Guaranteed Diversity & Quality for the Weighted CSP

Manon Ruffini*, Jelena Vucinic†, Simon de Givry‡, George Katsirelos§, Sophie Barbe¶ and Thomas Schiex||

*†‡||*MIAT*, Université de Toulouse
INRA, Auzeville-Tolosane, France
Email: firstname.name@inra.fr

§*MIA-Paris UMR518*
Paris, France
Email: georgios.katsirelos@inra.fr

¶*LISBP*, Université de Toulouse
CNRS, INRA, INSA, Toulouse, France
Email: sophie.barbe@insa-toulouse.fr

Abstract—**In many applications of constraint programming, it is often impossible to capture all the relevant information in one numerical criterion. In this case, it is useful to produce a set of high-quality yet diverse solutions. In this paper, motivated by a Computational Protein Design application, we consider the general problem of producing a diverse set of high-quality solutions of a given Weighted Constraint Satisfaction Problem, with guarantees both on solution quality and diversity. We use weighted automata decomposed in functions of bounded arity, incremental Cost Function Network solving, a simple form of predictive bounding, and compressed representations of distance constraints for improved efficiency. We show that this approach can be successfully applied to a variety of problems that include both Protein Design Problems but also large Bayesian networks represented as CFNs. We also show that our approach has the capacity to enumerate so-called local δ modes and that it does provide improved protein designs.**

Index Terms—**Diversity, Semi-metric, Constraint Programming, WCSP, Cost Function Networks, Local consistency, Automata, Dual Hidden representation, Protein Design**

I. Introduction

Constraint Programming (CP) is a modeling paradigm that is effective to build models of problems where solutions and non-solutions can be separated by a set of properties, or constraints, that solutions must satisfy. When more gradual information is available, capturing costs or uncertainty, two approaches can be followed: Constraint Optimization (COP [1]) reduces optimization to a series of feasibility requests by introducing specific cost variables and functional constraints to represent a criterion that must be sufficiently optimized. The main advantage is that existing CP tools can be directly used. Alternatively, Cost Function Networks (CFNs) directly model a criterion and feasibility constraints as a sum of cost functions, where infinite or intolerable costs capture infeasibility. When the criterion to optimize decomposes naturally in a sum of small functions, CFNs do not require the introduction of additional variables and constraints while providing dedicated powerful processing algorithms [2], [3].

When the optimized criterion is not a perfect representation of the real problem, optimization queries are often insufficient to identify a single universally satisfying assignment. In the CSP (Constraint Satisfaction Problem) or SAT problems, all solutions are equivalent. In COP or CFNs, there are often additional criteria that could not be captured in the model. In these cases, optimal solutions may possibly be not the best "in practice". This can happen if the criterion has been learned from data or when it is an approximate representation of an otherwise highly intractable mathematical model. In these cases, it intuitively makes sense to produce a set of diverse low-cost solutions.

In this paper, we are specifically motivated by the Computational Protein Design (CPD [4]) problem. A protein is a chain of simple molecules called amino acids. All amino acids consist of a constant common core of carbon, nitrogen and oxygen atoms and a variable and flexible side chain, with varying chemical properties. There are 20 natural side chains defining 20 natural amino acids. In a protein, the constant parts of successive amino acids are linked in a chain to form the *backbone* of the protein. In water, most proteins *fold* into a specific 3D shape which is determined by the sequence of its amino acids. The side chain of amino acids are flexible and can be rotated along up to 4 axes relative to the backbone. The 3D shape defined by the backbone and the set of rotations is called a *conformation* of the protein and determines its chemical function. In CPD, we are given a rigid 3D backbone and we need to identify sequences of amino acids that should fold in this target backbone. To solve this problem, it is usual to rely on a pairwise decomposable energy function which, given the nature and orientation of all its side chains, provides an approximation of the energy of the protein. It then becomes possible to identify the nature and conformation of the side chains by minimizing the protein energy (maximizing stability). The energy function being pairwise decomposable and the backbone assumed to be rigid, the protein energy can be described as a pairwise Cost Function Network with one variable for every position in the protein chain, with a domain that describes a discrete ensemble of possible side chain natures, each with a set of statistically preferred orientations (called a rotamer library). An assignment of minimum energy defines an amino-acid sequence that can then be synthesized and tested for proper function. Because proteins govern much of how cells work, in humans, animals, plants, and microbes, newly designed proteins have a large potential for applications in medicine, biofuels, green chemistry…

Funded by the Agence Nationale de la Recherche, ANR-16-CE40-0028.

In CPD, as in many other real problems, the criteria optimized only approximates the real criteria: the actual energy of the protein. This makes the protein design process unreliable: typical workflow includes the expensive production and experimental testing of several proteins. Ideally, this library should therefore be a set of diverse and low energy solutions. The hope is that diversity will improve the likelihood that a working protein is found. The diversity can be defined by the Hamming distance between sequences or can be a "chemical" diversity that can be estimated with existing protein dissimilarity matrices. Because of their important applications, protein sequences can also be subject to patents: in this case, a newly designed sequence must absolutely satisfy a Hamming distance constraint to existing patented sequences.

It has been recently shown that exact algorithms Maintaining *Soft Local Consistencies* (MSLC) during tree-search often outperformed alternative exact approaches for finding optimal CFN assignments [3]. On the rigid backbone CPD problem, this approach outperforms a variety of solvers including Integer Linear and Quadratic Programming solvers, Partial Weighted MaxSAT solvers or COP solvers as well as CPD-dedicated exact algorithms [5]. Recently, MSLC algorithms have even been used to bring to light the limitations of a CPD-dedicated Simulated Annealing implementation [6] and to produce a hyper-stable self-assembling protein [7].

In this paper, we are interested in producing a set of solutions that have low cost and that satisfy inter-solutions distance constraints. Beyond the specific Protein Design problem, diversity appears to be a very usual requirement and many approaches to this problem already exist. However, none fits our needs. After a quick tour of existing approaches, we precisely define the general problem we tackle and prove that it generalizes the closely related problem of producing a set of M δ-modes [8] (or local minima). Because of the previously shown efficiency of MSLC algorithms, we consider enforcing soft local consistency on such distance requirements and show that it is NP-hard, even for the weakest nontrivial soft local consistencies. We represent distance constraints using global (high-order) automaton-based functions and optimize a non-homogeneous decomposed representation by exploiting the distance function semantics. We explore how the dual/hidden representations of Constraint Satisfaction problems can be lifted to CFNs to accelerate filtering. Similarly to the incremental SAT framework, we solve a sequence of WCSPs (Weighted Constraint Satisfaction Problems), relying on a simple but effective predictive bounding approach. We evaluate the algorithm on a set of problems that includes very large Bayesian networks benchmark problems and CPD instances, both encoded as CFNs. Our results show that the protein design process can be improved thanks to diversity. We also compare our approach, when feasible, to a closely related recent algorithm computing δ-modes [9]. All problems and code are available at https://forgemia.inra.fr/thomas.schiex/toulbar2 in the 'tschiex/incremental' branch.

II. Background

We use capital letters to denote variables, lowercase letters for values and bold letters for sets, sequences or tuples.

Definition 1. *A Cost Function Network (CFN) is a pair $(\mathbf{X}, \mathbf{W})$ where $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a set of n variables and $\mathbf{W}$ is a set of cost functions. Each variable $X_i \in \mathbf{X}$ has a finite domain $\mathbf{D}_i$ of values that can be assigned to it. For a set of variables $\mathbf{S} \subseteq \mathbf{X}$, $\mathbf{D_S}$ denotes the Cartesian product of the domains of the variables in $\mathbf{S}$. For a given tuple of values $\mathbf{t}$, $\mathbf{t}[\mathbf{S}]$ denotes the projection of $\mathbf{t}$ over $\mathbf{S}$. A cost function $w_\mathbf{S} \in \mathbf{W}$, with scope $\mathbf{S} \subseteq \mathbf{X}$, is a function $w_\mathbf{S}$ that maps tuples in $\mathbf{D_S}$ to integer costs less than a maximum cost $k \in \mathbb{Z} \cup \{\infty\}$, used for forbidden tuples. We say that a CFN is in Normal Form if all $w_\mathbf{S} \in \mathbf{W}$ with $\mathbf{S} \neq \varnothing$ are non-negative and $\mathbf{W}$ contains one constant empty scope cost function $w_\varnothing$ with arbitrary value in $\mathbb{Z}$.*

A CFN $N = (\mathbf{X}, \mathbf{W})$ defines a joint integer cost function $J_N(\mathbf{X}) = \bigoplus_{w_\mathbf{S} \in \mathbf{W}} w_\mathbf{S}$, where $a \oplus b = \min(k, a+b)$ is the k-bounded addition. In the rest of the paper, we assume that all CFNs are in Normal Form (linear time computable) and in this case, $w_\varnothing$ is a lower bound on $J_N(\mathbf{X})$. A function $w_\mathbf{S}$ that uses only costs 0 and k is a *constraint*. An assignment of $\mathbf{X}$ that has cost strictly less than k is *feasible*. The Weighted Constraint Satisfaction Problem (WCSP) is to find a feasible assignment of $\mathbf{X}$ minimizing $J_N(\mathbf{X})$. The WCSP is decision NP-complete. If $k = 1$, the WCSP is the Constraint Satisfaction Problem (CSP). As described in [5], the CPD problem reduces to a binary WCSP where there is one variable per position to design in the sequence of the considered protein. Each value in a domain contains a pair that describes both the nature of the amino acid used and its conformation (typically a few hundred values per domain). The energy of the designed protein is described by unary and binary cost functions forming an almost complete graph. Typical protein design problems involve less than 100 mutable amino acids or variables.

One of the most successful techniques for solving the WCSP consists of maintaining Soft Local Consistencies (MSLC) during Branch and Bound Search [2], [3], [10]. These algorithms explore a (usually) binary tree where a CFN is associated with every node. The root is the CFN N to solve. To branch, an unassigned variable X_i is selected as well as a value r in its domain. On the left branch, X_i is assigned to r and on the right branch, r is removed from the possible values for X_i. To avoid exploring the whole tree, a lower bound lb on the cost of the best complete assignment below the current node is computed. If lb is larger than or equal to the cost k, the branch is pruned with no loss of information. If a leaf is reached, it is a feasible assignment and the upper bound k is updated to its cost, enforcing the fact that a better solution is sought. In practice, there are two crucial components in this combination: the branching variable selection, that should first assign variables that are likely to lead to strong pruning, and the use of $w_\varnothing$ as a lower bound, following the application of generalization of Arc Consistency to CFNs, called Soft Local

Consistencies [2].

Soft Local Consistencies (SLCs) are properties that can be enforced on a CFN $(\mathbf{X}, \mathbf{W})$, transforming it into an equivalent CFN $(\mathbf{X}, \mathbf{W}')$ that satisfies the enforced SLC property, has a non-decreased lower bound $w_{\varnothing}$ and possibly smaller domains: any value which can be easily shown to lead to assignments of cost k can be safely pruned. There is a variety of SLCs providing increasingly tight lower bounds: Node, $\varnothing$-inverse, Directional, Arc, Existential, Virtual and Optimal consistencies [2], [11]. Except for the first three, they all reduce to Arc Consistency [12] when $k = 1$ (CSP).

To measure the distance between solutions, we consider semi-metrics defined by a sum of variable-wise dissimilarities defined in a zero-diagonal symmetric positive matrix D. Given two assignments $\mathbf{t_S}$ and $\mathbf{t'_S}$ of a set of variables $\mathbf{S}$, we assume that $d(\mathbf{t_S}, \mathbf{t'_S}) = \sum_{X_i \in \mathbf{S}} D(\mathbf{t_S}[X_i], \mathbf{t'_S}[X_i])$. The properties of D guarantee that a semi-metric is defined. The Hamming distance is defined by $H(i,j) = \mathbb{1}(i \neq j)$. In biology, protein sequences are often compared using dedicated protein similarity matrices, such as the BLOSUM62 matrix. A protein similarity matrix S can be transformed into a dissimilarity matrix by $D(i,j) = \frac{1}{2}(S(i,i) + S(j,j)) - S(i,j)$.

Definition 2. *Given a set* $\mathbf{Z}$ *of solutions, we define its average dissimilarity as* $\bar{d}(\mathbf{Z}) = \frac{1}{|\mathbf{Z}|}\sum_{\mathbf{t} \neq \mathbf{t}' \in \mathbf{Z}} d(\mathbf{t}, \mathbf{t}')$ *and its minimum dissimilarity as* $\check{d}(\mathbf{Z}) = \min_{\mathbf{t} \neq \mathbf{t}' \in \mathbf{Z}} d(\mathbf{t}, \mathbf{t}')$.

Definition 3. *Given two sets of variables* $\mathbf{S}$ *and* $\mathbf{S}'$ *of the same cardinality, a dissimilarity matrix* D *and a diversity lower bound* δ, *we define the high-order cost function* $\textsc{Dist}(\mathbf{S}, \mathbf{S}', D, \delta)$ *which is equal to* 0 *if sign*$(\delta).d(\mathbf{S}, \mathbf{S}') \geq \delta$ *and* k *otherwise.*

Problem 1 (DIVERSESET)**.** *Given a CFN* $N = (\mathbf{X}, \mathbf{W})$, *a dissimilarity matrix* D, *an integer* M *and a dissimilarity threshold* δ, *the problem* $\textsc{DiverseSet}(N, D, M, \delta)$ *consists of producing a set* $\mathbf{Z}$ *of* M *solutions of* N *such that* $\forall \mathbf{t} \neq \mathbf{t}' \in \mathbf{Z}$, $\textsc{Dist}(\mathbf{t}, \mathbf{t}', D, \delta) = 0$ *and* $\sum_{\mathbf{t} \in \mathbf{Z}} J_N(\mathbf{t})$ *is minimum.*

For a CFN N with n variables, solving DIVERSESET requires to simultaneously decide the value of nM variables. It can be solved by making M copies of N with variable sets $\mathbf{X}^1$ to $\mathbf{X}^M$ and adding $\frac{M.(M-1)}{2}$ constraints $\textsc{Dist}(\mathbf{X}^i, \mathbf{X}^j, D, \delta)$ for all $1 \leq i < j \leq M$ (if k is finite, all occurrences of k must also be replaced by $M.(k-1)+1$). Because this problem may be very hard to solve on even tiny problems, we consider a closely related problem.

Problem 2 (DIVERSESEQ)**.** *Given a CFN* $N = (\mathbf{X}, \mathbf{W})$, *a dissimilarity matrix* D, *an integer* M *and a dissimilarity threshold* δ, *the problem* $\textsc{DiverseSeq}(N, D, M, \delta)$ *consists of producing a sequence* $\mathbf{Z}$ *of* M *solutions of* N *such that for any* $1 \leq i \leq M$, $\mathbf{Z}[i]$ *is such that* $\forall j < i, \textsc{Dist}(\mathbf{Z}[i], \mathbf{Z}[j], D, \delta) = 0$ *and* $\mathbf{Z}[i]$ *has minimum cost.*

DIVERSESEQ can be solved greedily by iteratively deciding n variables M times which, given the NP-hardness of WCSP, may be exponentially faster.

Definition 4. *Given a set of solutions* $\mathbf{Z}$, *we define the global cost function* $\textsc{Div}_{\min}(\mathbf{X}, \mathbf{Z}, D, \delta) = \bigoplus_{\mathbf{t} \in \mathbf{Z}} \textsc{Dist}(\mathbf{X}, \mathbf{t}, D, \delta)$.

We repeatedly solve the CFN $(\mathbf{X}, \mathbf{W} \cup \{\textsc{Div}_{\min}(\mathbf{X}, \mathbf{Z}, D, \delta)\})$ starting from $\mathbf{Z} = \varnothing$ and adding the solution found to $\mathbf{Z}$ iteratively until $|\mathbf{Z}| = M$ or no solution exists. Because the CFNs solved are increasingly constrained, the cost of successive solutions must be non-decreasing. If pre-existing (patented) solutions $\mathbf{t}$ exist, they can be taken into account by adding the corresponding $\textsc{Dist}(\mathbf{X}, \mathbf{t}, D, \delta)$ constraints before the first iteration.

III. Relation with Existing Work

In the case of Boolean functions, [13] considers the optimization of M or δ using average or minimum dissimilarity. The authors prove that enforcing Arc Consistency on a constraint requiring sufficient average dissimilarity $\bar{d}$ is polynomial but NP-complete for minimum dissimilarity $\check{d}$ and evaluate an algorithm for incremental production of a set maximizing $\bar{d}$. [14] and [15] later addressed the same problems using global constraints and knowledge compilation techniques. More recently, [16] proposed a COP approach to provide diverse high-quality solutions. Their approach however trades diversity for quality.

The idea of producing diverse solutions has also been explored in the related area of discrete stochastic Graphical Models (such as Markov Random Fields), which are closely related to CFNs. [17] exploits the fact that the Lagrangian relaxation of minimum dissimilarity constraints adds only unary cost functions. However, the duality gap is non zero even for simple dissimilarities and an exact method only available for submodular problems. This was extended in [18] using high-order functions (or potentials) to *approximately* optimize a trade-off between diversity and quality. More recently, [19] addressed the DIVERSESET problem, but using optimization techniques that provide no guarantee.

In the end, we observe that none of these approaches simultaneously provides guarantees on quality and diversity. Closest to our target, [20] considered the problem of incrementally producing the set of the best M δ-modes of the joint distribution $J_N(X)$.

Definition 5 ([9])**.** *A solution* $\mathbf{t}$ *is said to be a* δ*-mode iff there exists no better solution than* $\mathbf{t}$ *in the Hamming ball of radius* δ *centered in* $\mathbf{t}$ *(implying that* $\mathbf{t}$ *is a local minimum).*

In [8], [9], [20], [21], an exact dynamic programming algorithm, combined with a A* heuristic search and tree-decomposition was proposed to exactly solve this problem with the Hamming distance. This algorithm relies however on NP-complete lower bounds and is restricted to a fixed variable order. It however provides a diversity guarantee: indeed, a δ-mode will always be *strictly* more than δ away from another one and will be produced by greedily solving DIVERSESEQ.

Theorem 1. *Given* N, δ, *and* H *the Hamming dissimilarity matrix, for any* δ*-mode* $\mathbf{t}$, *there exists a value* M' *such that the solution of* $\textsc{DiverseSeq}(N, H, M', \delta + 1)$ *contains* $\mathbf{t}$.

Proof. If a δ-mode $\mathbf{t}$ is not in the solution of $\textsc{DiverseSeq}(N, H, M', \delta + 1)$, this must be because it gets forbidden by a $\textsc{Dist}$ constraint. Consider the iteration i which forbids $\mathbf{t}$ for the first time: a solution with a cost lower than the cost of $\mathbf{t}$ was produced (else $\mathbf{t}$ would have been produced instead) but this solution is strictly less than $\delta + 1$ away from $\mathbf{t}$ (since $\mathbf{t}$ gets forbidden). But this contradicts the fact that $\mathbf{t}$ is a δ-mode. □

For a sufficiently large M, the sequence $\mathbf{Z}$ solution of $\textsc{DiverseSeq}(N, H, M, \delta + 1)$ will therefore contain all δ-modes and possibly some extra solutions. Interestingly, it is not difficult to separate modes from non-modes.

Theorem 2. *Any assignment* $\mathbf{t}$ *of a CFN* $N = (\mathbf{X}, \mathbf{W})$ *is a* δ*-mode iff it is an optimal solution of the CFN* $(X, W \cup \{\textsc{Dist}(\mathbf{X}, \mathbf{t}, H, -\delta)\})$*. For bounded* δ*, this problem is in P.*

Proof. The function $\textsc{Dist}(\mathbf{X}, \mathbf{t}, H, -\delta)$ restricts $\mathbf{X}$ to be within δ of $\mathbf{t}$. If $\mathbf{t}$ is an optimal solution of $(X, W \cup \{\textsc{Dist}(\mathbf{X}, \mathbf{t}, H, -\delta\}))$ then there is no better assignment than $\mathbf{t}$ in the δ-radius Hamming ball and $\mathbf{t}$ is a δ-mode. For δ bounded, a CFN with n variables and at most d values in each domain, there is $O((nd)^{\delta})$ tuples within the Hamming ball and the problem of checking if $\mathbf{t}$ is optimal is in P. □

IV. Soft local consistency and $\textsc{Div}_{\min}$

In order to iteratively solve the CFN $(X, W \cup \{\textsc{Div}_{\min}(X, Z_{\min}, D, \delta)\})$ using an MSLC approach, we need to enforce some SLC on $\textsc{Div}_{\min}$. In the CSP case, [13] proved that enforcing arc consistency on $\textsc{Div}_{\min}$ is NP-complete. Since Soft Arc Consistency and stronger variants generalize Arc Consistency (AC) in CSP, we know that enforcing soft arc consistency on $\textsc{Div}_{\min}$ is NP-complete too. However, there are SLCs that are weaker than AC in CSP and one could hope that they could be enforced efficiently on $\textsc{Div}_{\min}$. One of the weakest non-naive soft local consistency is $\varnothing$-inverse consistency [22]. A cost function $w_{\mathbf{S}}$ is said to be $\varnothing$-inverse consistent if its minimum is 0. To enforce $\varnothing$-inverse consistency, one subtracts the minimum of $w_{\mathbf{S}}$ from $w_{\mathbf{S}}$ and adds it to $w_{\varnothing}$. These operations that shift costs between scopes are called *equivalence preserving transformations* (EPT) [2]. They generalize the AC *Revise* operation to CFNs to enforce soft local consistencies.

Theorem 3. $\varnothing$*-inverse consistency is NP-hard to enforce on* $\textsc{Div}_{\min}(\mathbf{X}, \mathbf{Z}, D, \delta)$*.*

Proof. Checking AC on $\textsc{Div}_{\min}$ is NP-complete. By definition of AC, $\textsc{Div}_{\min}$ is AC iff for any $X_i \in \mathbf{X}$, for any $r \in D_i$, there exists an assignment of $\mathbf{X} - \{X_i\}$ that has cost 0 or, equivalently, that if we reduce the domain X_i to $\{r\}$, $\textsc{Div}_{\min}$ is $\varnothing$-inverse consistent. It is therefore possible to reduce AC to a linear number of calls to a $\varnothing$-inverse consistency oracle. □

For this reason, in this paper, we do not consider enforcing Soft Local Consistencies on $\textsc{Div}_{\min}$ directly but instead exploit the fact that $\textsc{Div}_{\min}$ is defined as a combination of $\textsc{Dist}(\mathbf{X}, \mathbf{t}, D, \delta)$ functions on which SLC can be independently enforced in polynomial time.

V. Using automata

For a CSP, given an order on its variables, any constraint is defined by its finite set of authorized tuples. This set defines a regular language which can be encoded into a finite state automaton [23] and AC enforced on the corresponding $\textsc{Regular}$ constraint in time linear in the automaton size. The $\textsc{Div}_{\min}$ and $\textsc{Dist}$ constraints can therefore be encoded as automata. The NP-completeness of $\textsc{Div}_{\min}$ means that there is no compact automaton for $\textsc{Div}_{\min}$ (unless P=NP). For $\textsc{Dist}$, it is well-known that a compact automaton exists [23] and that the $\textsc{Regular}$ constraint can be decomposed in a set of ternary constraints such that enforcing AC on the decomposition is equivalent to enforcing AC on the original high-order constraint [24].

Inside a CFN, equivalence preserving transformations used to enforce SLC can shift costs between the scopes of various cost functions. Therefore, any function initially defining a constraint, such as the $\textsc{Div}_{\min}$ constraints, will quickly be transformed into a function using costs other than just $\{0, k\}$ costs. SLC can still be enforced in such a $\textsc{WeightedRegular}$ global cost function using a *weighted* automaton [25] defining the weighted language of all tuples with their associated cost.

A weighted automaton $\mathcal{A}$ encoding $\textsc{Dist}(\mathbf{X}, \mathbf{t}, D, \delta)$:

- has $(\delta + 1).(n + 1)$ states s_i^d that represent the fact that the first i values of $\mathbf{X}$ have distance d to the first i values of t (or distance $\geq \delta$ for states s_i^{δ}).
- for every value r of X_i, there is a 0-cost transition from s_i^d to $s_{i+1}^{\min(d+D(r,t[i+1]),\delta)}$.
- the starting state is s_0^0 and the accepting state is s_n^{δ}

This weighted automaton contains $O(n.(\delta + 1).d)$ transitions. An assignment $\mathbf{t}'$ of $\mathbf{X}$ is accepted by this automaton iff $d(\mathbf{t}, \mathbf{t}') \geq \delta$.

It has been previously been shown that SLCs can be enforced on $\textsc{WeightedRegular}$ in polytime in the automaton size using min-cost flow or dynamic programming algorithms [25]. As in the CSP case, the $\textsc{WeightedRegular}$ cost function can also be decomposed into a sequence of ternary cost functions [26]. This can be achieved by adding $n + 1$ CFN state variables $Q_i, 0 \leq i \leq n$ and n ternary functions $w^{\mathcal{A}}_{\{Q_i, X_{i+1}, Q_{i+1}\}}$ such that $w^{\mathcal{A}}_{\{Q_i, X_{i+1}, Q_{i+1}\}} = c$ iff there is a transition from state Q_i to Q_{i+1} through value X_{i+1} with cost c in the automaton. Variables Q_0 and Q_n have restricted domains containing respectively only the starting and accepting states.

Contrarily to the CSP case, enforcing an SLC on the decomposition may be weaker than on the high-order function. To preserve the strength of SLCs, the order of the variables used to build the automaton must be consistent with the order used for Directional AC [26]. This condition is however easy to satisfy for the $\textsc{Div}_{\min}$ and $\textsc{Dist}$ constraints as their definitions make them order insensitive: the set of solutions forbidden by a constraint $\textsc{Dist}(\mathbf{X}, \mathbf{t}, D, \delta)$ does not depend on the order on $\mathbf{X}$ and $\textsc{Div}_{\min}$ inherits this property from

DIST. The ordering condition of [26] can therefore always be satisfied by reordering variables in the scope of DIST or $\text{DIV}_{\min}$ constraint as required. This makes decomposition an attractive approach for DIST and $\text{DIV}_{\min}$ encoding.

In the case of the DIST constraint, each state variable has $(\delta+1).(n+1)$ values and the cost table of the ternary w^A function has size $(\delta+1)^2.(n+1)^2.d$ where d is the domain size. $\text{DIV}_{\min}$ instead would require a $(\delta+1)^{2M}.(n+1)^2.d$ table. To make SLC enforcing faster, we exploit the properties of DIST and D to reduce this complexity.

VI. Compressing the Dist encoding

In this section, we show how the encoding of a DIST constraint in a sequence of $n+1$ ternary constraints described in cost tables of size $(\delta+1)^2.(n+1)^2.d$ can be reduced along several lines. For DIST, we know that states s_i^d can only be reached after i transitions and are specific to variable Q_i: the domains of all variables $Q_i, 0 < i < n$ can be restricted to the $\delta+1$ states s_i^d. Furthermore, our semi-metrics being defined by a non-decreasing sum of non-negative elements of D, any state s_i^d can reach the accepting state s_n^δ iff the maximum dissimilarity (denoted md_i) that can be achieved from variable i to variable n is larger than $\delta - d$. All such maximum dissimilarities can be pre-computed in one pass over all variables in $\mathbf{X}$ as $md_1 = 0; md_i = md_{i-1} + max_{r,s \in D_i \times D_{i+1}} D(r,s)$. In the Hamming case, the distance can increase by 1 at most, this is just $n-i$. A symmetric argument holds for the starting state s_0^0. These simplifications reduce the ternary cost tables to $O((\delta+1)^2.d)$.

A. Dual and hidden representations for DIST

For a given dissimilarity matrix D, we denote as $\#D$ the number of distinct values that appear in D. If variables have domains of maximum size d and ignoring the useless 0 matrix, we know that $2 \leq \#D \leq 1 + \frac{d.(d-1)}{2}$. However, distance matrices are usually more structured. For Hamming, we have $\#H = 2$ (the BLOSUM62 protein similarity matrix contains 12 different similarity levels). In the Hamming case, this means that a state s_i^d can only reach states s_{i+1}^d or s_{i+1}^{d+1}. To exploit the sparsity of the transition matrix, we need to make it visible. This can be achieved using extended variants of the Dual or Hidden encoding of Constraint Networks [27].

In Constraint Satisfaction, the *dual* representation of a constraint network (a CFN $(\mathbf{X}, \mathbf{W})$ with $k = 1$) is a new network $(\mathbf{X}', \mathbf{W}')$ that contains one variable $X_\mathbf{S}$ for every function $w_\mathbf{S} \in W$ with a domain defined by all tuples $\mathbf{t} \in D^\mathbf{S}$ such that $w_\mathbf{S}(t) \neq k$. Furthermore, for any pair of functions $w_\mathbf{S}, w_{\mathbf{S}'} \in W$ such that $\mathbf{S} \cap \mathbf{S}' \neq \varnothing$, there is a function involving $X_\mathbf{S}$ and $X_{\mathbf{S}'}$ that maps all pairs of tuples in the domains $X_\mathbf{S}$ and $X_{\mathbf{S}'}$ to 0 if $\mathbf{t}[\mathbf{S} \cap \mathbf{S}'] = \mathbf{t}'[\mathbf{S} \cap \mathbf{S}']$ and k otherwise.

The *hidden* representation of $(\mathbf{X}, \mathbf{W})$ is a CFN $(\mathbf{X}'', \mathbf{W}'')$ that contains all the variables in $\mathbf{X}$ plus the variables $X_\mathbf{S}$ from the dual network. For any dual variable $X_\mathbf{S}$, and every variable $X_i \in \mathbf{S}$, $\mathbf{W}''$ contains a function involving X_i and $X_\mathbf{S}$ that maps a pair $(a, \mathbf{t})$ to 0 if $\mathbf{t}[\{X_i\}] = a$ and k otherwise.

These transformations are known to preserve the set of solutions and their costs [27], [28]. Instead of applying these transformations on all the functions of a CFN, we consider the idea of applying suitable transformations to the reduced $w^A_{Q_i,X_i,Q_{i+1}}$ functions only.

The dual variable of $w^A_{Q_i,X_i,Q_{i+1}}$ is a variable X_i^A that contains all pairs (s, s') of $Q_i \times Q_{i+1}$ such that there is a transition from s to s'. For Hamming case, this variable has at most $2\delta+1$ values. It is connected to variable X_i by a pairwise function that maps a pair $((s, s'), a)$ from the domain of X_i^A and X_i to 0 iff there is a transition from s to s' with label a and k otherwise. It contains $O(d.\delta)$ pairs only.

In this new dual representation, for every pair of dual variables X_{i-1}^A and X_i^A, we add a function involving these two variables that maps a pair $((s_{i-1}, s'_{i-1}), (s_i, s'_i))$ to 0 iff $s'_{i-1} = s_i$ or k otherwise. In the worst case, this function has size $O(\#D^2.\delta^2)$ and $O(\delta^2)$ in the Hamming case. Only n extra variables are required.

In the new hidden representation, we keep variables Q_i and create two pairwise functions involving each Q_i and respectively X_i^A and X_{i-1}^A that map a pair $(s'', (s, s'))$ to 0 iff $s'' = s$ for the function connecting Q_i to X_i^A (respectively $s'' = s'$ for the function connecting Q_i to X_{i-1}^A). In the worst case, these functions have size $O(\#D.\delta^2)$ and $O(\delta^2)$ in the Hamming case.

These dedicated dual and hidden representations require the description of $O(\delta.d + \#D^2.\delta^2)$ and $O(\delta.d + \#D.\delta^2)$ tuples respectively ($O(\delta.d + \delta^2)$ in the Hamming case) instead of the $O(d.\delta^2)$ tuples in $w^A_{Q_i,X_i,Q_{i+1}}$.

VII. Greedy DiverseSeq

To tackle the $\text{DIVERSESEQ}(N, D, M, \delta)$ problem, one can use the following greedy approach: starting from the CFN N, we solve it using a MSLC Branch and Bound algorithm. If a solution $\mathbf{t}$ is found, it is added to the ongoing solution sequence $\mathbf{Z}$. If M solutions have been produced, we stop there. Otherwise, a $\text{DIST}(\mathbf{X}, \mathbf{t}, D, \delta)$ constraint is added to the previously solved problem and we loop and solve the problem again. If no solution exists, the sequence $\mathbf{Z}$ can provably not be extended to length M and the problem has no solution (but a shorter sequence has been produced).

We have improved this basic schema in three different ways:

- since the problems solved are increasingly constrained, all the equivalence preserving transformations and pruning that have been applied to enforce Soft Local Consistencies at iteration $i-1$ are still valid in the following iterations. Instead of restarting from a problem $N = (\mathbf{X}, \mathbf{W} \cup_{1 \leq j < i} \{\text{DIST}(\mathbf{X}, \mathbf{Z}[j], D, \delta\})$, we reuse the problem solved at iteration $i-1$ after it has been made locally consistent, add the $\text{DIST}(\mathbf{X}, \mathbf{Z}[i-1], D, \delta)$ constraint and reinforce Soft Local Consistency. Similarly to incremental SAT solvers, adaptive variable ordering heuristics that have been trained at iteration $i-1$ are reused at iteration i.
- since the problems solved are increasingly constrained, we know that the optimal cost oc^i obtained at iteration

i cannot have a lower cost than the optimum cost oc^{i-1} reported at iteration $i-1$. When large plateaus are present in the energy landscape, this allows stopping the search as soon as a solution of cost oc^{i-1} is reached, avoiding a useless repeated proof of optimality.

- even if there are no plateaus in the energy landscape, there may be large regions with similar variations in energy. In this case, the difference in energy between oc^{i-1} and oc^{i} will remain similar for several iterations. Let $\Delta_i^h = \max_{\max(2,i-h)\leq j<i}(oc^j - oc^{j-1})$ be the maximum variation observed in the last h iterations (we used $h = 5$). At iteration i, we can first solve the problem with a temporary upper bound $k' = \min(k, oc_{i-1}+2.\Delta_i^h)$ that should preserve a solution. If $k' < k$, this will lead to increased determinism, additional pruning and possibly exponential savings. Otherwise, if no solution is found, the problem is solved again with the original upper bound k. We call this *predictive bounding*.

Each of these three improvements has the capacity to offer exponential time savings and all are used in the following experiments (unless mentioned otherwise).

VIII. Experiments

We implemented the iterative approach described above in its direct (ternary) decomposition as well as in the dedicated hidden and dual versions for Hamming $\textsc{Div}_{\min}$ decomposed in a conjunction of $\textsc{Dist}$ functions above the CFN open source C++ solver toulbar2 [3]. Toulbar2 implements a variety of preprocessing algorithms dedicated to exact optimization. We had to deactivate all preprocessing algorithms that do not preserve suboptimal solutions: variable elimination, dead-end-elimination and variable merging were all deactivated at the root node. We chose to enforce the strong Virtual Arc Consistency [2] on the first problem solved. The computational cost of VAC, in $O(\frac{e.d^2.k}{\varepsilon})$, where ε is the smallest representable cost in the fixed decimal point representation used, is important but is amortized over the M resolutions. During tree search, the default EDAC local consistency was used. All the experiments were performed on one core of a Xeon E5-2680 CPU at 2.50GHz with $\varepsilon = 10^{-6}$. The source code and benchmark problems are available on the GitLab repository https://forgemia.inra.fr/thomas.schiex/toulbar2, in the 'tschiex/incremental' branch.

A. Bayesian networks

To show that our algorithm is not restricted to CPD problems and to compare it with the related δ-mode algorithm of [9], we first experimented on discrete Bayesian networks from http://www.bnlearn.com/bnrepository in the medium, large, very large, and massive categories. We used all the complete networks, with no change. Solving the Maximum Probability Explanation (MPE) problem on a Bayesian net (BN) reduces easily to the WCSP: the probability in a BN is the product of conditional probabilities: minimizing the sum of the $-\log$ of probabilities is equivalent to maximizing the product.

			dual			**hidden**	**3ary**
name	n	d	4-d	3-m	CPU	4-d	4-d
alarm	37	4	147	28	0.38	128	141
andes	223	2	102	8	4.96	87	110
barley	48	67	126	92	2.74	125	113
child	20	6	172	8	1.65	153	154
diabetes	413	21	0	0	TO	0	0
hailfinder	56	11	109	24	0.65	92	102
hepar2	70	4	96	1	TO	87	93
insurance	27	5	186	30	0.40	164	153
link	724	4	117	117	6.69	113	93
mildew	35	100	107	25	7.71	100	97
munin	1041	21	16	2	TO	16	14
pathfinder	109	63	118	15	2.46	90	124
pigs	441	3	162	162	6.28	134	138
water	32	4	183	17	0.48	161	174
win95	76	2	132	21	0.93	115	119

TABLE I: For each network, we give the network name, number of variables, and maximum domain size. Then for the dual encoding, we report the number of 4-diverse solutions found in 5 minutes, the number of 3-modes among these, and the CPU-time taken to produce four 3-modes, as in [9]. We then give the number of 4-diverse solutions produced in the same amount of time using the hidden and ternary encoding.

Table I below reports the number of variables n, the maximum domain size d and the number of 4-diverse solutions found within a maximum of 300 CPU seconds per instance for each representation. Even though this is not our main target, in the dual (fastest) case, we also computed the number of 4-diverse solutions that are 3-modes for comparison with [9]. As shown in Th. 2, checking if an assignment $\mathbf{t}$ is a δ-mode of CFN $(\mathbf{X}, \mathbf{W})$ can be checked in polytime by solving $(\mathbf{X}, \mathbf{W} \cup \{\textsc{Dist}(\mathbf{X}, \mathbf{t}, D, -\delta)\}$. We report the total CPU-time taken to find four 3-modes. These times include both the time to find all 4-diverse solutions and to filter out those that are not 3-modes (the filtering time was always small compared to the time needed to produce the 4-diverse solutions: checking all 4-diverse solutions of all problems took less than 6 minutes overall). For the two problems (child and alarm) that we share with the benchmarks used in [9], we see very similar CPU-times. We could not compare our approach on the 3 remaining problems used by [9]. They were learned from data and the authors could not provide us with the estimated parameters (these problems are small: the largest has 45 variables and Boolean domains ; it however took $380s$ to the algorithm of [9] to provide four 3-modes on it). The energy landscape of each problem has a strong influence on the fraction of $(\delta + 1)$-diverse solutions that are δ-modes: on the *link* and *pigs* problems, which present a large plateau at optimum, our approach is very fast. Instead, the *hepar2* instance shows a very large nonflat basin around the optimum, which offers plenty of 4-diverse solutions but just one local minimum.

We also compared our greedy $\textsc{DiverseSeq}$-solving approach to a globally optimal approach which solves the harder $\textsc{DiverseSet}$ problem. As described in Section II, we made $M = 4$ copies of the *child* medium-size Bayesian network. We decomposed each $\textsc{Dist}(\mathbf{X}^i, \mathbf{X}^j, H, \delta)$ constraint on copy variable sets $\mathbf{X}^i$ and $\mathbf{X}^j$ in a sequence of ternary cost functions using extra Boolean variable set $\mathbf{E}^{i,j}$, with $\forall 1 \leq l \leq n, \mathbf{E}^{i,j}[l] = \mathbb{1}(\mathbf{X}^i[l] \neq \mathbf{X}^j[l])$ supplementary disequality constraints. The resulting DIVERSESET child problem has 320 variables. We solved it with toulbar2 version 1.0, varying δ from 1 to 15. It took more than 23 hours (resp. 3 hours) to solve the $\delta = 15$ (resp. $\delta = 14$) DIVERSESET problem. Instead, solving DIVERSESEQ took 0.28 second for $\delta = 15$ and produced a sequence of fifteen solutions in 1.1 seconds. This efficiency comes with little side-effects in terms of cost: Figure 1 shows the total cost $\sum_{\mathbf{t} \in \mathbf{Z}} J_N(\mathbf{t})$ for the two approaches. A solution to the DIVERSESEQ problem can still be a solution of the DIVERSESET problem for large Hamming distances (up to $\delta = 11$) and remains less than 12.9% above the optimum for $\delta = 15$.

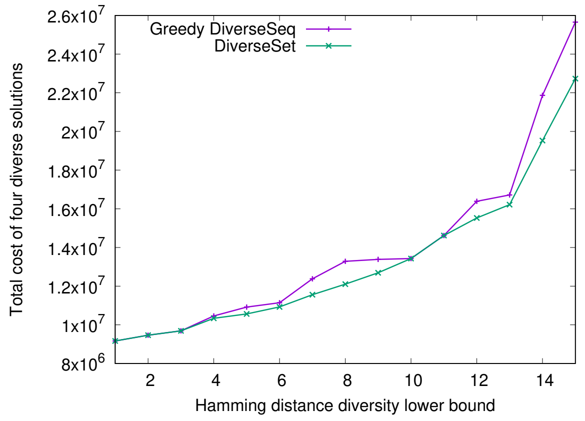

Fig. 1: Comparison between greedy DIVERSESEQ and DIVERSESET on the *child* Bayesian network.

B. Computational Protein Design

Following our initial and main motivation for protein design, we extracted a set of 20 prepared protein backbones for full redesign from the benchmark set built by [6]. We selected the 20 proteins that had required the least CPU-time to solve in this paper, as indicated in Column S of the Excel sheet provided in the supplementary information of the paper) (protein 1aho, 1b9w, 1f94, 1hyp, 1uln, 1uoy, 1yzm, 2cg7, 2erw, 2fht, 2fjz, 2gkt, 2pne, 2pst, 2qt4, 3ca7, 3i8z, 3rdy, 3vdj and 4pti). We used the provided scripts with the Rosetta `ref2015` energy function [29] to build the CFN model for each backbone. The number of variables in these problems ranges from 46 to 109 with maximum domain sizes from 329 to 414 values. For each problem, we generated sets of $M = 10$ diverse solutions with small ($\delta = 1$) and large ($\delta \in \{7, 8, 9, 10\}$) diversity requirements. For $\delta = 1$, the set of solutions produced is just the set of the 10 best sequences.

In terms of CPU, the maximum CPU-time spent on one problem was 32 minutes when predictive bounding was not used. This maximum CPU-time was reduced to 17 minutes when predictive bounding was activated, with an average time of 201s per problem. This shows that predictive bounding provides a simple and efficient boost, and that real CPD instances can be solved in a reasonable time, even when relatively large diversity requirements are required.

δ	1	7	8	9	10
nsr	45.5	47.0	46.6	46.9	46.7
p-value		0.004	0.007	0.008	0.016
nssr	60.1	61.1	61.5	61.9	61.3
p-value		0.015	0.003	0.002	0.021

TABLE II: For each δ in $\{1, 7, 8, 9, 10\}$ we give the average over all proteins of the best nsr (line 2) and nssr (line 4) among the 10 δ-diverse sequences produced. Lines 3 and 5 give the p-values, testing for improvement of nsr (resp nssr) for each δ compared to $\delta = 1$.

In the case of CPD problems using real protein backbones obtained by X-ray crystallography, it is possible to measure the improvements that diversity brings. Traditional metrics to evaluate protein design protocols are the so-called "native sequence recovery" (nsr) and "native similarity sequence recovery" (nssr) which measure how much the redesigned protein resembles the natural protein in terms of sequence identity (nsr: percentage of positions with the same amino acid) or similarity (nssr: percentage of positions with a positive score in the BLOSUM62 similarity matrix). If solution diversity helps, the maximum nsr/nssr over the 10 sequences should improve when δ is large compared to a small δ, as long as the cost (which represents energy in this case) remains close to the optimum. Even with $\delta = 10$, the maximum difference in energy we observed with the global minimum energy never exceeded 2.7 kcal/mol (with an average of 1.21 kcal/mol), which is reasonably small. We therefore compared, for each protein, the best nsr and nssr that could be obtained with $\delta \in \{7, 8, 9, 10\}$ to the best obtained with $\delta = 1$. As the Table II shows, we observe a systematic increase in the average best nsr and nssr for $\delta > 1$ compared to $\delta = 1$ (p-values for a unilateral Wilcoxon signed ranked test comparing the sample of 20 best nsr/nssr for each δ compared to $\delta = 1$) showing that large diversity requirements do lead to improved results.

IX. CONCLUSION

Producing a sequence of diverse solutions is a very usual requirement when an approximate or learned model is used for optimal decoding. In this paper, we show that using an incremental CFN approach using diversity constraints represented as weighted automata that are densely encoded in a dedicated dual encoding together with predictive bounding, it is possible to produce sequences of solutions that satisfy guarantees on diversity on large Bayesian networks (in the "massive" category of the BN repository) as well as for large Computational Protein Design instances. This guarantee is obtained on dense problems with non-submodular functions while also guaranteeing that each new solution produced is the best given the previously identified solutions.

We also showed that the stream of diverse solutions that our algorithm produces can be filtered and each solution efficiently identified as being a δ-mode or not. Compared to the results of [9], our approach looks computationally at least as efficient, while providing a much more extensive report on the shape of the energy landscape around the optimum.

On real protein design problems, we observe that sufficiently large diversity requirements do improve the quality of sequence libraries when native proteins are fully redesigned.

Although guaranteed diversity is necessary for the context of *e.g.* legal requirements, in the context of optimizing an approximate or learned function, the requirement for an optimal cost solution may be considered as exaggerated. However, given that even computationally expensive approaches with asymptotic convergence results such as Simulated Annealing may fail with unbounded error [6], some finite time guarantee remains desirable. The requirement for optimality that we have used in our experiments can be trivially relaxed to a relative or absolute approximation guarantee using artificially tightened pruning rules as originally proposed in [30]. This is already implemented in the toulbar2 solver (using the `-agap` flag).

There are two directions that could extend our work. The first direction is brought to light by the *diabetes* problem for which even the first element of DIVERSESEQ could not be produced in five minutes of CPU-time. This problem has tree-width less than 4 and can be solved by toulbar2 in less than 40 seconds if one exploits a min-fill tree decomposition during Branch and Bound search [10], [31], [32]. It would be therefore desirable to show that the decomposed ternary or binary functions we use for encoding DIST can be arranged in such a way that tree-width can be preserved or not exaggeratedly increased. The second more obvious direction would be to identify a relaxed version of the NP-hard $\text{DIV}_{\min}$ constraint that would produce tighter bounds thanks to SLCs. Despite non-negligible efforts in this direction using relaxed Multi-Valued Decision Diagrams [33], this has eluded us up to now.

REFERENCES

[1] J.-C. Régin, T. Petit, C. Bessiere, and J.-F. Puget, "An original constraint based approach for solving over constrained problems," in *International Conference on Principles and Practice of Constraint Programming*. Springer, 2000, pp. 543–548.

[2] M. Cooper, S. de Givry, M. Sanchez, T. Schiex, M. Zytnicki, and T. Werner, "Soft arc consistency revisited," *Artificial Intelligence Journal*, vol. 174, pp. 449–478, 2010.

[3] B. Hurley, B. OSullivan, D. Allouche, G. Katsirelos, T. Schiex, M. Zytnicki, and S. de Givry, "Multi-language evaluation of exact solvers in graphical model discrete optimization," *Constraints*, pp. 1–22, 2016.

[4] S. Traoré, D. Allouche, I. André, S. De Givry, G. Katsirelos, T. Schiex, and S. Barbe, "A new framework for computational protein design through cost function network optimization," *Bioinformatics*, vol. 29, no. 17, pp. 2129–2136, 2013.

[5] D. Allouche, I. André, S. Barbe, J. Davies, S. De Givry, G. Katsirelos, B. O'Sullivan, S. Prestwich, T. Schiex, and S. Traoré, "Computational protein design as an optimization problem," *Artificial Intelligence*, vol. 212, pp. 59–79, 2014.

[6] D. Simoncini, D. Allouche, S. de Givry, C. Delmas, S. Barbe, and T. Schiex, "Guaranteed discrete energy optimization on large protein design problems," *Journal of chemical theory and computation*, vol. 11, no. 12, pp. 5980–5989, 2015.

[7] H. Noguchi, C. Addy, D. Simoncini, S. Wouters, B. Mylemans, L. Van Meervelt, T. Schiex, K. Y. Zhang, J. Tame, and A. Voet, "Computational design of symmetrical eight-bladed β-propeller proteins," *IUCrJ*, vol. 6, no. 1, 2019.

[8] C. Chen, H. Liu, D. Metaxas, and T. Zhao, "Mode estimation for high dimensional discrete tree graphical models," in *Proceedings of Advances in neural information processing systems*, 2014, pp. 1323–1331.

[9] C. Chen, C. Yuan, Z. Ye, and C. Chen, "Solving m-modes in loopy graphs using tree decompositions," in *Proc. of the International Conference on Probabilistic Graphical Models*, 2018, pp. 145–156.

[10] D. Allouche, S. De Givry, G. Katsirelos, T. Schiex, and M. Zytnicki, "Anytime hybrid best-first search with tree decomposition for weighted csp," in *Proc. of Principles and Practice of Constraint Programming (CP'15)*. Springer, 2015, pp. 12–29.

[11] T. Schiex, "Arc consistency for soft constraints," in *Principles and Practice of Constraint Programming - CP 2000*, ser. LNCS, vol. 1894, Singapore, Sep. 2000, pp. 411–424.

[12] F. Rossi, P. van Beek, and T. Walsh, Eds., *Handbook of Constraint Programming*. Elsevier, 2006.

[13] E. Hebrard, B. Hnich, B. O'Sullivan, and T. Walsh, "Finding diverse and similar solutions in constraint programming," in *Proceedings of AAAI 2005*, vol. 5, 2005, pp. 372–377.

[14] E. Hebrard, B. O'Sullivan, and T. Walsh, "Distance constraints in constraint satisfaction." in *IJCAI*, vol. 2007, 2007, pp. 106–111.

[15] T. Hadžić, A. Holland, and B. OSullivan, "Reasoning about optimal collections of solutions," in *International Conference on Principles and Practice of Constraint Programming*. Springer, 2009, pp. 409–423.

[16] T. Petit and A. C. Trapp, "Finding diverse solutions of high quality to constraint optimization problems," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[17] D. Batra, P. Yadollahpour, A. Guzman-Rivera, and G. Shakhnarovich, "Diverse m-best solutions in markov random fields," in *European Conference on Computer Vision*. Springer, 2012, pp. 1–16.

[18] A. Prasad, S. Jegelka, and D. Batra, "Submodular meets structured: Finding diverse subsets in exponentially-large structured item sets," in *Advances in Neural Information Processing Systems*, 2014, pp. 2645–2653.

[19] A. Kirillov, B. Savchynskyy, D. Schlesinger, D. Vetrov, and C. Rother, "Inferring m-best diverse labelings in a single one," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1814–1822.

[20] C. Chen, V. Kolmogorov, Y. Zhu, D. Metaxas, and C. Lampert, "Computing the m most probable modes of a graphical model," in *Proc. of Artificial Intelligence and Statistics*, 2013, pp. 161–169.

[21] C. Chen, C. Yuan, and C. Chen, "Solving m-modes using heuristic search." in *Proc. of IJCAI'16*, 2016, pp. 3584–3590.

[22] M. Zytnicki, C. Gaspin, S. de Givry, and T. Schiex, "Bounds Arc Consistency for Weighted CSPs," *Journal of Artificial Intelligence Research*, vol. 35, pp. 593–621, 2009.

[23] G. Pesant, "A regular language membership constraint for finite sequences of variables," in *International conference on principles and practice of constraint programming*. Springer, 2004, pp. 482–495.

[24] C. Bessière and P. Van Hentenryck, "To be or not to be ... a global constraint," in *Proceedings of CP'03*, 2003, pp. 789–794.

[25] J. H. M. Lee and K. L. Leung, "Consistency Techniques for Global Cost Functions in Weighted Constraint Satisfaction," *Journal of Artificial Intelligence Research*, vol. 43, pp. 257–292, 2012.

[26] D. Allouche, C. Bessiere, P. Boizumault, S. De Givry, P. Gutierrez, J. H. Lee, K. L. Leung, S. Loudni, J.-P. Métivier, T. Schiex *et al.*, "Tractability-preserving transformations of global cost functions," *Artificial Intelligence Journal*, vol. 238, pp. 166–189, 2016.

[27] F. Bacchus and P. Van Beek, "On the conversion between non-binary and binary constraint satisfaction problems," in *Proceedings of AAAI 1998*, 1998, pp. 310–318.

[28] J. Larrosa and R. Dechter, "On the dual representation of non-binary semiring-based CSPs," in *CP2000 workshop on soft constraints*, 2000.

[29] R. Das and D. Baker, "Macromolecular modeling with rosetta," *Annu. Rev. Biochem.*, vol. 77, pp. 363–382, 2008.

[30] I. Pohl, "Heuristic search viewed as path finding in a graph," *Artificial intelligence*, vol. 1, no. 3-4, pp. 193–204, 1970.

[31] P. Jégou and C. Terrioux, "Decomposition and good recording," in *Proc. of ECAI-04*, Valencia, Spain, 2004, pp. 196–200.

[32] S. de Givry, T. Schiex, and G. Verfaillie, "Exploiting Tree Decomposition and Soft Local Consistency in Weighted CSP," in *Proc. of AAAI-06*, Boston, MA, 2006.

[33] D. Bergman, W.-J. Van Hoeve, and J. N. Hooker, "Manipulating mdd relaxations for combinatorial optimization," in *International Conference on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems*. Springer, 2011, pp. 20–35.