is used to define the domains $\mathcal{D}$: $\forall_{i:S_i=H} : \mathcal{D}(X_i) = O$ and $\forall_{i:S_i=P} : \mathcal{D}(X_i) = L \setminus O$. Thus, if we find a solution of such a CSP, i.e. an assignment $a_i \in \mathcal{D}(X_i)$ for each variable that satisfies all constraints in $\mathcal{C}$, it will minimize the energy function in Eq. 1, i.e. an optimal structure.

3 Representative Optimal Structures

Revisiting the CSP we can see, that P monomers are constrained only by the SAW constraints. Imagine a sequence with a long tail of P monomers. Each valid placement of the subchain in front of the tail can be combined with a combinatorial number of possible SAWs of the tail. This leads to the immense degeneracy in the HP model.

Therefore, we set up an *equivalence relation* $\overset{H}{\sim}$ on structures (Eq. 2) that decomposes the set of all (optimal) structures into equivalence classes. In the following, the number of equivalence classes of optimal structures is denoted as *core-degeneracy*. As given by Eq. 2, structures from different equivalence classes differ in at least one H monomer placement.

$$C \overset{H}{\sim} \hat{C} \Leftrightarrow \forall_{i|S_i=H} : C_i = \hat{C}_i. \tag{2}$$

The representative enumeration (that corresponds to core-degeneracy calculation) can be done via an extension of the CPSP approach presented in Sec. 2. Instead of calculating all optimal structures, we want to calculate only one representative per equivalence class. This has to be ensured at two stages: (I) the solutions of each single CSP for a given H-core have to be different according to Eq. 2, and (II) the solutions from two CSPs for two different H-cores have to be different as well. The second condition (II) holds by definition, because $\overset{H}{\sim}$ is only defined on the H monomer placements that are constrained by different H-cores from $\mathcal{O}$ (differing in at least one position). In the following, we will discuss how to achieve the difference for solutions of a single CSP (I).

Note that the core-degeneracy, i.e. the number of different placements of H-monomers, or core-configurations, in optimal structures of a sequence, is *not equal* to the number of different H-cores, which are the sets of lattice points that are occupied by H-monomers. The latter number is easily obtained from the standard prediction algorithm, described in Sec. 2. It equals the number of cores, where the sequence is successfully threaded on.

Restricted Search for Enumeration of Representatives

The standard way to solve a CSP is a combination of domain filtering (i.e. constraint propagation) and depth first search. This results in a binary tree where each node represents a subproblem of the initial CSP (root) and edges represent the additional constraints added to derive the two subproblems from its predecessor node (CSP). The constraints c and $\neg c$ added to derive the leave nodes of a certain CSP are often of the form $c = (X_i \equiv d)$ by selecting a variable X_i from $\mathcal{X}$ and a value $d \in \mathcal{D}(\mathcal{X}_i)$ according to some heuristics. The constraint solver traverse the binary tree until a solution was found or an inconsistency of a constraint from $\mathcal{C}$ was detected.

Therefore, a straightforward way to enumerate only one representative for each equivalence class can be sketched as follows: first, we restrict the search of the solving process onto the H associated variables. Then, we perform a single check for satisfiability, i.e. search for a single assignment of P monomer variables fulfilling all constraints in $\mathcal{C}$. Thus, we get only one P monomer placement for a given H monomer assignment if any exists.

The drawback of this approach is that we restrict the variable order of the search heuristics. But the performance of the CPSP approach mainly depends on the search heuristics applied to select a certain variable or value from its domain. It turned out that a mixed assignment of H and P associated variables yields the best runtimes. These heuristics can not be applied within the sketched procedure where we have to first assign H-associated variables, then P-associated ones. Thus, a lower CPSP performance is expected. But, we have to do less search which results in much faster runtimes than enumerating all optimal structures.

4 Representative Optimal Structures with Side Chains

Recently, we have introduced the extension of the CPSP approach [20] to *HP models including side chains* [7]. Here, each amino acid of a protein sequence is represented by two monomers: C_i^b representing the backbone atoms, and C_i^s representing the atoms of the side chain. Beneath the SAW condition on the backbone monomers C_i^b, we constrain each side chain to be neighbored to its backbone, i.e. $\forall_{1 \leq i \leq n} : (C_i^s - C_i^b) \in N_L$. An example structure is given in Fig. 1b). The applied energy function E' exploits only HH-contacts of side chain monomers C_i^s:

$$E'(S, C^s) = \sum_{1 \leq i < j \leq n} \begin{cases} -1 & : (C_i^s - C_j^s) \in N_L \wedge S_i = S_j = H \\ 0 & : \text{otherwise} \end{cases} \tag{3}$$

Therefore, the side chain models show an even higher degeneracy than the backbone-only models discussed so far, since all backbone monomers C_i^b are unconstrained by the energy function as well. Thus, an equivalence relation $\overset{H}{\approx}$ that focuses on the monomers constrained by the energy function is even more striking in HP models including side chains. The relation $\overset{H}{\approx}$ is given by

$$(C^b, C^s) \overset{H}{\approx} (\hat{C}^b, \hat{C}^s) \Leftrightarrow \forall_{i:S_i=H} : C_i^s = \hat{C}_i^s \tag{4}$$

Therefore, we will enforce that structures from one equivalence class show the same H monomer side chain positioning. The CPSP approach for HP models including side chains differs only in the CSP formulation from the original approach for backbone-only models [20]. This allows for the application of the same approach discussed in the previous section to enumerate non-equivalent optimal structure representatives. Thus, we restrict search to the H associated side chain variables first and only check for satisfiability on the remaining variables.

5 Results and Discussion

We exemplify the enumeration of representatives for backbone-only and side chain models. We focus on the comparison of the resulting core-degeneracy of a sequence and its overall number of optimal structures, i.e. degeneracy, because we are interested in a reduced set of optimal structures, e.g. for local search initialization (see introduction). All following results are given for HP-sequences of length 27 in 3D cubic lattice. Since the enumeration and check of all 2^{27} sequences ($> 10^8$) is computationally not feasible, we restrict each study to a large randomly chosen subset of 10^5 and 10^4 sequences, respectively.

The program HPREP implements the approach from section 3. It is integrated into the CPSP-tools package [22] version 2.4.0 and available online[1].

As discussed in Sec. 2, the structure threading step of the CPSP approach screens through a precomputed list of appropriate H-cores in decreasing number of contacts stored in a database. Therefore, it might occur that the available list from the database is exceeded without any solution, i.e. no optimal structure was computed. Still, the energy of the last

[1] `http://cpsp.informatik.uni-freiburg.de`

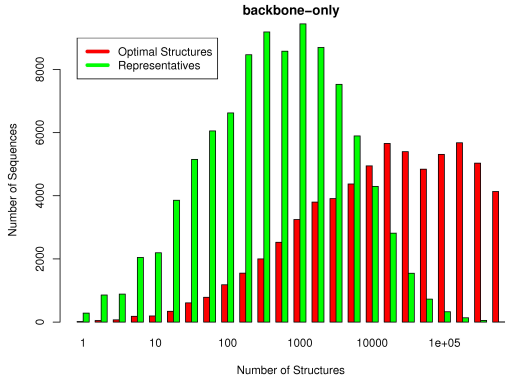


Fig. 2. Backbone-only models : Histogram of core-degeneracy (green) and degeneracy (red) with cut-off $\leq 10^6$. (Plots refer to 3D cubic lattice and sequence length 27.)

H-core tried is a close lower bound on the energy this sequence can adopt. In the following, $\mathcal{B}$ denotes the subset of sequences where the current H-core database is not sufficient and thus the CPSP approach can give only a lower bound for now. The number of sequences in $\mathcal{B}$ is quite small. It is reasonable to assume that the degeneracy distribution among $\mathcal{B}$ is the same as for the remaining sequences or on average even higher.

Backbone-only models

We tested 10^5 random sequences in the backbone-only model in the 3D cubic lattice. Here, only 66% show a degeneracy below 10^6. $\mathcal{B}$ comprises about 4% of the sequences. The remaining 30% can adopt even more than 10^6 structures with minimal energy.

Figure 2 summarizes the results: in *red* the degeneracy and in *green* the core-degeneracy distribution with cut-off 10^6 is presented. Thus, in *red* the degeneracy distribution comprises 66% of the sequences as given above. In contrast, *all* sequences show a number of optimal equivalence classes below 10^6 (in *green*)! The average degeneracy is reduced from 124800 (with cutoff 10^6) to a mean core-degeneracy of 4856. This reduction within two orders of magnitude results in reasonably small sets of representative structures e.g. to be utilized in local search initializations. Furthermore, the enumeration of representatives is on average six times faster than the enumeration of all optimal structures with a mean runtime of 2 seconds (Opteron 2356 - 2.3 GHz).

This increase of small sets of representatives compared to the complete sets of optimal structures shows the advantage of the approach: core-degeneracy does not show the huge combinatorial explosion of degeneracy. This gets even more striking in HP models including side chains, as shown in the next section.

Models including side chains

The degeneracy in HP models including side chains is much higher than for backbone-only models. This results from the simple energy function (Eq. 3) that does not constrain the backbone or P monomers. Therefore, an immense number of optimal structures is present. From the 10^4 HP-sequences tested only 408 show a degeneracy below 10^6. $\mathcal{B}$ comprises again about 3.1% of the sequences.

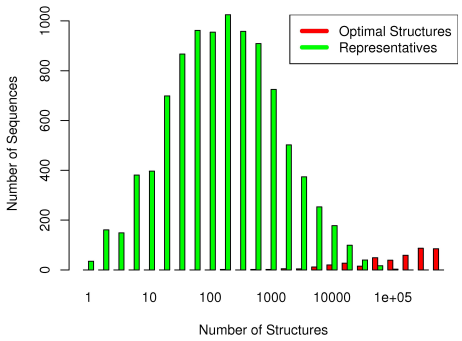


Fig. 3. Models with side chains : Histogram of the core-degeneracy *(green)* and degeneracy *(red)* with cut-off $\leq 10^6$. Note: only 408 sequences out of 10^4 showed a degeneracy below 10^6 as given in the text. (Plots refer to 3D cubic lattice and sequence length 27.)

When investigating core-degeneracy the picture changes completely: *All* of the sequences tested have less than 10^6 representatives. Figure 3 summarizes the distribution. The average number of representatives is about 1550, which is again at least three orders of magnitude smaller than the average degeneracy. Since we have only a very rough lower bound of 10^6 on the average degeneracy (due to the cut-off), the real reduction ratio is expected to be even higher.

6 Conclusions

The introduced equivalence relations for HP models enables a energy function driven partitioning of structures. The presented CPSP approach extension enables an efficient calculation of representatives for all equivalence classes of optimal structures, i.e. calculation of a sequence's core-degeneracy. Using our implementation HPREP, we showed that sequences show several orders of magnitudes less optimal equivalence classes than optimal structures. This is most striking in models including side chains.

The sets of representatives are usually small. Furthermore, representatives show different hydrophobic core arrangements. Therefore, they are well placed to be used for the initialization of local search procedures that utilize more complex energy functions [29]. This emulates the hydrophobic collapse in the folding process.

Since a sequence's degeneracy is a measure of structural stability [12], we propose another application of our approach. The core-degeneracy might be used as a more reasonable *measure of stability* in the HP model compared to degeneracy. It ignores the HP model specific degeneracy blow-up due to unconstrained subchains of P monomers (see section 3). Thus, a structural stability analysis could be based on the presented equivalence classes instead of all possible structures.

References

1. Vishwas R. Agashe, M. C. R. Shastry, and Jayant B. Udgaonkar. Initial hydrophobic collapse in the folding of barstar. *Nature*, 377:754–757, 1995.
2. Rolf Backofen and Sebastian Will. Optimally compact finite sphere packings — hydrophobic cores in the FCC. In *Proc of CPM'01*, volume 2089 of *LNCS*, pages 257–272, 2001.

3. Rolf Backofen and Sebastian Will. A constraint-based approach to fast and exact structure prediction in three-dimensional protein models. *J Constraints*, 11(1):5–30, Jan 2006.
4. Bonnie Berger and Tom Leighton. Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete. *J Comp Biol*, 5:27–40, 1998.
5. K. A. Dill, S. Bromberg, K. Yue, K. M. Fiebig, D. P. Yee, P. D. Thomas, and H. S. Chan. Principles of protein folding – a perspective of simple exact models. *Protein Science*, 4:561–602, 1995.
6. S. Bromberg K. A. Dill. Side-chain entropy and packing in proteins. *Protein Sci*, 3(7):997–1009, 1994.
7. Kit F. Lau and Ken A. Dill. A lattice statistical mechanics model of the conformational and sequence spaces of proteins. *Macromol.*, 22(10):3986–3997, 1989.
8. Martin Mann, Daniel Maticzka, Rhodri Saunders, and Rolf Backofen. Classifying protein-like sequences in arbitrary lattice protein models using LatPack. *HFSP Journal*, 2(6):396, 2008.
9. Martin Mann, Cameron Smith, Mohamad Rabbath, Marlien Edwards, Sebastian Will, and Rolf Backofen. CPSP-web-tool : a server for 3D lattice protein studies. *Bioinformatics*, 25(5):676–677, 2009.
10. Martin Mann, Sebastian Will, and Rolf Backofen. CPSP-tools - exact and complete algorithms for high-throughput 3D lattice protein studies. *BMC Bioinformatics*, 9:230, 2008.
11. S. Miyazawa and R. L. Jernigan. Residue-residue potentials with a favorable contact pair term and an unfavorable high packing density term, for simulation and threading. *J Mol Biol*, 256(3):623–44, 1996.
12. David Shortle, Hue S. Chan, and Ken A. Dill. Modeling the effects of mutations on the denatured states of proteins. *Prot Sci*, 1:201–215, 1992.
13. Kathleen Steinhöfel, Alexandros Skaliotis, and Andreas A. Albrecht. Stochastic protein folding simulation in the d-dimensional HP-model. In *Proc of BIRD'07*, pages 381–394. Springer, 2007.
14. Abu Dayem Ullah, Leonidas Kapsokalivas, Martin Mann, and Kathleen Steinhöfel. Protein folding simulation by two-stage optimization. In *Proc. of ISICA'09*, CCIS, Wuhan, China, Oct 2009. Springer. (accepted).
15. Sebastian Will and Martin Mann. Counting protein structures by DFS with dynamic decomposition. In *Proc of WCB'06*, pages 83–90, 2006.

Constraint-based Local Move Definitions for Lattice Protein Models Including Side Chains

Martin Mann[1], Mohamed Abou Hamra[2], Kathleen Steinhöfel[3], and Rolf Backofen[1]

[1] University of Freiburg, Bioinformatics, 79110 Freiburg, Germany,
{mmann,backofen}@informatik.uni-freiburg.de,
[2] German University in Cairo, 5th Settlement New Cairo City, Cairo, Egypt,
mohamed.abdel-fattah@student.guc.edu.eg,
[3] King's College London, Department of Computer Science, London WC2R 2LS, UK,
kathleen.steinhofel@kcl.ac.uk,

Abstract. The simulation of a protein's folding process is often done via stochastic local search, which requires a procedure to apply structural changes onto a given conformation. Here, we introduce a constraint-based approach to enumerate lattice protein structures according to k-local moves in arbitrary lattices. Our declarative description is much more flexible for extensions than standard operational formulations. It enables a generic calculation of k-local neighbors in backbone-only and side chain models. We exemplify the procedure using a simple hierarchical folding scheme.

1 Introduction

The *in silico* determination of a protein's functional fold is a well established problem in bioinformatics. Since X-ray or NMR studies are still time consuming and expensive, computational methods for *ab initio* protein structure prediction are needed. Despite research over the last decades, a direct calculation of minimal energy structures in full atom resolution is currently not feasible. Thus, heuristics and a wide variation of protein models have been developed to identify fundamental principles guiding the process of structure formation. A common abstraction of proteins are lattice protein models [3,13,14,20]. Their discretized structure space enables efficient folding simulations [29,31] while maintaining good modelling accuracy [25].

Folding simulations are often based on stochastic local searches, e.g. Monte Carlo simulations [29]. Different procedures, so called *move sets*, have been developed to calculate the structural changes along the simulation, i.e. to enumerate the structural neighborhood of a certain structure. A method often applied in literature are *k-local moves* [28] that allow for structural changes within a successive interval of fixed length k. They are discussed in detail in Sec 3. Dotu and co-workers have used local moves for backbone-only HP models within a constraint-based large neighborhood search for optimal protein structures [9]. Lesh *et al.* introduced *pull moves* [15] that are widely used in recent studies [19,29]. *Pivot moves* allow for the rotation or reflection of subchains at an arbitrary Pivot position of the structure [17], while Zhang *et al.* suggested a sequential regrowth of structure fragments to enhance folding simulations [31].

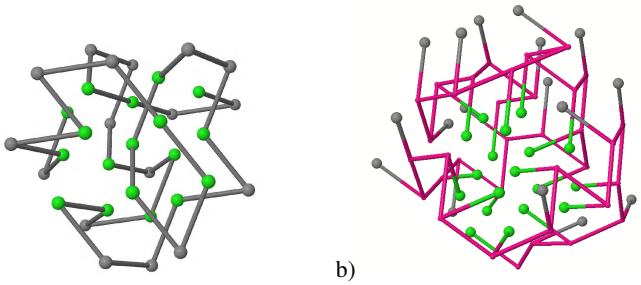
All named move sets are currently restricted to backbone-only lattice protein models, i.e. only the C_α-trail of the protein is modeled. For more realistic protein models incorporating

side chains, often a combination of different move sets is applied. Betancourt combined Pivot moves on the backbone with a new FEM move set [5], while Dima and Thirumalai have used a combination of 2-local moves on the backbone with a simple relocation of the side chain [8]. An exception is the advanced CABS model by Kolinski and co-workers [13], which represents the side chain in higher detail and requires more complex moves.

Here, we introduce a generic and flexible approach to enable folding simulations in backbone-only and side chain models using any k-local moves (i.e. any interval length k) in arbitrary lattices. The constraint programming (CP) based formulation focuses on a description of the targeted structural neighbors instead of an operational encoding of the moves possible. The introduced scheme is therefore easy to extend with new directives or can be used for other applications, e.g. fragment re-localization [31], as discussed later. Beneath applications in studies of the whole energy landscape [21], the approach is well placed to be applied within a local search following the framework of Pesant and Gendreau [26]. We apply our move set for side chain models within a simple folding simulation procedure in the style of [29] and evaluate the results with known protein structures.

2 Preliminaries

Given a lattice $L \subseteq \mathbb{Z}^3$ and an according neighborhood relation $\overset{L}{\sim}$ between coordinates of L. A *backbone-only* lattice protein of length n is described by (S, C) where $S \in \Sigma^n$ denotes the sequence over some alphabet Σ (e.g. the 20 proteinogen amino acids) and $C \in L^n$ the lattice nodes occupied. A valid lattice protein structure satisfies connectivity of successive monomers $\forall_{1 \leq i < n} : C_i \overset{L}{\sim} C_{i+1}$ and their self-avoidingness $\forall_{1 \leq i < j \leq n} : C_i \neq C_j$. A *side chain* lattice protein is defined by $(S, C^{\mathrm{b}}, C^{\mathrm{s}})$, i.e. a sequence $S \in \Sigma^n$, the backbone positions $C^{\mathrm{b}} \in L^n$ and the side chain positions $C^{\mathrm{s}} \in L^n$. The side chain position C^{s} represents the centroid of the amino acid's side chain atoms. A valid lattice protein structure including side chains satisfies connectivity of successive backbone monomers $\forall_{1 \leq i < n} : C^{\mathrm{b}}_i \overset{L}{\sim} C^{\mathrm{b}}_{i+1}$, the connection of backbone and side chain for each amino acid $\forall_{1 \leq i \leq n} : C^{\mathrm{b}}_i \overset{L}{\sim} C^{\mathrm{s}}_i$, and the selfavoidingness of all monomers $\forall_{1 \leq i < j \leq n} : C^{\mathrm{b}}_i \neq C^{\mathrm{b}}_j \wedge C^{\mathrm{s}}_i \neq C^{\mathrm{s}}_j \wedge C^{\mathrm{b}}_i \neq C^{\mathrm{s}}_j \wedge C^{\mathrm{b}}_i \neq C^{\mathrm{s}}_i$. We consider the contact based energy functions $E^{\mathrm{b}}(S, C) = \sum_{1 \leq i < j \leq n}^{(C_i \overset{L}{\sim} C_j)} e(S_i, S_j)$ for backbone-only and $E^{\mathrm{s}}(S, C^{\mathrm{b}}, C^{\mathrm{s}}) = \sum_{1 \leq i < j \leq n}^{(C^{\mathrm{s}}_i \overset{L}{\sim} C^{\mathrm{s}}_j)} e(S_i, S_j)$ for side chain lattice proteins for a given energy contribution function $e : \Sigma \times \Sigma \to \mathcal{R}$. Note, the energy function for side chain proteins considers (as in [7]) the contacts between side chain positions only! e^{20} denotes an empirical 20 amino acid contact potential as described in [4,23]. e^{HP} represents the energy contribution function of the Hydrophobic-Polar (HP) model [14], i.e. it returns -1 if both amino acids are hydrophobic, 0 otherwise. Our hydrophobic/polar (H/P) assignment follows [29]. An *optimal structure* minimizes the energy function. In the following, we denote a structure *HP-optimal* if it minimizes the energy function based on e^{HP}. Figure 1 exemplifies HP-optimal structures for both lattice protein models. In the following, we assume a scaled lattice such that neighbored positions in the lattice have a distance of $3.8\mathring{A}$, the average C_α-atom distance in proteins.



a) b)

Fig. 1. HP-optimal structures of HPPHHPPPHPHHPHHPPHPHPPHHHPHHPPHPHPH in the face-centered-cubic lattice. (a) backbone-only model with energy -50, (b) side chain model with energy -55. Colors: green - H monomers, gray - P monomers, red - backbone in side chain models. Calculation and visualization are done using the CPSP-package [22].

3 Constraint-based Local Move Set Definition

To enable folding simulations we need a definition of structural changes that encodes the structural neighborhood of a given lattice protein structure. Here, we follow the idea of k-local moves, that confine the difference between the initial and the neighbored structure to a consecutive interval of maximal length of k. Therefore, we define the k-neighborhood $\mathcal{N}_k(C)$ of a given structure C as:

$$\mathcal{N}_k(C) = \left\{\text{valid structures } C' \mid \exists_{1 \leq s \leq n} : \forall_{j \notin [s,\ldots,(s+k-1)]} : C_j = C'_j\right\} \tag{1}$$

In order to enumerate all valid structural neighbors $C' \in \mathcal{N}_k(C)$ of a given lattice protein C, we have to enumerate the neighbors for all possible interval lengths $1 \leq k' \leq k$ and interval starts $1 \leq s \leq (n - k' + 1)$. Since we want to calculate each neighboring structure only once, we have to enhance the k-local move definition to *strict k-local moves*. Here, we enforce in addition that both ends (C'_s and C'_{s+k-1}) of the successive interval of length k are changed, i.e. a strict k-local move does not cover a k'-local move with $k' < k$, as a normal k-local move in accordance with Eq. 1 does. This ensures a unique enumeration of structural neighbors for an increasing k'.

In the following, we will introduce the Constraint Satisfaction Problems (CSP) that describe all valid structural neighbors $C' \in \mathcal{N}_k(C)$ of a given lattice protein C according to strict k-local moves in a lattice L. A CSP is given by $(\mathcal{X}, \mathcal{D}, \mathcal{C})$, where we denote the set of variables $\mathcal{X}$, their domains $\mathcal{D}$, and a set of constraints $\mathcal{C}$. A solution of a CSP is an assignment $a_i \in \mathcal{D}(X_i)$ for each variable that satisfies all constraints in $\mathcal{C}$. To simplify the presentation, we utilize a binary neighboring constraint $\mathrm{neigh}(X, Y)$ that ensures $\forall_{d_x \in \mathcal{D}(X)} : \exists_{d_y \in \mathcal{D}(Y)} : (d_x \overset{L}{\sim} d_y)$ and vice versa. Furthermore, we use the global $\mathrm{all\text{-}different}$ constraint by Régin [27] to enforce pairwise differences within a set of variables.

3.1 CSP for Backbone-only Models

Given a valid backbone-only lattice protein structure C of length n, a move interval length $k \leq n$, and the start of the interval $1 \leq s \leq (n - k + 1)$. We define k variables X_i, one for each position of the interval, with $\mathcal{D}(X_i) = L \setminus \{C_1, \ldots, C_{s-1}, C_{s+k}, \ldots, C_n\}$. These

variables have to form a valid structure, therefore we post $\text{all-different}(X_1, \ldots, X_k)$ and $\forall_{1 \leq i < k} : \text{neigh}(X_i, X_{i+1})$. Since we describe a substructure, it has to be connected to the interval borders: if $s > 1 : \text{neigh}(X_1, C_{s-1})$ and if $(s + k - 1) < n : \text{neigh}(X_k, C_{s+k})$. Finally, we enforce that both ends of the interval are different from the old placement, i.e. $X_1 \neq C_i$ and $X_k \neq C_{i+k-1}$, to enumerate strict k-local move neighbors only.

The presented CSP is similar to the work of Dotu *et al.* [9], but in contrast ensures the uniqueness of each move. Thus, each neighbored structure is available only via a single interval. This is of high importance to enable a non-redundant enumeration of a structure's neighborhood in the fold space to access its energy landscape [21].

3.2 CSP for Models Including Side Chains

Given a valid side chain lattice protein structure $(C^{\text{b}}, C^{\text{s}})$ of length n, a move interval length $k \leq n$, and the start of the interval $1 \leq i \leq (n - k + 1)$. We define k variables X_i^{b} and X_i^{s}, two for each position of the interval, with $\mathcal{D}(X_i^{\text{b}}) = \mathcal{D}(X_i^{\text{s}}) = L \setminus \{C_1^{\text{b}}, \ldots, C_{s-1}^{\text{b}}, C_{s+k}^{\text{b}}, \ldots, C_n^{\text{b}}, C_1^{\text{s}}, \ldots, C_{s-1}^{\text{s}}, C_{s+k}^{\text{s}}, \ldots, C_n^{\text{s}}\}$. To ensure a valid structure, we enforce $\text{all-different}(X_1^{\text{b}}, \ldots, X_k^{\text{b}}, X_1^{\text{s}}, \ldots, X_k^{\text{s}})$, $\forall_{1 \leq i < k} : \text{neigh}(X_i^{\text{b}}, X_{i+1}^{\text{b}})$, and $\forall_{1 \leq i \leq k} : \text{neigh}(X_i^{\text{b}}, X_i^{\text{s}})$. Since we describe a substructure, it has to be connected to the interval borders: if $s > 1 : \text{neigh}(X_1^{\text{b}}, C_{s-1}^{\text{b}})$ and if $(s + k - 1) < n : \text{neigh}(X_k^{\text{b}}, C_{s+k}^{\text{b}})$. Finally, we warrant the strictness of the k-local moves and enforce that both ends of the interval differ from the old backbone or side chain placement, i.e. $(X_1^{\text{b}} \neq C_i^{\text{b}} \vee X_1^{\text{s}} \neq C_i^{\text{s}})$ and $(X_k^{\text{b}} \neq C_{i+k-1}^{\text{b}} \vee X_k^{\text{s}} \neq C_{i+k-1}^{\text{s}})$.

4 Application

In the following, we applied the introduced move set to folding simulations of side chain lattice protein models in the 3D face-centered-cubic (FCC) lattice. In the FCC lattice, two lattice points l_1 and l_2 are neighbored, if and only if
$(l_1 - l_2) \in \{\pm(1,1,0), \pm(1,0,1), \pm(0,1,1), \pm(1,-1,0), \pm(1,0,-1), \pm(0,1,-1)\}$.
Thus, each point of the FCC lattice has 12 neighbored positions. k-local moves are known to be non-ergodic for backbone-only models [16] depending on k, the used lattice, and the protein length. We expect the same for models including side chains, but using the FCC and an intermediate k should shift the problem to long chain lengths. Thus, we apply 3-local moves, i.e. with a maximal interval length $k = 3$ such that up to 6 monomers are moved (2 per amino acid). The implementation is based on Gecode [11]. To evaluate the structural difference between two structures $(C^{\text{b}}, C^{\text{s}})$ and $(\hat{C}^{\text{b}}, \hat{C}^{\text{s}})$ we calculate the distance and coordinate root mean square deviation (dRMSD and cRMSD) as given by Eq. 2 and 3, respectively. The needed superpositioning utilizes Kabsch's algorithm [12]. We apply the contact based energy function E^{s} that evaluates (only) side chain monomer contacts using the e^{20} contact energy potentials from Sec. 2 similar to the backbone-only studies in [4,29]. In the following, we use C as an abbreviation for $(C^{\text{b}}, C^{\text{s}})$.

$$\text{dRMSD} : \sqrt{\frac{\sum_{i<j}(|C_i^{\text{b}} - C_j^{\text{b}}| - |\hat{C}^{\text{b}}{}_i - \hat{C}^{\text{b}}{}_j|)^2 + (|C_i^{\text{s}} - C_j^{\text{s}}| - |\hat{C}^{\text{s}}{}_i - \hat{C}^{\text{s}}{}_j|)^2 + \sum_i (|C_i^{\text{b}} - C_i^{\text{s}}| - |\hat{C}^{\text{b}}{}_i - \hat{C}^{\text{s}}{}_i|)^2}{n^2}} \tag{2}$$

$$\text{cRMSD} : \sqrt{\frac{\sum_i (|C_i^{\text{b}} - \hat{C}^{\text{b}}{}_i|)^2 + (|C_i^{\text{s}} - \hat{C}^{\text{s}}{}_i|)^2}{2 \cdot n}} \tag{3}$$

PDB ID - chain	Sequences (original and HP transform)
1BAZ-A	SKMPQVNLRWPREVLDLVRKVAEENGRSVNSEIYQRVMESFKKEGRIGA
	PPHPPHPHPHPPPHHPHHPPHPPPPPPPHPPPHHPPHHPPHPPPPPHPP
1J8E-A	GSHSCSSTQFKCNSGRCIPEHWTCDGDNDCGDYSDETHANCTNQ
	PPPPHPPPPHPHPPPPHHPPPHPHPPPPPHPPHPPPPPPPHPPP
1RH6-A	MYLTLQEWNARQRRPRSLETVRRWVRESRIFPPPVKDGREYLFHESAVKVDLNRP
	HHHPHPPHPPPPPPPPHPPHPPHHPPPPHHPPPHPPPPPHHHPPPPHPHPHPPP
1Z0J-B	IEEELLLQQIDNIKAYIFDAKQCGRLDEVEVLTENLRELKHTLAKQKGGTD
	HPPPHHHPPHPPHPPHHHPPPPHPPHPPHPHHPPPHPPHPPPHPPPPPPPP
2DS5-A	GKLLYCSFCGKSQHEVRKLIAGPSVYICDECVDLCNDIIREEI
	PPHHHHPHHPPPPPPHPPHHPPPPHHHHPPHHPHHPPHHPPPH
2EQ7-C	LAMPAAERLMQEKGVSPAEVQGTGLGGRILKEDVMRH
	HPHPPPPPHHPPPPHPPPPHPPPPHPPPHHPPPHHPP
2HBA-A	MKVIFLKDVKGMGKKGEIKNVADGYANNFLFKQGLAIEATPANLKALEAQKQ
	HPHHHHPPHPPHPPPPPHPPHPPPHPPPHHHPPPHPHPPPPPPHPPHPPPPP

PDB ID - chain	n	C_{fit} to C_{PDB} dRMSD	C_{fit} to C_{PDB} cRMSD	$E(C_{fit})$	$E(g(C_{fit}))$	$g(C_{fit})$ to C_{fit} dRMSD	$g(C_{fit})$ to C_{fit} cRMSD
1BAZ-A	49	0.886 $\AA$	1.725 $\AA$	-3.73	-31.51	4.050 $\AA$	6.565 $\AA$
1J8E-A	44	0.928 $\AA$	1.939 $\AA$	-3.54	-30.76	3.865 $\AA$	6.857 $\AA$
1RH6-A	55	0.921 $\AA$	1.791 $\AA$	1.33	-38.17	4.192 $\AA$	8.243 $\AA$
1Z0J-B	51	0.917 $\AA$	2.095 $\AA$	2.05	-35.95	3.185 $\AA$	6.640 $\AA$
2DS5-A	43	0.901 $\AA$	1.750 $\AA$	-4.35	-34.36	4.658 $\AA$	7.755 $\AA$
2EQ7-C	37	0.905 $\AA$	1.813 $\AA$	-3.07	-20.58	2.328 $\AA$	4.751 $\AA$
2HBA-A	52	0.890 $\AA$	1.780 $\AA$	-3.04	-30.62	3.224 $\AA$	6.015 $\AA$

Table 1. Used sequences, their HP transforms, length n, the quality of the fitted lattice protein model, and the according energies.

We derived a protein data from the Pisces web server [30] on June 23rd 2009. Only complete X-ray structures of 2.0$\AA$ resolution or better with an R-value of 0.3 that contain side-chain data were considered. We used a 30% sequence identity cut-off. Since we are applying a simple contact-based energy function we filtered for short globular shaped proteins. Table 1 summarizes the used sequences and their corresponding Protein Data Bank (PDB) identifiers etc.

For each full atom PDB structure C_{PDB}, we derived a lattice protein structure C_{fit} that minimizes the dRMSD to C_{PDB}. This was done using LATFIT from the LATPACK-tools package v1.7.0[4] [19]. Table 1 summarizes the resulting dRMSD and cRMSD values.

Since the applied energy function is still a rough abstraction of the forces that guide the real folding process into C_{PDB}, no energy minimizing folding strategy will find the fitted lattice protein structure C_{fit}. Thus, we map C_{fit} to the according local minimum in the energy landscape. The mapping is done via a steepest decent or *gradient walk*. Starting from a given structure, at each step the neighbored structure with lowest energy is chosen for the next step until no such neighbor exists. Therefore, a gradient walk ends in a local minimum of the energy landscape, which we denote $g(C)$ for a given start structure C.

[4] Freely available at http://www.bioinf.uni-freiburg.de/Software/LatPack/

PDB ID - chain	average values $\langle E(C_{HP})\rangle$	minimal values $\min E(g(C_{HP}))$	$\min E(r(C_{HP}))$
1BAZ-A	-10.67	-33.07	-34.60
1J8E-A	-12.45	-29.33	-32.35
1RH6-A	-13.09	-35.12	-37.59
1Z0J-B	-13.42	-34.71	-37.69
2DS5-A	-6.97	-31.00	-32.53
2EQ7-C	-6.55	-21.64	-25.10
2HBA-A	-11.07	-30.91	-35.56

PDB ID	$g(C_{HP})$ vs. $g(C_{fit})$		$r(C_{HP})$ vs. $g(C_{fit})$	
- chain	dRMSD	cRMSD	dRMSD	cRMSD
1BAZ-A	4.736 Å	8.797 Å	4.762 Å	9.360 Å
1J8E-A	3.384 Å	7.508 Å	3.196 Å	7.052 Å
1RH6-A	4.190 Å	9.645 Å	4.242 Å	10.156 Å
1Z0J-B	5.609 Å	10.166 Å	6.232 Å	11.438 Å
2DS5-A	3.588 Å	8.679 Å	3.425 Å	7.639 Å
2EQ7-C	3.427 Å	7.247 Å	4.177 Å	8.401 Å
2HBA-A	3.832 Å	8.848 Å	4.194 Å	9.075 Å

Table 2. Resulting energies and a structural comparison of the folding results.

The $g(C_{fit})$ structures represent our "true" model to benchmark the following folding scheme. The energies of C_{fit} and $g(C_{fit})$ and their structural differences to each other and to C_{PDB} are given in Table 1.

The folding simulation procedure applied follows the idea of [29]. For each amino acid sequence S, we derive an according HP-sequence S_{HP} using the translation table used in [29]. The derived S_{HP} are given in Table 1. Following the observation of the hydrophobic collapse [1], we calculated HP-optimal structure representatives utilizing the CPSP-approach [3,22,20] and its latest extension HPREP [18]. The resulting HP-optimal structures are named C_{HP}. For each C_{HP} we run gradient walks and evaluated the resulting local minima found. The corresponding energies are listed in Table 2. Furthermore, we performed a structural comparison of the resulting $g(C_{HP})$ structures to our "true" models $g(C_{fit})$ from the fitting. The RMSD values are given in Table 2.

In addition, we executed for each C_{HP} *random descending walks* in order to sample the local minima of the energy landscape accessible from the collapsed starting structures. Here, at each step a random neighbor with lower energy is selected following a uniform distribution until no such neighbor exists. The lowest reached local minimum of all random descending walks starting at C is denoted by $r(C)$. Energy and structural differences are given in Table 2.

5 Discussion

The gradient walks using 3-local moves starting from the fitted structures C_{fit} revealed that the currently applied contact based energy function using the energy potentials e^{20}, originally derived for backbone-only models [4], does not reflect the real forces present for models including side chains. This can be observed when comparing the energies $E(C_{fit})$

to $E(g(C_{fit}))$ (see Table 1). An energy function that results in a smaller difference would be preferable, i.e. it would be a better model for the real forces guiding the folding process to C_{PDB}. In addition we could show, that the derived structures from our simple energy-optimizing folding simulation procedure are still quite dissimilar to the energy-optimized lattice fits of the real structures (see Table 2). We assume this mainly results from the simple energy function as well.

To improve the results, we plan to apply more advanced energy functions, e.g. following [13]. Most important: the energy function has to consider the backbone positioning as well, which is not done by the contact-based energy functions from Sec. 2. Additionally, we want to apply distance based energy potentials that allow for a more realistic energy evaluation. Another direction of ongoing research is to further constrain the allowed structures. Here, we will directly benefit from the CP-based formulation of k-local moves. Since we are formulating a CSP on valid structural neighbors, it is quite easy to post additional structural constraints. For instance, we can enforce a restriction on the allowed relative torsion angles along the protein chain (as e.g. done in [24]), that follows the observation of a limited degree of freedom in nature.

6 Conclusions and Summary

We introduced a CP-based approach to enumerate k-local neighbors of a lattice protein structure in backbone-only and side chain lattice protein models. The generic approach can be applied for any local move length k within arbitrary lattices. Thus, it enables a fast prototyping of new folding simulation schemes or can be easily extended with additional constraints, e.g. restricted torsion angles. The CSP formulation enables the enumeration of the whole k-local move neighborhood $\mathcal{N}_k(C)$ of a given structure C or the calculation of a random neighboring structure $C_r \in \mathcal{N}_k(C)$ when applying a randomized search as possible in Gecode [11]. The application of symmetry breaking search [2] can be used to avoid the enumeration of symmetric structures, increasing the efficiency of folding simulations [10]. We plan the incorporation of the k-local move neighbor enumeration into our C++ energy landscape library (ELL) [21]. This will open an easy interface for folding simulations in arbitrary lattices utilizing any energy function of interest. Furthermore, this will enable full kinetics studies based on the energy landscape topology.

We will utilize the flexibility of the CP-based approach to incorporate additional structural constraints into the neighborhood generation. Following [24,23], it is beneficial to restrict torsion angles along the backbone or to exploit secondary structure information.

Another advantage of the CP-based approach is its extensibility to constraint optimization problems (COP). Currently, we plan to incorporate the energy function as the objective into the CSP, as e.g. done in [6,9]. Thus, by solving a COP while optimizing the energy function, we can directly calculate the lowest energy neighbor of a structure following the framework of Pesant and Gendreau [26], which is needed e.g. for a gradient walk in the energy landscape as done in Sec. 4. Furthermore, this would enable an extension of the work of Zhang *et al.* [31]. They showed (for backbone-only models) that the performance of Monte Carlo folding simulations can be significantly increased using a greedy sequential regrowth of subchains. Thus, we plan to directly apply the sketched COP to calculate the optimal fragments for lattice proteins including side chains. Finally, the presented CP-based move set formulation can be easily extended to any other local move definition of interest.

References

1. Vishwas R. Agashe, M. C. R. Shastry, and Jayant B. Udgaonkar. Initial hydrophobic collapse in the folding of barstar. *Nature*, 377:754–757, 1995.
2. Rolf Backofen and Sebastian Will. Excluding symmetries in constraint-based search. *Constraints*, 7(3):333–349, 2002.
3. Rolf Backofen and Sebastian Will. A constraint-based approach to fast and exact structure prediction in three-dimensional protein models. *J Constraints*, 11(1):5–30, Jan 2006.
4. Marco Berrera, Henriette Molinari, and Federico Fogolari. Amino acid empirical contact energy definitions for fold recognition in the space of contact maps. *BMC Bioinformatics*, 4:8, 2003.
5. M. R. Betancourt. Efficient monte carlo trial moves for polypeptide simulations. *J Chem Phys*, 123(17), 2005.
6. Raffele Cipriano, Alessandro Dal Palù, and Agostino Dovier. A hybrid approach mixing local search and constraint programming applied to the protein structure prediction problem. In *Proc of WCB'08*, 2008.
7. S. Bromberg K. A. Dill. Side-chain entropy and packing in proteins. *Protein Sci*, 3(7):997–1009, 1994.
8. R.I. Dima and D. Thirumalai. Exploring protein aggregation and self-propagation using lattice models: Phase diagram and kinetic. *Prot. Sci.*, 11(5):1036–1049, 2002.
9. Ivan Dotu, Manuel Cebrián, Pascal Van Hentenryck, and Peter Clote. Protein structure prediction with large neighborhood constraint programming search. In *Proc of CP'08*, volume 5202 of *LNCS*, pages 82–96. Springer, 2008.
10. Xiangchao Gan, Leonidas Kapsokalivas, Andreas A. Albrecht, and Kathleen Steinhöfel. A symmetry-free subspace for ab initio protein folding simulations. In *Proc. of BIRD'08*, volume 13 of *CCIS*, pages 128–139. Springer, 2008.
11. Gecode: Generic constraint development environment, 2007. Available as an open-source library from `www.gecode.org`.
12. W. Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica*, A32:922–923, 1976.
13. Sebastian Kmiecik and Andrzej Kolinski. Characterization of protein-folding pathways by reduced-space modeling. *PNAS*, 104(30):12330–12335, 2007.
14. Kit Lau and Ken Dill. A lattice statistical mechanics model of the conformational and sequence spaces of proteins. *Macromolecules*, 22(10):3986–3997, 1989.
15. Neal Lesh, Michael Mitzenmacher, and Sue Whitesides. A complete and effective move set for simplified protein folding. In *Proc of RECOMB'03*, pages 188–195. ACM, 2003.
16. Neal Madras and Alan D. Sokal. Nonergodicity of local, length-conserving Monte Carlo algorithms for the self-avoiding walk. *Journal of Statistical Physics*, 47(3-4):573–595, 1987.
17. Neal Madras and Alan D. Sokal. The pivot algorithm: A highly efficient Monte Carlo method for the self-avoiding walk. *J Stat Phys*, 50(1-2):109–186, 1988.
18. Martin Mann, Rolf Backofen, and Sebastian Will. Equivalence classes of optimal structures in HP protein models including side chains. In *Proc of WCB'09*, 2009.
19. Martin Mann, Daniel Maticzka, Rhodri Saunders, and Rolf Backofen. Classifying protein-like sequences in arbitrary lattice protein models using LatPack. *HFSP Journal*, 2(6):396, 2008.
20. Martin Mann, Cameron Smith, Mohamad Rabbath, Marlien Edwards, Sebastian Will, and Rolf Backofen. CPSP-web-tool : a server for 3D lattice protein studies. *Bioinformatics*, 25(5):676–677, 2009.
21. Martin Mann, Sebastian Will, and Rolf Backofen. The energy landscape library - a platform for generic algorithms. In *Proc of BIRD'07*, volume 217, pages 83–86. OCG, 2007.
22. Martin Mann, Sebastian Will, and Rolf Backofen. CPSP-tools - exact and complete algorithms for high-throughput 3D lattice protein studies. *BMC Bioinformatics*, 9:230, 2008.

23. Alessandro Dal Palu, Agostino Dovier, and Federico Fogolari. Constraint Logic Programming approach to protein structure prediction. *BMC Bioinformatics*, 5(1):186, 2004.
24. Alessandro Dal Palù, Sebastian Will, Rolf Backofen, and Agostino Dovier. Constraint based protein structure prediction exploiting secondary structure information. In *Proc of CILC'04*, pages 16–17, 2004.
25. Britt H. Park and Michael Levitt. The complexity and accuracy of discrete state models of protein structure. *J Mol Biol*, 249:493–507, 1995.
26. Gilles Pesant and Michel Gendreau. A constraint programming framework for local search methods. *Journal of Heuristics*, 5(3):255–279, 1999.
27. Jean-Charles Régin. A filtering algorithm for constraints of difference in CSPs. In *Proc. of 12th National Conference on AI*, pages 362–367, 1994.
28. A. Sali, E. Shakhnovich, and M. Karplus. Kinetics of protein folding. A lattice model study of the requirements for folding to the native state. *J Mol Biol*, 235(5):1614–1636, 1994.
29. Abu Dayem Ullah, Leonidas Kapsokalivas, Martin Mann, and Kathleen Steinhöfel. Protein folding simulation by two-stage optimization. In *Proc. of ISICA'09*, CCIS, Wuhan, China, Oct 2009. Springer. (accepted).
30. G. Wang and Roland L. Dunbrack. Pisces: a protein sequence culling server. *Bioinformatics*, 19(12):1589–91, 2003.
31. J. Zhang, S. C. Kou, and J. S. Liu. Biopolymer structure simulation and optimization via fragment regrowth Monte Carlo. *J Chem Phys*, 126(22):225101, 2007.

Author Index