

1 Graphical models: queries, complexity, algorithms

2 M.C. Cooper 


3 Université Fédérale de Toulouse, ANITI, IRIT, Toulouse, France

4 cooper@irit.fr

5 S. de Givry 

6 Université Fédérale de Toulouse, ANITI, INRAE, UR 875, Toulouse, France

7 Simon.de-Givry@inrae.fr

8 T. Schiex¹ 

9 Université Fédérale de Toulouse, ANITI, INRAE, UR 875, Toulouse, France

10 <http://www.inrae.fr/mia/T/schiex>

11 Thomas.Schiex@inrae.fr

12 — Abstract —

13 Graphical models (GMs) define a family of mathematical models aimed at the concise description of
14 multivariate functions using decomposability. We restrict ourselves to functions of discrete variables
15 but try to cover a variety of models that are not always considered as “Graphical Models”, ranging
16 from functions with Boolean variables and Boolean co-domain (used in automated reasoning) to
17 functions over finite domain variables and integer or real co-domains (usual in machine learning and
18 statistics). We use a simple algebraic semi-ring based framework for generality, define associated
19 queries, relationships between graphical models, complexity results, and families of algorithms, with
20 their associated guarantees.

21 **2012 ACM Subject Classification** Theory of computation → Discrete optimization; Mathematics of
22 computing → Graph algorithms

23 **Keywords and phrases** Computational complexity, tree decomposition, graphical models, submodu-
24 larity, message passing, local consistency, artificial intelligence, valued constraints, optimization

25 **Digital Object Identifier** 10.4230/LIPIcs.STACS.2020.XX

26 **Category** Tutorial

27 **Supplement Material** <http://github.com/toulbar2/toulbar2>

28 **Funding** The authors have received funding from the French "Investing for the Future – PIA3"
29 program under the Grant agreement ANR-19-PI3A-000, in the context of the ANITI project.

30 *M.C. Cooper*: Funded by ANR-18-CE40-0011

31 *S. de Givry*: Funded by ANR-16-CE40-0028.

32 *T. Schiex*: Funded by ANR-16-CE40-0028.

33 **1** Introduction

34 Graphical models are descriptions of decomposable multivariate functions. A variety of
35 famous frameworks in Computer Science, Logic, Constraint Satisfaction/Programming,
36 Machine Learning, Statistical Physics and Artificial Intelligence can be considered as specific
37 sub-classes of graphical models. In this paper, we restrict ourselves to models describing
38 functions of variables having each a finite (therefore discrete) domain with a totally ordered
39 co-domain, that may be finite or not. This excludes, for example, “Gaussian Graphical
40 Models” which are used in statistics, describing continuous probability densities.

¹ Corresponding author



41 A graphical model over a set of variables is defined by a finite set of small functions
 42 which are combined together using a binary associative and commutative operator, defining
 43 a joint multivariate function. The combined functions may be small because they involve
 44 few variables or because they are described using a restricted language.

45 As an example, propositional logic (PL) is organized around the idea of describing logical
 46 properties as functions over Boolean variables. From the values of these variables, one should
 47 be able to determine if a property is true or not. A usual way to achieve this is to use clauses
 48 (disjunction of variables or their negation) as “small functions” and combine them with
 49 conjunction, defining the Conjunctive Normal Form. The resulting joint function defines the
 50 truth value of the formula. Various queries on such graphical models have been considered:
 51 existence of a variable assignment that optimizes the joint function, the SAT/PL problem,
 52 model counting (the #P-complete #-SAT/PL problem), etc.

53 If we instead use tensors to describe Boolean functions over sets of variables of bounded
 54 cardinality and combine these functions with conjunction again, we obtain a constraint
 55 network (CN) and the existence of an assignment that optimizes the joint function is the
 56 Constraint Satisfaction Problem (CSP/CN).

57 In the extreme, we may use tensors to describe small real (in \mathbb{R}) functions combined
 58 using addition or multiplication, enabling the description of discrete probability distributions,
 59 as done with Markov Random Fields (MRFs) and Bayesian Networks (BNs) [74, 16]. The
 60 associated optimization problem is the Maximum A Posteriori (MAP/MRF) problem and
 61 weighted counting allows to compute the partition function [74] (or normalizing constant),
 62 another #P-complete problem with important applications in statistical physics and machine
 63 learning. The terminology of “graphical models” comes from the stochastic facet of GMs.
 64 Stochastic graphical models are of specific interest because they can be learned (or estimated)
 65 from data. Assuming that an i.i.d. sample of an unknown probably distribution is available,
 66 one may look for a graphical model that gives maximum probability to the sample. This
 67 maximum-likelihood approach has attractive asymptotic properties [74]. Efficient approximate
 68 estimation algorithms based on convex optimization exist [96]. Because of the unavoidable
 69 sampling noise, exact optimization algorithms are usually not sought. In this paper, we
 70 assume that the graphical models are known, either because they describe known rules or
 71 functions or have been previously estimated from data.

72 In this stochastic case, algorithms that provide some form of guarantee remain desirable,
 73 especially if the considered graphical model combines logical (deterministic) information,
 74 describing known/required properties, with probabilistic information learned from data.
 75 Indeed, logical information cannot be approximated without a complete loss of semantics.

76 In the rest of this paper, we first give an algebraic definition of what a graphical
 77 model is and consider the most usual queries. We then introduce the two main families of
 78 algorithms that can be used to exactly solve such queries: tree search and non-serial dynamic
 79 programming. Restricted to small sub-problems, we show how dynamic programming can
 80 provide fast approximate algorithms, called message-passing or local consistency enforcing
 81 algorithms, including associated polynomial classes. We then consider some empirically
 82 useful hybrid algorithms, with associated solvers, based on these approaches and recent
 83 applications in structural biology.

84 **2 Notations, Definition**

85 Variables are denoted as capital variables X, Y, Z, \dots that may be optionally indexed (X_i or
 86 just i). Variables can be assigned values from their domain. The domain of a variable X

is denoted as D_X or D_i for variable X_i . Actual values are represented as $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{g}, \mathbf{r}, \mathbf{t}, 1 \dots$ and an unknown value denoted as $u, v, w, x, y, z \dots$. Sequences or sets of objects are denoted in bold. A sequence of variables is denoted as $\mathbf{X}, \mathbf{Y}, \mathbf{Z}, \dots$. A sequence of values is denoted as $\mathbf{u}, \mathbf{v}, \mathbf{w}, \mathbf{x}, \mathbf{y}, \mathbf{z} \dots$. The domain of a sequence of variables \mathbf{X} is denoted as $D_{\mathbf{X}}$, the Cartesian product of the domains of all the variables in \mathbf{X} . An assignment $\mathbf{u}_{\mathbf{X}}$ is an element of $D_{\mathbf{X}}$ which defines an assignment for all the variables in \mathbf{X} . Finally, we denote by $\mathbf{u}_{\mathbf{X}}[\mathbf{Y}]$ (or $\mathbf{u}_{\mathbf{Y}}$) the *projection* of $\mathbf{u}_{\mathbf{X}}$ on $\mathbf{Y} \subseteq \mathbf{X}$: the sequence of values that the variables in \mathbf{Y} takes in $\mathbf{u}_{\mathbf{X}}$.

► **Definition 1** (Graphical Model). A Graphical Model $\mathcal{M} = \langle \mathbf{V}, \Phi \rangle$ with co-domain B and a combination operator \oplus is defined by:

- a sequence of n variables \mathbf{V} , each with an associated finite domain of size less than d .
- a set of e functions (or factors) Φ . Each function $\varphi_{\mathbf{S}} \in \Phi$ is a function from $D_{\mathbf{S}} \rightarrow B$. \mathbf{S} is called the scope of the function and $|\mathbf{S}|$ its arity.

\mathcal{M} defines a joint function:

$$\Phi_{\mathcal{M}}(\mathbf{v}) = \bigoplus_{\varphi_{\mathbf{S}} \in \Phi} \varphi_{\mathbf{S}}(\mathbf{v}[\mathbf{S}])$$

In this definition we assume that the co-domain B is totally ordered (by \prec) with a minimum and maximum element. Arbitrary elements of B will be denoted by Greek letters (α, β, \dots) . The combination operator \oplus is required to be associative, commutative, monotonic and has an identity element $\mathbf{0} \in B$, the minimum element of B . The maximum element \top of B is required to be absorbing ($\alpha \oplus \top = \top$). This set of axioms defines what is called a valuation structure [105], introduced in AI to represent graded beliefs or costs in Constraint Networks. It is closely related to triangular co-norms, often using $B = [0, 1]$ [72]. It is also known as a tomonoid in algebra and includes tropical algebra [56]. Commutativity and associativity make the combination insensitive to the order of application of \oplus . Monotonicity captures the fact that adding more information can only strengthen it. Finally, the specific roles of the maximum and minimum elements of B are here to express deterministic information: $\mathbf{0}$ represents the fact that a function has reached an absolute minimum and this has no effect on the value of the joint function value Φ . Instead, any combined function taking value \top will lead to a joint function that is also equal to \top . For simplicity, we assume that elements of B can be represented in constant space and that $a \oplus b$ can be computed in constant time. An important additional property of \oplus is often considered: idempotency ($\alpha \oplus \alpha = \alpha$). As section 6.1 will show, it has strong algorithmic implications. Finally, such valuation structures are said to be *fair* if, for any elements of B such that $\beta \preceq \alpha$, there exists a maximum γ such that $\beta \oplus \gamma = \alpha$ (such a γ may not exist in infinite structures). This element γ is denoted $\alpha \ominus \beta$ and defines a pseudo-inverse operator \ominus (we have $\beta \oplus (\alpha \ominus \beta) = \alpha$).

Structure (GM)	B	$a \oplus b$	\prec	$\mathbf{0}$	\top	Idemp.	$a \ominus b$
Boolean (PL,CN)	$\{t, f\}$	$a \wedge b$	$t < f$	t	f	yes	a
Additive (GAI)	$\bar{\mathbb{N}}$	$a + b$	$<$	0	$+\infty$	no	$a - b$
Weighted (CFN)	$\{0, 1, \dots, k\}$	$\min(k, a+b)$	$<$	0	k	no	$(a=k ? k : a-b)$
Probabilistic (MRF,BN)	$[0, 1]$	$a \times b$	$>$	1	0	no	a/b
Possibilistic (PCN)	$[0, 1]$	$\max(a, b)$	$<$	0	1	yes	$\max(a, b)$
Fuzzy (FCN)	$[0, 1]$	$\min(a, b)$	$>$	1	0	yes	$\min(a, b)$

■ **Table 1** Some valuation structures. Idemp. indicates the idempotency of \oplus . See [33] for details.

Valuation structures have been analyzed in detail [33, 31]. We know that any fair and countable valuation structure can be viewed as a stack of additive/weighted valuation

121 structures, which interact with each other as an idempotent structure (thus using $\oplus = \max$).
 122 The case $\oplus = \max$ being very close to the case of classical Constraint Networks [89, page
 123 293], most of the work has since then focused on the weighted and probabilistic cases.

124 The mathematical definition of graphical models above needs to be further refined by
 125 defining how the functions $\varphi_{\mathcal{S}} \in \Phi$ are represented. Thanks to the assumption of finite
 126 domains, a universal representation of functions can be obtained using tensors (or multi-
 127 dimensional tables) that map any sequence of values $\mathbf{u}_{\mathcal{S}} \in D_{\mathcal{S}}$ to an element of B . This
 128 representation requires $O(d^{|\mathcal{S}|})$ space where d is the maximum domain size. When useful,
 129 alternative specialized representations may be used, with possibly different complexities for
 130 various elementary operations and queries.

131 Altogether, this definition of graphical models covers a variety of well-studied frameworks,
 132 including constraint networks [102] (tensors + Boolean), propositional logic [14] (Boolean
 133 domains, clauses + Boolean), generalized additive independence models [7] (tensors + addi-
 134 tive), weighted propositional logic (Boolean domains, clauses + additive), Markov Random
 135 Fields [71, 74] (tensors + Probabilistic), Bayesian networks [74] (MRFs with normalized
 136 tensors describing conditional probabilities, organized according to a directed acyclic graph),
 137 Possibilistic and Fuzzy Constraint Networks [48] (tensors + Possibilistic/Fuzzy). There are
 138 various related models, such as *Ceteris Paribus* networks (or CP-nets [47]) that could be
 139 covered using a weaker set of axioms, possibly too weak to prove interesting results.

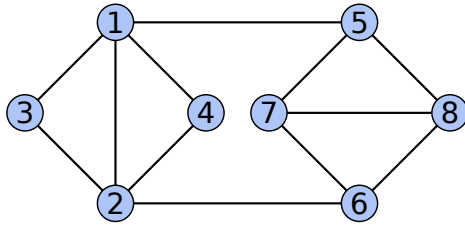
140 We are biased in the paper towards the family of Cost Function Networks (CFN) which
 141 are GMs using tensors (extended with so-called “global functions”, see e.g. [82, 3]) and the
 142 Weighted structure. This combines the ability of combining additive finite weights with a
 143 finite upper bound cost that appears naturally in many situations.

144 ► **Definition 2.** *Two graphical models $\mathcal{M} = \langle \mathbf{V}, \Phi \rangle$ and $\mathcal{M}' = \langle \mathbf{V}, \Phi' \rangle$, with the same
 145 variables and valuation structure are equivalent iff they define the same joint function:
 146 $\forall \mathbf{v} \in D_{\mathbf{V}}, \Phi_{\mathcal{M}}(\mathbf{v}) = \Phi_{\mathcal{M}'}(\mathbf{v})$.*

147 CFNs have a tight relationship with stochastic graphical models. Since $\log(a \times b) =$
 148 $\log(a) + \log(b)$, a $-\log$ transform applied to elements of B followed by suitable scaling,
 149 shifting and rounding can transform any stochastic graphical model into a Cost Function
 150 Network defining a joint function that represents a controlled approximation of the $-\log()$ of
 151 the joint function of the stochastic model. The function \log being monotonic, MAP/MRF
 152 and WCSP/CFN are essentially the same problems.

153 ► **Definition 3.** *Given two graphical models $\mathcal{M} = \langle \mathbf{V}, \Phi \rangle$ and $\mathcal{M}' = \langle \mathbf{V}, \Phi' \rangle$, with the same
 154 variables and valuation structure, \mathcal{M} is a relaxation of \mathcal{M}' iff $\forall \mathbf{v} \in D_{\mathbf{V}}, \Phi_{\mathcal{M}}(\mathbf{v}) \preceq \Phi_{\mathcal{M}'}(\mathbf{v})$.*

155 In a graphical model, the set of variables \mathbf{V} and the set of scopes \mathcal{S} of the different
 156 functions $\varphi_{\mathcal{S}} \in \Phi$ define a hyper-graph whose vertices are the variables of \mathbf{V} and the
 157 hyper-edges the different scopes. In the case in which the maximum arity is limited to 2,
 158 this hyper-graph is a graph, called the constraint graph or problem graph (hence the name
 159 graphical model). When larger arities are present, one can use the 2-section of the hypergraph
 160 as an approximation of it, often called the primal or moral graph of the GM. The bipartite
 161 incidence graph of this hypergraph has elements of \mathbf{V} and Φ as vertices, an edge connecting
 162 $X \in \mathbf{V}$ to $\varphi_{\mathcal{S}}$ iff $X \in \mathcal{S}$ and is often called the factor graph [1] of the graphical model.
 163 This hyper-graph or its incidence graph only captures the overall structure of the model but
 164 neither the domains nor the functions. Therefore, a more fine-grained representation, known
 165 as the micro-structure graph, is often used to represent (pairwise) graphical models, vertices
 166 representing values and weighted edges the value of functions on pairs of values.



■ **Figure 1** Simple graph example (from Wikipedia DSATUR). Optimum value and corresponding solution for Min-Cut ($s = 1, t = 8$): $\Phi_{\mathcal{M}}(aaaabbbb) = 2$, Max-Cut: $\Phi_{\mathcal{M}}(aabbbbaa) = 2$ (i.e., a maximum cut involving $12 - 2 = 10$ edges), Vertex Cover: $\Phi_{\mathcal{M}}(bbaaaabb) = 4$, Max-Clique: $\Phi_{\mathcal{M}}(aaabbbb) = 5$ (i.e., a maximum clique of size $8 - 5 = 3$), 3-Coloring: $\Phi_{\mathcal{M}}(abcccccab) = 0$, and Min-Sum 3-Coloring: $\Phi_{\mathcal{M}}(bcaaaabc) = 14$.

3 Queries

A query over a graphical model asks to compute simple statistics over the function such as its minimum or average value. Such queries already cover a wide range of practical usages. Assuming that `true` \prec `false`, the SAT/PL [14] and CSP/CN [102] problems are equivalent to finding the minimum of the joint function, which is also the case for (Weighted) Max-SAT, WCSP/CFN, MAP/MRF (or MPE/BN aka Maximum Probability Explanation in Bayesian Networks). In these optimization problems, we want to compute $\min_{\mathbf{v} \in D_{\mathcal{V}}} \Phi(\mathbf{v})$. Alternatively, for numerical B , counting problems require to compute $\sum_{\mathbf{v} \in D_{\mathcal{V}}} \Phi(\mathbf{v})$. A relatively general situation can be captured by introducing a so-called “elimination” operator \otimes , assumed to be associative, commutative and such that \oplus distributes over \otimes : $\alpha \oplus (\beta \otimes \gamma) = (\alpha \oplus \beta) \otimes (\alpha \oplus \gamma)$. The two operations (\otimes, \oplus) and their associated properties have been studied, with some variations, under a variety of names, often in relation with non-serial dynamic programming [111, 17, 1, 73, 72, 56]. The query to answer becomes $\bigotimes_{\mathbf{v} \in D_{\mathcal{V}}} \bigoplus_{\varphi_{\mathcal{S}} \in \Phi} \varphi_{\mathcal{S}}(\mathbf{v}[\mathcal{S}])$.

A powerful toolbox on graphical models can be built from three simple operations: assignment (or conditioning), combination and elimination.

Given a function $\varphi_{\mathcal{S}}$, it is possible to assign a variable $X_i \in \mathcal{S}$ with a value $a \in D_i$. We obtain a function on $\mathcal{T} = \mathcal{S} - \{X_i\}$ defined by $\varphi_{\mathcal{T}}(\mathbf{v}) = \varphi_{\mathcal{S}}(\mathbf{v} \cup \{X_i = a\})$. If we assign all the variables of a function, we obtain a function with an empty scope, denoted φ_{\emptyset} . This operation has negligible complexity (we directly access a part of the original cost function).

The second operation is the equivalent of the relational join operation in databases: it combines two functions $\varphi_{\mathcal{S}}$ and $\varphi_{\mathcal{S}'}$ into a single function, which is equivalent to their combination by \oplus . The resulting function has the scope $\mathcal{S} \cup \mathcal{S}'$ and is defined by $(\varphi_{\mathcal{S}} \oplus \varphi_{\mathcal{S}'}) (\mathbf{v}) = \varphi_{\mathcal{S}}(\mathbf{v}[\mathcal{S}]) \oplus \varphi_{\mathcal{S}'}(\mathbf{v}[\mathcal{S}'])$. The calculation of the combination of the two functions is exponential in time and space ($O(d^{|\mathcal{S} \cup \mathcal{S}'|})$). Observe that we can replace a set of functions by their combination without changing the joint function defined by the graphical model (preserving equivalence).

Finally, the elimination operation consists in summarising the information from a function $\varphi_{\mathcal{S}}$ on a subset of variables $\mathcal{T} \subset \mathcal{S}$. The elimination of the variables of $\mathcal{S} - \mathcal{T}$ in $\varphi_{\mathcal{S}}$ leads to its so-called projection (or marginal) onto \mathcal{T} , the function $\varphi_{\mathcal{T}}(\mathbf{u}) = \bigotimes_{\mathbf{v} \in D_{\mathcal{S}-\mathcal{T}}} \varphi_{\mathcal{S}}(\mathbf{u} \cup \mathbf{v})$. Since elimination requires enumerating the whole domain $D_{\mathcal{S}}$ of $\varphi_{\mathcal{S}}$, it has time complexity $O(d^{|\mathcal{S}|})$. The resulting function requires $O(d^{|\mathcal{T}|})$ space. If $\mathcal{S} - \mathcal{T}$ is a singleton $\{X_i\}$, we will denote by $\varphi_{\mathcal{S}}[-X_i] = \varphi_{\mathcal{S}}[\mathcal{S} - \{X_i\}]$ the result of the elimination of the single variable X_i .

These complexities are given for functions represented as tensors. They may change if other languages such as clauses, analytic representation or compact data structures such as weighted automata or decision diagrams [38, 51] are used to represent functions.

We close this section by expressing simple graph problems as WCSP/CFNs.

Given an undirected graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ with vertex set \mathcal{V} and edge set \mathcal{E} , we can define

- 205 the following WCSP/CFN $\mathcal{M} = \langle \mathbf{V}, \Phi \rangle$ problems, having one variable per vertex:
- 206 ■ $s-t$ Min-Cut: $\forall i \in \mathbf{V} \setminus \{s, t\}, D_i = \{a, b\}, D_s = \{a\}, D_t = \{b\}$, and $\Phi = \{\varphi_{ij} \mid (i, j) \in \mathbf{E}\}$,
 207 with $\varphi_{ij}(a, a) = \varphi_{ij}(b, b) = 0$ and $\varphi_{ij}(a, b) = \varphi_{ij}(b, a) = 1$.
- 208 ■ Max-Cut: $\forall i \in \mathbf{V}, D_i = \{a, b\}, \Phi = \{\varphi_{ij} \mid (i, j) \in \mathbf{E}\}$, with $\varphi_{ij}(a, a) = \varphi_{ij}(b, b) = 1$ and
 209 $\varphi_{ij}(a, b) = \varphi_{ij}(b, a) = 0$.
- 210 ■ Vertex Cover: $\forall i \in \mathbf{V}, D_i = \{a, b\}, \Phi = \{\varphi_i \mid i \in \mathbf{V}\} \cup \{\varphi_{ij} \mid (i, j) \in \mathbf{E}\}$, with $\varphi_i(a) = 0$,
 211 $\varphi_i(b) = 1, \varphi_{ij}(a, b) = \varphi_{ij}(b, a) = \varphi_{ij}(b, b) = 0$ and $\varphi_{ij}(a, a) = \top$.
- 212 ■ Max-Clique: $\forall i \in \mathbf{V}, D_i = \{a, b\}, \Phi = \{\varphi_i \mid i \in \mathbf{V}\} \cup \{\varphi_{ij} \mid (i, j) \notin \mathbf{E}\}$, with $\varphi_i(a) = 0$,
 213 $\varphi_i(b) = 1, \varphi_{ij}(a, b) = \varphi_{ij}(b, a) = \varphi_{ij}(b, b) = 0$ and $\varphi_{ij}(a, a) = \top$.
- 214 ■ Graph Coloring with 3 colors: $\forall i \in \mathbf{V}, D_i = \{a, b, c\}, \Phi = \{\varphi_{ij} \mid (i, j) \in \mathbf{E}\}$, with $\forall u, v$,
 215 $\varphi_{ij}(u, v) = 0$ if $u \neq v$ and $\varphi_{ij}(u, v) = \top$ otherwise.
- 216 ■ Min-Sum Coloring with 3 colors: $\forall i \in \mathbf{V}, D_i = \{a, b, c\}, \Phi = \{\varphi_i \mid i \in \mathbf{V}\} \cup \{\varphi_{ij} \mid$
 217 $(i, j) \in \mathbf{E}\}$, with $\varphi_i(a) = 1, \varphi_i(b) = 2, \varphi_i(c) = 3$, and $\forall u, v, \varphi_{ij}(u, v) = 0$ if $u \neq v$ and
 218 $\varphi_{ij}(u, v) = \top$ otherwise.
- 219 An example of optimal solution for each problem is given in Fig.1.

220 4 Tree search

221 One of the most basic algorithms to answer the query $\bigotimes_{v \in D_{\mathbf{V}}} \bigoplus_{\varphi_S \in \Phi}$ relies on conditioning
 222 and can be described as brute force tree-search. It relies on the fact that if all variables in a
 223 graphical model are assigned (all variables have domain size 1), the answer can be obtained
 224 by just computing Φ on the assignment. If a given model has a variable $X_i \in \mathbf{V}$ with a larger
 225 domain, the query can be reduced to solving two queries on simpler models, one where X_i is
 226 assigned to some value $a \in D_i$ and the other where a is removed from D_i . The result of these
 227 queries is then combined using \otimes . This can be understood as the exploration of a binary
 228 tree whose root is the original graphical model and where leaves are fully assigned models
 229 on which the query can be answered. This tree can be explored using various strategies
 230 (Depth-First, Breadth-First, Iterative [95],...), DFS has $O(nd)$ space and $\theta(\exp(n))$ time.

231 In the case of optimization ($\otimes = \min$) on a GM \mathcal{M} it is possible to exploit any available
 232 lower bound on the values of the leaves below a given node in the tree for pruning. If this
 233 lower bound is larger than or equal to the value of the joint function on a best known leaf,
 234 then the sub-tree can be pruned. If \mathcal{M} contains a constant function φ_{\emptyset} with empty scope,
 235 then the value of this function provides such a lower bound, thanks to monotonicity and
 236 non-negativity. With pruning, the order in which this tree is explored (the choice of the
 237 branching variable X_i and value a) becomes crucial for empirical efficiency.

238 5 Non-serial Dynamic Programming

239 An alternative approach for answering the query above consists in using non-serial dynamic
 240 programming (DP) [11], also called Variable-Elimination (VE), bucket elimination [44],
 241 among other names [74, 111, 1]. The process relies on the distributivity of \otimes over \oplus and has
 242 also been described as the Generalized Distributive Law[1]. To our knowledge, its axiomatic
 243 requirements in a general situation were first studied in [111]. The fundamental idea itself is
 244 reminiscent of famous elimination algorithms (e.g. Gaussian elimination) and was already
 245 described in 1972 under the name of Non-Serial Dynamic Programming [11], which we use.
 246 This book also defines a graph parameter (DIMENSION), and is the same as tree-width [19].

247 In its simplest variable elimination variant, non serial DP works variable per variable,
 248 replacing a variable X and all functions that involve it by a function on the neighbors of

249 X . Following the stochastic GMs terminology, this new function will be called a “message”.
 250 Given a graphical model $\mathcal{M} = \langle \mathbf{V}, \Phi \rangle$, a variable $X \in \mathbf{V}$, let Φ^X be the set of functions
 251 $\varphi_{\mathcal{S}} \in \Phi$ such that $X \in \mathcal{S}$ and T the neighbors of X in the graph of the problem, we define
 252 the message $m_{\mathbf{T}}^{\Phi^X}$ from Φ^X to \mathbf{T} as:

$$253 \quad m_{\mathbf{T}}^{\Phi^X} = \left(\bigoplus_{\varphi_{\mathcal{S}} \in \Phi^X} \varphi_{\mathcal{S}} \right)[-X] \quad (1)$$

254 By distributivity, it is now possible to rewrite our query as:

$$255 \quad \bigotimes_{\mathbf{v} \in D_{\mathbf{V}}} \bigoplus_{\varphi_{\mathcal{S}} \in \Phi} \varphi_{\mathcal{S}}(\mathbf{v}[\mathcal{S}]) = \bigotimes_{\mathbf{v} \in D_{\mathbf{V}-\{X\}}} \bigoplus_{\varphi_{\mathcal{S}} \in \Phi - \Phi^X \cup \{m_{\mathbf{T}}^{\Phi^X}\}} \varphi_{\mathcal{S}}(\mathbf{v}[\mathcal{S}])$$

256 This equality shows that our initial query can be answered by solving a similar query on
 257 a graphical model that has one variable less and where all functions involving this variable
 258 have been replaced by a new function (or message) $m_{\mathbf{T}}^{\Phi^X}$. This operation can be applied
 259 to every variable successively until a final graphical model with no variable and a constant
 260 function φ_{\emptyset} is obtained. The value of this function is the answer to the initial query.

261 When tensors are used to represent functions, the successive applications of combination
 262 and elimination show that computing $m_{\mathbf{T}}^X$ is $O(d^{|\mathbf{T}+1|})$ time and space but space can be
 263 easily reduced to $O(d^{|\mathbf{T}|})$ if elimination is performed incrementally. This can be different
 264 with other representations. In propositional logic, if ℓ is a literal over variable X and if Φ^X
 265 contains the two clauses $\varphi_{\mathcal{S}} = (\ell \vee \mathbf{L})$ and $\varphi_{\mathcal{S}'} = (-\ell \vee \mathbf{L}')$, where \mathbf{L} and \mathbf{L}' are clauses, then
 266 the message $m_{\mathbf{T}}^X$ obtained by combining the two clauses and eliminating X is the function
 267 represented by the clause $\mathbf{L} \vee \mathbf{L}'$: the resolution principle [100] performs efficient variable
 268 elimination [45].

269 It is well-known that the order in which the variables are eliminated can have a strong
 270 influence on the complexity of the whole process. Each elimination creates a new function
 271 whose scope is known. As each step is exponential in the number of variables in the
 272 neighbourhood of the eliminated variable X which come after X in the elimination order,
 273 one may prefer to simulate the process (play the elimination game) and get an estimate of
 274 the global complexity without actually performing the calculations. It is the step for which
 275 the number w of future neighbors is maximum which determines the complexity (spatial and
 276 temporal): this complexity is exponential in w . The parameter w is called the induced width
 277 of the graph of the GM but is better known as its tree-width for the given elimination order.
 278 Minimizing w is known to be NP-hard, but useful heuristics exist [18].

279 This algorithm defines the first non-trivial tractable class for the query above: graphical
 280 models with a bounded tree-width graph.

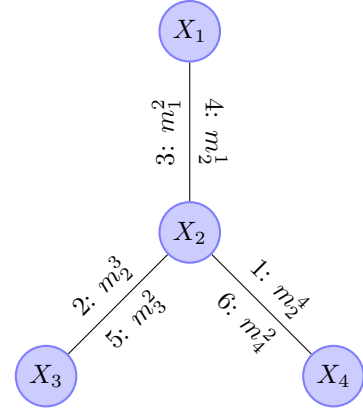
281 5.1 Message passing

282 The bounded tree-width class is specifically interesting for pairwise models \mathcal{M} with an acyclic
 283 graph (tree-width one). In this case, the query above can be solved by rooting the tree in
 284 any variable $X \in \mathbf{V}$ and iteratively eliminating the leaves of this graph until only the root
 285 remains. In this case, messages $m_{\mathbf{T}}^X$ are such that $|\mathbf{T}| = 1$ and the message that corresponds
 286 to the elimination of variable X_i with parent X_j through edge φ_{ij} is just denoted m_j^i (a
 287 message from i to j). Denoting by $neigh(X)$ the set of neighbor variables of X in the tree,
 288 the message computed by applying Eq. 1 is:

$$289 \quad m_j^i = \bigotimes_{X_i} (\varphi_i \oplus \varphi_{ij} \bigoplus_{X_o \in neigh(X_i), o \neq j} m_o^i)$$

290 In the end, one variable X remains and the joint function Φ it defines describes the
 291 so-called *marginal* of the joint function defined by \mathcal{M} . This is a function of X that answers
 292 the query for every possible assignment of X .

These marginal functions are of specific interest in counting queries over stochastic models as they provide marginal probabilities. Specific algorithms have therefore been developed to compute the marginals over all the variables of acyclic models. This requires only two steps (instead of one per variable). In this algorithm, messages are computed and kept aside and no function is removed from the graphical model. We therefore have an initially empty, growing list L of computed messages. If a variable has received messages from all its neighbors but one, it becomes capable of computing the message for this variable and adds it to the list of computed messages. The algorithm therefore starts by computing messages from leaves and ultimately computes two messages m_j^i and m_i^j for every edge φ_{ij} . For any variable X_i , the marginal of the joint function Φ on X_i is directly accessible as $\varphi_i \oplus_{o \in \text{neigh}(X_i)} m_i^o$. It is easy to check that the gathered messages for every vertex X are exactly those that would have been done if the tree had been rooted in X .



■ **Figure 2** The graph of a GM with three pairwise functions. The marginal of Φ on X_2 can be computed by combining all three incoming messages: m_2^3, m_2^1, m_2^4 .

294 If the initial graph is cyclic, a tree decomposition of the graph can be identified and
 295 the algorithm above used to send messages m_S^C by combining all functions in cluster C
 296 and projecting to separator S . The resulting algorithms similarly computes marginals
 297 over clusters which can be further projected onto every variable inside [74]. This has two
 298 advantages over the pure variable elimination algorithm above: the space complexity is
 299 $O(\exp(s))$ where s is the size of the largest separator in the tree decomposition and all the
 300 marginals are accessible. In its “one-pass” variant, this algorithm is the “block-by-block”
 301 elimination introduced in 1972 by Bertelé and Brioschi [11]. Despite the improved space
 302 complexity (but unchanged time complexity), this algorithm is restricted to problems with
 303 very small tree-width (especially with large domain sizes).

6 Loopy Belief propagation and Local Consistency

305 The Loopy Belief Propagation heuristic can be used when the algorithm above cannot finish
 306 in reasonable time. In a pairwise model, the list L is initialized with all possible constant
 307 empty messages $m_i^j = m_j^i = \mathbf{0}$ for all $\varphi_{ij} \in \Phi$. Every message is then updated using the
 308 same equation as above: $m_j^i = \otimes_{X_i} (\varphi_i \oplus \varphi_{ij} \oplus_{X_o \in \text{neigh}(X_i), o \neq j} m_i^o)$. These updates can be
 309 done synchronously (new versions are all computed simultaneously and replace those of the
 310 previous iteration) or asynchronously. This is repeated until quiescence or after a finite
 311 number of iterations as the algorithm has no guarantee of termination in general. Loopy
 312 BP was introduced by Pearl [97] and later used for signal coding and encoding defining
 313 Turbo-decoding [10]. Despite its limited theoretical properties, it is one of the most used
 314 algorithms over stochastic graphical models, probably the most cited (implemented in every
 315 GSM phone). It has been the object of intense study [119]. It can be used to get heuristic
 316 answers for optimization or counting queries under the name of the max-sum, min-sum or
 317 sum-prod algorithm [1].

6.1 Local consistency and Filtering

For optimization ($\otimes = \min$), if \oplus is either idempotent or fair, these algorithms can be transformed into well-behaved polytime reformulation techniques that provide incremental lower bounds, for Branch and Bound algorithms.

► **Theorem 4.** *Suppose that \oplus is idempotent, and consider $\mathcal{M} = \langle \mathbf{V}, \Phi \rangle$ and $\mathcal{M}' = \langle \mathbf{V}, \Phi' \rangle$ a relaxation of \mathcal{M} . Then $\mathcal{M}'' = \langle \mathbf{V}, \Phi \cup \Phi' \rangle$ is equivalent to \mathcal{M} .*

Proof. From the axioms of an idempotent valuation structure, consider two valuations $\mathbf{0} \preceq \beta \preceq \alpha \in B$. Then $\alpha = \alpha \oplus \mathbf{0} \preceq \alpha \oplus \beta \preceq \alpha \oplus \alpha = \alpha$ by identity, monotonicity and idempotency successively. Then $\Phi_{\mathcal{M}''} = \Phi_{\mathcal{M}} \oplus \Phi_{\mathcal{M}'}$. The results follows from the fact that \mathcal{M}' is a relaxation of \mathcal{M} . ◀

► **Theorem 5.** *If $\otimes = \min$, any message $m_{\mathbf{T}}^X$ computed by elimination is a relaxation of Φ^X and therefore of \mathcal{M} .*

This follows directly from the definition of a message and the fact that $\otimes = \min$. Together, these results show that any message (computed by Eq. 1) can be added to an idempotent graphical model, without changing its meaning. This result applies to the Boolean, Possibilistic and Fuzzy cases. In the Boolean case, we recover the trivial fact that a logical consequence (relaxation) of a formula can be added to it without changing its semantics (as achieved by the resolution principle [100] and the seminal Davis and Putnam algorithm [40]). It also immediately provides algorithms for Possibilistic and Fuzzy graphical models [49].

These observations are very useful in the context of message passing algorithms. If \oplus is idempotent, it becomes possible to add the generated messages to the set of functions Φ of the graphical model with the guarantee that equivalence will be preserved. This idea underlies all the “(Boolean) constraint propagation” algorithms (e.g., unit propagation in PL and arc consistency filtering in CNs) that have been actively developed in the last decades. We now give a non-standard definition of Arc Consistency in binary idempotent graphical models to illustrate this:

► **Definition 6.** *A graphical model $\mathcal{M} = \langle \mathbf{V}, \Phi \rangle$ with idempotent \oplus is arc-consistent iff every variable $X \in \mathbf{V}$ is arc consistent w.r.t. every function $\varphi_{\mathbf{S}}$ s.t. $X \in \mathbf{S}$. A variable X_i is arc-consistent w.r.t. a function φ_{ij} iff the message m_i^j is a relaxation of φ_i .*

This local consistency property can be satisfied (or not) by an idempotent graphical model. When it's not satisfied, there is an obvious brute force algorithm that can transform the non AC model into an equivalent AC model: it suffices to apply message passing and directly combine all messages m_i^j in \mathcal{M} until the φ_i do not change. One can show that this algorithm must stop in polytime and that the process is confluent (there is only one fixpoint). This algorithm is not optimal however and several generations of AC algorithms for e.g. Constraint Networks have been proposed until the first time optimal algorithm (AC4 [90]), the first space and time optimal (AC6 [13]) and the simplest empirically most efficient time/space optimal variant (AC2001 aka AC3.1) came out [12, 120].

Arc consistency and Unit Propagation in SAT are fundamental in efficient CSP and SAT provers. Adding messages (new unary functions) to the problem makes these unary functions increasingly tight. Ultimately, it may be the case that some $\varphi_i \in \Phi$ becomes such that $\forall a \in D_i, \varphi_i(a) = \top$. Because equivalence is preserved, it is then known that the joint function of the original graphical model is always equal to \top (it is inconsistent in logical terminology). Naturally, being a polytime process, arc consistency enforcing cannot provide

363 such an inconsistency proof in all cases (assuming $P \neq NP$). Stronger, more expensive,
 364 messages can be obtained using messages m_T^C combining functions in C projected onto
 365 T . In pairwise GMs, when C is defined by all functions whose scope is included in any
 366 set of cardinality less than i projected on any sub-scope of cardinality $i - 1$, this is called
 367 i -consistency enforcing [30, 5], which solves idempotent GMs with tree-width less than i in
 368 polynomial time.

369 6.2 The non idempotent fair case

370 When optimizing, if \oplus is not idempotent, it becomes impossible to simply add messages
 371 to the graphical model being processed and preserve equivalence. However, the pseudo-
 372 difference operator \ominus of fair structures makes it possible to add the message to the model and
 373 compensate for this by subtracting the message from its source [104]. Such a generalization
 374 may lead to loss of guaranteed termination since information can both increase and decrease
 375 at different local levels. While usual local consistency enforcing takes a set of functions and
 376 adds the message m obtained by eliminating a subset of variables in the problem, what has
 377 been called an *Equivalence Preserving Transformation* (EPT) combines a set of functions
 378 into a new function φ , computes the associated message (or any relaxation of it) and replaces
 379 the original set of functions by m and $\varphi \ominus m$. When scopes are unchanged, this has been
 380 called reparameterizations in MRFs [117].

381 If the problem of computing an optimal *sequence* of EPTs (in the sense that applying
 382 these messages reformulate the GM into another one that has a maximum φ_\emptyset) is NP-hard
 383 for integer costs, finding an optimal *set* of messages using rational costs can be reduced to
 384 a linear programming instance of polynomial size [32], a reduction which was previously
 385 published in 1976 [110], in Russian. Schlesinger's team has produced a variety of results on
 386 GMs that have been reviewed in English [118]. The LP-based algorithm has the ability to
 387 exactly optimize additive or weighted [29] GMs with a tree-structure or with only submodular
 388 functions but solving the LP is empirically too slow to be of practical use in most cases. This
 389 LP has been shown to be universal, in the sense that any reasonable LP can be reduced to
 390 (the dual of) such an LP in linear time, with a constructive proof [98].

391 ► **Definition 7.** *Function φ_S is submodular if $\forall \mathbf{u}, \mathbf{v} \in D_S, \varphi_S(\min(\mathbf{u}, \mathbf{v})) + \varphi_S(\max(\mathbf{u}, \mathbf{v})) \leq$*
 392 *$\varphi_S(\mathbf{u}) + \varphi_S(\mathbf{v})$.*

393 This assumes that domains are ordered, max and min being applied pointwise. If it exists, a
 394 witness order for submodularity can be easily found [109] defining the polynomial class of
 395 weighted GMs with permuted submodular functions [108].

396 The best known general algorithm for Boolean submodular function minimization is
 397 $O(en^3 \log^{O(1)} n)$ [83]. For CFNs, practical algorithms can be obtained by exploiting a
 398 connection with idempotent GMs (here CNs). Any weighted function φ_S can be transformed
 399 into a Boolean function $\text{Bool}(\varphi_S)$ which is **true** iff φ_S is non **0**.

400 ► **Definition 8** ([34, 29]). *Given a weighted GM (or CFN) $\mathcal{M} = \langle \mathbf{V}, \Phi \rangle$, $\text{Bool}(\mathcal{M}) =$*
 401 *$\langle \mathbf{V}, \{\text{Bool}(\varphi_S) \mid |S| > 0\} \rangle$ (a constraint network).*

402 ► **Definition 9** (Virtual Arc Consistency (VAC)[34]). *A weighted GM $\mathcal{M} = \langle \mathbf{V}, \Phi \rangle$ is Virtual*
 403 *Arc Consistent iff enforcing AC on $\text{Bool}(\mathcal{M})$ does not prove inconsistency.*

404 If enforcing AC on $\text{Bool}(\mathcal{M})$ yields an inconsistency proof, it is possible to extract a
 405 minimal sub-proof and transform it in a set of EPTs that will increase φ_\emptyset when applied on
 406 \mathcal{M} . The repeated application of this principle leads to an $O(ed^2m/\varepsilon)$ time, $O(ed)$ space VAC

407 enforcing algorithm (where ε is a required precision). It can be seen as a sophistication of
 408 max-flow algorithms using “augmenting proofs” [77, 118]. On Min-Cut problems, proofs can
 409 have the expected augmenting path structure. Related algorithms, using a Block Coordinate
 410 Descent approach to approximately solve the LP above have also been investigated in Image
 411 processing: the TRW-S algorithm [75] fixpoints satisfy the VAC property (called Weak Tree
 412 Agreement in MRFs) which can be exploited to detect strongly persistent assignments [57]
 413 (subsets of variables having the same value in all optimal solutions). On GMs with Boolean
 414 variables, the bound φ_\emptyset is related to the max-flow roof-dual lower bound of Quadratic
 415 Pseudo-Boolean Optimization [20].

416 All these algorithms ultimately produce strengthened lower-bounds φ_\emptyset . In the con-
 417 text of Branch & Bound, looser approximations of the dual are often used because of
 418 their high incrementality. One of the most useful is existential directional arc consistency,
 419 with $O(ed^2 \max(nd, m))$, $O(nd)$ space complexity on pairwise models [81], that solves tree-
 420 structured problems.

421 **7** Hybrid algorithms

422 In this section, we rapidly review a subset of the large number of hybrid algorithms that have
 423 been designed to rigorously answer optimization queries over graphical models. The arena of
 424 existing algorithms is currently clearly dominated by a family of algorithms that combines
 425 tree search with local consistency enforcing. Local consistency enforcing strengthens the
 426 obvious lower bound defined by φ_\emptyset . It reformulates the graphical model associated with
 427 each node of the search-tree incrementally. It provides improved information with tightened
 428 unary φ_i . This information can be used in cheap variable and value ordering heuristics, to
 429 decide which variable to explore first, along which value. These heuristics are crucial for
 430 the empirical efficiency of the algorithms (as evaluated on large sets of benchmarks, which
 431 are increasingly available in this data era). They have become adaptive [91, 21]: they
 432 are parameterized and these parameters change during tree search, trying to identify and
 433 favor the selection of variables that belong to the regions of the graphical model that contain
 434 strong information, variables which once assigned would lead to rapid backtracking.

435 These algorithms can be (empirically) enhanced by hybrid branching strategies mixing
 436 depth and best-first, restarts (preserving information not invalidated by restart), stronger
 437 message passing at the root node, dominance analysis (showing a value \mathbf{a} can always be
 438 replaced by value \mathbf{b} without degrading Φ allows us to remove \mathbf{a}).

439 In the idempotent \oplus case, conflict analysis [15] is a powerful technique that allows to
 440 produce a new informative relaxation (or logical consequence) of the model when a conflict
 441 occurs during tree search (a backtrack must occur). Initially developed [42] and improved [107]
 442 for Constraint Networks, this has been adapted to SAT and is now a fundamental ingredient
 443 of the modern CDCL (Conflict Directed Clause Learning) SAT provers that obsoleted the
 444 90’s generation of advanced DPLL (Davis, Putnam, Logemann, Loveland) provers [39].

445 At this point, one may note that for optimization at least, the empirically most efficient
 446 algorithms are algorithms that have $O(\exp(n))$ worst-case complexity while algorithms with
 447 better worst-case complexity (based on non-serial dynamic programming) are restricted to
 448 a small subset of problems having (small) bounded tree-width (depending on d). A series
 449 of proposals tried to define algorithms that combined the empirical efficiency of enhanced
 450 tree-search algorithms with tree-width based bounds. To the best of our knowledge, the first
 451 algorithm along this line is the Pseudo-Tree search algorithm [53], for solving the CSP in
 452 constraint networks. Defined in 1985, this algorithm relies on a so-called pseudo-tree (defined

below). It is shown to have a complexity which is exponential in the height of the pseudo-tree and uses only linear space. This pseudo-tree height was later shown to be related by a $\log(n)$ factor to induced width (a synonymous of tree-width) in 1995 [9, 106]. Pseudo-tree height seems to be the exact equivalent of tree-depth [93].

► **Definition 10** (pseudo-tree, pseudo-tree height [53]). *A pseudo-tree arrangement of a graph G is a rooted tree with the same vertices as G and the property that adjacent vertices in G reside in the same branch of the tree. The pseudo-tree height of G is the minimum, over all pseudo-tree arrangements of G of the height of the pseudo-tree arrangement.*

The idea was revived in the context of counting problems in the “Recursive Conditioning” algorithm, relying on the related notion of d-tree [37]. This was quickly followed by the proposal of two algorithms for CSP/CN, WCSP/CFN and MAP/MRF relying either on pseudo-trees, leading to the family of AND-OR tree and graph search algorithms [85, 87, 86, 88] or tree-decompositions, leading to the Backtrack Tree-Decomposition algorithm (BTD [66, 67, 41]). These algorithms enhance the original pseudo-tree search along two lines: they add some form of local consistency enforcing to prune the search tree, with associated non trivial per-cluster lower-bound management and may also memorize information, leading to algorithms with worst-case time/space complexity that reach those of the best elimination algorithms, with much better empirical complexity, thanks to pruning and variable ordering heuristics.

These algorithms take as input a graphical model (we assume a pairwise GM for simplicity) and a tree decomposition of its graph. The tree decomposition is rooted in a chosen cluster. Then a usual (hybrid) tree-search algorithm is used but the variable assignment order is constrained by a rule that forbids to assign a variable from a cluster if all the variables of its parent cluster have not already been assigned. When this happens, the variables in the separators between the parent cluster and any of its sons are assigned: the functions that connect the parent and son clusters can be cheaply eliminated, the clusters disconnect and can be solved independently given the current assignment of the separator. This principle is applied recursively on sub-problems. For every first visit of a given assignment of a separator, it is possible to memorize the value of the query for this assignment. When the same assignment is revisited later, it can be directly reused. This is what the AND/OR graph search and BTD algorithms do, at the cost of an $O(d^s)$ space complexity. Their time complexity, in the simplest case of the CSP problem, is just exponential in the tree-width instead of the pseudo-tree height. Alternatively, no space is used in the AND/OR tree search algorithm and the worst-case space and time complexity become those of the Pseudo-Tree Search algorithm (but with improved empirical complexity thanks to *mini-bucket elimination*, a specific form of message passing that provides the bounds required for pruning [43, 46]).

Thanks to the bounds and reformulations provided by local consistency enforcing, these algorithms will typically explore a tiny fraction of the separators, empirically requiring much less space than a pure non-serial dynamic programming algorithm. The constraint that the rooted tree decomposition imposes on variable assignment ordering will however play a possibly negative role. Finally, because space is an abrupt constraint in practice, tree-decompositions with large separators may be unexploitable in practice if a space-intensive approach is used. One usual way to deal with space constraints is to use a bounded amount of space for *caching* separator assignments and corresponding query answers and have some dedicated cache management strategy. In the end, the best tree-decompositions for hybrid algorithms such as BTD or Pseudo-Tree Search will usually not be minimum tree-width decompositions (if they could be computed). The usual approach here is to rely on simple heuristics such as min-degree, Maximum Cardinality Search [101] or min-fill to heuristically

501 build a decomposition. This may be randomized and iterated [70]. The result may then be
 502 improved by cluster-merging operations, trying to minimize separator sizes while preserving
 503 sufficient cluster size so that variable ordering heuristics do not get too constrained. Not
 504 only we do not have any final definition of what is an optimal tree decomposition but
 505 even a simple min-fill heuristics may be too long in practice to be useful (on large GMs,
 506 executing the polytime min-fill algorithm on the GM graph may take more time than solving
 507 the later NP-hard query on the GM). For this reason, new approaches that directly and
 508 efficiently build empirically useful decompositions for GM optimization queries are now being
 509 introduced [64, 63]. To better exploit the fact that different assignments may lead to different
 510 subproblems, with different graphs at the micro-structure level, dynamic decomposition is
 511 also explored. Some of these strategies rely on iterated (hyper)graph-partitioning using
 512 large-graphs dedicated heuristics such as Metis [69] or PaToH [24, 80], in the spirit of the
 513 seminal Nested Dissection algorithm [54].

Exact Method	Time	Space
Depth First Branch & Bound	$O(\exp(n))$	$O(n)$
Hybrid Best-First Search [4]	$O(\exp(n))$	$O(\exp(n))$
Variable elimination [11]	$O(n \exp(w))$	$O(n \exp(w))$
Block-by-block elimination [11]	$O(n \exp(w))$	$O(n \exp(s))$
Pseudo-Tree Search [53, 9]	$O(n \exp(w \log(n)))$	$O(n)$
AND/OR tree search [85, 87]	$O(n \exp(w \log(n)))$	$O(n)$
AND/OR graph search [86, 88]	$O(n \exp(w))$	$O(n \exp(s))$
BTD and variants [66, 113, 41, 103, 4]	$O(k n \exp(w))$	$O(n \exp(s))$

■ **Table 2** Time and space complexities of exact methods for a CFN with tree-width w , s maximum separator size ($s \leq w \leq n$), and initial problem upper bound k .

514 In the *idempotent* cases, the exploitation of tree-decompositions is implicit in algorithms
 515 relying on a combination of clause/constraint learning and restarts (confirmed in practice [62]).

516 ► **Theorem 11** ([6, 62]). *A standard randomized CDCL SAT-solver with a suitable learning*
 517 *strategy will decide the consistency of any pairwise Constraint Network instance with tree-*
 518 *width w with $O(n^{2w} d^{2w})$ expected restarts.*

519 8 Additional complexity results

520 Although NP-hard in general, under certain restrictions, calculating the global value of a GM
 521 can be achieved in polynomial time. We have seen that this is the case when the tree-width
 522 of the primal graph is bounded by a constant. In terms of graph structure, the family of GMs
 523 with bounded tree-width has been clearly the most exploited in practice (with the closely
 524 related branch-width [99]). In the non-pairwise case, parameters such as hypertree-width are
 525 implicitly exploited by hybrid algorithms such as BTD [65].

526 In the case of CFNs, restrictions on the language of cost functions can also define tractable
 527 classes. Restrictions which are not exclusively concerned with the graph, nor exclusively
 528 concerned with the language of cost functions define what are known as hybrid classes. We
 529 consider language-based tractable classes first.

530 The characterisation of all tractable languages of cost functions was a long-standing
 531 problem which has recently been solved thanks to the characterisation of tractable languages
 532 in the last remaining (and most difficult) case of Boolean valuation structures [22, 121]. This

533 latter result resolved positively the so-called Feder and Vardi Conjecture [52] that there is
 534 a P/NP-complete dichotomy for the constraint satisfaction problem parameterized by the
 535 language of possible constraint relations [23].

536 There is only one non-trivial class of cost functions defining a tractable subproblem of
 537 CFNs which comes out of the study of MAX-SAT: the class of submodular functions (see
 538 Definition 7). The class of submodular functions includes all unary functions, the binary
 539 functions $\sqrt{x^2 + y^2}$, $((x \geq y) ? (x - y)^t : k)$ (for $t \geq 1$), $K - xy$, the cut function of a graph and
 540 the rank function of a matroid [55]. Efficient algorithms have been developed in Operations
 541 Research to minimize submodular functions [83, 25]. Another approach [34] consists in
 542 establishing virtual arc consistency (which preserves the submodularity of the cost functions).
 543 By the definition of VAC, the arc-consistency closure \mathcal{Q} of $\text{Bool}(\mathcal{P})$ is non-empty and, by
 544 definition of submodularity, its cost functions are both *min-closed* and *max-closed* [61]; to
 545 find a solution of \mathcal{Q} (which is necessarily an optimal solution of \mathcal{P}), it suffices to assign
 546 always the minimum (or always the maximum) value in each domain of \mathcal{Q} .

547 A CFN can be coded as an integer programming problem (whose variables include
 548 $v_{ia} \in \{0, 1\}$ which is equal to 1 if and only if $X_i = a$ in the original CFN instance [60]). The
 549 linear relaxation of this integer programming problem (in which v_{ia} is now a real number
 550 in the interval $[0, 1]$) has integer optimal solutions if all cost functions are submodular,
 551 meaning that the CFN can be solved by linear programming. The dual of this relaxation
 552 is exactly the linear program used by OSAC [29] to find an optimal transformation (set of
 553 EPTs) [118]. CFNs restricted to a language of finite-valued cost functions over the valuation
 554 structure $\mathbb{Q}^{\geq 0} \cup \{\infty\}$ is tractable if and only if it is solved by this linear program [114].
 555 This notably includes languages of cost functions that are submodular on arbitrary lattices.
 556 State-of-the-art results concerning the tractability of languages of cost functions are covered
 557 in detail in a recent survey article [78].

558 These theoretical results demonstrate the importance of submodularity and linear pro-
 559 gramming in the search for tractable subproblems of the CFN. However, we should mention
 560 that there are tractable languages of cost functions other than the class of submodular func-
 561 tions. For example, replacing the functions min and max in the definition of submodularity
 562 by *any* pair of commutative conservative functions f, g (where conservative means that $\forall x, y,$
 563 $f(x, y), g(x, y) \in \{x, y\}$) gives rise to a tractable class [28].

564 *Hybrid classes* [36] may be defined by restrictions on the micro-structure of the CFN. To
 565 illustrate this, we describe the hybrid tractable class defined by the *joint winner property*
 566 (JWP) [35]. A class of *binary* CFNs satisfies the JWP if for any three variable-value
 567 assignments (to three distinct variables), the multiset of pairwise costs imposed by the binary
 568 cost functions does not have a unique minimum. If there is no cost function explicitly defined
 569 on a pair of variables, then it is considered to be a constant- $\mathbf{0}$ binary cost function. Note
 570 that the unary cost functions in a CFN that satisfies the JWP can be arbitrary. It has been
 571 shown that there is a close link between the JWP and M^\sharp -convex functions studied in [92].
 572 Indeed, a function satisfying the JWP can be transformed into a function represented as the
 573 sum of two quadratic M -convex functions, which can be minimized in polynomial time via an
 574 M -convex intersection algorithm. This leads to a larger tractable class of binary finite-valued
 575 CFNs, which properly contains the JWP class [59].

576 **9 Solvers and applications**

577 Thanks to a simple algebraic formulation of graphical models and queries over GMs, this
 578 review covers a very large variety of algorithms and approaches that have been developed

579 in probabilistic reasoning, propositional logic and constraint programming. On the non
580 stochastic side, SAT and CSP solvers have had a significant impact in several areas such
581 as Electronic Design Automation (where SAT solvers are used as oracles to solve complex
582 Bounded Model Checking problems), Software Verification (where suitable Abstraction
583 Refinement Strategies are used to manage possibly very large instances [27, 58]) or for task
584 scheduling [8] or planning [116] among many others. Stochastic variants are one of many
585 modeling tools available in Machine Learning [16], but Markov Random Fields have been
586 more specifically applied in Image processing for tasks such as segmentation [84]. In most
587 Image Processing applications, a variable of the GM is associated with a single pixel, leading
588 to very large MAP/MRF optimization problems that are never solved exactly (with a proof
589 or theoretical guarantee), using instead heuristic or primal/dual approaches offering a final
590 optimality gap [68]. However, the class of submodular problems with Boolean domains,
591 solvable in polytime using a min-cut (max-flow) algorithm has been exploited intensely under
592 the name of the “Graph-Cut” algorithm [76, 122].

593 In this final section, we rapidly describe recent results we obtained using an exact
594 WCSP/CFN solver. Toulbar2 implements most of all the algorithms we have described above
595 to solve the WCSP/CFN problem, which is essentially equivalent (up to a $-\log$ transform
596 and bounded precision description of real numbers) to MAP/MRF. Toulbar2 is becoming
597 increasingly famous for its ability to solve “Computational Protein Design” problem instances.
598 Proteins are linear polymers made of elementary bricks called amino acids. Each amino
599 acid a in a protein can be chosen among a fixed alphabet of 20 natural amino acids and
600 each choice in this alphabet is defined by the combination of a fixed “backbone” part and
601 a variable side-chain, a highly flexible part on the molecule. The computational protein
602 design problem starts from the three-dimensional description of a protein backbone and
603 asks to determine the nature and orientation of all side-chains that minimizes the energy of
604 the resulting molecule (a minimum of energy defining a most stable choice for a given rigid
605 backbone). The combination of dedicated pairwise separable (non-convex) energy functions
606 with a discretization of orientations in side chains (so-called rotamer libraries) allows to
607 reduce this problem to a pure WCSP/CFN problem. On such problems, Toulbar2 has been
608 used to bring to light the limitations of dedicated highly optimized Simulated annealing
609 implementations [112], being capable of providing guaranteed optimal solutions for problems
610 with search space larger than 400^{100} in reasonable time on a single core, on problems that
611 cannot be tackled by state-of-the-art ILP, Weighted MaxSAT or quadratic boolean solvers,
612 even those based on powerful SDP-based bounds [2]. The instances are not permuted
613 submodular [109] and do not have a sparse graph that would make dynamic programming
614 feasible. This has enabled the design of a real new self-assembling protein [94].

615 Conclusion

616 Algorithms for processing graphical models have been explored mostly independently in
617 the stochastic and deterministic cases. In this review, we have tried, using simple algebra,
618 to cover a large set of modeling frameworks with associated NP-hard queries and showed
619 that there is a strong common core for several algorithms in either case: non serial dynamic
620 programming. Restricted to sub-problems, combined with tree-search, learned heuristics,
621 and other tricks, these bricks have generated impressive algorithmic progress in the exact
622 solving of SAT/PL, CSP/CN, WCSP/CFN aka MAP/MRF or MPE/BN problems and even
623 for #P-complete problems [26, 50].

References

-
- 625 1 Srinivas M Aji and Robert J McEliece. The generalized distributive law. *IEEE transactions*
626 *on Information Theory*, 46(2):325–343, 2000.
- 627 2 David Allouche, Isabelle André, Sophie Barbe, Jessica Davies, Simon de Givry, George Kat-
628 sirelos, Barry O’Sullivan, Steve Prestwich, Thomas Schiex, and Seydou Traoré. Computational
629 protein design as an optimization problem. *Artificial Intelligence*, 212:59–79, 2014.
- 630 3 David Allouche, Christian Bessiere, Patrice Boizumault, Simon De Givry, Patricia Gutierrez,
631 Jimmy HM Lee, Ka Lun Leung, Samir Loudni, Jean-Philippe Métivier, Thomas Schiex, and
632 Yi Wu. Tractability-preserving transformations of global cost functions. *Artificial Intelligence*,
633 238:166–189, 2016.
- 634 4 David Allouche, Simon De Givry, George Katsirelos, Thomas Schiex, and Matthias Zytnicki.
635 Anytime hybrid best-first search with tree decomposition for weighted CSP. In *Principles and*
636 *Practice of Constraint Programming*, pages 12–29. Springer, 2015.
- 637 5 Albert Atserias, Andrei Bulatov, and Victor Dalmau. On the power of k -consistency. In
638 *International Colloquium on Automata, Languages, and Programming*, pages 279–290. Springer,
639 2007.
- 640 6 Albert Atserias, Johannes Klaus Fichte, and Marc Thurley. Clause-learning algorithms with
641 many restarts and bounded-width resolution. *Journal of Artificial Intelligence Research*,
642 40:353–373, 2011.
- 643 7 Fahiem Bacchus and Adam Grove. Graphical models for preference and utility. In *Proceedings of*
644 *the Eleventh conference on Uncertainty in artificial intelligence*, pages 3–10. Morgan Kaufmann
645 Publishers Inc., 1995.
- 646 8 Philippe Baptiste, Claude Le Pape, and Wim Nuijten. *Constraint-based scheduling: applying*
647 *constraint programming to scheduling problems*, volume 39. Springer Science & Business Media,
648 2012.
- 649 9 Roberto Bayardo and Daniel Miranker. On the space-time trade-off in solving constraint
650 satisfaction problems. In *Proc. of the 14th IJCAI*, pages 558–562, Montréal, Canada, 1995.
- 651 10 Claude Berrou, Alain Glavieux, and Punya Thitimajshima. Near shannon limit error-correcting
652 coding and decoding: Turbo-codes. 1. In *Proceedings of ICC’93-IEEE International Conference*
653 *on Communications*, volume 2, pages 1064–1070. IEEE, 1993.
- 654 11 Umberto Bertelé and Francesco Brioschi. *Nonserial Dynamic Programming*. Academic Press,
655 1972.
- 656 12 C. Bessière and J-C. Régin. Refining the basic constraint propagation algorithm. In *Proc.*
657 *IJCAI’2001*, pages 309–315, 2001.
- 658 13 Christian Bessière. Arc-consistency and arc-consistency again. *Artificial Intelligence*, 65:179–
659 190, 1994.
- 660 14 Armin Biere, Marijn Heule, and Hans van Maaren, editors. *Handbook of Satisfiability*, volume
661 185. IOS press, 2009.
- 662 15 Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh. Conflict-driven clause learning
663 sat solvers. *Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications*,
664 pages 131–153, 2009.
- 665 16 Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- 666 17 Stefano Bistarelli, Ugo Montanari, and Francesca Rossi. Semiring-based constraint satisfaction
667 and optimization. *Journal of the ACM (JACM)*, 44(2):201–236, 1997.
- 668 18 H L Bodlaender and A M C A Koster. Treewidth Computations I. Upper Bounds. Technical
669 Report UU-CS-2008-032, Utrecht University, Department of Information and Computing
670 Sciences, Utrecht, The Netherlands, September 2008. URL: <http://www.cs.uu.nl/research/techreps/repo/CS-2008/2008-032.pdf>.
- 671
- 672 19 Hans L Bodlaender. A partial k -arboretum of graphs with bounded treewidth. *Theoretical*
673 *computer science*, 209(1-2):1–45, 1998.
- 674 20 E. Boros and P. Hammer. Pseudo-Boolean Optimization. *Discrete Appl. Math.*, 123:155–225,
675 2002.

- 676 21 Frédéric Boussemart, Fred Hemery, Christophe Lecoutre, and Lakhdar Sais. Boosting system-
677 atic search by weighting constraints. In *ECAI*, volume 16, page 146, 2004.
- 678 22 Andrei A. Bulatov. A dichotomy theorem for nonuniform csps. In Umans [115], pages 319–330.
679 URL: <https://doi.org/10.1109/FOCS.2017.37>, doi:10.1109/FOCS.2017.37.
- 680 23 Andrei A. Bulatov. A short story of the CSP dichotomy conjecture. In *34th Annual ACM/IEEE*
681 *Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-*
682 *27, 2019*, page 1. IEEE, 2019. URL: <https://doi.org/10.1109/LICS.2019.8785678>, doi:
683 10.1109/LICS.2019.8785678.
- 684 24 Ümit Çatalyürek and Cevdet Aykanat. Patch (partitioning tool for hypergraphs). *Encyclopedia*
685 *of Parallel Computing*, pages 1479–1487, 2011.
- 686 25 Deeparnab Chakrabarty, Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. Subquadratic
687 submodular function minimization. In Hamed Hatami, Pierre McKenzie, and Valerie King,
688 editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing,*
689 *STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 1220–1231. ACM, 2017. URL:
690 <https://doi.org/10.1145/3055399.3055419>, doi:10.1145/3055399.3055419.
- 691 26 Supratik Chakraborty, Kuldeep S Meel, and Moshe Y Vardi. A scalable approximate model
692 counter. In *International Conference on Principles and Practice of Constraint Programming,*
693 pages 200–216. Springer, 2013.
- 694 27 Edmund Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-
695 guided abstraction refinement. In *International Conference on Computer Aided Verification,*
696 pages 154–169. Springer, 2000.
- 697 28 David A. Cohen, Martin C. Cooper, and Peter Jeavons. Generalising submodularity and horn
698 clauses: Tractable optimization problems defined by tournament pair multimorphisms. *Theor.*
699 *Comput. Sci.*, 401(1-3):36–51, 2008. URL: <https://doi.org/10.1016/j.tcs.2008.03.015>,
700 doi:10.1016/j.tcs.2008.03.015.
- 701 29 M. Cooper, S. de Givry, M. Sanchez, T. Schiex, M. Zytnicki, and T. Werner. Soft arc
702 consistency revisited. *Artificial Intelligence*, 174:449–478, 2010.
- 703 30 M C. Cooper. An optimal k -consistency algorithm. *Artificial Intelligence*, 41:89–95, 1989.
- 704 31 M C. Cooper. High-order consistency in Valued Constraint Satisfaction. *Constraints*, 10:283–
705 305, 2005.
- 706 32 M C. Cooper, S. de Givry, and T. Schiex. Optimal soft arc consistency. In *Proc. of IJCAI’2007,*
707 pages 68–73, Hyderabad, India, January 2007.
- 708 33 Martin Cooper and Thomas Schiex. Arc consistency for soft constraints. *Artificial Intelligence*,
709 154(1-2):199–227, 2004.
- 710 34 Martin C Cooper, Simon de Givry, Martí Sánchez, Thomas Schiex, and Matthias Zytnicki.
711 Virtual Arc Consistency for Weighted CSP. In *AAAI*, volume 8, pages 253–258, 2008.
- 712 35 Martin C. Cooper and Stanislav Zivny. Hybrid tractability of valued constraint problems. *Artif.*
713 *Intell.*, 175(9-10):1555–1569, 2011. URL: <https://doi.org/10.1016/j.artint.2011.02.003>,
714 doi:10.1016/j.artint.2011.02.003.
- 715 36 Martin C. Cooper and Stanislav Zivny. Hybrid tractable classes of constraint problems. In
716 Krokhn and Zivny [79], pages 113–135. URL: <https://doi.org/10.4230/DFU.Vol17.15301.4>,
717 doi:10.4230/DFU.Vol17.15301.4.
- 718 37 Adnan Darwiche. Recursive conditioning. *Artificial Intelligence*, 126(1-2):5–41, 2001.
- 719 38 Adnan Darwiche and Pierre Marquis. Compiling propositional weighted bases. *Artificial*
720 *Intelligence*, 157(1-2):81–113, 2004.
- 721 39 Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-
722 proving. *Communications of the ACM*, 5(7):394–397, 1962.
- 723 40 Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal*
724 *of the ACM (JACM)*, 7(3):201–215, 1960.
- 725 41 S. de Givry, T. Schiex, and G. Verfaillie. Exploiting Tree Decomposition and Soft Local
726 Consistency in Weighted CSP. In *Proc. of the National Conference on Artificial Intelligence,*
727 *AAAI-2006*, pages 22–27, 2006.

- 728 42 Rina Dechter. Learning while searching in constraint satisfaction problems. In *Proc. of*
729 *AAAI'86*, pages 178–183, Philadelphia, PA, 1986.
- 730 43 Rina Dechter. Mini-buckets: A general scheme for generating approximations in automated
731 reasoning. In *IJCAI*, pages 1297–1303, 1997.
- 732 44 Rina Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*,
733 113(1–2):41–85, 1999.
- 734 45 Rina Dechter and Irina Rish. Directional resolution: The Davis-Putnam procedure, revisited.
735 *KR*, 94:134–145, 1994.
- 736 46 Rina Dechter and Irina Rish. Mini-buckets: A general scheme for bounded inference. *Journal*
737 *of the ACM (JACM)*, 50(2):107–153, 2003.
- 738 47 C. Domshlak, F. Rossi, KB Venable, and T. Walsh. Reasoning about soft constraints and
739 conditional preferences: complexity results and approximation techniques. In *Proc. of the 18th*
740 *IJCAI*, pages 215–220, Acapulco, Mexico, 2003.
- 741 48 Didier Dubois, H elene Fargier, and Henri Prade. The calculus of fuzzy restrictions as a basis
742 for flexible constraint satisfaction. In *Second IEEE International Conference on Fuzzy Systems*,
743 pages 1131–1136. IEEE, 1993.
- 744 49 Didier Dubois, H elene Fargier, and Henri Prade. Propagation and satisfaction of fuzzy
745 constraints. In Yager R.R. and Zadeh L.A., editors, *Fuzzy Sets, Neural Networks and Soft*
746 *Computing*, pages 166–187. Kluwer Acad., 1993.
- 747 50 Stefano Ermon, Carla Gomes, Ashish Sabharwal, and Bart Selman. Taming the curse of
748 dimensionality: Discrete integration by hashing and optimization. In *International Conference*
749 *on Machine Learning*, pages 334–342, 2013.
- 750 51 H elene Fargier, Pierre Marquis, Alexandre Niveau, and Nicolas Schmidt. A knowledge
751 compilation map for ordered real-valued decision diagrams. In *Twenty-Eighth AAAI Conference*
752 *on Artificial Intelligence*, 2014.
- 753 52 Tom as Feder and Moshe Y. Vardi. The computational structure of monotone monadic
754 SNP and constraint satisfaction: A study through datalog and group theory. *SIAM J.*
755 *Comput.*, 28(1):57–104, 1998. URL: <https://doi.org/10.1137/S0097539794266766>, doi:
756 10.1137/S0097539794266766.
- 757 53 Eugene C. Freuder and Michael J. Quinn. Taking advantage of stable sets of variables in
758 constraint satisfaction problems. In *Proc. of the 9th IJCAI*, pages 1076–1078, Los Angeles,
759 CA, 1985.
- 760 54 Alan George. Nested dissection of a regular finite element mesh. *SIAM Journal on Numerical*
761 *Analysis*, 10(2):345–363, 1973.
- 762 55 M. Gondran and M. Minoux. *Graphes et algorithmes*. Lavoisier, EDF R&D, 2009.
- 763 56 Michel Gondran and Michel Minoux. *Graphs, dioids and semirings: new models and algorithms*,
764 volume 41. Springer Science & Business Media, 2008.
- 765 57 Stefan Haller, Paul Swoboda, and Bogdan Savchynskyy. Exact map-inference by confining
766 combinatorial search with lp relaxation. In *Thirty-Second AAAI Conference on Artificial*
767 *Intelligence*, 2018.
- 768 58 Marijn JH Heule and Oliver Kullmann. The science of brute force. *Commun. ACM*, 60(8):70–79,
769 2017.
- 770 59 Hiroshi Hirai, Yuni Iwamasa, Kazuo Murota, and Stanislav Zivny. A tractable class of binary
771 VCSPs via M-convex intersection. *ACM Trans. Algorithms*, 15(3):44:1–44:41, 2019. URL:
772 <https://doi.org/10.1145/3329862>, doi:10.1145/3329862.
- 773 60 Barry Hurley, Barry O’Sullivan, David Allouche, George Katsirelos, Thomas Schiex, Matthias
774 Zytnicki, and Simon de Givry. Multi-language evaluation of exact solvers in graphical model
775 discrete optimization. *Constraints*, pages 1–22, 2016.
- 776 61 Peter Jeavons and Martin C. Cooper. Tractable constraints on ordered domains. *Artif.*
777 *Intell.*, 79(2):327–339, 1995. URL: [https://doi.org/10.1016/0004-3702\(95\)00107-7](https://doi.org/10.1016/0004-3702(95)00107-7), doi:
778 10.1016/0004-3702(95)00107-7.

- 779 62 Peter Jeavons and Justyna Petke. Local consistency and SAT-solvers. *Journal of Artificial*
780 *Intelligence Research*, 43:329–351, 2012.
- 781 63 Philippe Jégou, Hanan Kanso, and Cyril Terrioux. Towards a dynamic decomposition of CSPs
782 with separators of bounded size. In *International Conference on Principles and Practice of*
783 *Constraint Programming*, pages 298–315. Springer, 2016.
- 784 64 Philippe Jégou, Hélène Kanso, and Cyril Terrioux. On the relevance of optimal tree decomposi-
785 tions for constraint networks. In *2018 IEEE 30th International Conference on Tools with*
786 *Artificial Intelligence (ICTAI)*, pages 738–743. IEEE, 2018.
- 787 65 Philippe Jégou, Samba Ndojh Ndiaye, and Cyril Terrioux. A new evaluation of forward
788 checking and its consequences on efficiency of tools for decomposition of CSPs. In *2008 20th*
789 *IEEE International Conference on Tools with Artificial Intelligence*, volume 1, pages 486–490.
790 IEEE, 2008.
- 791 66 Philippe Jégou and Cyril Terrioux. Hybrid backtracking bounded by tree-decomposition of
792 constraint networks. *Artificial Intelligence*, 146(1):43–75, 2003.
- 793 67 Philippe Jégou and Cyril Terrioux. Decomposition and good recording for solving Max-CSPs.
794 In *Proceedings of the 16th European Conference on Artificial Intelligence, IOS Press*, pages
795 196–200, 2004.
- 796 68 Joerg Kappes, Bjoern Andres, Fred Hamprecht, Christoph Schnorr, Sebastian Nowozin,
797 Dhruv Batra, Sungwoong Kim, Bernhard Kausler, Jan Lellmann, Nikos Komodakis, et al. A
798 comparative study of modern inference techniques for discrete energy minimization problems.
799 In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages
800 1328–1335, 2013.
- 801 69 George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning
802 irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.
- 803 70 Kaley Kask, Andrew Gelfand, Lars Otten, and Rina Dechter. Pushing the power of stochastic
804 greedy ordering schemes for inference in graphical models. In *Twenty-Fifth AAAI Conference*
805 *on Artificial Intelligence*, 2011.
- 806 71 Ross Kindermann. *Markov random fields and their applications*. American Mathematical
807 Society, 1980.
- 808 72 E.P. Klement, R. Mesiar, and E. Pap. *Triangular Norms*. Kluwer Academic Publishers, 2000.
- 809 73 Juerg Kohlas and Nic Wilson. Semiring induced valuation algebras: Exact and approximate
810 local computation algorithms. *Artificial Intelligence*, 172(11):1360–1399, 2008.
- 811 74 Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*.
812 MIT press, 2009.
- 813 75 Vladimir Kolmogorov. Convergent tree-reweighted message passing for energy minimization.
814 *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(10):1568–1583, 2006.
- 815 76 Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimized via graph
816 cuts? *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 2:147–159, 2004.
- 817 77 V.K. Koval and M.I. Schlesinger. Dvumernoe programmirovaniye v zadachakh analiza izo-
818 brazheniy (two-dimensional programming in image analysis problems),. *USSR Academy of*
819 *Science, Automatics and Telemechanics*, 8:149–168, 1976.
- 820 78 Andrei A. Krokhin and Stanislav Zivny. The complexity of valued CSPs. In *The Constraint*
821 *Satisfaction Problem: Complexity and Approximability* [79], pages 233–266. URL: <https://doi.org/10.4230/DFU.Vol17.15301.9>, doi:10.4230/DFU.Vol17.15301.9.
- 822 79 Andrei A. Krokhin and Stanislav Zivny, editors. *The Constraint Satisfaction Problem: Complex-*
823 *ity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*. Schloss Dagstuhl - Leibniz-Zentrum
824 fuer Informatik, 2017. URL: <http://www.dagstuhl.de/dagpub/978-3-95977-003-3>.
- 825 80 Jean-Marie Lagniez and Pierre Marquis. An improved decision-DNNF compiler. In *IJCAI*,
826 pages 667–673, 2017.
- 827 81 J. Larrosa, S. de Givry, F. Heras, and M. Zytnicki. Existential arc consistency: getting closer
828 to full arc consistency in weighted CSPs. In *Proc. of the 19th IJCAI*, pages 84–89, Edinburgh,
829 Scotland, August 2005.
- 830

- 831 **82** Jimmy Ho-Man Lee and Ka Lun Leung. Consistency techniques for flow-based projection-safe
832 global cost functions in weighted constraint satisfaction. *Journal of Artificial Intelligence*
833 *Research*, 43(1):257–292, 2012.
- 834 **83** Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A faster cutting plane method and its
835 implications for combinatorial and convex optimization. In *2015 IEEE 56th Annual Symposium*
836 *on Foundations of Computer Science*, pages 1049–1065. IEEE, 2015.
- 837 **84** Stan Z Li. *Markov random field modeling in image analysis*. Springer Science & Business
838 Media, 2009.
- 839 **85** R. Marinescu and R. Dechter. AND/OR branch-and-bound for graphical models. In *Proc. of*
840 *the 19th IJCAI*, page 224, Edinburgh, Scotland, 2005.
- 841 **86** Radu Marinescu and Rina Dechter. Memory intensive branch-and-bound search for graphical
842 models. In *Proc. of AAAI’06*, page 1200. Menlo Park, CA; Cambridge, MA; London; AAAI
843 Press; MIT Press; 1999, 2006.
- 844 **87** Radu Marinescu and Rina Dechter. And/or branch-and-bound search for combinatorial
845 optimization in graphical models. *Artificial Intelligence*, 173(16-17):1457–1491, 2009.
- 846 **88** Radu Marinescu and Rina Dechter. Memory intensive and/or search for combinatorial
847 optimization in graphical models. *Artificial Intelligence*, 173(16-17):1492–1524, 2009.
- 848 **89** P. Meseguer, F. Rossi, and T. Schiex. Soft constraints processing. In F. Rossi, P. van Beek,
849 and T. Walsh, editors, *Handbook of Constraint Programming*, chapter 9. Elsevier, 2006.
- 850 **90** R. Mohr and T.C. Henderson. Arc and path consistency revisited. *Artificial Intelligence*,
851 28(2):225–233, 1986.
- 852 **91** Matthew W Moskewicz, Conor F Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik.
853 Chaff: Engineering an efficient sat solver. In *Proceedings of the 38th annual Design Automation*
854 *Conference*, pages 530–535. ACM, 2001.
- 855 **92** Kazuo Murota. *Discrete Convex Analysis*. SIAM, 2003.
- 856 **93** Jaroslav Nešetřil and Patrice Ossona De Mendez. Tree-depth, subgraph coloring and homo-
857 morphism bounds. *European Journal of Combinatorics*, 27(6):1022–1041, 2006.
- 858 **94** Hiroki Noguchi, Christine Addy, David Simoncini, Staf Wouters, Bram Mylemans, Luc
859 Van Meervelt, Thomas Schiex, Kam YJ Zhang, Jeremy RH Tame, and Arnout RD Voet.
860 Computational design of symmetrical eight-bladed β -propeller proteins. *IUCrJ*, 6(1), 2019.
- 861 **95** Abdelkader Ouali, David Allouche, Simon de Givry, Samir Loudni, Yahia Lebbah, Lakhdar
862 Loukil, and Patrice Boizumault. Variable neighborhood search for graphical model energy
863 minimization. *Artificial Intelligence*, 278:103194, 2020. doi:[https://doi.org/10.1016/j.
864 artint.2019.103194](https://doi.org/10.1016/j.artint.2019.103194).
- 865 **96** Youngsuk Park, David Hallac, Stephen Boyd, and Jure Leskovec. Learning the network
866 structure of heterogeneous data via pairwise exponential Markov random fields. *Proceedings*
867 *of machine learning research*, 54:1302, 2017.
- 868 **97** Judea Pearl. *Probabilistic Reasoning in Intelligent Systems, Networks of Plausible Inference*.
869 Morgan Kaufmann, Palo Alto, 1988.
- 870 **98** Daniel Prusa and Tomas Werner. Universality of the local marginal polytope. In *Proceedings*
871 *of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1738–1743, 2013.
- 872 **99** Neil Robertson and Paul D. Seymour. Graph minors. X. Obstructions to tree-decomposition. *J.*
873 *Comb. Theory, Ser. B*, 52(2):153–190, 1991. URL: [https://doi.org/10.1016/0095-8956\(91\)
874 90061-N](https://doi.org/10.1016/0095-8956(91)90061-N), doi:10.1016/0095-8956(91)90061-N.
- 875 **100** J. Alan Robinson. A machine-oriented logic based on the resolution principle. *Journal of the*
876 *ACM*, 12:23–44, 1965.
- 877 **101** D.J. Rose. Tringulated graphs and the elimination process. *Journal of Mathematical Analysis*
878 *and its Applications*, 32, 1970.
- 879 **102** F. Rossi, P. van Beek, and T. Walsh, editors. *Handbook of Constraint Programming*. Elsevier,
880 2006.
- 881 **103** M. Sanchez, D. Allouche, S. de Givry, and T. Schiex. Russian doll search with tree decomposi-
882 tion. In *Proc. IJCAI’09*, pages 603–608, San Diego (CA), USA, 2009.

- 883 104 T. Schiex. Arc consistency for soft constraints. In *Principles and Practice of Constraint*
884 *Programming - CP 2000*, volume 1894 of *LNCS*, pages 411–424, Singapore, September 2000.
- 885 105 T. Schiex, H. Fargier, and G. Verfaillie. Valued constraint satisfaction problems: hard and
886 easy problems. In *Proc. of the 14th IJCAI*, pages 631–637, Montréal, Canada, August 1995.
- 887 106 Thomas Schiex. A note on csp graph parameters. Technical report, Citeseer, 1999.
- 888 107 Thomas Schiex and Gérard Verfaillie. Nogood recording for static and dynamic constraint
889 satisfaction problems. *International Journal on Artificial Intelligence Tools*, 3(02):187–207,
890 1994.
- 891 108 D. Schlesinger. Exact Solution of Permuted Submodular MinSum Problems. In *Energy*
892 *Minimization Methods in Computer Vision and Pattern Recognition*, number 4679/2007 in
893 *LNCS*, pages 28–38, August 2007.
- 894 109 Dmitrij Schlesinger. Exact solution of permuted submodular minsum problems. In *International*
895 *Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*,
896 pages 28–38. Springer, 2007.
- 897 110 M.I. Schlesinger. Sintaksicheskiy analiz dvumernykh zritelnykh signalov v usloviyakh pomekh
898 (Syntactic analysis of two-dimensional visual signals in noisy conditions). *Kibernetika*, 4:113–
899 130, 1976.
- 900 111 G. Shafer. An axiomatic study of computation in hypertrees. Working paper 232, University
901 of Kansas, School of Business, Lawrence, 1991.
- 902 112 David Simoncini, David Allouche, Simon de Givry, Céline Delmas, Sophie Barbe, and Thomas
903 Schiex. Guaranteed discrete energy optimization on large protein design problems. *Journal of*
904 *Chemical Theory and Computation*, 11(12):5980–5989, 2015. doi:10.1021/acs.jctc.5b00594.
- 905 113 C. Terrioux and P. Jegou. Bounded backtracking for the valued constraint satisfaction
906 problems. In *Proc. of the Ninth International Conference on Principles and Practice of*
907 *Constraint Programming (CP-2003)*, 2003.
- 908 114 Johan Thapper and Stanislav Živný. The complexity of finite-valued CSPs. *J. ACM*, 63(4):37:1–
909 37:33, 2016. URL: <https://doi.org/10.1145/2974019>, doi:10.1145/2974019.
- 910 115 Chris Umans, editor. *58th IEEE Annual Symposium on Foundations of Computer Science,*
911 *FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*. IEEE Computer Society, 2017. URL:
912 <https://ieeexplore.ieee.org/xpl/conhome/8100284/proceeding>.
- 913 116 Peter Van Beek and Xinguang Chen. Cplan: A constraint programming approach to planning.
914 In *AAAI/IAAI*, pages 585–590, 1999.
- 915 117 Martin Wainwright, Tommi Jaakkola, and Alan Willsky. Tree-based reparameterization
916 analysis of belief propagation and related algorithms for approximate inference on graphs
917 with cycles. In *Proceedings IEEE International Symposium on Information Theory*, page 113.
918 IEEE, 2002.
- 919 118 T. Werner. A Linear Programming Approach to Max-sum Problem: A Review. *IEEE*
920 *Trans. on Pattern Recognition and Machine Intelligence*, 29(7):1165–1179, July 2007. URL:
921 <http://dx.doi.org/10.1109/TPAMI.2007.1036>.
- 922 119 Jonathan S Yedidia, William T Freeman, and Yair Weiss. Bethe free energy, Kikuchi ap-
923 proximations, and belief propagation algorithms. *Advances in neural information processing*
924 *systems*, 13, 2001.
- 925 120 Yuanlin Zhang and Roland HC Yap. Making AC-3 an optimal algorithm. In *IJCAI*, volume 1,
926 pages 316–321, 2001.
- 927 121 Dmitriy Zhuk. A proof of CSP dichotomy conjecture. In Umans [115], pages 331–342. URL:
928 <https://doi.org/10.1109/FOCS.2017.38>, doi:10.1109/FOCS.2017.38.
- 929 122 Stanislav Živný and Peter G Jeavons. Classes of submodular constraints expressible by graph
930 cuts. *Constraints*, 15(3):430–452, 2010.