Annals of Mathematics and Artificial Intelligence (2020) 88:907–949
https://doi.org/10.1007/s10472-020-09698-z

Leveraging cluster backbones for improving MAP inference in statistical relational models

Mohamed-Hamza Ibrahim[1,2] · **Christopher Pal**[1] · **Gilles Pesant**[1]

Published online: 7 May 2020

Abstract

A wide range of important problems in machine learning, expert system, social network analysis, bioinformatics and information theory can be formulated as a maximum a-posteriori (MAP) inference problem on statistical relational models. While off-the-shelf inference algorithms that are based on local search and message-passing may provide adequate solutions in some situations, they frequently give poor results when faced with models that possess high-density networks. Unfortunately, these situations always occur in models of real-world applications. As such, accurate and scalable maximum a-posteriori (MAP) inference on such models often remains a key challenge. In this paper, we first introduce a novel family of extended factor graphs that are parameterized by a smoothing parameter $\chi \in [0, 1]$. Applying belief propagation (BP) message-passing to this family formulates a new family of **W**eighted **S**urvey **P**ropagation algorithms (WSP-χ) applicable to relational domains. Unlike off-the-shelf inference algorithms, WSP-χ detects the "backbone" ground atoms in a solution cluster that involve potentially optimal MAP solutions: the cluster backbone atoms are not only portions of the optimal solutions, but they also can be exploited for scaling MAP inference by iteratively fixing them to reduce the complex parts until the network is simplified into one that can be solved accurately using any conventional MAP inference method. We also propose a lazy variant of this WSP-χ family of algorithms. Our experiments on several real-world problems show the efficiency of WSP-χ and its lazy variants over existing prominent MAP inference solvers such as MaxWalkSAT, RockIt, IPP, SP-Y and WCSP.

Keywords Maximum-a-posteriori inference · Markov logic network · Survey propagation

Mathematics Subject Classification (2010) 68T01 "General topics in artificial intelligence"

✉ Mohamed-Hamza Ibrahim
mohamed.ibrahim@polymtl.ca

Christopher Pal
christopher.pal@polymtl.ca

Gilles Pesant
gilles.pesant@polymtl.ca

[1] Department of Computer and Software Engineering, École Polytechnique de Montréal, Montréal, Québec, Canada

[2] Department of Mathematics, Faculty of Science, Zagazig University, Zagazig, Egypt

1 Introduction

Inspired by the ability of statistical relational learning (SRL) [17] to combine the power of first-order logic with probabilistic graphical models [46], Markov logic networks (MLN) [59] have recently emerged as a popular SRL framework and they have been actively explored for a wide variety of real-world applications in AI, particularly those that involve both relational and uncertain inference. Commonly in MLNs it has been convenient to convert formulas to conjunctive normal form (CNF) and to propositionalize the theory to a grounded network of clauses. Then, one often wishes to apply a probabilistic maximum a-posteriori (MAP) inference procedure for reasoning about uncertain queries. In such cases, the logical structure of the grounded network renders the MAP inference equivalent to solving a weighted MAX-SAT problem (i.e., where one finds the most probable truth assignment or "MAP solution" that maximizes the total weight of satisfied clauses) [52]. A simple approach to tackle MAP inference is to use off-the-shelf local search algorithms that are designed to efficiently solve weighted MAX-SAT instances. For example MaxWalkSAT [25, 66] was recently applied as a conventional MAP inference method in probabilistic graphical models, including those models instantiated by underlying MLNs [59].

In relational domains, there are often millions of atoms, each involved in hundreds of thousands of clauses, which are normally translated into grounded networks featuring high densities.[1] From the point of view of satisfiability, if the density of the grounded network is close to a satisfiability threshold, then the (MAP) assignments in the solution space are clustered.[2] This is known as a "clustering phenomenon" in statistical physics [41]. It has been shown that such clustering in the solution space exists not only for uniform random satisfiability, but also for some structured satisfiability instances that follow realistic and natural distributions [18, 20, 53, 76], similar to real-world problems that are modeled as MLNs (e.g., social networks, as shown in Kambhampati & Liu, 2013).

In general, using local search algorithms for MAP inference frequently has the following limitations due to the clustering of the solution space:

Limitation 1. Usually, the existence of many clusters is an indication of a rugged energy landscape, which then also gives rise to many local optima. This often hinders the performance of most local search algorithms because they can get stuck in a local optimum [45]. MaxWalkSAT (a stochastic local search algorithm) is one exception because it performs a combination of random and greedy steps which prevent it from getting stuck in a local optimum.

Limitation 2. Another possible consequence of the clustering of the solution space is that the search space fractures dramatically with a proliferation of metastable' clusters [9]. This acts as a dynamic trap for local search algorithms, including MaxWalkSAT, since they can get stuck in one such 'metastable' cluster at a local optimum [9, 30]. On the flip side of the issue, if the clauses capture complex structures (e.g., relational dependencies) then the clusters tend to decompose into an exponentially small fraction of (MAP) assignments [41]. So there is not even any guarantee of getting into a cluster that contains an optimal solution. Clearly, this all greatly weakens the possibility that a local search can converge to an optimal solution, particularly for SRL applications [60].

[1] The density is the ratio of ground clauses to ground atoms.

[2] A cluster of assignments is generally defined as a set of connected components of the solution space, where the assignments belonging to the same cluster are close to each other (e.g., in terms of Hamming distance) [41].

It is well known that techniques such as marginalization-decimation based on the max-product BP algorithm [72, 74], can be used to help local search find an optimal MAP solution, but it is also believed that BP does not converge due to strong attraction in many directions [28, 35]. That is, max-product BP fails because its local computations may obtain locally optimal assignments corresponding to different clusters, but these cannot be combined to find a global MAP solution [35].

Fortunately, it is likely that in each cluster there is a certain fraction of 'frozen' ground atoms [1] that are fixed in all (MAP) solutions within the cluster, while the others can be varied subject to some ground clauses. These frozen ground atoms are known as 'cluster backbones' [34]. In the world of satisfiability, survey propagation (SP) is a common heuristic method that can be efficiently used in a decimation procedure [35] to obtain cluster backbones when solving hard SAT instances [8]. For example, SP can obtain frozen variables, which are $45 - 50\%$ of all variables, without making any mistake [35].

The main objective of this paper is to bring SP's success and SRL closer together for the benefit of MAP inference. That is, to remedy the aforementioned limitations of local search algorithms, which are due to search space clustering, we introduce Weighted Survey Propagation-inspired Decimation (WSP-Dec), a family of message-passing algorithms for applying MAP inference to SRL models and Markov logic in particular. In this paper we focus on Markov logic. But, without difficulty, the WSP-Dec framework can be applied to other representations that are special cases of SRL models, including standard graphical models defined in terms of factor graphs. We achieve this as follows:

- First, we describe a novel family of extended factor graphs that are specified by the parameter $\chi \in [0, 1]$. These factor graphs are re-parameterized in such a way that they define positive joint probabilities over generalized complete assignments called max-cores. These max-cores are natural interpretations of core assignments [40] that satisfy a set of clauses with maximal weights.
- We then show that applying BP message-passing to this family recovers a family of Weighted SP algorithms (WSP-χ), ranging from pure Weighted SP (WSP-1) to standard max-product BP (WSP-0).
- The marginals computed by any WSP-χ algorithm can be used to obtain backbones (i.e., frozen ground atoms) of a cluster involving potentially optimal MAP solutions.
- The cluster backbones are not only a portion of the optimal solutions in the cluster, but they also can be used to enlarge the evidence database and shrink the query set. Therefore, iteratively fixing them results in a reduction of the complex parts of the grounded network, which afterwards can be simplified to a scalable one that is then solved accurately using any conventional MAP method. Hence, integrating WSP-χ as pre-processing with a MaxWalkSAT algorithm in a decimation procedure produces a family of Weighted Survey Propagation-Inspired decimation (WSP-Dec) algorithms for solving the task of full MAP inference on SRL models.
- We then explain lazy variants of WSP-χ family of algorithms.
- Our experiments on three real-world applications show the promise of WSP-Dec algorithms and their lazy variants as opposed to using MaxWalkSAT alone.

We have organized the rest of the paper in the following manner. In Section 2, we examine further work related to the MAP inference problem as well as survey propagation. In Section 3, we review the basic notations, Markov logic and SP technique. Then in Section 4, we demonstrate the family of WSP-χ algorithms. In Section 5, we explain how WSP-χ can be applied to improve MAP inference in Markov logic. In Section 6, we conduct a thorough

experimental study, which is followed by a discussion in Section 7. Finally, in Section 8, we present our conclusions and discuss directions for future research.

2 Related work

Developing accurate and scalable MAP inference in probabilistic graphical models (PGMs) is a key challenge that would affect a wide range of applications in machine learning, constraint satisfaction, information theory, computational biology, and other sub-disciplines of artificial intelligence. The Viterbi algorithm [16] was first proposed to address MAP inference — in hidden Markov models — for decoding applications. A generalization was then proposed to other types of PGMs in other applications [54], and extended by the appearance of a clique tree algorithm [37]. Since then, several pioneering research directions in AI have been explored for improving MAP inference.

The first research area uses the fact that MAP inference can be cast as the task of minimizing an energy function — in computer vision applications [71], the energy function defined over terms enforces some kind of spatial coherence; another one penalizes solutions that are inconsistent with the observed data. Under this line of analysis, hill-climbing methods such as iterated conditional modes (ICM) [6], or simulated annealing (SA) [19] become directly applicable to solve the MAP inference problem. However, in practice these algorithms were proven to be inefficient. For example, in computer vision applications [71], this is due to the slowdown that comes from the considerable amount of computation in their early adaptation.

Furthermore, when the PGM features logical structures like those instantiated as SRL models, finding the MAP solution of the model can be obtained by solving its weighted satisfiability problem using any satisfiability solver [21]. This in fact opens the way to the use of local search algorithms for MAP inference. For instance, MaxWalkSAT [25, 66] was proposed to tackle the MAP problem in MLNs [59]. However, local search algorithms are non-systematic, and they often come with no performance guarantees. Additionally, it was observed that if the density of the underlying graphical model was close to a certain threshold then the search space of MAP assignments would become clustered [18, 20, 24, 53, 76]. However, none of those approaches took into consideration the clustering that occurred in the search space when solving the MAP inference problem, and this makes them vulnerable to getting stuck in a local maximum at one of the metastable clusters [9, 30].

Our WSP-χ approach can serve as a pre-processing step to handle the clustered search space when solving the MAP problem. Thus it can be used in conjunction with all the above search algorithms to increase the possibility of guaranteeing optimality.

In a complementary context, another fruitful area of research is how to relate Boolean satisfiability and CSP methods to local search for MAP inference, most notably survey propagation [8]. The SP algorithm has shown a surprising ability to accurately solve extremely large and hard SAT instances close to the satisfiability threshold [8, 35]. In an attempt to more fully understand the success of SP, it was first shown that SP can be viewed as belief propagation (BP) on a factor graph defined over solution clusters represented by covers having positive probability [7]. Subsequently, SP has been generalized [40] to work on extended MRFs, where solution clusters can be elegantly represented as cores having positive probability. It has also been extended to an algorithm called SP-y [5] to handle the maximum satisfiability (Max-SAT) problem. Recently, relaxed SP [11] has been introduced as an extension to SP and SP-y for weighted satisfiability. However, the applicability

of these approaches to complex relational problems is still limited. Our WSP-χ approach can be seen as akin to these SP-based inference methods. But WSP-χ is different in being built on different re-parameterized extended factor graphs that are used to address relational MAP inference.

Another traditional research area involves the use of message-passing algorithms to tackle MAP inference. Here the mainstream interest is in max-product loopy BP. From a theoretical perspective, it was shown that the marginals obtained after the convergence of max-product BP can be used to derive a MAP solution that is globally optimum for an acyclic network, and locally optimum for a loopy network [74]. Accordingly, parameterized message passing [72] has been proposed to obtain beliefs corresponding to max-marginals. This work then empirically extended by developing a convexified message-passing algorithm for MAP inference called Tree-reweighted max-product message-passing (TRW) algorithm [73]. However, TRW does not monotonically increase its objective function and therefore may not converge to accurate results. As a remedy, a variant of TRW, called TRW-S [32] — which involved asynchronous message passing — has been proposed to improve the objective monotonically. However, TRW-S still cannot guarantee convergence to a global optimum since it can get stuck in local optima. From the optimization perspective, TRW and its variant TRW-S can be in fact considered the first connection between LP relaxation and message-passing algorithms. This stimulates some other works to extend TRW by using sophisticated LP relaxation methods. For instance, a general TRW-based approach [58] has been proposed to iteratively improve the solution by searching the nearest improvements using a proximal optimization method. However, it was subsequently proven that the use of a simple LP relaxation [36] would produce a better (i.e., tighter) relaxation than the complicated one that is based on a proximal optimization method [58]. Other work has extended TRW to Tree-Reweighted Belief Propagation (TRBP) that uses the marginals derived from a convex BP to obtain optimal MAP solutions [75].

At a high level, our WSP-χ approach shares with the previously mentioned approaches the derivation of a parameterized BP message passing procedure such that the obtained beliefs correspond to marginals. However, WSP-χ is different, since its marginals are defined over max-cores, which are representations of clusters of candidate MAP solutions. In addition, in WSP-χ, there is a cooling parameter y that plays a similar role to the temperature parameter in simulated annealing [19]. This cooling parameter can be tuned to reduce the strengths of factors. We show empirically how this greatly improves the convergence of WSP-χ on the associated extended factor graph. This gives WSP-χ different characteristics that enable it to avoid getting stuck in local optima and be more likely to convergence to a global optimum.

Another promising research area that has been recently explored seeks to improve the scalability of inference by using a restricted class of probabilistic graphical architectures that compromise between expressiveness and tractability. For example, the sum-product network (SPN) [56] was initially proposed for tractable marginal inference: In its basic form, SPN is not a conditional dependency graph, but a graphical representation for computing the partition function. Later on, [47] developed a relational formalization of SPNs (i.e., RSPNs) that is applicable to SRL models. One limitation of RSPN is that the relational atoms can not be modeled directly, but through aggregations. This often requires the user to know and specify the fixed parts for decomposing the training and testing mega-examples [47], which may be difficult in the problems with complicated decomposing structures (e.g., Hyperlink analysis application) — This is in fact consistent with our experimental evaluations on WebKB dataset in Section 6.3.4. Apart from marginal inference, [55] demonstrated that the

tractable MAP inference without hidden variables is a form of max probable explanation (MPE) on selective SPNs. However, it is widely known that many SPN learning algorithms can not guarantee selectivity [61] and without selectivity the MPE inference is still is NP-hard [55]. Subsequently, [42] proposed an efficient MAX inference method in non-selective SPNs — MAX is a simpler form of MAP inference problem under the assumption that neither evidence nor hidden variables exist.

Our WSP-χ approach can not be directly applied on SPNs since its current form is adapted to factor graph as well as MLN. While the ground RSPN can be converted into an equivalent factor graph in linear time, the resulting graph model may blow up exponentially and might have a high tree-width, which could yield a computationally intractable MAP inference.

Another promising research area seeks to improve the scalability of MAP inference on large probabilistic networks. Here, mainstream work attempts to take advantage of some local structural properties in the network like determinism [22, 51], symmetry [2], sparseness [57], and type hierarchy [29] to scale inference. For instance, Lifted Inference uses the symmetry present in the network's structure to shrink its size. The main idea is to deal with groups of indistinguishable variables rather than individual variables. FOVE-P [15] was the first to propose lifted variable elimination, which is a modification of FOVE [14] that incorporates partial inversion for lifting MAP inference in SRL models. Recently, a lifted MaxWalkSAT algorithm [62] has been proposed for MAP inference in Markov logic networks (MLNs). Subsequently, a generalization of lifted MaxWalkSAT [63] was introduced for lifting MAP inference in MLNs. However, we could repeal the merit of lifting inference if symmetries break when the variables become correlated by the virtue of depending asymmetrically on evidence [27], since in this case lifted inference becomes close to propositionalized inference. As such, other extensions to liftable first-order MLN models had been introduced. For instance, [26] proposed a top-down liftable algorithm with a greatly computational complexity better than propositional algorithms. However, a recent study [39] showed that the algorithm can not handle the symmetries break if the models are highly connected and contain a massive amount of evidence axioms and variables (which is often the case in real-world applications). Lazy Inference is another efficient way to scale MAP inference. Lazy MaxWalkSAT [68] was proposed as the first lazy algorithm for MAP inference in Markov logic networks, where the sparseness was used to ground the theory lazily to reduce memory and time consumption. Other work proposed a Cutting Plane Inference (CPI) [60] as an efficient and accurate algorithm for MAP inference. A concrete example is RockIt [50], one of the most prominent bottom-up inference engines that shows powerful results in several recent applications like ensemble matching [43] and cause analysis [65]. The Xeggora method [4] was recently proposed as an evolutionary extension of RockIt to exploit further symmetries (that were ignored in RockIt) by performing higher order aggregations. However, the virtue of higher order aggregations appears only in models with lengthy formulas, which rarely exist in real-life applications. In addition, although CPI-based methods avoid grounding the whole MLN, they only work well for some types of MLNs where the separation step returns a small set of constraints.

Our WSP-χ approach is different from the above scaling algorithms, since it is based on the use of backbones (or frozen variables) to fix complex parts of the network by enlarging the evidence database and reducing the set of queries, and thereby simplifying the network. But it is also orthogonal to their benefits, and thus can be combined with them (as it will be shown in Section 6, where we combine WSP-χ with lazy inference).

3 Background

To set the stage for our work, in this section we first express our notation and basic concepts using an explanatory example presented in Table 1.

3.1 Notation and definitions

A first-order knowledge base (KB) is a set of formulas in first-order logic, typically in conjunctive normal form (CNF). After propositional grounding, we get a formula $\mathcal{F}$ which is a conjunction of m ground clauses. We use $f_i \in \mathcal{F}$ to denote a ground clause which is a disjunction of literals built from $\mathcal{X} = \{X_1, \cdots, X_n\}$, a set of n Boolean random variables representing ground atoms. For instance, and as represented in Table 1, the CNF $\mathcal{F}$ is the conjunction of the grounded clauses $\{f_a, f_b, f_c, f_d\}$ which are defined on ground atoms $\{X_1, X_2, X_3\}$. We call an assignment of truth values to the n variables in $\mathcal{X}$ a complete assignment $x \in \{0, 1\}^n$ (e.g., $x = \{X_1 = 0, X_2 = 0, X_3 = 1\}$). We also use $\pi_x(f_i)$ to denote the partial assignment of x for clause f_i (e.g., $\pi_{x=\{0,0,1\}}(f_c) = \{X_2 = 0, X_3 = 1\}$).

Definition 1 For a ground clause f_i, we define $s_{i,j}$ (resp. $u_{i,j}$) as the value of a ground atom $X_j \in \{1, 0\}$ that satisfies (resp. violates) f_i. Therefore, f_i is satisfied if and only if at least one of its ground atoms X_j is equal to $s_{i,j}$.

For example, f_a is satisfied iff $s_{c,2} = 0$ or $s_{c,3} = 1$. Each f_i is associated with the pair $(w_i, \mathcal{X}_{f_i})$, where w_i is its weight and $\mathcal{X}_{f_i}$ is the set of variables appearing in its scope (e.g., f_c is associated with the pair $(3.2, \mathcal{X}_{f_c} = \{X_2, X_3\})$. Both "+" and "−" will be used to denote the positive and negative appearance of the ground atoms. Then from the above definition, we have the following sets:

$$\mathcal{F}^+(j) = \{f_i \in \mathcal{F}(j); s_{i,j} = 1\}, \tag{1a}$$

$$\mathcal{F}^-(j) = \{f_i \in \mathcal{F}(j); s_{i,j} = 0\}, \tag{1b}$$

$$\mathcal{F}^s_{f_i}(j) = \{f_k \in \mathcal{F}(j) \setminus \{f_i\}; s_{i,j} = s_{k,j}\}, \tag{1c}$$

$$\mathcal{F}^u_{f_i}(j) = \{f_k \in \mathcal{F}(j) \setminus \{f_i\}; s_{i,j} \neq s_{k,j}\} \tag{1d}$$

where $\mathcal{F}(j) = \{\mathcal{F}^+(j) \cup \mathcal{F}^-(j)\}$ is the set of whole ground clauses involving X_j. $\mathcal{F}^+(j)$ (resp. $\mathcal{F}^-(j)$) is the set of ground clauses in $\mathcal{F}(j)$ that contain X_j positively (resp. negatively). For instance, $\mathcal{F}^+(X_2) = \emptyset$ and $\mathcal{F}^-(X_2) = f_c, f_d$. We use $\mathcal{F}^s_{f_i}(j)$ (resp. $\mathcal{F}^u_{f_i}(j)$) to denote the subset of ground clauses that agree (resp. disagree) with f_i about X_j. For

Table 1 An explanatory Markov logic

Rule	First-order logic	Clausal form	Weight
f_a	$\neg X_1(a)$	$\neg X_1$	1.2
f_b	$\neg X_3(a)$	$\neg X_3$	0.7
f_c	$X_2(a) \Rightarrow X_3(a)$	$\neg X_2 \vee X_3$	3.2
f_d	$X_2(a) \Rightarrow X_1(a)$	$X_1 \vee \neg X_2$	∞

It involves grounding clauses $\{f_a, f_b, f_c, f_d\}$ defined on ground atoms $\{X_1(a), X_2(a), X_3(a)\}$

instance, $\mathcal{F}^s_{f_c}(X_2) = \{f_d\}$ and $\mathcal{F}^u_{f_c}(X_1) = \{f_a\}$. We use $\Theta_j = \left[\theta_j^+, \theta_j^-\right]$ to denote the marginal of X_j, where θ_j^+ and θ_j^- represent its positive and negative marginal probabilities, respectively.

Definition 2 Maneva and Mosel [40] We say that a variable X_j is the unique satisfying variable for a clause $f_i \in \mathcal{F}$ if it is assigned $s_{i,j}$ whereas all other variables in the clause are assigned $u_{i,j}$. A variable X_j is constrained by the clause f_i if it is the unique satisfying variable for f_i. That is to say, it satisfies an indicator function as follows:

$$\text{CON}_{i,j}(\pi_x(f_i)) = \text{Ind}(X_j \text{ is the unique variable satisfying } f_i) \tag{2}$$

Where Ind(Predicate) returns 1 if the Predicate is true and 0 otherwise. For instance, if we assign $X_2 = 1$ and $X_3 = 1$ then X_3 is the unique satisfying variable for f_c. That is, $\text{CON}_{c,3}(\pi_x(f_c)) = 1$. The variable X_j is said to be unconstrained if it has value 0 or 1 and it violates $\text{CON}_{i,j}(\pi_x(f_i))$ for each clause $f_i \in \mathcal{F}_{X_j}$ that involves it. For instance, without assigning values to both X_1 and X_3, if $X_2 = 1$, then it is said to be unconstrained for both f_c and f_d. This in turns implies that only X_1 and X_3 could later on serve as the unique satisfying variables for f_c and f_d. On contrary, the variable X_j is said to be joker for f_i (and has a joker value $*$) if f_i is already satisfied by other variables and it does not care about X_j, so the variable X_j is free to take either value 0 or 1. In other words, a variable with the value $*$ can be interpreted as being undecided, while variables with values 0 or 1 can be interpreted as being decided on what they want to be. That is to say, without assigning values to X_1 and X_3, if $X_2 = 0$ then both f_c and f_d will be satisfied. This renders both X_1 and X_3 to be joker variables. Accordingly, we can redefine when a complete assignment taking values in $\{0, 1, *\}$ satisfies or violates clauses as follows:

Definition 3 Chieu and Lee [10] A complete assignment $x \in \{0, 1, *\}^n$:

- Satisfies a clause f_i if and only if (i) f_i contains a constrained variable X_j that is set to $s_{i,j}$, or (ii) f_i contains at least two unconstrained joker variables.
- Violates a clause f_i if and only if all its variables $X_j \in \mathcal{X}_{f_i}$ are set to $u_{a,i}$.
- Is invalid for a clause f_i if and only if exactly one variable X_j takes a joker value $*$ and all its other variables $X_k \in \mathcal{X}_{f_i} \setminus X_j$ are set to $u_{i,k}$.

The complete assignment x is valid for a $\mathcal{F}$ if it is valid for all of its clauses. For instance, the complete assignment $x = \{X_1 = 0, X_2 = *, X_3 = 0\}$ is invalid because of its invalidity for clauses f_c and f_d. Note that the above definition of the invalid complete assignment reflects the interpretation that the joker value "$*$" is a "don't care" state for variable X_j: clauses involving a variable $X_j = *$ should be already satisfied by other variables, and the value of X_j does not matter. This in fact means that $X_j = *$ cannot be the last remaining possibility of satisfying any clause. So, if the other variables violate the clause and the remaining variable X_j is a joker, then the complete assignment is neither satisfying nor violating the clause because in this case we have two possibilities for $X_j = *$: one makes the complete assignment satisfying and the other makes it violating. Thus we say that the complete assignment is invalid. In the case where a clause contains two variables set to joker, the clause can be satisfied by either one of these two variables, so the other variable can take the "don't care" value.

Definition 4 Maneva and Mosel [40] A core is a valid complete assignment $x \in \{0, 1, *\}^n$ that satisfies all the clauses, and contains no unconstrained variables equal 0 or 1.

Note that the core can be seen as a combinatorial representative of valid complete assignments of a cluster. For example, the core $x = \{0, 1, *, 1, *\}$ is the combinatorial assignment that compresses the cluster of four complete assignments: $\{0, 1, 0, 1, 0\}$, $\{0, 1, 0, 1, 1\}$, $\{0, 1, 1, 1, 0\}$, and $\{0, 1, 1, 1, 1\}$. When a variable takes a fixed value 0 or 1 in the core x, then it is said to be *frozen* with respect to the core x (i.e., fixed in all the valid assignments of the cluster represented by core x) [1]. Otherwise it is *unfrozen*. A frozen variable is also called a *backbone* variable [30], and it can be extended for optimization problems [70].

Definition 5 Achlioptas and Ricci-Tersenghi [1] *Cluster backbones* refer to the set of frozen variables in a core x. In other words, it is the set of backbone variables in all valid complete assignments (i.e., solutions) within the cluster represented by x [34].

Thus, one relatively crude but useful way to obtain a portion of a solution in the cluster is by finding the cluster backbones of its core.

3.2 Markov logic

Markov logic [59] is a set of pairs (f_i, w_i), where f_i is a first-order logic formula (or clause in CNF) and w_i is its associated weight. Together with finite sets of constants that are domains to atoms, we can build a Markov logic network (MLN) that has the following:

- One binary node for each possible grounding of each atom appearing in each clause. The value of the node is 1, if the ground atom is true, and 0 otherwise.
- One feature for each possible ground clause f_i. The value of this feature is 1, if the ground formula is true, and 0 otherwise. The weight of the feature is the weight w_i associated with f_i, where the weight attached to each ground clause reflects its strength dependency.

The power of MLNs appears in their ability to bridge the gap between logic and probability theory. Thus it has become one of the preferred probabilistic graphical models for representing both probabilistic and deterministic knowledge, with deterministic dependencies represented as hard clauses, and probabilistic ones represented as soft clauses. MLNs compactly represent the joint distribution over non-evidence ground atoms $\mathcal{X}$ given the evidence ones E as: $P(\mathcal{X}|E) = \mathcal{Z}_E^{-1} e^{\sum_i w_i \cdot f_i}$, where $\mathcal{Z}_E$ is the normalizing constant. For any complete assignment (or possible world) $x \in \{0, 1\}^n$, to have a non-zero probability, all the hard ground clauses have to be satisfied.

To understand the semantics of Markov logic, recall the explanatory example in Table 1. In this example, Markov logic enables us to model the KB as shown in Fig. 1. It considers the rules $\{f_a, f_b, f_c\}$ as soft (i.e., associated with soft weights) whether the rule f_d is considered as hard (i.e., assigned an infinite weight in a sense that it must be satisfied).

3.3 Relational MAP inference

Given the grounded network of the MLN, the objective of the MAP inference task [48] is to find the most probable truth assignment (or MAP solution) of the non-evidence atoms $\mathcal{X}$ that maximizes the sum of weights of satisfied clauses, given other atoms as evidence E:

$$x^{MAP} = \arg\max_x P(\mathcal{X} = x|E) \tag{3}$$

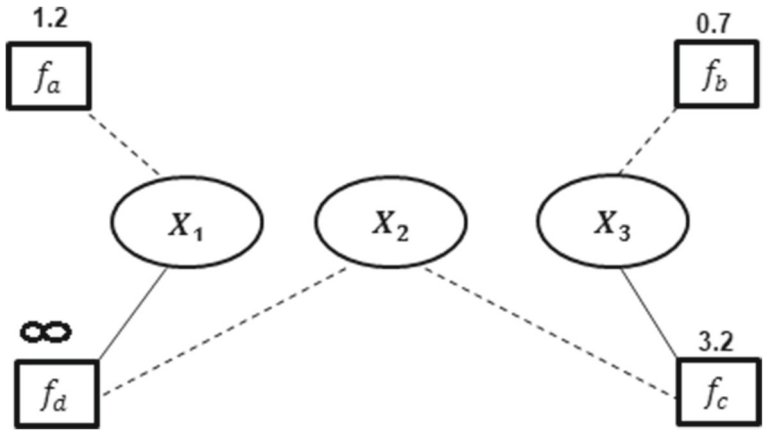

Fig. 1 The factor graph representation of the explanatory Markov logic presented in Table 1. The dashed and solid lines represent "-" and "+" appearance of the atoms, respectively

Taking into account the logical structures of a MLN, MAP inference corresponds to an instance of solving the weighted maximum satisfiability problem, and therefore it can be carried out efficiently using a weighted satisfiability solver like MaxWalkSAT [25, 63, 66], which is currently the state-of-the-art MAP inference in SRL systems like ALCHEMY [31], TUFFY [49], LoMRF [69], ProbCog and Pracmln, etc.

Algorithm 1 MaxWalkSAT for solving MAP inference in MLN.

Input: Set of clauses and their weights ($\mathcal{F}$,$\mathcal{W}$), set of query atoms $\mathcal{X}$, maximum number of tries $\mathcal{I}_t$, maximum number of flips $\mathcal{I}_f$, threshold ϑ, and probability p.
Output: MAP solution x^{MAP} or no solution found (FAILURE)

1: **for** $i \leftarrow 1$ to $\mathcal{I}_t$ **do**
2: $Temp_{Sol} \leftarrow$ randomly generated assignment to all atoms in $\mathcal{X}$
3: **if** the sum of weights ($\sum_k w_k$) of satisfied clauses in $\mathcal{F}$ by $Temp_{Sol} > \vartheta$ **then**
4: $x^{MAP} \leftarrow Temp_{Sol}$
5: **Return** x^{MAP}
6: **end if**
7: **for** $j \leftarrow 1$ to $\mathcal{I}_f$ **do**
8: $f_{\text{UNSAT}} \leftarrow$ Selected unsatisfied clause of $\mathcal{F}$
9: **With probability** p**:**
10: $Temp_{Sol} \leftarrow$ flip a random atom in f_{UnSAT}
11: **With probability** $1 - p$**:**
12: $Temp_{Sol} \leftarrow$ flip the best atom in f_{UnSAT} that maximizes the sum of satisfied clause weights
13: Go to line 3
14: **end for**
15: **end for**
16: **Return** FAILURE;

Algorithm 1 gives the pseudo-code for MaxWalkSAT algorithm. The algorithm takes as input the set of clauses $\mathcal{F}$ and their weights $\mathcal{W}$, the set of ground atoms $\mathcal{X}$ on which we need to compute its MAP assignment, and the probability of random step p. Up to a maximum number of tries $\mathcal{I}_t$, it iteratively starts by randomly generating a possible MAP assignment to $\mathcal{X}$ (line 2). If the total sum of weights, for the clauses that are satisfied by the generated MAP assignment, is larger than the prespecified threshold ϑ, then it returns the obtained optimal MAP solution (lines 3-6). Otherwise, MaxWalkSAT is used to perform a stochastic local search through two steps. In the first step, it picks an unsatisfied clause uniformly at random (line 8). Here, a simple heuristic could be used to order the false ground clauses based on their weights, and then uniformly selects the ones which have high weights. In the second step, it iteratively flips the truth values of the atoms in the selected false ground clauses. At each flip, the atom is chosen randomly with a certain probability p (lines 9-10); otherwise, with probability $(1 - p)$ it applies a greedy hill-climbing heuristic to select the atom that maximizes the sum of satisfied clause weights when flipped (lines 11-12).

3.4 Decimation based on survey propagation

Survey Propagation (SP) [40] can be viewed as a BP message passing algorithm on a factor graph that defines non-zero marginals over cores representing solution clusters. That is, unlike standard BP, the variables' marginals obtained from SP correspond to surveys over the clusters in the solution space. These marginals in fact provide information about the fraction of clusters in which each variable is free or frozen [5, 8, 11]. Thus one efficient way to exploit the SP's marginals (also called biases) is to apply a marginalization-decimation algorithm based on SP [8, 35].

Algorithm 2 gives the pseudo-code for SP-guided decimation algorithm. It starts by randomly initializing the survey messages (line 1). It then deploys its message-passing process on the factor graph (line 2) as follows : (1) Each factor $f_i \in \mathcal{F}$ passes a survey message $\eta_{f_i \to X_j}$ containing a real number to each of its neighboring variables X_j in the factor graph. This survey is the probability that the factor f_i warns the variable X_j against violating it. That is, if the survey message is close to 1, then the factor f_i is warning the variable X_j against taking a value that will violate the factor; if the survey is close to 0, then the factor f_i does not care about the value taken by X_j since it is satisfied by other variables in $\mathcal{X}_{f_i} \setminus \{X_j\}$. (2) Each variable X_j sends to a neighboring factor f_i, a message $[\mu^s_{X_j \to f_i}, \mu^u_{X_j \to f_i}, \mu^*_{X_j \to f_i}]$, where the three components of the message correspond to the probability that X_j will be warned by other factors to take a value that will satisfy f_i, violate f_i, or freely take any value (i.e., joker). SP alternatively exchanges its messages between factors and atoms until converging to a fixed-point of survey messages $\{\eta^*_{f_i \to X_j}\}$. If non-trivial surveys are found (i.e., $\{\eta^*_{f_i \to X_j} \neq 0\}$), then it uses these surveys to compute the marginals of the query atoms in $\mathcal{X}$ (lines 4-5). It consequently applies a decimation procedure as follows: (1) extract the fraction of frozen atoms with the largest biased marginals (e.g., $|\theta^+_j - \theta^-_j| > 0.5$) - (line 6); (2) fix the frozen atoms to their most likely values (i.e., assign a value 1 if $(\theta^+_j > \theta^-_j)$ and a value 0 if $(\theta^+_j < \theta^-_j)$), and then reduce the factor graph (i.e., simplify the problem's formula)-(lines 7-8); (3) Add the fixed frozen atoms as a partial assignment to x^{MAP} (line 9). Note that fixing the frozen atoms potentially results in determining a partial assignment of optimum solution cluster. So the goal of the decimation is to use SP to find a cluster that contains potential optimal/sub-optimal solutions. Once the cluster is found, the problem is relatively easy to solve. As such, if all survey messages are

Algorithm 2 The SP-guided decimation algorithm for solving MAP inference in MLN.

Input: The MLN factor graph $\mathcal{G}$ representing the set of clauses and their weights $(\mathcal{F}, \mathcal{W})$ and the set of query atoms $\mathcal{X}$.
Output: MAP solution x^{MAP}

`// I.Initial phase:`
1: Randomly initialize SP survey messages
2: Run SP on factor graph $\mathcal{G}$ `// Passing messages between clauses and atoms`
3: **if** (SP converges) **then** `// reaching a fixed-point of survey messages` $\{\eta^*_{f_i \to X_j}\}$
4: **if** non-trivial surveys ($\{\eta^*_{f_i \to X_j} \neq 0\}$) are found **then**
5: Use $\{\eta^*_{f_i \to X_j}\}$ to compute the atoms' marginals $(\Theta_j = \left[\theta_j^+, \theta_j^-\right],\ \forall X_j \in \mathcal{X})$

`// II. Decimation phase: fix frozen variables with the largest biased marginals`
6: Frozen atoms $\leftarrow$ Select ground atoms with the largest $|\theta_j^+ - \theta_j^-|$
7: Fix each frozen atom to the value 1 if $(\theta_j^+ > \theta_j^-)$ and the value 0 if $(\theta_j^+ < \theta_j^-)$
8: Use fixed frozen atoms to simplify the formula
9: x^{MAP}.Append(fixed frozen atoms)
10: **else** `// all surveys are trivial` $(\{\eta^*_{f_i \to X_j} = 0\})$
11: Output the simplified sub-formula and run MaxWalkSAT on it `// call Algorithm 1`
12: **end if**
13: **else**
14: **Return** SP does not converge or **Go to** step 1
15: **end if**
16: If the problem is solved completely the **return** the obtained MAP solution x^{MAP}, Otherwise, **go to** step 2 to repeat the decimation process on the simplified factor graph.

trivial (line 10), then an assignment to the rest of the atoms can be found using any conventional algorithm (e.g., Walk-SAT algorithm) that works well within a given cluster. Hence, to obtain the complete x^{MAP} solution, we call MaxWalkSAT algorithm [25, 66] to solve the simplified sun-formula (lines 11-14). From the satisfiability viewpoint, this mechanism has shown to be one of the best incomplete solvers in solving hard K-SAT instances efficiently [5, 8, 11, 35].

4 WSP-χ framework

At a conceptual level our overall WSP-χ approach consists of the following three key elements. First, we extend the factor graph used to represent a given problem using mega-node random variables which behave identically to the parent set of variables participating in a factor. Second, we perform WSP inference to update an approximation over the original variables in the extended factor graph. Third, we use the marginals computed over original variables to identify a subset of all non-evidence atoms that are backbone ground atoms.

4.1 Factor Graph Re-parameterization

To set up the framework of WSP-χ, the first thing we do is to generalize the natural interpretation of core assignments (see Definition 4) to be applicable to MAP inference in SRL models. That is, we introduce a new notion of max-core (denoted as "$\mathcal{W}$-core"):

Definition 6 A max-core ($\mathcal{W}$-core) is a valid complete assignment $x \in \{0, 1, *\}^n$ such that:

- The total weight of its satisfying ground clauses equals $\mathcal{W}$
- It contains no unconstrained ground atoms equaling 0 or 1
- It satisfies all the hard ground clauses (if there are both hard and soft ground clauses involved in the model).

Intuitively, the simple interpretation of a max-core is that it is a combinatorial object providing a representative generalization of MAP solutions within a cluster. Additionally, the core defined in Definition 4 is a particular case of the $\mathcal{W}$-core wherein $\mathcal{W}$ is the sum of all the ground clauses.

Now the second thing to be done in WSP-χ's framework is to modify the factor graph such that it defines a joint probability over complete assignments that are max-cores. Specifically, we need to re-parameterize the factor graph in such a way that carrying out a BP on the new parameterization is equivalent to running a weighted variant of SP on the original factor graph for MAP inference. In Fig. 2(*left*), we consider a simple example of a factor graph $\mathcal{G}$ that involves four ground clauses $\mathcal{F} = \{f_a, f_b, f_c, f_d\}$, and three ground atoms $\{X_1, X_2, X_3\}$. We re-parameterize this factor graph by transforming it into an *extended factor graph* $\hat{\mathcal{G}}$, shown in Fig. 2(*right*), as follows:

First, we extend the domain of each ground atom X_j to take values in $\{0, 1, *\}$.

Second, we add an *auxiliary mega-node* P_j (dashed circle) corresponding to each ground atom node X_j. Each of these mega-nodes P_j captures the parent set $P_j(x)$ of X_j. Although there could be several ways to define $P_j(x)$, we use a definition similar to the one used by Maneva et al. (2007)

$$P_j(x) = \{f \in \mathcal{F} | \text{CON}_{j,f}(\pi_x(f)) = 1\} \tag{4}$$

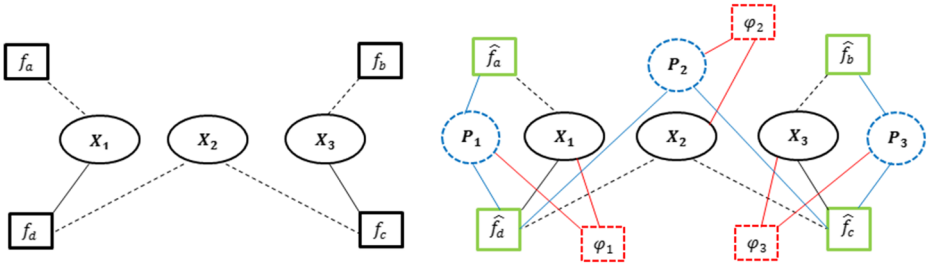

Fig. 2 (*Left*) Our explanatory factor graph $\mathcal{G}$ which involves the grounding clauses $\mathcal{F} = \{f_a, f_b, f_c, f_d\}$ that are defined on the ground atoms $\{X_1, X_2, X_3\}$. The dashed and solid lines represent "-" and "+" appearance of the atoms, respectively. (*Right*) It corresponding extended factor graph $\hat{\mathcal{G}}$, after adding auxiliary mega-node variables $\mathcal{P} = \{P_1, P_2, P_3\}$ and auxiliary factor nodes $\Phi = \{\varphi_1, \varphi_2, \varphi_3\}$, which yields a set of extended factors $\hat{\mathcal{F}} = \left\{\hat{f}_a, \hat{f}_b, \hat{f}_c, \hat{f}_d\right\}$

That is, $P_j(x)$ is the set of ground clauses for which X_j is the unique satisfying variable in x (i.e., the set of ground clauses constraining X_j to its value). $\mathcal{P} = \{P_j\}_{j=1}^{n}$ is the set of auxiliary mega-nodes in $\mathcal{G}$, with $n = 3$ in the example factor graph of Figure 2(*right*).

Now, since we expand the arguments of each factor f_i by including auxiliary mega-node variables that correspond to their ground atoms $\mathcal{X}_{f_i}$, we then have an *extended factor* $\hat{f}_i$. $\hat{\mathcal{F}} = \{\hat{f}_i\}_{i=1}^{m}$ is the set of extended factors in $\mathcal{G}$, with $m = 4$ in the example factor graph. Note that this extension implies that the complete assignments $\{x\}$ in the original factor graph $\mathcal{G}$ extended to corresponding configurations $\{\rho\}$ in the extended factor graph. Each configuration ρ takes the form $\rho = \{X_j, P_j(X)\}_{X_j \in \mathcal{X}}$ such that the projection $\pi_{\mathcal{X}}(\rho)$ of ρ onto variables $\mathcal{X}$ produces a complete assignment x to $\mathcal{G}$, representing a candidate for a max-core.

Each $\hat{f}_i$ defines the following function for each configuration ρ

$$\hat{f}_i(\rho) = \xi(\pi_{\mathcal{X}_{f_i}}(\rho)) \times \prod_{X_k \in \mathcal{X}_{f_i}} \delta([f_i \in P_k(x)], \mathrm{CON}_{k,f_i}(\pi_x(f_i))) \tag{5}$$

where ξ is a reward function that assigns different values to the projection $\pi_{\mathcal{X}_{f_i}}(\rho)$ of x onto the original factor f_i as follows:

$$\xi(\pi_{\mathcal{X}_{f_i}}(\rho)) = \begin{cases} e^{\hat{w}_i \cdot y} & \text{If } \pi_{\mathcal{X}_{f_i}}(\rho) \text{ satisfies } f_i. \\ \hat{v}_i & \text{If } \pi_{\mathcal{X}_{f_i}}(\rho) \text{ violates } f_i \\ 0 & \text{If } \pi_{\mathcal{X}_{f_i}}(\rho) \text{ is invalid for } f_i. \end{cases} \tag{6}$$

Where $\hat{w}_i$ is the weight associated with original factor f_i, and y is a cooling parameter that plays a similar role to the y in the SP-y algorithm [5]: it allows the clauses to be violated at a certain price. As the value of y increases, there is more possibility to violate clauses with minimal weights but the survey messages (i.e., message-passing process) will prefer configurations that violate a minimum number of clauses. $\hat{v}_i$ is a violation value that equals $e^{-\hat{w}_i \cdot y}$. For ease of representation, $\hat{v}_i$ can be adjusted to equal 1, if f_j is a soft ground clause, and 0, if it is a hard ground clause. Therefore the role of the ξ function is to provide a reward $e^{\hat{w}_i \cdot y}$ into the joint probability of a valid max-core x if it satisfies f_i, and to penalize it by 0 if x violates f_j, which is a hard ground clause. The second term in (5) involves a multiplication of Kronecker delta function δ:

$$\delta(\overbrace{[f_i \in P_k(x)]}^{\kappa_1}, \overbrace{\mathrm{CON}_{k,f_i}(\pi_x(f_i))}^{\kappa_2}) = \begin{cases} 1 & \kappa_1 = \kappa_2 \\ 0 & \kappa_1 \neq \kappa_2 \end{cases} \tag{7}$$

which represents a constraint that enforces the consistency between the parent set $P_k(x)$ and the set of ground clauses constraining the ground atom X_k to its value.

Third, we attach an *auxiliary factor node* φ_j (dashed square) that connects each ground atom node X_j with its corresponding auxiliary mega-node P_j. Each φ_j defines the following function for any configuration ρ:

$$\varphi_j(\rho(X_j, P_j)) = \begin{cases} \gamma & \text{If } P_j(x) = \emptyset, \text{ and } X_j \neq *, \\ \chi & \text{If } P_j(x) = \emptyset, \text{ and } X_j = *, \\ 1 & \text{for other valid } (X_j, P_j). \end{cases} \tag{8}$$

where χ and γ are smoothing parameters restricted with a condition $\gamma + \chi = 1$. In the first case, the function φ_j assigns a value γ to a complete assignment x that is an invalid max-core (since the projection of ρ onto $\mathcal{X}$ has an unconstrained variable X_j set to 1 or 0, it represents an invalid max-core). In the second case, if the complete assignment x represents

a valid max-core then it is assigned a positive value χ. In the third case the validity of (X_j, P_j) means that if $X_j = 1$ (resp. $X_j = 0$), then $P_j(x) \subseteq \mathcal{F}^+(j)$ (resp. $P_j(x) \subseteq \mathcal{F}^-(j)$). $\Phi = \{\varphi_j\}_{j=1}^n$ is the set of auxiliary factors in $\hat{\mathcal{G}}$, with $n = 3$ in the explanatory factor graph of Fig. 2(*right*). Now it should be noted that since the values of a complete assignment x of $\mathcal{X}$ determine uniquely the values of $\{P_j(x)\}_{X_j \in \mathcal{X}}$ in ρ, then the distribution over ρ is a distribution over x. Hence the extended factor graph defines the joint probability over complete assignment x as follows:

$$p(x) = p(\pi_{\mathcal{X}}(\rho)) = \prod_{\hat{f}_i \in \hat{\mathcal{F}}} \hat{f}_i(\rho) \prod_{X_j \in \mathcal{X}} \varphi_j(\rho(X_j, P_j)) \propto \gamma^{n_0} \chi^{n_*} \prod_{f_i \in S(x)} e^{\hat{w}_i \cdot y} \tag{9}$$

where n_0 is the number of unconstrained ground atoms in x having the value 1 or 0, and n_* is the number of unconstrained ground atoms in x having joker value $*$. $S(x)$ represents the set of all ground clauses satisfied by x.

4.2 WSP-χ message-passing

The message passing process of WSP-χ proceeds by iteratively sending two types of messages on the extended factor graph $\hat{\mathcal{G}}$ until convergence. These messages are slightly different from simply running the standard BP algorithm, but their structures are categorized based on the value of the set of auxiliary mega-nodes $\mathcal{P}$ in $\hat{\mathcal{G}}$ as follows:

- Factor-to-variable message ($\eta_{\hat{f}_i \to X_j}$) each extended factor $\hat{f}_i$ sends to each X_j of its ground atoms $\mathcal{X}_{\hat{f}_i}$ a message that can be parameterized as a vector of four components:

$$\eta_{\hat{f}_i \to X_j} = \left[\eta^s_{\hat{f}_i \to X_j},\ \eta^u_{\hat{f}_i \to X_j},\ \eta^*_{\hat{f}_i \to X_j},\ 0 \right] \tag{10}$$

where

- $\eta^s_{\hat{f}_i \to X_j}$ represents the probability of the warning that X_j is constrained to be uniquely satisfying for $\hat{f}_i$ (i.e., when $X_j = s_{i,j}$, $P_j \subseteq \mathcal{F}^s_{\hat{f}_i}(j) \cup \{\hat{f}_i\}$),
- $\eta^u_{\hat{f}_i \to X_j}$ is the probability that X_j can violate $\hat{f}_i$ (i.e., if $X_j = u_{i,j}$, $P_j \subseteq \mathcal{F}^u_{\hat{f}_i}(j)$),
- $\eta^*_{\hat{f}_i \to X_j}$ represents the probability that $\hat{f}_i$ does not care about the value of X_j. That is, X_j is either unconstrained by $\hat{f}_i$ or at least one other variable X_k in $\mathcal{X}_{\hat{f}_i} \setminus \{X_j\}$ satisfies $\hat{f}_i$ and X_j equals $*$ (i.e., when $X_j = *$, $P_j = \emptyset$ or $X_j = s_{i,j}$, $P_j \subseteq \mathcal{F}^s_{\hat{f}_i}(j)$),
- Otherwise, we use 0 as a catch-all case if none of the three previous cases occur.

It should be noted that $\eta^s_{\hat{f}_i \to X_j}$, $\eta^u_{\hat{f}_i \to X_j}$ and $\eta^*_{\hat{f}_i \to X_j}$ are elements of [0, 1].

- Variable-to-factor message ($\mu_{X_j \to \hat{f}_i}$) Here each ground atom X_j sends to its extended factors $\hat{f}_i \in \mathcal{F}(j)$ a message that consists of three components:

$$\mu_{X_j \to \hat{f}_i} = \left[\mu^s_{X_j \to \hat{f}_i},\ \mu^u_{X_j \to \hat{f}_i},\ \mu^*_{X_j \to \hat{f}_i} \right] \tag{11}$$

where

- $\mu^{s}_{X_j \to \hat{f}_i}$ represents the probability that X_j is warned by other extended factors $\mathcal{F}_{\hat{f}_i}(j)$ to satisfy $\hat{f}_i$ (i.e., if X_j is constrained to satisfy $\hat{f}_i$),
- $\mu^{u}_{X_j \to \hat{f}_i}$ represents the probability that X_j is warned by other extended factors $\mathcal{F}_{\hat{f}_i}(j)$ to violate $\hat{f}_i$ (i.e., if X_j is constrained to violate $\hat{f}_i$),
- $\mu^{*}_{X_j \to \hat{f}_i}$ is the probability that X_j is unconstrained to $\hat{f}_i$ (i.e., if X_j is unconstrained to $\hat{f}_i$ or equals $*$).

Now in the following Section 4.3, we demonstrate the update equations that are used to repeatedly compute the components of the messages in (10) and (11).

4.3 WSP-χ's update equations

In this subsection we settle for explaining the closed-form of the update equations for WSP-χ's message passing that is expressed in (10) and (11). We refer the reader to Appendix A wherein we provide a detailed derivation of these update equations.

4.3.1 Variable-to-factor Updates

We will begin with the update equations for the messages from variables to factors, given in (12a), (12b) and (12c) as follows:

$$\mu^{s}_{X_j \to \hat{f}_i} = \left[\prod_{\hat{f}_k \in \mathcal{F}^{s}_{\hat{f}_i}(j)} \left(\eta^{s}_{\hat{f}_k \to X_j} + \eta^{*}_{\hat{f}_k \to X_j} \right) \right] \prod_{\hat{f}_k \in \mathcal{F}^{u}_{\hat{f}_i}(j)} \eta^{u}_{\hat{f}_k \to X_j} \tag{12a}$$

$$\mu^{u}_{X_j \to \hat{f}_i} = \left[\prod_{\hat{f}_k \in \mathcal{F}^{u}_{\hat{f}_i}(j)} \left(\eta^{s}_{\hat{f}_k \to X_j} + \eta^{*}_{\hat{f}_k \to X_j} \right) - \prod_{\hat{f}_k \in \mathcal{F}^{s}_{\hat{f}_i}(j)} \eta^{*}_{\hat{f}_k \to X_j} \right] \prod_{\hat{f}_k \in \mathcal{F}^{s}_{\hat{f}_i}(j)} \eta^{u}_{\hat{f}_k \to X_j} \tag{12b}$$

$$\begin{aligned} \mu^{*}_{X_j \to \hat{f}_i} = & \left[\prod_{\hat{f}_k \in \mathcal{F}^{s}_{\hat{f}_i}(j)} \left(\eta^{s}_{\hat{f}_k \to X_j} + \eta^{*}_{\hat{f}_k \to X_j} \right) - \prod_{\hat{f}_k \in \mathcal{F}^{u}_{\hat{f}_i}(j)} \eta^{*}_{\hat{f}_k \to X_j} \right] \prod_{\hat{f}_k \in \mathcal{F}^{u}_{\hat{f}_i}(j)} \eta^{u}_{\hat{f}_k \to X_j} \\ & + \prod_{\hat{f}_k \in \mathcal{F}^{s}_{\hat{f}_i}(j) \cup \mathcal{F}^{u}_{\hat{f}_i}(j)} \eta^{*}_{\hat{f}_k \to X_j} \end{aligned} \tag{12c}$$

As shown in (12a), we update $\mu^{s}_{X_j \to \hat{f}_i}$ by multiplying two parts: the first one represents the probability that X_j satisfies all its constrained factors $\mathcal{F}^{s}_{\hat{f}_i}(j)$ and the second is the probability that X_j violates all its violating factors $\mathcal{F}^{u}_{\hat{f}_i}(j)$.

According to (12b), we update $\mu^{u}_{X_j \to \hat{f}_i}$ by multiplying the probability that X_j violates all its constrained factors $\mathcal{F}^{s}_{\hat{f}_i}(j)$ with the probability that X_j satisfies all its violating factors $\mathcal{F}^{u}_{\hat{f}_i}(j)$.

From (12c), we update $\mu^{*}_{X_j \to \hat{f}_i}$ by considering the probability that X_j is unconstrained from either its satisfying factors $\mathcal{F}^{s}_{\hat{f}_i}(j)$ or its violating factors $\mathcal{F}^{u}_{\hat{f}_i}(j)$.

4.3.2 Factor-to-variable updates

The update equations for the messages from factors to variables are given in (13a), (13b) and (13c) as follows:

$$\eta^{s}_{\hat{f}_i \to X_j} = \left[\prod_{X_k \in \mathcal{X}_{\hat{f}_i} \setminus \{X_j\}} \mu^{u}_{X_k \to \hat{f}_i} \right] \times e^{\hat{w}_i \cdot y} \tag{13a}$$

$$\eta^{u}_{\hat{f}_i \to X_j} = \left[\prod_{X_k \in \mathcal{X}_{\hat{f}_i} \setminus \{X_j\}} \left(\mu^{u}_{X_k \to \hat{f}_i} + \mu^{*}_{X_k \to \hat{f}_i}\right) + \prod_{X_i \in \mathcal{X}_{\hat{f}_i} \setminus \{X_j, X_k\}} \mu^{u}_{X_i \to \hat{f}_i} \right.$$
$$\left. \sum_{X_k \in \mathcal{X}_{\hat{f}_i} \setminus \{X_j\}} \left(\mu^{s}_{X_k \to \hat{f}_i} - \mu^{*}_{X_k \to \hat{f}_i}\right) \right] - \left[\overbrace{\left(1 - e^{-\hat{w}_i \cdot y}\right)}^{\text{penalty}} \prod_{X_k \in \mathcal{X}_{\hat{f}_i} \setminus \{X_j\}} \mu^{u}_{X_k \to \hat{f}_i} \right] \tag{13b}$$

$$\eta^{*}_{\hat{f}_i \to X_j} = \left[\prod_{X_k \in \mathcal{X}_{\hat{f}_i} \setminus \{X_j\}} \left(\mu^{u}_{X_k \to \hat{f}_i} + \mu^{*}_{X_k \to \hat{f}_i}\right) \right] - \prod_{X_k \in \mathcal{X}_{\hat{f}_i} \setminus \{X_j\}} \mu^{u}_{X_k \to \hat{f}_i} \tag{13c}$$

That is, from (13a), we update $\eta^{s}_{\hat{f}_i \to X_j}$ by considering the product of the messages of all other ground atoms of $\hat{f}_i$ except X_j that are violating it. Then we reward the result by multiplying it with the factor term $e^{\hat{w}_i \cdot y}$ (refer to (6) for the definition of the reward term).

From (13b), updating $\eta^{u}_{\hat{f}_i \to X_j}$ requires considering the difference between two parts. The first includes the probability that two (or more) ground atoms in $\hat{f}_i$ satisfies it plus the probability that there is exactly one of $\hat{f}_i$'s ground atoms (except X_j) satisfying $\hat{f}_i$ and all other ground atoms are violating it. The second part involves the probability that all other ground atoms are violating f_i. This latter probability is penalized with the factor $(1-e^{-w_j \cdot y})$ since the same part is rewarded in (13a) when satisfying $\hat{f}_i$.

In (13c), we simply update $\eta^{*}_{\hat{f}_i \to X_j}$ by considering the messages on which the other ground atoms in $\hat{f}_i$ except X_j are either unconstrained or satisfying $\hat{f}_i$ minus the probability that they are violating $\hat{f}_i$.

4.3.3 Marginal updates

The marginals can be calculated from the factor to variable messages once the algorithm converges, to obtain estimates of the marginals over max-covers. The calculation of the marginals are given in (14a), (14b) and (14c) as follows:

$$\theta^{+}_{j} = \mathcal{Z}_j^{-1} \prod_{\hat{f}_i \in \mathcal{F}^{-}(j)} \eta^{u}_{\hat{f}_i \to X_j} \times \left[\prod_{\hat{f}_i \in \mathcal{F}^{+}(j)} \left(\eta^{s}_{\hat{f}_i \to X_j} + \eta^{*}_{\hat{f}_i \to X_j}\right) - \prod_{\hat{f}_i \in \mathcal{F}^{+}(j)} \eta^{*}_{\hat{f}_i \to X_j} \right] \tag{14a}$$

$$\theta^{-}_{j} = \mathcal{Z}_j^{-1} \prod_{\hat{f}_i \in \mathcal{F}^{+}(j)} \eta^{u}_{\hat{f}_i \to X_j} \times \left[\prod_{\hat{f}_i \in \mathcal{F}^{-}(j)} \left(\eta^{s}_{\hat{f}_i \to X_j} + \eta^{*}_{\hat{f}_i \to X_j}\right) - \prod_{\hat{f}_i \in \mathcal{F}^{-}(j)} \eta^{*}_{\hat{f}_i \to X_j} \right] \tag{14b}$$

$$\theta^{*}_{j} = \mathcal{Z}_j^{-1} \prod_{\hat{f}_i \in \mathcal{F}(j)} \eta^{*}_{\hat{f}_i \to X_j} \tag{14c}$$

where $\mathcal{Z}_j$ is the normalizing constant, given the evidence E.

In (14a), we compute the unnormalized positive marginal of a ground atom X_j by multiplying the violating incoming messages from the ground clauses in which X_j appears negatively by the result of subtracting all unconstrained incoming messages from the satisfying incoming messages from the ground clauses in which X_j appears positively.

Similarly, as shown in (14b), we calculate the unnormalized negative marginal by multiplying the violating incoming messages from the factors in which X_i appears positively by the result of subtracting all unconstrained incoming messages from the satisfying incoming messages from the ground clauses in which X_j appears negatively.

Finally, as shown in (14c), we calculate the unnormalized joker marginal by multiplying all the unconstrained incoming messages from all factors in which X_j appears.

The simple interpretation to the positive marginal θ_j^+, in (14a), is that it estimates the probability of randomly selecting a max-core (i.e., a cluster), and finding that X_j is frozen to "+". This max-core satisfies the ground clauses of total maximal weights on which X_j appears positively and violate the ground clauses of total minimal weights on which X_j appears negatively. In other words, θ_j^+ approximates the fraction of the max-cores that contains potentially the optimal MAP solutions in which $X_j =$"+".

Hence, when θ_j^+ is greater than θ_j^-, it is an indication that the fraction of clusters containing X_j frozen to "+" have MAP solutions better optimized (i.e., having more total weights) than the fraction of clusters in which X_j is frozen to "−". This means that the probability of getting inside a cluster that involves optimal solutions increases when X_j is frozen to "+" compared to X_j frozen to "−". This intuitively implies that fixing $X_j =$"+" is more likely to be a part of the optimal MAP solution than $X_j =$"−".

4.4 Family of extended factor graphs

In a promising attempt at understanding the success of SP, it was suggested that the solutions of random formulas typically do not possess non-trivial cores [40] (the core assignment is non-trivial if it has at least one frozen variable). This implies that the variants of SP(χ) are most effective for values of χ close to and not necessarily equal to 1. That is, pure SP, denoted as SP(1), is not always the most effective method that one usually wants to use, and that other versions of BP could be preferable. This is because the near-core assignments which are the ones of maximum weight in this case, may correspond to quasi-solutions of the cavity equations [45]. However, this explanation has been shortly dismissed by experiments that ensure that non-trivial cores simply do exist for large formulas [33]. This means that the pure SP is surprisingly the most accurate at computing marginals over these cores despite the existence of many cycles in the formulas [33]. Recently, it has been shown that the cores can represent singleton clusters (with very small number of variables taking the $*$ value). These cores are called *degenerate covers* [11]. In addition, it has been proven that, in many structured weighted Max-SAT problems, cores are often degenerate — see Lemma (2) in [11].

All of the aforementioned observations motivate the studying of a full family of WSP-χ extended factor graphs at various values of the smoothing parameter (i.e., $0 \leq \chi \leq 1$). This can be helpful to investigate if the MAP solutions possess non-trivial max-cores or not. From the satisfiability perspective, we believe that this can be beneficial for more understanding about the combinatorial properties of solution space of the structured SAT problems in relational domains. It can also be used to study the satisfiability threshold of these problems, and classifying them according to connectivity of the solution space.

Based on that, if we now adjust the value of the smoothing parameter[3], $\chi \in [0, 1]$, in the auxiliary factor nodes (see (8)), we consequently have a family of extended factor graphs that can be categorized into three cases:

1. **Set $\chi \neq 0$ and $\gamma \neq 0$:** we have a subset of extended factor graphs parameterized by $0 < \chi < 1$. That is, for each value of $\chi \in (0, 1)$, we have an extended factor graph that defines a positive joint distribution over max-cores as defined in (9). Hence, running WSP-χ's message-passing (as explained in Section 4.2) recovers a family of Weighted SP algorithms corresponding to the values of $\chi \in (0, 1)$.
2. **Set $\chi = 0$ and $\gamma = 1$:** In this case the the max-cores with $n_* = 0$ are the only ones having a positive joint distribution as given in (9). This means that the ground atoms are not allowed to take a joker value $*$, but take only values 1 or 0. This implies that the extended factor graph $\hat{\mathcal{G}}$ is equivalent to the original factor graph $\mathcal{G}$ in such a way that renders running WSP-χ's message-passing on $\hat{\mathcal{G}}$ equivalent to loopy max-product BP on $\mathcal{G}$.
3. **Set $\chi = 1$ and $\gamma = 0$:** Here each auxiliary factor node (as defined in (8)) takes the form

$$\varphi_j(\rho(X_j, P_j)) = \begin{cases} 0 & \text{If } P_j(X) = \emptyset, \text{ and } X_j \neq *, \\ 1 & \text{If } P_j(X) = \emptyset, \text{ and } X_j = *, \\ 1 & \text{for other valid } (X_j, P_j) \end{cases} \tag{15}$$

   where in the first case the function φ_j assigns a value 0 only if x is an invalid max-core, i.e., $n_0 > 0$ (from Definition 6, when the complete assignment has unconstrained ground atom set to 1 or 0 then it can not be a max-core). Otherwise it assigns a positive value 1 when x is a valid max-core. Therefore, the extended factor graph in this case has a joint distribution over max-core x, as defined in the following theorem 1.

Theorem 1 *The underlying joint distribution defined by the extended factor graph, $\hat{\mathcal{G}}$ with $\chi = 1$, is positive only over valid max-cores.*

Proof The complete assignment x in extended factor graph $\hat{\mathcal{G}}$ that is not max-core will be either invalid or will involve unconstrained ground atoms set to value 1 or 0. For invalid complete assignments, the distribution is zero because of the definition of ξ in (6). Additionally, for complete assignments with unconstrained ground atoms set to value 1 or 0 the distribution will be zero because of the definition of φ_j in (15). Thus, from (9), for each valid max-core x we have a joint probability of the form:

$$p(x) \propto \prod_{f_i \in S(x)} e^{\hat{w}_i \cdot y} \tag{16}$$

where $S(x)$ represents only the set of the ground clauses satisfied by x. This means that the joint probability is always positive for valid max-core x. □

It is worth noting that Theorem 1 also implicitly shows that any invalid must have a zero joint probability. This is because its joint factorization will be multiplied by either the $(\gamma = 0)^{n_0}$ with $n_0 > 0$, or 0 penalty term in the reward function. Let us support the comprehension of Theorem 2 by using an illustrative example represented in Table 2, which depicts the

[3]Note that since $\gamma + \chi = 1$ then $\gamma = 1 - \chi$, so we can specify the setting of the auxiliary factor node using only one parameter χ.

Table 2 (*Top*) The joint probabilities of complete assignments {1, 0, 1} and {1, 1, 1} in the original factor graph

X_1	X_2	X_3	$p(X_1, X_2, X_3)$			
1	0	1	$e^{(\hat{w}_c+\hat{w}_d)\cdot y}$			
1	1	1	$e^{(\hat{w}_c+\hat{w}_d)\cdot y}$			
.	.	.	.			
X_1	X_2	X_3	P_1	P_2	P_3	$p(X_1, X_2, X_3, P_1, P_2, P_3)$
1	0	1	f_d	$\emptyset$	f_c	$\gamma \times e^{(\hat{w}_c+\hat{w}_d)\cdot y}$
1	1	1	f_d	$\emptyset$	f_c	$\gamma \times e^{(\hat{w}_c+\hat{w}_d)\cdot y}$
1	*	1	f_d	$\emptyset$	f_c	$\chi \times e^{(\hat{w}_c+\hat{w}_d)\cdot y}$
.	.	.	.	.	.	.

(*Bottom*) The (solution cluster-based) joint probabilities of their corresponding configurations ρ in the extended factor graph, where $\hat{w}_c$ (resp. $\hat{w}_d$) are the weights associated with the factors f_c (resp. f_d) that are satisfied by the underlying complete assignments

expansion of complete assignments {1, 0, 1} and {1, 1, 1} in our explanatory factor graph $\mathcal{G}$ to their corresponding ρ configurations in the extended factor graph $\hat{\mathcal{G}}$ in Fig. 2.

Let us consider the complete assignment {1, 0, 1} in $\mathcal{G}$, this assignment has $X_1 = 1$ constrained by f_d, $X_3 = 1$ constrained by f_c, and $X_2 = 0$ is not a unique satisfying variable to $\{f_c, f_d\}$. The assignment {1, 1, 1} implies the same constrained ground clauses for X_1 and X_3, but not for X_2 which can not be constrained to 1 by any ground clause. In the extended factor graph $\hat{\mathcal{G}}$, we can notice that flipping X_2 value from 0 to 1 will not violate or satisfy any additional ground clauses. Thus, we have additionally the complete assignment {1, ∗, 1}. Now, each one of the three assignments {1, 0, 1}, {1, 1, 1} and {1, ∗, 1} can be seen as a candidate for a max-core with a certain probability. However, since both candidate assignments $\rho = \{1, 0, 1\}$ and $\rho = \{1, 1, 1\}$ have identical parent sets of the form ($P_1 = \{\hat{f}_d\}$, $P_2 = \emptyset$, and $P_3 = \{\hat{f}_c\}$) then they yield an invalid max-core because they violate the second condition in the concept of max-core (see Definitions 3 and 6). That is, the two factors $\{\hat{f}_c, \hat{f}_d\}$ involve the unconstrained ground atom X_2 set to value 1 or 0 (i.e., $n_0 = 2$). According to the extended factor graph parameterization, this situation implies computing the auxiliary factors $(\varphi_j(\rho(X_j, P_j))_{j=1}^{3})$ based on (15), and it is a commitment to set $\chi = 1$ and $\gamma = 0$. Now since $n_0 > 0$ for these these two invalid max-core, then based on (9), their joint probability equals to zero because the multiplication by γ. On the contrary, $x = \{1, *, 1\}$ is a valid max-core and its corresponding ρ configurations have positive joint probability multiplied by χ and not γ since $n_0 = 0$. When $n_0 > 0$, we are obliged to assign γ with a non-negative value. Note that, in this case, decreasing the value of γ, which is equivalent to increasing χ's value, implies increasing the probability that both {1, 0, 1} and {1, 1, 1} assignments can not be a max-core. It also increases the probability that {1, ∗, 1} can be a max-core. (It will be shown in Section 4.4 that the best parameterization to $\hat{\mathcal{G}}$ is to choose $\chi = 1$ and $\gamma = 0$.).

Consequently, performing WSP-χ's message-passing on $\hat{\mathcal{G}}$ with $\chi = 1$ produces a pure version of Weighted SP algorithm (WSP-1).

Theorem 2 *When $y \to \infty$, then WSP-1 estimates marginals corresponding to the stationary point of the Bethe free energy on a uniform distribution over max-cores.*

Proof Assume that we have a max-core (i.e., $\mathcal{W}$-core) of total weight $\mathcal{W}$, and a more optimal one (($\mathcal{W}+\epsilon$)-core) of larger weight $\mathcal{W}+\epsilon$. According to (16), the ratio probability of the two max-cores is:

$$\frac{P(\mathcal{W}+\epsilon\text{-core})}{P(\mathcal{W}\text{-core})} = e^{\epsilon \cdot y} \tag{17}$$

Note that from (17), the ratio probability is still positive only over max-cores. Now, as y tends to ∞ in (17), the two max-cores has equal probabilities. This means that each max-core will have the same joint probability which is e^{∞}. This in turn implies that the extended factor graph defines a uniform joint distribution over valid max-cores. Hence, running WSP-1's message-passing over the extended factor graph representing (16) estimates marginals over uniformly distributed max-cores. □

From the above formulations, we have a parameterized family of Weighted Survey Propagation algorithms (WSP-χ), ranging from a traditional max-product BP (WSP-0) to pure weighted SP (WSP-1). In Section 6.3.3, our experimental evaluation shows the success of pure WSP-1 on real-world problems for finding the most accurate MAP solutions. This is consistent with experimental results in [11, 33], ensuring that non-trivial cores often do exist for large formulas of structured problems.

5 Using WSP-χ as a general procedure for solving MAP inference

As demonstrated in the previous subsection, the marginals obtained from WSP-χ correspond to surveys over max-cores representing clusters of MAP solutions. That is to say, they provide information about the fraction of clusters in which ground atoms are frozen or unfrozen in their MAP solutions. Thus, a direct use of that information is to apply a marginalization-decimation algorithm [35] based on WSP-χ, recovering a family of WSP-χ inspired decimation (**WSP-Dec**) algorithms for solving the task of full MAP inference (i.e., finding an assignment to all non-evidence atoms) on SRL models.

5.1 WSP-χ inspired decimation

For convenience, and without loss of generality, we focus on MLN when explaining how WSP-Dec finds a MAP solution.

As clarified in Algorithm 3, WSP-Dec is a two-stage strategy. In the first stage, the goal is to use WSP-χ for scaling the MLN's grounded network and obtaining a portion of the optimal MAP solution, as follows: We first assign the smoothing parameter χ the value $\hat{\chi}$ to specify the WSP-$\hat{\chi}$ algorithm from the family WSP-χ that will be used as preprocessing. We then adjust its setting for both the cooling parameter $\hat{y}$ and magnetization threshold $\hat{\mathcal{T}}$ (line 1). The specified WSP-$\hat{\chi}$ starts by initializing its messages uniformly at random, as in line 2. It then iteratively applies a set of decimation steps until reaching a trivial fixed point of the messages. At each decimation step, lines 8-11, it iteratively updates its messages, using (12a), (12b), (12c), (13a), (13b), and (13c), until either exceeding the maximum number of iterations or converging to a non-trivial fixed point of the messages

Algorithm 3 WSP-Dec for MAP inference in MLN.

Input: Set of Clauses and their Weights $(\mathcal{F}, \mathcal{W})$, set of query atoms $\mathcal{X}$, Evidence database $\mathcal{DB}$, Maximum number of iterations $\mathcal{I}_{\max}$, cooling parameter $\hat{y}$, Magnetization threshold $\hat{\mathcal{T}}$, smoothing parameter $\hat{\chi}$.
Output: MAP solution x^{MAP}.

1: Set the parameters $y = \hat{y}$, $\mathcal{T} = \hat{\mathcal{T}}$ and $\chi = \hat{\chi}$;
`// Using WSP-`$\hat{\chi}$ `as a pre-processing`
2: $\eta_{\hat{f}_i \leftarrow X_j} \in \mathcal{U}[0, 1]$, $\forall X_j \in \mathcal{X}$, $\forall f_i \in \mathcal{F}$; `// Messages initialization;`
3: **repeat** `// Updating the messages`
4: Use $\eta_{\hat{f}_i \rightarrow X_j}$ to update $\mu_{X_j \rightarrow \hat{f}_i}$; `// Using Eqs.(12a),(12b), and (12c)`
5: Use $\mu_{X_j \rightarrow \hat{f}_i}$ to update $\eta_{\hat{f}_i \rightarrow X_j}$; `// Using Eqs.(13a),(13b), and (13c)`
6: **until** (Convergence or termination of $\mathcal{I}_{\max}$)
7: **Return** a fixed point of the messages $\hat{\eta}_{\hat{f}_i \rightarrow X_j}$;
8: **if** (non-trivial $\hat{\eta}_{\hat{f}_i \rightarrow X_j} \neq 0$ are found) **then**
9: **for** each $X_j \in \mathcal{X}$ **do**
10: Compute: $\Theta_j = [\theta_j^+, \theta_j^-]$; `// Using Eqs.(14a),(14b),(14c), (33a), (33b), and (33c)`
11: **end for**
12: $\beta \leftarrow$ Select[ground atoms X_js having ($|\theta_j^+ - \theta_j^-| > \mathcal{T}$)] `// obtain frozen ground atoms`
13: $\mathcal{X}^* \leftarrow \mathcal{X} \setminus \{\beta\}$; `// remove` β `from queries`
14: $\mathcal{DB}^* \leftarrow \mathcal{DB} \cup \{\beta\}$; `// add` β `to evidence database`
15: $\beta^* \leftarrow \beta^* \cup \beta$; `// store all` β `as a portion of` x^{MAP}
16: Simplify the model clauses $\mathcal{F}$ into $\mathcal{F}^*$; `// Fix frozen ground atoms (`β`)`
17: $\mathcal{F} \leftarrow \mathcal{F}^*$; $\mathcal{X} \leftarrow \mathcal{X}^*$ and Go to (2);
18: **else if** (trivial $\hat{\eta}_{\hat{f}_i \rightarrow X_j} = 0$ are found) **then**
19: Run MaxWalkSAT on the simplified grounded network constructed by $(\mathcal{X}^*, \mathcal{F}^*, \mathcal{DB}^*)$;
20: **end if**
21: $x^{MAP} \leftarrow$ Combination of returns from steps 15 and 19;
22: **Return** x^{MAP};

(lines 4-7). It then estimates the marginals of ground query atoms in $\mathcal{X}$ using (14a), (14b), (14c), (33a), (33b), and (33c). Subsequently, as in line 12, it uses the computed marginals to identify frozen ground atoms (i.e, cluster backbones): the fraction of ground atoms in $\mathcal{X}$ that have a magnetization $|\theta_j^+ - \theta_j^-| \geq \mathcal{T}$. Afterwards, it fixes the frozen ground atoms to their more likely truth values (i.e., magnetized values). In addition, as in line 14, it adds them to the evidence database $\mathcal{DB}$: those whose $(\theta_j^+ < \theta_j^-)$ are added to $\mathcal{DB}$ as false evidence, and those whose $(\theta_j^+ > \theta_j^-)$ are added as true evidence. At this point, it should be noted that the advantage of fixing the frozen ground atoms is two-fold: shrinking the set of query atoms (i.e., $\mathcal{X} \setminus \{\beta\}$), and enlarging the evidence database (i.e., $\mathcal{DB} \cup \{\beta\}$). This in turn results in reducing the set of ground clauses, and therefore simplifying the grounded network that instantiates the MAP inference problem (line 16). Eventually, as in line 18, if it reaches a trivial fixed point of the messages: those that often produce demagnetized marginals (i.e.,

marginals that are not biased to either positive or negative values) and yield paramagnetic solutions (paramagnetic solution refers to a generalized complete assignment that is not biased to any value for all variables). Then either the complex parts of the grounded network have been decimated by fixing the frozen ground atoms and/or the remaining query ground atoms define a simple MAP inference that can be efficiently solved using any off-the-shelf local search algorithm (e.g., MaxWalkSAT).

In the second stage, we run the MaxWalkSAT algorithm (line 19) to solve the remaining simplified MAP inference problem. The output returned from MaxWalkSAT combined with the total set of the frozen ground atoms obtained from the WSP will provide the overall MAP solution (line 21).

5.2 Combining WSP-χ with lazy MAP inference

One key advantage of WSP-χ algorithms is that they can be combined with other state-of-the-art approaches which greatly improve the scalability of MAP inference such as Lazy and Lifted inference. Algorithm 4 shows how to combine WSP-χ with a Lazy MAP inference, which yields **Lazy-WSP-Dec**. Lazy-WSP-Dec mainly differs from the WSP-Dec of Algorithm 3 in both the initial set of underlying query atoms and clauses, and the local search algorithm that will be used to solve the simplified MAP inference (line 19). That is, Lazy-WSP-Dec starts by grounding the network lazily, and maintaining only active ground clauses and their active ground atoms that are sufficient to answer the queries. Note that a ground clause is active if it can be made unsatisfied by flipping none or at least one of its active atoms. Thus, by default, an unsatisfied ground clause is always active. An atom is active if it was flipped at some point in the search, or if it is in the initial set of active atoms. Lazy-WSP-Dec then calls the specified WSP-$\hat{\chi}$ algorithm, steps 1-17 of algorithm 3, to scale the lazy ground network, which was built using those active clauses and atoms, by fixing the frozen active atoms. After reaching a trivial fixed point, it runs Lazy-MaxWalkSat, as in line 6, on the simplified network instead of propositional MaxWalkSAT as in algorithm 3.

Algorithm 4 Combining WSP-Dec with lazy MAP inference.

1: $\mathcal{X} \leftarrow$ (atoms in clauses unsatisfied by $\mathcal{DB}$); `// Consider only active atoms`
2: $\mathcal{F} \leftarrow$ (clauses activated by $\mathcal{X}$); `// Consider only active clauses`
3: $\mathcal{W} \leftarrow$ Weights associated with $\mathcal{F}$;
`// Call WSP-χ: steps 1-17 in algorithm 3.`
4: $[\mathcal{X}^*, \mathcal{F}^*, \mathcal{DB}^*] \leftarrow$ WSP-$\hat{\chi}(\mathcal{X}, \mathcal{F}, \mathcal{W})$; `//Simplifying the lazy grounded network`
5: **if** (trivial $\hat{\eta}_{\hat{f}_i \rightarrow X_j} = 0$ are found) **then**
6: Run Lazy-MaxWalkSAT on the lazy grounded network constructed by $(\mathcal{X}^*, \mathcal{F}^*, \mathcal{DB}^*)$;
7: **end if**

6 Experimental evaluation

The goal of our experimental evaluation is to investigate the following key questions.

- (**Q1**) Is the WSP-Dec algorithm competitive with the state-of-the-art inference algorithms for finding an optimal MAP solution?
- (**Q2**) Is WSP-χ powerful enough to significantly reduce the size of grounded networks compared to the prominent state-of-the-art scalable methods such as Lazy Inference?
- (**Q3**) How is the behavior of WSP-Dec influenced by the choice of the cooling parameter y and magnetization threshold $\mathcal{T}$?
- (**Q4**) How is the performance of WSP-Dec algorithm affected by tuning the value of the smoothing parameter χ in WSP-χ?
- (**Q5**) Does the combination of WSP-χ with Lazy inference, i.e., Lazy-WSP-Dec, improve the efficiency of WSP-χ based MAP inference?
- (**Q6**) Is the WSP-Dec algorithm efficient compared to state-of-the-art MAP inference algorithms on other tractable graphical models such as relational sum-product networks (RSPNs)?

We experiment on three tasks: a protein interaction, an hyperlink analysis, and an entity resolution in a citation matching domain. We used both the MLNs and datasets available from the Alchemy web page[4]

Protein Interaction We used the *Yeast* dataset that captures information about a protein's location, function, phenotype, etc. It contains four subsets, each of which contains the information of about 450 proteins.

- **MLN:** We used the MLN model [13] that involves singleton rules for predicting the interaction relationship, and rules describing how protein functions relate to interactions between proteins (i.e. two interacting proteins tend to have similar functions). The final knowledge base has 7 atoms and 8 first-order formulas.
- **Query:** The goal of MAP inference is to predict MAP solution of interaction relations (i.e., *Interaction, and Function*). All other atoms (e.g., location, protein-class, enzyme, etc.) are considered evidence atoms.

Hyperlink Analysis We used the *WebKB* dataset that consists of labeled web pages from the computer science departments of four universities. It features 4165 web pages and 10,935 web links, along with the words on the webpages, anchors of the links, and neighborhoods around each link. Each web page is marked with some subset of the categories: person, student, faculty, professor, department, research project, and course.

- **MLN:** We used the MLN model [38] that involves only formulas linking words to page classes, and page classes to the classes of linked pages. The final knowledge base contains 3 atoms and 6 formulas.
- **Query:** The goal of MAP inference is to predict MAP solution of the web pages point to each other predict and their categories from the web pages' words and link structures, given their topics.

[4] http://alchemy.cs.washington.edu/

Entity Resolution We used the *Cora* dataset which consists of 1295 citations of 132 different computer science papers. Recently, the dataset was cleaned and split into five subsets for cross-validation [67].

- **MLN:** We used the MLN model [67] that involves formulas stating regularities such as: if two citations are the same, their fields are the same; if two fields are the same, their citations are the same. Also it has formulas representing transitivity, which are assigned very high weight. The final knowledge base contains 10 atoms and 32 formulas.
- **Query:** The goal of MAP inference is to predict MAP solution to predict which pairs of citations refer to the same citation (*SameBib*), and similarly for author, title and venue fields (*SameTitle, SameAuthor and SameVenue*). The other atoms are considered evidence atoms.

6.1 Methodology

To evaluate WSP-Dec, we compare its results with the following prominent inference algorithms:

1. **MaxWalkSAT** (MWS) algorithm [25, 66] and its lazy version (Lazy-MWS) which is the state-of-the-art MAP inference in SRL systems like TUFFY [49], LoMRF [69] and ALCHEMY [31]. We here use MWS to serve here as a good baseline in our experiments
2. **Weighted constraint satisfaction problem** (WCSP) [3, 23], an exact solver that is currently used as the state-of-the-art MAP inference in PRACMLN engine[5]
3. **RockIt**[6] inference query engine [50]. It applies integer linear programming-based meta algorithms such as cutting plane inference and cutting plane aggregation. It also leverages parallelism and symmetry to scale up MAP inference in SRL models
4. **IPP**[7] algorithm [64], which is an integer polynomial programming based solver. IPP first converts the MAP problem to an Intger Linear Program and then it uses Gurobi[8] solver to solve it exactly
5. **SP-Y** algorithm [5], which is a popular survey propagation algorithm for solving MAP inference and MAX-SAT problems

In addition, to obtain robust answers to the proposed questions, we apply the following:

- Varying the number of objects in the domains, following the methodology previously used for MLNs [57]
- Varying the cooling parameter, following the methodology used for relaxed SP [10]
- Varying the magnetization threshold $\mathcal{T} \in \{0.2, 0.5\}$. Thus we mainly considered two WSP-Dec algorithms, **WSP-Dec-0.2 and WSP-Dec-0.5**, on which we selected the ground atom as a frozen if its magnetization is greater than 0.2 and 0.5, respectively.

We then conduct the experimental evaluations in the following manner. In the training phase, we learned the weights using a preconditioned scaled conjugate gradient (PSCG) algorithm [38] by performing a four-way cross-validation for protein interaction task, and a five-way cross-validation for both the link prediction and entity resolution tasks. In the testing phase, we carried out a MAP inference on the held-out dataset using six

[5]Publicly available at: http://www.pracmln.org/index.html
[6]Publicly available at: https://github.com/jnoessner/rockIt
[7]Publicly available at: http://www.utdallas.edu/∼somdeb.sarkhel/software.html
[8]Publicly available at: http://www.gurobi.com/

underlying inference algorithms (WSP-Dec-0.2, WSP-Dec-0.5, MWS, Lazy-WSP-Dec-0.2, Lazy-WSP-Dec-0.5, Lazy-MWS) to produce the MAP solution.

All of the experiments were run on a cluster of nodes with 3.0 GHz Intel CPUs, 3 GB of RAM, RED HAT Linux 5.5. We used the MWS algorithm and its lazy version as implemented in the Alchemy software [31], and took advantage of the SP code[9] to implement our WSP-χ as an extension to the PRACMLN[10] software [31]. That is, we implemented WSP-χ as a separate program to find the frozen atoms, then output of this program was passed as additional evidence to PRACMLN engine for the MAP inference.

6.2 Metrics

In order to compare the performance and scalability of the testbed algorithms we considered three metrics:

- The quality of MAP solution as a function of the running time.
- The quality of MAP solution as a function of cooling parameter.
- The average percentage of fixed (frozen) ground atoms.

where the quality of MAP solution is measured by computing the average cost of unsatisfied ground clauses by the obtained MAP solution. It is worth noting that solving the MAP inference here is equivalent to solving a weight MAX-SAT problem where the goal is to find the MAP solution that maximizes the total weight of the satisfying clauses (which is identical to minimize the total weight of the unsatisfying clauses). Thus considering the average cost of unsatisfied ground clauses serves as a quite good measurement to test the quality of the obtained MAP solution.

6.3 Results

We conduct our empirical evaluations through three experiments.

6.3.1 Experiment I

Figures 3, 4 and 5 display the average cost of unsatisfied clauses (smaller is better) as a function of time for the six underlying inference algorithms at three different numbers of objects in the domains of Cora, WebKB, and Yeast datasets respectively. Notation used to label each of these figures is as: MLN-*number-of-objects* (*number of ground clauses in the propositional MLN*). The absence of some algorithms in some plots means that no results have been obtained since they ran out of memory.

Overall WSP-Dec algorithm (at thresholds 0.2 and 0.5) is the most accurate of all the algorithms the compared, achieving the best solution quality on the three underlying datasets. It finishes at least 19% more accurately than IPP on both WebKB and Cora datasets, and 28% more accurately on the Yeast dataset. RockIT came close behind WSP-Dec algorithm only on both WebKB-50 and Cora-250. WSP-Dec dominates both WCSP and MaxWalkSAT on the three datasets. IPP was also very competitive with RockIT and

[9] Available: http://users.ictp.it/~zecchina/SP/

[10] Publicly available at: http://www.pracmln.org/index.html

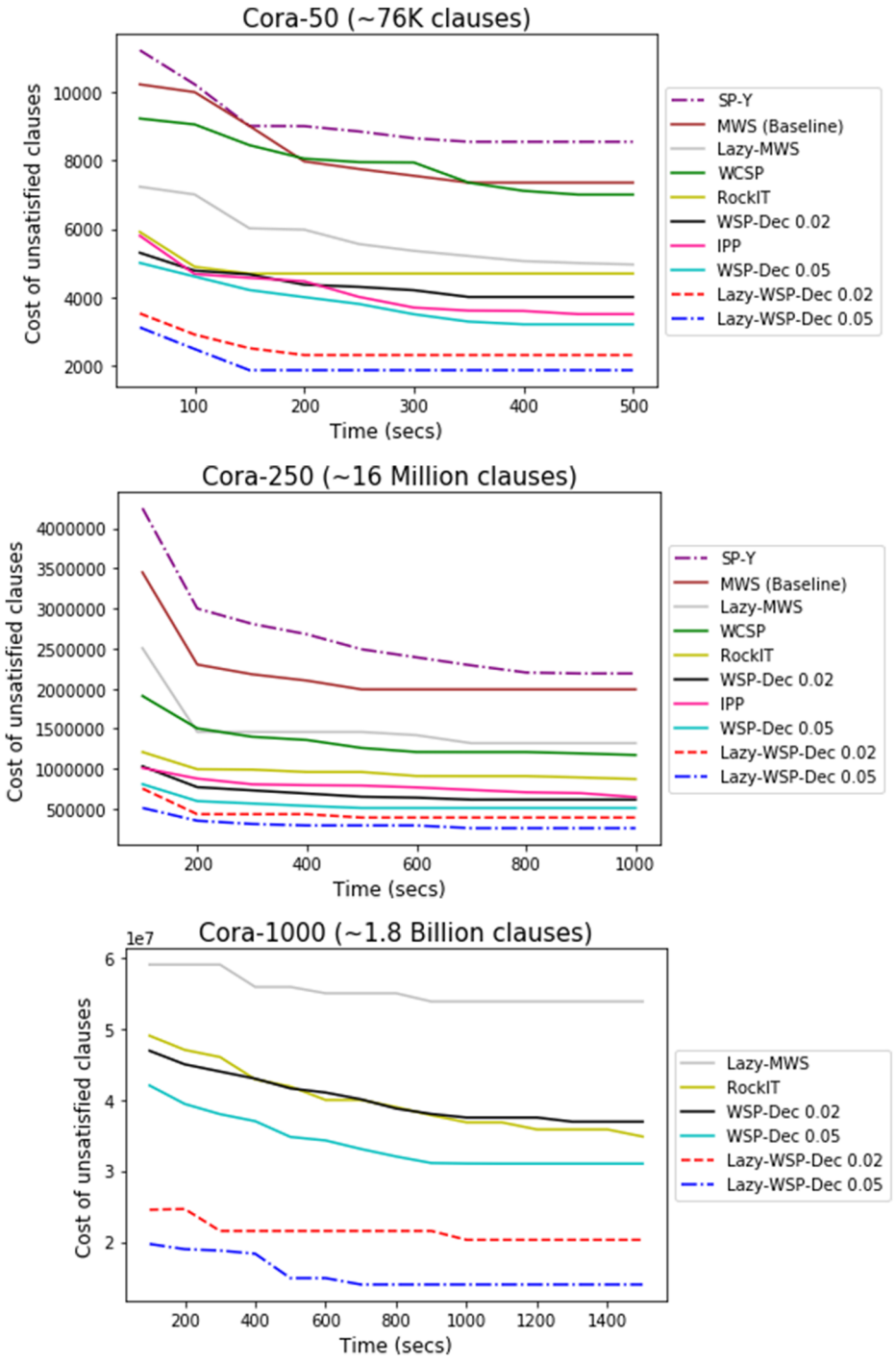

Fig. 3 Cost vs. Time: average cost of unsatisfied clauses (smaller is better) against time for Cora at 50 objects (*top*), 250 objects (*middle*) and 1000 objects (*bottom*)

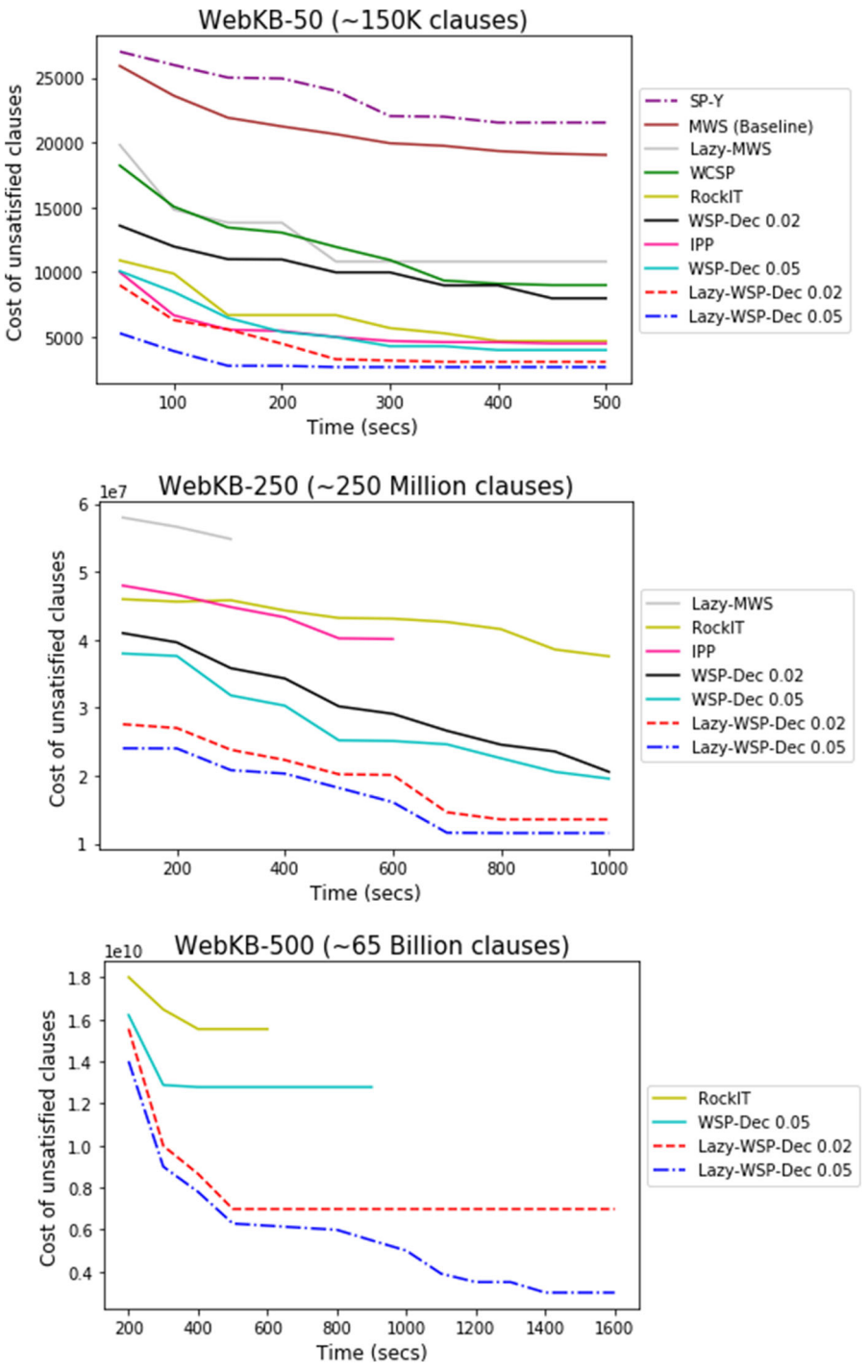

Fig. 4 Cost vs. Time: average cost of unsatisfied clauses (smaller is better) against time for WebKB at 50 objects (*top*), 250 objects (*middle*) and 500 objects (*bottom*)

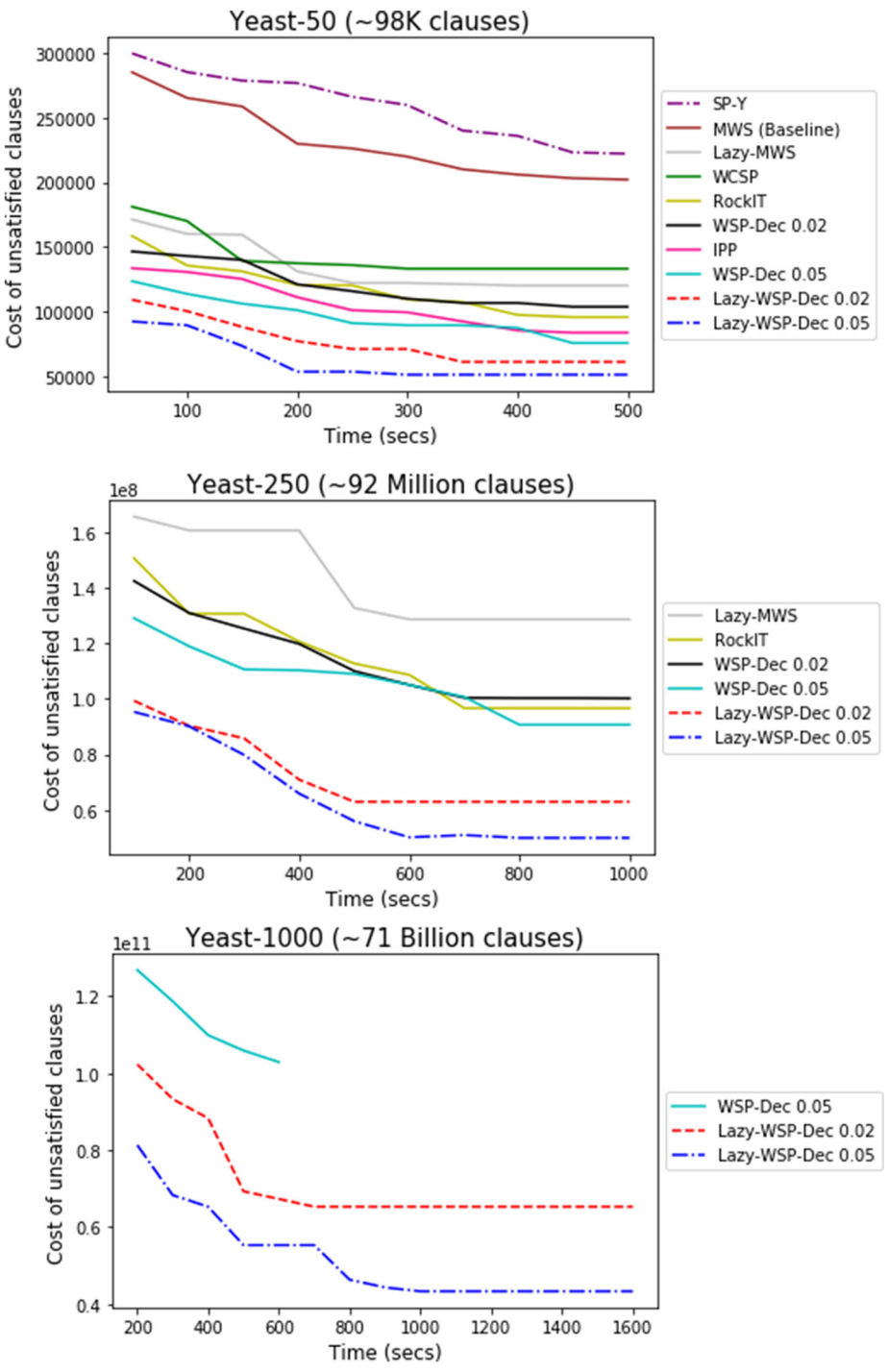

Fig. 5 Cost vs. Time: average cost of unsatisfied clauses (smaller is better) against time for Yeast at 50 objects (*top*), 250 objects (*middle*) and 1000 objects (*bottom*)

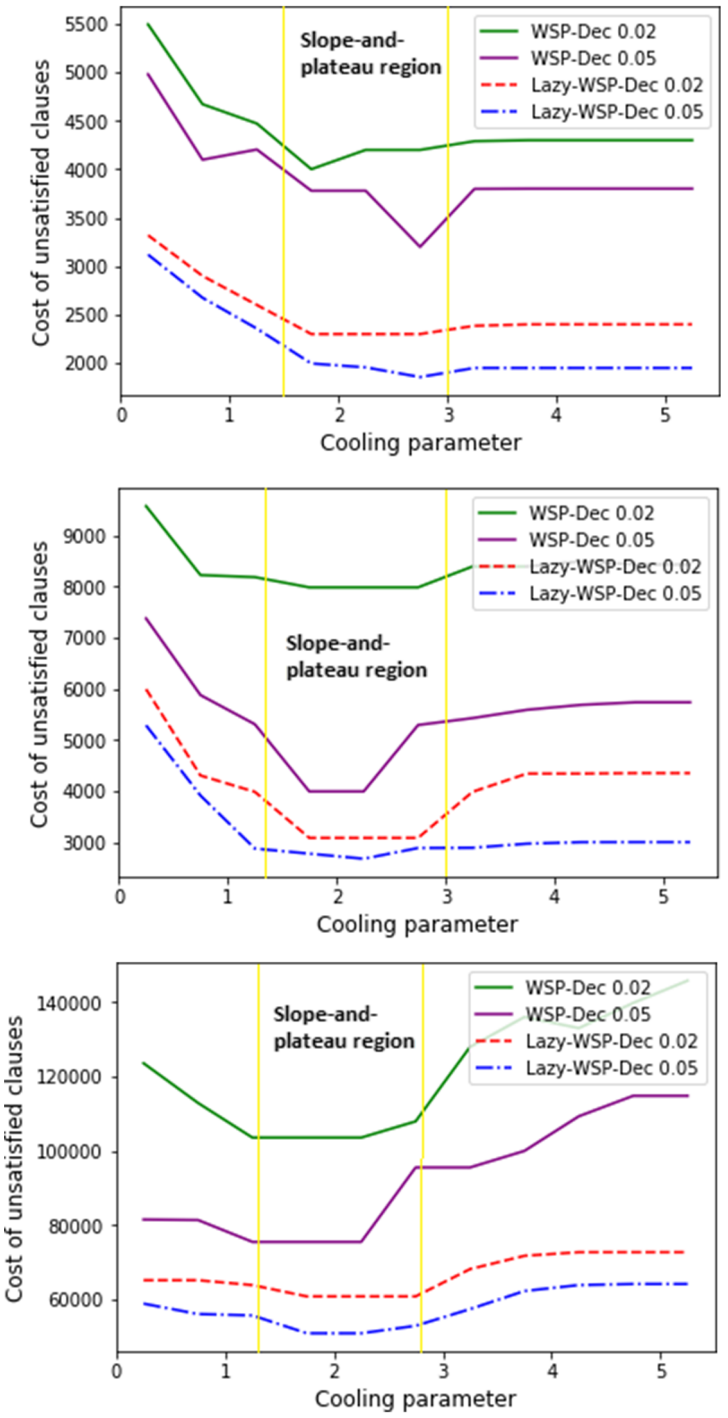

Fig. 6 Cost vs. cooling parameter: average cost of unsatisfied clauses against different values of cooling parameter y of WSP-Dec algorithm on Cora (*top*), WebKB (*middle*) and Yeast (*bottom*)

WSP-Dec-0.02 when grounding the Cora, WebKB and Yeast models with small number of objects (i.e., 50 objects). In addition, lazy-WSP-Dec (at thresholds 0.2, 0.5) substantially outperformed SP-Dec algorithm on all underlying tested datasets. Clearly, the WSP-Dec algorithms at threshold 0.5 were marginally more accurate than the WSP-Dec algorithms at threshold 0.2 on both Cora, WebKB and Yeast datasets.

In terms of scalability, Lazy-WSP-Dec at thresholds 0.2, 0.5 and WSP-Dec at threshold 0.5 were able to handle all full datasets, whereas all other tested inference algorithms ran out of memory with 1000 objects in the Yeast dataset. Additionally, both Lazy-WSP-Dec at thresholds 0.2, 0.5 dominated IPP, which ran out of memory with 1000 objects in both Cora and 500 objects in the WebKB dataset. Apart from lazy variants of WSP-Dec, propositional WSP-Dec at thresholds 0.2, 0.5 finishes at least four orders of magnitude faster than the WCSP solver with objects 250 in Cora dataset, and at least two times faster than RockIT solver with objects 250 in WebKB dataset.

6.3.2 Experiment II

Figure 6 displays the average cost of unsatisfied clauses as a function of different cooling parameter's values ($y \in [0.25, 5.25]$) of the WSP-Dec algorithm and the Lazy-WSP-Dec algorithm at thresholds 0.2, 0.5 for 50 objects in the domains of the three underlying datasets. The result shows that the WSP-Dec algorithm reaches a slope-and-plateau region where its quality of solution increases and then starts to decease. In the three tested datasets, this slope-and-plateau region occurred when the cooling y parameter's value is approximately between 1.5 and 3 (*answering the part of Q3 relates to cooling parameter*).

6.3.3 Experiment III

Table 3 records both the average cost of unsatisfied clauses and the percentage of the frozen ground atoms that are fixed by the WSP-Dec family of algorithms at four pairs of settings of (χ, γ) on Cora, Web-KB, and Yeast. Overall, the most successful pair of WSP-Dec algorithm was $(\chi = 1, \gamma = 0)$ in all tested datasets. For this setting, the decimation step fixed approximately $30\% - 51\%$ of the query ground atoms (i.e., it obtains a portion that is $30 - 51\%$ of the optimal MAP solution). By contrast, the poorest pair setting was $(\chi = 0, \gamma = 1)$, the decimation step of the WSP-Dec algorithm fixed at most 10% of the query ground atoms, reaching to small portions of the MAP solutions. In addition, the algorithm with the pair $(\chi = 0.5, \gamma = 0.5)$ was more accurate than the pair $(\chi = 0.25, \gamma = 0.75)$ in terms of both quality of MAP solution, and it has a larger amount of fixed frozen atoms (*conclusive answer to Q4*).

6.3.4 Experiment IV

This experiment was performed to answer Q6. That is, we are interested in comparing our WSP-Dec to the prominent **LeanRSPN-based** MAP inference method on relational sum-product networks (RSPNs) [47] —RSPN is a relational tractable representation of sum-product network [56]. To guarantee a fair scalability comparison, we used 200 objects to ground the models. In the training phase, we re-ran experiment I for WSP-Dec algorithms while in LeanRSPN-based method we used the Nath et. al's python implementation of LearnRSPN algorithm [47] to train the RSPN model. In the testing phase, and using the learned models, we repeatedly assigned all tested algorithms an identical running time (i.e., 1000 secs) sufficient to judge the inference behavior. At the testing of LeanRSPN-based

Table 3 The percentage of the frozen ground atoms (i.e., cluster backbones) that are fixed (fixed%) and the average cost of unsatisfied clauses (Cost) for a family of WSP-Dec at different choices of smoothing pairs (χ, γ) on Cora, Web-KB, and Yeast

		WSP-Dec algorithms							
		$\chi = 1, \gamma = 0$		$\chi = 0.5, \gamma = 0.5$		$\chi = 0.25, \gamma = 0.75$		$\chi = 0, \gamma = 1$	
Datasets	No. Obj.	fixed%	Cost	fixed%	Cost	fixed%	Cost	fixed%	Cost
Cora	50	**43.1**	3.2×10^3	30.8	4.4×10^3	21.1	5.4×10^3	6.4	6.6×10^3
	250	**40.5**	5.1×10^5	29.7	8.2×10^5	18.8	1.2×10^6	5.6	1.5×10^6
	1000	**51.7**	3.1×10^7	**36.5**	6.1×10^7	24.3	7.9×10^7	10	9.7×10^7
Web-KB	50	**35.5**	3.9×10^3	26.2	9.1×10^3	16.0	1.4×10^4	7.0	1.6×10^4
	250	**41.0**	1.9×10^7	30.0	5.2×10^7	17.9	9.5×10^7	8.3	1.2×10^8
	500	**48.2**	1.2×10^{10}	**35.3**	3.4×10^{10}	22.5	6.8×10^{10}	9.9	9.3×10^{10}
Yeast	50	**47.0**	7.5×10^4	**34.0**	1.3×10^5	21.6	5.8×10^5	9.0	9.6×10^5
	250	**38.4**	9.0×10^7	28.9	1.2×10^8	16.7	3.6×10^8	7.8	8.1×10^8
	1000	30.5	1.0×10^{11}	22.6	1.6×10^{11}	13.4	2.7×10^{11}	4.3	7.1×10^{11}

The cooling parameter y assigned a value 2 and the threshold takes a value 0.5. The values appear in bold are the cases in which WSP-Dec algorithm variants are able to fix more than one-third of the total number of ground atoms (i.e., $> 33\%$)

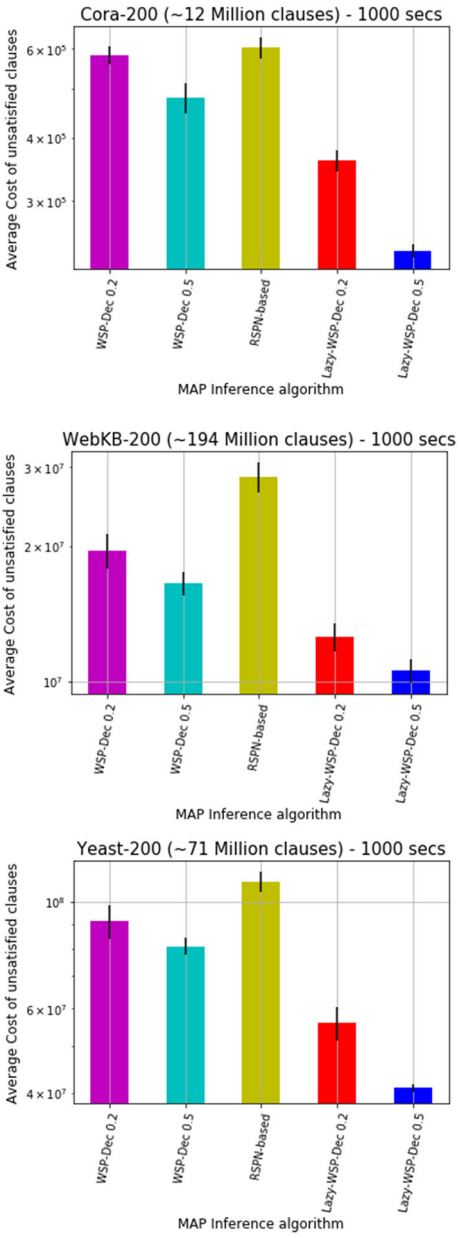

Fig. 7 Log scale of the average cost of unsatisfied clauses (smaller is better) of WSP-Dec variants and LeanRSPN-based MAP inference algorithm for Cora (*top*), WebKB (*middle*) and Yeast (*bottom*) after grounding the models with 200 objects

inference, we leveraged the MPE inference implemented in SPFlow library[11] [44] to obtain the MAP solution.

Figures 7 reports the log scale of the average cost of unsatisfied clauses obtained after 5 runs of tested MAP inference algorithms, on which at each run we used different ran-

[11] Publicly available at: https://github.com/SPFlow/SPFlow

dom initialization. In terms of performance, the results show that lazy variants of WSP-Dec algorithm (at thresholds 0.2 and 0.5) are the most accurate of all the tested algorithms. LeanRSPN-based MAP inference came close behind WSP-Dec 0.2 on Cora dataset while it was considerably less accurate on Yeast and WebKB datasets since the sizes of networks dramatically increase. From the model representation viewpoint, the preliminary results of this experiment clearly show that our proposed relational parameterized factor graphs (i.e., WSP-χ) are potentially more tractable than RSPN in relational domain.

7 Discussion

Overall the results clearly show that WSP-χ based algorithms substantially improve the accuracy and scalability of the propositional MaxWalkSAT algorithm for MAP inference. This is due to, first, finding the frozen ground atoms, which provides a large portion of the optimal MAP solution. Second, fixing the frozen atoms simplifies the MAP inference task into another one that can be solved accurately using any conventional MAP inference algorithm.

WSP-Dec algorithms were also very competitive with respect to Lazy MAP Inference whenever a substantial amount of frozen atoms were obtained. This can be attributed to the fact that fixing the frozen atoms enlarges the evidence database and shrinks the query set which provides great implications for reducing the effective size of the grounded network. Moreover, the WSP-Dec algorithm dominated both propositional MaxWalkSAT and Lazy-MaxWalkSAT on all tested data sets when it combined with Lazy Inference. This is because the result of such combination is the exploitation of both sparseness in Lazy and frozen atoms from WSP to scale up the MAP inference.

The magnetization threshold $\mathcal{T}$ offers a trade-off for WSP-Dec algorithm in terms of the amount of frozen atoms/the quality of obtained MAP solution: on one hand decreasing $\mathcal{T}$'s value enables one to obtain a large amount of frozen ground atoms and therefore improve the scalability, but on the other hand some of those frozen atoms could be inaccurate and that can effect the quality of the final obtained MAP solution. Thus, in the presence of a huge grounded network, one can choose to slightly scarify the quality of the solution by decreasing $\mathcal{T}$ just to enable the WSP-Dec algorithm to find a MAP solution. For instance, with 1000 objects in the domain of Yeast dataset - on which its MLN features diversity in the weights) - WSP-Dec 0.5 has difficulties reaching a MAP solution, whereas WSP-Dec 0.2 algorithm can find one.

The results in Experiment II show that the behaviour of the WSP-Dec algorithm over varying cooling parameter y is consistent with theorem 2, ensuring that as long as WSP-χ converges, its performance improves as cooling parameter y increases. That is to say, the marginals computed by the WSP-Dec algorithm will prefer MAP solutions that satisfy the clauses with a maximum total weight. However, at a certain point of increasing y, WSP can fail to converge to accurate results which may be attributed to overshooting the optimal solution. Also, choosing a small value of y can slow the convergence. Experimentally, we find that we need to take it to a sufficiently large value between 1.5 and 3 to obtain high convergence to an accurate result. This range of y is very close to the work of Battaglia et al. (2004), who found that $y = 2.5$ worked very well for the SP-y algorithm. Although one can use a bisection method to numerically obtain the ideal cooling value y beforehand, we believe that the value of y can be dynamically tuned to favor the convergence of the WSP-Dec algorithm, which will be an interesting analysis for future research.

From efficiency perspective, the results in experiment IV hypothetically emphasize that designing a tractable graphical model representation, based on certain characteristics (e.g., determinism, sparseness, independences, etc.), often has a key impact on the MAP solution's quality. That is, RSPN compactly leverages context-specific independence (i.e., the children of a sum node have different decomposition in their product nodes) for an efficient marginal inference. While the foundation of context-specifig independence makes RSPNs more expressiveness than low-treewidth tractable models (and therefore provide accurate marginal probability and partition function), this foundation however hinders the principle of exploiting relational dependencies. This in turn could frequently leads RSPN to provide inaccurate MAP assignment in relational domains — cf. the theoretical analysis of SPNs presented in [12, 42]. On contrary, our WSP-χ representation is designed for the specific purpose of relational MAP inference. It basically exploits both relational dependencies and local logical structures to heuristically guide traditional local search-based MAP inference methods. This could explain why WSP-χ model provide more accurate MAP solutions than RSPNs.

Furthermore, the performance of WSP-χ algorithms on the extended factor graph significantly effected by the choice of smoothing parameter χ, and this appears clearly in the results of Table 3. That is to say, given the pool of WSP-χ algorithms, increasing the value of χ (or equivalently decreasing the value of γ) allows the algorithm to obtain more frozen ground atoms, which results in enlarging the evidence database and shrinking the query set, and therefore improving the scalability besides finding a larger portion for the optimal MAP solution. Thus, when setting χ to the most (i.e., $\chi = 1$ and $\gamma = 0$), a call to MaxWalkSAT might in fact not be needed or only needed to solve an easy MAP inference on a scalable grounded network. On the contrary, decreasing χ to the most (i.e., $\chi = 0$ and $\gamma = 1$) reduces WSP-χ to traditional BP, and this makes the calling of MaxWalkSAT often faces an hard MAP inference on a simplified grounded network that is very close to the propositional one. In addition, the success of pure WSP-1 for finding the most accurate results, supports the conjecture that MAP solutions of relational problems typically do possess non-trivial max-cores for large structured formulas.

8 Conclusion

It is widely known that many real-world problems can be formulated using expressive SRL models that feature logical structures with very high densities, and this renders them identical to hard satisfiability instances. We believe a clear gap in the present literature on MAP inference in SRL exists with respect to taking into consideration the fact that the solution space of such problems is frequently clustered when the density of the underlying model is high or close to a critical threshold.

Ignoring the clustering that occurs in the solution space can engender intricacies of getting stuck in a metastable cluster at local optimum when handling the inference using some state-of-the-art techniques like local search, max-product message-passing or LP relaxation based algorithms. The novelty in this paper lies in twofold: First, we present a new family of extended factor graphs WSP-χ associated with a family of Weighted Survey Propagation algorithms applicable to SRL models. The objective of WSP-χ is to identify the backbones of a cluster containing potentially optimal MAP solutions. This introduces the WSP-χ family as a set of pre-processing methods that can help in finding the optimal solution in the presence of clustered solution spaces. Second, we propose lazy variants of the WSP-χ family of algorithms to improve scalability for MAP inference. Using real-world domains

such as protein interaction, hyperlink analysis and entity resolution, we have experimentally shown that WSP-χ and its lazy variants are able to greatly improve quality and scalability of MAP inference when integrated with propositional MaxWalkSAT and its lazy version.

To conclude, the approach of WSP-χ represents an improvement in relational MAP inference. By obtaining cluster backbones that determine which particular clusters contain potentially optimal solutions, not only is a large portion of the optimal solution provided in the cluster, but also fixing them helps getting inside the cluster by enlarging the evidence database and shrinking the query set. This therefore reduces the graph network into a scalable one that can be solved accurately using any conventional MAP inference method.

In the future we plan to apply WSP-χ to the Tuffy system to get an approximate and exact solution. Also we plan to combine WSP-χ with other inference algorithms such as Lifted Inference. We intend to derive an online MAP inference for WSP-χ that dynamically tunes the smoothing and cooling parameters to favor a better convergence for the WSP-Dec algorithm. We will also try to use WSP-Dec for solving CSPs, since a decimation combined with WSP-χ based message-passing can be viewed as a depth-first search combined with a "highest bias" heuristic. Finally, we will attempt to derive a new adaptation of WSP-χ-based decimation that can be directly applied to other tractable representations of probabilistic graphical model like relational sum-product networks.

Appendix A: Derivation of WSP-χ's update equations

Here we derive the update equations for WSP-χ's message passing. For simplicity, and without lose of generality, we consider the derivation of WSP-1 — a pure version of WSP-χ on $\hat{\mathcal{G}}$ when setting $\chi = 1$ and $\gamma = 0$ in (8).

A1. Variable-to-factor

Let us start here by computing the update of the component $\mu^s_{X_j \to \hat{f}_i}$. This component represents the probability that X_j is constrained by other extended factors to satisfy $\hat{f}_i$, and therefore, it is specified by the event that the variable $X_j = s_{i,j}$ and its mega-node $P_j = Z^j \cup \{\hat{f}_i\}$. If we use $P_j = Z^j \cup \{\hat{f}_i\}$ as a notation representing the following event for a ground atom X_j

$$\hat{f}_i \in P_j \ \text{ and } \ Z^j = P_j \setminus \{\hat{f}_i\} \subseteq \mathcal{F}^s_{\hat{f}_i}(j) \tag{18}$$

Then we can compute $\mu^s_{X_j \to \hat{f}_i}$ as follows:

$$\mu^s_{X_j \to \hat{f}_i} = \sum_{Z^j \subseteq \mathcal{F}^s_{\hat{f}_i}(j)} \left\{ \eta_{\hat{f}_i \to X_j} \middle| X_j = s_{i,j}, P_j = Z^j \cup \{\hat{f}_i\} \right\} \tag{19a}$$

$$\mu^s_{X_j \to \hat{f}_i} = \sum_{Z^j \subseteq \mathcal{F}^s_{\hat{f}_i}(j)} \prod_{\hat{f}_k \in Z^j} \eta^s_{\hat{f}_k \to X_j} \prod_{\hat{f}_k \in \mathcal{F}^s_{\hat{f}_i}(j) \setminus Z^j} \eta^*_{\hat{f}_k \to X_j} \prod_{\hat{f}_k \in \mathcal{F}^u_{\hat{f}_i}(j)} \eta^u_{\hat{f}_k \to X_j} \tag{19b}$$

$$= \left[\prod_{\hat{f}_k \in \mathcal{F}^s_{\hat{f}_i}(j)} \left(\eta^s_{\hat{f}_k \to X_j} + \eta^*_{\hat{f}_k \to X_j} \right) \right] \prod_{\hat{f}_k \in \mathcal{F}^u_{\hat{f}_i}(j)} \eta^u_{\hat{f}_k \to X_j} \tag{19c}$$

Similarly for $\mu^{u}_{X_j \to \hat{f}_i}$. This component is specified by the event that $X_j = u_{i,j}$ and its mega-node $P_j \subseteq \mathcal{F}^{u}_{\hat{f}_i}(j)$. Thus, we have:

$$\mu^{u}_{X_j \to \hat{f}_i} = \sum_{Z^j \subseteq \mathcal{F}^{u}_{\hat{f}_i}(j)} \left\{ \eta_{\hat{f}_i \to X_j} \middle| X_j = u_{i,j},\, P_j = Z^j \right\} \tag{20a}$$

$$= \sum_{Z^j \subseteq \mathcal{F}^{u}_{\hat{f}_i}(j), Z^j \neq \emptyset} \prod_{\hat{f}_k \in Z^j} \eta^{s}_{\hat{f}_k \to X_j} \prod_{\hat{f}_k \in \mathcal{F}^{u}_{\hat{f}_i}(j) \setminus Z^j} \eta^{*}_{\hat{f}_k \to X_j} \prod_{\hat{f}_k \in \mathcal{F}^{s}_{\hat{f}_i}(j)} \eta^{u}_{\hat{f}_k \to X_j} \tag{20b}$$

$$- \prod_{\hat{f}_k \in \mathcal{F}^{s}_{\hat{f}_i}(j)} \eta^{*}_{\hat{f}_k \to X_j} \prod_{\hat{f}_k \in \mathcal{F}^{s}_{\hat{f}_i}(j)} \eta^{u}_{\hat{f}_k \to X_j}$$

$$= \left[\prod_{\hat{f}_k \in \mathcal{F}^{u}_{\hat{f}_i}(j)} \left(\eta^{s}_{\hat{f}_k \to X_j} + \eta^{*}_{\hat{f}_k \to X_j} \right) - \prod_{\hat{f}_k \in \mathcal{F}^{s}_{\hat{f}_i}(j)} \eta^{*}_{\hat{f}_k \to X_j} \right] \prod_{\hat{f}_k \in \mathcal{F}^{s}_{\hat{f}_i}(j)} \eta^{u}_{\hat{f}_k \to X_j} \tag{20c}$$

Finally, computing $\mu^{*}_{X_j \to \hat{f}_i}$ is specified by the event that $X_j = s_{i,j}$ with $P_j = \mathcal{F}^{s}_{\hat{f}_i}(j)$, and $X_j = *$ with $P_j = \emptyset$. Thus we have the following:

$$\mu^{*}_{X_j \to \hat{f}_i} = \sum_{Z^j \subseteq \mathcal{F}^{s}_{\hat{f}_i}(j)} \left\{ \eta_{\hat{f}_i \to X_j} \middle| X_j = s_{i,j},\, P_j = Z^j \right\} + \left\{ \eta_{\hat{f}_i \to X_j} \middle| X_j = *,\, P_j = \emptyset \right\} \tag{21a}$$

$$= \sum_{Z^j \subseteq \mathcal{F}^{s}_{\hat{f}_i}(j), Z^j \neq \emptyset} \prod_{\hat{f}_k \in \mathcal{F}^{s}_{\hat{f}_i}(j)} \eta^{s}_{\hat{f}_k \to X_j} \prod_{\hat{f}_k \in \mathcal{F}^{s}_{\hat{f}_i}(j)} \eta^{*}_{\hat{f}_k \to X_j} \prod_{\hat{f}_k \in \mathcal{F}^{u}_{\hat{f}_i}(j)} \eta^{u}_{\hat{f}_k \to X_j}$$

$$- \prod_{\hat{f}_k \in \mathcal{F}^{u}_{\hat{f}_i}(j)} \eta^{*}_{\hat{f}_k \to X_j} \prod_{\hat{f}_k \in \mathcal{F}^{s}_{\hat{f}_i}(j)} \eta^{u}_{\hat{f}_k \to X_j} + \prod_{\hat{f}_k \in \mathcal{F}^{s}_{\hat{f}_i}(j)} \eta^{*}_{\hat{f}_k \to X_j} \prod_{\hat{f}_k \in \mathcal{F}^{u}_{\hat{f}_i}(j)} \eta^{*}_{\hat{f}_k \to X_j} \tag{21b}$$

$$= \left[\prod_{\hat{f}_k \in \mathcal{F}^{s}_{\hat{f}_i}(j)} \left(\eta^{s}_{\hat{f}_k \to X_j} + \eta^{*}_{\hat{f}_k \to X_j} \right) - \prod_{\hat{f}_k \in \mathcal{F}^{u}_{\hat{f}_i}(j)} \eta^{*}_{\hat{f}_k \to X_j} \right] \prod_{\hat{f}_k \in \mathcal{F}^{u}_{\hat{f}_i}(j)} \eta^{u}_{\hat{f}_k \to X_j}$$

$$+ \prod_{\hat{f}_k \in \mathcal{F}^{s}_{\hat{f}_i}(j) \cup \mathcal{F}^{u}_{\hat{f}_i}(j)} \eta^{*}_{\hat{f}_k \to X_j} \tag{21c}$$

A2. Factor-to-Variables

Let us start here with the component $\eta^{s}_{\hat{f}_i \to X_j}$. This component implies that $X_j = s_{i,j}$ and $\hat{f}_i \in P_j$, and that the only possible assignment for the other ground atoms $X_k \in \mathcal{X}_{\hat{f}_i} \setminus \{X_j\}$ is $u_{i,k}$ and their mega-nodes are $P_k \subseteq \mathcal{F}^{u}_{\hat{f}_i}(k)$. That is, it takes the form:

$$\eta^{s}_{\hat{f}_i \to X_j} = \prod_{X_k \in \mathcal{X}_{\hat{f}_i} \setminus \{X_j\}} \left(\overbrace{\sum_{P_k \subseteq \mathcal{F}^{u}_{\hat{f}_i}(k)} \left\{ \mu_{X_k \to \hat{f}_i} \middle| X_k = u_{i,k},\, P_k \subseteq \mathcal{F}^{u}_{\hat{f}_i}(k) \right\}}^{\text{From Eq. (20}a\text{) this equals } \mu^{u}_{X_k \to \hat{f}_i}} \right) \times \overbrace{e^{\hat{w}_i \cdot y}}^{\text{a reward term, see (6)}} \tag{22}$$

Note that since the component $\eta^s_{\hat{f}_i \to X_j}$ is constrained to satisfy $\hat{f}_i$, we multiply right hand side of (22) by the term $e^{\hat{w}_i \cdot y}$ which is the reward term of satisfying $\hat{f}_i$. Now, using the definition of $\mu^u_{X_k \to \hat{f}_i}$ from (20a) into (22), we obtain the following:

$$\eta^s_{\hat{f}_i \to X_j} = \left[\prod_{X_k \in \mathcal{X}_{\hat{f}_i} \setminus \{X_j\}} \mu^u_{X_k \to \hat{f}_i} \right] \times e^{\hat{w}_i \cdot y} \tag{23}$$

Now moving to the component $\eta^u_{\hat{f}_i \to X_j}$. This component represents the probability that X_j can violate $\hat{f}_i$. That is to say, we have $X_j = u_{i,j}$ and $P_j \subseteq \mathcal{F}^u_{\hat{f}_i}(j)$. This probability implies a combination of three possibilities (having weights labeled as W_1, W_2 and W_3) for the other ground atoms $X_k \in \mathcal{X}_{\hat{f}_i} \setminus \{X_j\}$ in a potential complete assignment:

1. There is one ground atom in $\mathcal{X}_{\hat{f}_i} \setminus \{X_j\}$ satisfying $\hat{f}_i$, and all the other ground atoms are violating it

$$\mathrm{W}_1 = \sum_{X_k \in \mathcal{X}_{\hat{f}_i} \setminus \{X_j\}} \overbrace{\sum_{Z^k \subseteq \mathcal{F}^s_{\hat{f}_i}(k)} \left\{ \mu_{X_k \to \hat{f}_i} \middle| X_k = s_{i,k},\, P_k = Z^k \cup \{\hat{f}_i\} \right\}}^{\text{From Eq. (19}a\text{) this equals } \mu^s_{X_k \to \hat{f}_i}}$$

$$\times \prod_{X_i \in \mathcal{X}_{\hat{f}_i} \setminus \{X_k, X_j\}} \overbrace{\sum_{Z^i \subseteq \mathcal{F}^u_{\hat{f}_i}(i)} \left\{ \mu_{X_i \to \hat{f}_i} \middle| X_i = u_{i,i},\, P_i = Z^i \right\}}^{\text{From Eq. (20}a\text{) this equals } \mu^u_{X_i \to \hat{f}_i}} \tag{24a}$$

$$= \sum_{X_k \in \mathcal{X}_{\hat{f}_i} \setminus \{X_j\}} \mu^s_{X_k \to \hat{f}_i} \times \prod_{X_i \in \mathcal{X}_{\hat{f}_i} \setminus \{X_k, X_j\}} \mu^u_{X_i \to \hat{f}_i} \tag{24b}$$

2. There are two or more ground atoms in $\mathcal{X}_{\hat{f}_i} \setminus \{X_j\}$ satisfying $\hat{f}_i$ or equal joker $*$, and all other ground atoms are violating it

$$\mathrm{W}_2 = \sum_{X_k \in \mathcal{X}_{\hat{f}_i} \setminus \{X_j\}} \left[\sum_{Z^k \subseteq \mathcal{F}^s_{\hat{f}_i}(k)} \left\{ \mu_{X_k \to \hat{f}_i} \middle| X_k = s_{i,k},\, P_k = Z^k \right\} + \left\{ \mu_{X_k \to \hat{f}_i} \middle| X_k = *,\, P_k = \emptyset \right\} \right]$$

$$\times \prod_{X_i \in \mathcal{X}_{\hat{f}_i} \setminus \{X_k, X_j\}} \sum_{Z^i \subseteq \mathcal{F}^u_{\hat{f}_i}(i)} \left\{ \mu_{X_i \to \hat{f}_i} \middle| X_i = u_{i,i},\, P_i = Z^i \right\} \tag{25a}$$

$$= \prod_{X_k \in \mathcal{X}_{\hat{f}_i} \setminus \{X_j\}} \left[\mu^u_{X_k \to \hat{f}_i} + \mu^*_{X_k \to \hat{f}_i} \right] - \sum_{X_k \in \mathcal{X}_{\hat{f}_i} \setminus \{X_j\}} \mu^*_{X_k \to \hat{f}_i} \times \prod_{X_i \in \mathcal{X}_{\hat{f}_i} \setminus \{X_k, X_j\}} \mu^u_{X_i \to \hat{f}_i}$$

$$- \prod_{X_k \in \mathcal{X}_{\hat{f}_i} \setminus \{X_j\}} \mu^u_{X_k \to \hat{f}_i} \tag{25b}$$

Note that the weight assigned to the event that each ground atom is either satisfying or $*$ is $\prod_{X_k \in \mathcal{X}_{\hat{f}_i} \setminus \{X_j\}} \left[\mu^u_{X_k \to \hat{f}_i} + \mu^*_{X_k \to \hat{f}_i} \right]$, and the weight W_2 is given by subtracting from this quantity the weight assigned to the event that there are not at least two joker ground atoms $*$ or satisfying. This event is a combination of two disjoint events that either all other ground atoms in $X_k \in \mathcal{X}_{\hat{f}_i} \setminus \{X_j\}$ are violating (which weight

$\prod_{X_k \in \mathcal{X}_{\hat{f}_i} \setminus \{X_j\}} \mu^u_{X_k \to \hat{f}_i}$) or that only one ground atom is $*$ or satisfying (with weight $\sum_{X_k \in \mathcal{X}_{\hat{f}_i} \setminus \{X_j\}} \mu^*_{X_k \to \hat{f}_i} \times \prod_{X_i \in \mathcal{X}_{\hat{f}_i} \setminus \{X_k, X_j\}} \mu^u_{X_i \to \hat{f}_i}$).

3. All other ground atoms in $\mathcal{X}_{\hat{f}_i} \setminus \{X_j\}$ are violating $\hat{f}_i$. So here, there is a penalty term $e^{-\hat{w}_i \cdot y}$ of violating $\hat{f}_i$ when updating the message:

$$\mathrm{W}_3 = \left[\prod_{X_k \in \mathcal{X}_{\hat{f}_i} \setminus \{X_j\}} \overbrace{\sum_{Z^k \subseteq \mathcal{F}^u_{\hat{f}_i}(k)} \left\{ \mu_{X_k \to \hat{f}_i} \middle| X_k = s_{i,k},\, P_k = Z^k \right\}}^{\text{From Eq. (6) this equals } \mu^u_{X_k \to \hat{f}_i}} \right] \times \overbrace{e^{-\hat{w}_i \cdot y}}^{\text{A penalty term, see (6)}} \tag{26a}$$

$$= \left[\prod_{X_k \in \mathcal{X}_{\hat{f}_i} \setminus \{X_j\}} \mu^u_{X_k \to \hat{f}_i} \right] \times e^{-\hat{w}_i \cdot y} \tag{26b}$$

Now, bringing together the weight forms of W_1, W_2, and W_3 from (24b), (25b) and (26b) results in:

$$\eta^u_{\hat{f}_i \to X_j} = \left[\prod_{X_k \in \mathcal{X}_{\hat{f}_i} \setminus \{X_j\}} \left(\mu^u_{X_k \to \hat{f}_i} + \mu^*_{X_k \to \hat{f}_i} \right) + \prod_{X_i \in \mathcal{X}_{\hat{f}_i} \setminus \{X_j, X_k\}} \mu^u_{X_i \to \hat{f}_i} \right.$$
$$\left. \sum_{X_k \in \mathcal{X}_{\hat{f}_i} \setminus \{X_j\}} \left(\mu^s_{X_k \to \hat{f}_i} - \mu^*_{X_k \to \hat{f}_i} \right) \right] - \left[\overbrace{(1 - e^{-\hat{w}_i \cdot y})}^{\text{penalty}} \prod_{X_k \in \mathcal{X}_{\hat{f}_i} \setminus \{X_j\}} \mu^u_{X_k \to \hat{f}_i} \right] \tag{27}$$

Finally, the component $\eta^*_{\hat{f}_i \to X_j}$ represents the probability that X_j can be unconstrained by $\hat{f}_i$. This probability is a combination of two possibilities: either X_j is satisfying $\hat{f}_i$ and all other ground atoms are unconstrained, or X_j is unconstrained (i.e., $X_j = *$ with $P_i = \emptyset$). So we have:

$$\eta^*_{\hat{f}_i \to X_j} = \sum_{X_k \in \mathcal{X}_{\hat{f}_i} \setminus \{X_j\}} \left[\sum_{Z^k \subseteq \mathcal{F}^s_{\hat{f}_i}(k)} \left\{ \mu_{X_k \to \hat{f}_i} \middle| X_k = s_{i,k},\, P_k = Z^k \right\} + \left\{ \mu_{X_k \to \hat{f}_i} \middle| X_k = *,\, P_k = \emptyset \right\} \right] x \tag{28}$$

Note that the first part of (25a) and (25b) is identical to (28). Thus, we substitute the computation of this part from (25a) and (25b) into (28), and we have:

$$\eta^*_{\hat{f}_i \to X_j} = \left[\prod_{X_k \in \mathcal{X}_{\hat{f}_i} \setminus \{X_j\}} \left(\mu^u_{X_k \to \hat{f}_i} + \mu^*_{X_k \to \hat{f}_i} \right) \right] - \prod_{X_k \in \mathcal{X}_{\hat{f}_i} \setminus \{X_j\}} \mu^u_{X_k \to \hat{f}_i} \tag{29}$$

A3. Estimating the Marginals

Now let us explain the derivation of ground atoms' marginals over max-cores in $\hat{\mathcal{G}}$. Computing the unnormalized positive marginal of a ground atom X_j requires multiplying the

satisfying income messages from the ground clauses in which X_j appears positively by the violating income messages from the ground clauses in which X_j appears negatively:

$$\tilde{\theta}_j^+ = \prod_{\hat{f}_i \in \mathcal{F}^s(j)} \sum_{\mathcal{F}(i)} \left\{ \eta_{\hat{f}_i \to X_j} \middle| X_j = s_{i,j}, P_j = \mathcal{F}^s(j) \right\} \times \prod_{\hat{f}_i \in \mathcal{F}^u(j)} \sum_{\mathcal{F}(i)} \left\{ \eta_{\hat{f}_i \to X_j} \middle| X_j = u_{i,j}, P_j = \mathcal{F}^u(j) \right\} \quad (30a)$$

$$= \prod_{\hat{f}_i \in \mathcal{F}^+(j)} \sum_{\mathcal{F}(i)} \left\{ \eta_{\hat{f}_i \to X_j} \middle| X_j = +, P_j = \mathcal{F}^+(j) \right\} \times \prod_{\hat{f}_i \in \mathcal{F}^-(j)} \sum_{\mathcal{F}(i)} \left\{ \eta_{\hat{f}_i \to X_j} \middle| X_j = -, P_j = \mathcal{F}^-(j) \right\} \quad (30b)$$

$$= \prod_{\hat{f}_i \in \mathcal{F}^-(j)} \eta^u_{\hat{f}_i \to X_j} \times \left[\prod_{\hat{f}_i \in \mathcal{F}^+(j)} \left(\eta^s_{\hat{f}_i \to X_j} + \eta^*_{\hat{f}_i \to X_j} \right) - \prod_{\hat{f}_i \in \mathcal{F}^+(j)} \eta^*_{\hat{f}_i \to X_j} \right] \quad (30c)$$

Similarly, we can obtain the unnormalized negative marginal by multiplying the satisfying income messages from the factors in which X_i appears negatively by the violating income messages from the factors in which X_i appears positively:

$$\tilde{\theta}_j^- = \prod_{\hat{f}_i \in \mathcal{F}^+(j)} \eta^u_{\hat{f}_i \to X_j} \times \left[\prod_{\hat{f}_i \in \mathcal{F}^-(j)} \left(\eta^s_{\hat{f}_i \to X_j} + \eta^*_{\hat{f}_i \to X_j} \right) - \prod_{\hat{f}_i \in \mathcal{F}^-(j)} \eta^*_{\hat{f}_i \to X_j} \right] \quad (31)$$

Finally, we can estimate the unnormalized joker marginal by multiplying all the unconstrained incoming messages from all factors in which X_j appears:

$$\tilde{\theta}_j^* = \prod_{\hat{f}_i \in \mathcal{F}(j)} \left\{ \eta_{\hat{f}_i \to X_j} \middle| X_j = *, P_j = \emptyset \right\}$$

$$= \prod_{\hat{f}_i \in \mathcal{F}(j)} \eta^*_{\hat{f}_i \to X_j} \quad (32a)$$

Now by normalizing the quantities in (30c), (31) and (32a), we obtain the marginal of X_j as follows:

$$\theta_j^+ = \mathcal{Z}_j^{-1} \tilde{\theta}_j^+ \quad (33a)$$

$$\theta_j^- = \mathcal{Z}_j^{-1} \tilde{\theta}_j^- \quad (33b)$$

$$\theta_j^* = \mathcal{Z}_j^{-1} \tilde{\theta}_j^* \quad (33c)$$

and

$$\mathcal{Z}_j = \tilde{\theta}_j^+ + \tilde{\theta}_j^- + \tilde{\theta}_j^* \quad (34)$$

where $\mathcal{Z}_i$ is the normalizing constant, given the evidence E.

References

1. Achlioptas, D., Ricci-Tersenghi, F.: Random formulas have frozen variables. SIAM J Comput **39**(1), 260–280 (2009). SIAM
2. Ahmadi, B., Kersting, K., Mladenov, M., Natarajan, S.: Exploiting symmetries for scaling loopy belief propagation and relational training, vol. 92 (2013)
3. Allouche, D., de Givry, S., Schiex, T.: Toulbar2 an Open Source Exact Cost Function Network Solver. Technical report, INRIA (2010)
4. Amirian, M.M., Ghidary, S.S.: Xeggora: Exploiting immune-to-evidence symmetries with full aggregation in statistical relational models. J. Artif. Intell. Res. **66**, 33–56 (2019)
5. Battaglia, D., Kolár, M., Zecchina, R.: Minimizing energy below the glass thresholds. Phys. Rev. E **70**, 36107–36118 (2004)
6. Besag, J.: On the statistical analysis of dirty pictures. J R Stat Soc Series B stat Methodol **48**(3), 259–279 (1986)

7. Braunstein, A., Zecchina, R.: Survey and belief propagation on random k-sat. In: Proceedings of the 7th International Conference on Theory and Applications of Satisfiability Testing, vol. 2919, pp. 519–528. Springer, Vancouver (2004)
8. Braunstein, A., Mézard, M., Zecchina, R.: Survey propagation: an algorithm for satisfiability. Random Struct. Algorithm. **27**(2), 201–226 (2005)
9. Chavas, J., Furtlehner, C., Mézard, M., Zecchina, R.: Survey-propagation decimation through distributed local computations. J. Stat. Mech. Theory Exper. **2005**(11), 11016–11027 (2005). IOP Publishing
10. Chieu, H.L., Lee, W.S.: Relaxed survey propagation for the weighted maximum satisfiability problem. J. Artif. Intell. Res. (JAIR) **36**, 229–266 (2009)
11. Chieu, H.L., Lee, W.S., Teh, Y.W.: Cooled and relaxed survey propagation for mrfs. In: Proceedings of the 21st Annual Conference on Neural Information Processing Systems: Advances in Neural Information Processing Systems, vol. 20, pp. 297–304, Vancouver. Curran Associates, Inc. (2007)
12. Conaty, D., Maua, D., de Campos, C.: Approximation complexity of maximum a posteriori inference in sum-product networks. In: Proceedings of The 33rd Conference on Uncertainty in Artificial Intelligence, AUAI (2017)
13. Davis, J., Domingos, P.: Deep Transfer via Second-Order Markov Logic. In: Proceedings of the 26Th International Conference on Machine Learning (ICML-09), Montreal (2009)
14. De Salvo Braz, R., Amir, E., Roth, D.: Lifted first-order probabilistic inference. In: Proceedings of the 19th International joint conference in artificial intelligent, pp. 1319–1325. AAAI Press (2005)
15. De Salvo Braz, R., Amir, E., Roth, D.: Mpe and partial inversion in lifted probabilistic variable elimination. In: Proceedings Of The Twenty-first National Conference On Artificial Intelligence, vol. 6, pp. 1123–1130. AAAI press, Boston (2006)
16. Forney, G.D.: The viterbi algorithm. Proc. IEEE **61**(3), 268–278 (1973). IEEE computer Society
17. Getoor, L., Taskar, B.: Introduction to Statistical Relational Learning. Adaptive Computation and Machine Learning. The MIT Press (2007)
18. Gomes, C., Hogg, T., Walsh, T., Zhang, W.: Tutorial - Phase Transitions and Structure in Combinatorial Problems. In: Proceedings Of The Eighteenth National Conference On Artificial Intelligence. AAAI Press, Edmonton (2002)
19. Granville, V., Krivánek, M., Rasson, J.P.: Simulated annealing: a proof of convergence. IEEE Trans. Pattern Anal. Mach. Intell. **16**(6), 652–656 (1994). IEEE computer society
20. Hartmann, A.K., Weigt, M.: Phase transitions in combinatorial optimization problems: basics, algorithms and statistical mechanics. Wiley, New York (2006)
21. Huynh, T.N., Mooney, R.J.: Max-margin weight learning for markov logic networks. In: Machine Learning and Knowledge Discovery in Databases, vol. 5781, pp. 564–579. Springer (2009)
22. Ibrahim, M.H., Pal, C., Pesant, G.: Exploiting determinism to scale relational inference. In: Proceedings of the Twenty-Ninth National Conference on Artificial Intelligence (AAAI'15), pp. 1756–1762. AAAI Press, Austin (2015)
23. Jain, D., Maier, P., Wylezich, G.: Markov Logic as a Modelling Language for Weighted Constraint Satisfaction Problems. In: Eighth International Workshop on Constraint Modelling and Reformulation, in conjunction with CP (2009)
24. Kambhampati, S.C., Liu, T.: Phase transition and network structure in realistic sat problems. In: Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, pp. 1619–1620. AAAI Press, Washington (2013)
25. Kautz, H., Selman, B., Jiang, Y.: A general stochastic approach to solving problems with hard and soft constraints. Satisfiab Problem Theory Appl. **17**, 573–586 (1997)
26. Kazemi, S.M., Kimmig, A., Van den Broeck, G., Poole, D.: New liftable classes for first-order probabilistic inference. In: Advances in Neural Information Processing Systems, pp. 3117–3125 (2016)
27. Kersting, K.: Lifted probabilistic inference. In: Proceedings of 20th European Conference on Artificial Intelligence (ECAI–2012), vol. 27-31, pp. 33–38. ECCAI, Montpellier (2012)
28. Khosla, M., Melhorn, K., Panagiotou, K.: Message Passing Algorithms, PhD thesis. Citeseer (2009)
29. Kiddon, C., Domingos, P.: Coarse-to-fine inference and learning for first-order probabilistic models. In: Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, pp. 1049–1056. AAAI Press, San Francisco (2011)
30. Kilby, P., Slaney, J., Thiébaux, S., Walsh, T.: Backbones and backdoors in satisfiability. In: Proceedings of the The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, vol. 5, pp. 1368–1373. AAAI Press, Pittsburgh (2005)
31. Kok, S., Singla, P., Richardson, M., Domingos, P., Sumner, M., Poon, H., Lowd, D.: The Alchemy System for Statistical Relational AI. In: Technical Report Department of Computer Science and Engineering, University of Washington, Seattle. http://alchemy.cs.washington.edu (2007)
32. Kolmogorov, V.: Convergent tree-reweighted message passing for energy minimization. IEEE Trans. Pattern Anal. Mach. Intell. **28**(10), 1568–1583 (2006)

33. Kroc, L., Sabharwal, A., Selman, B.: Survey propagation revisited. In: Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence, pp. 217–226. AUAI Press, Vancouver (2007)
34. Kroc, L., Sabharwal, A., Selman, B.: Counting solution clusters in graph coloring problems using belief propagation. In: Proceedings of 22nd Conference on Neural Information Processing Systems: Advances in Neural Information Processing Systems, vol. 21, pp. 873–880. Curran Associates Inc., Vancouver (2008)
35. Kroc, L., Sabharwal, A., Selman, B.: Message-passing and local heuristics as decimation strategies for satisfiability. In: Proceedings of the 2009 ACM symposium on Applied Computing, pp. 1408–1414. ACM (2009)
36. Kumar, M.P., Torr, P.H.: Efficiently solving convex relaxations for map estimation. In: Proceedings of the 25th international conference on Machine learning, pp. 680–687. ACM, Helsinki (2008)
37. Lauritzen, S.L., Spiegelhalter, D.J.: Local computations with probabilities on graphical structures and their application to expert systems. J. R. Stat. Soc. Ser. B (Methodol.) **50**, 157–224 (1988)
38. Lowd, D., Domingos, P.: Efficient weight learning for markov logic networks. In: Proceedings of 11th European Conference on Principles and Practice of Knowledge Discovery in Databases PKDD 2007, pp. 200–211. Springer, Warsaw (2007)
39. Lüdtke, S., Schröder, M., Krüger, F., Bader, S., Kirste, T.: State-space abstractions for probabilistic inference: a systematic review. J. Artif. Intell. Res. **63**, 789–848 (2018)
40. Maneva, E., Mossel, E., Wainwright, M.J.: A new look at survey propagation and its generalizations. J. ACM (JACM) **54**(4), 17–21 (2007). ACM
41. Mann, A., Hartmann, A.: Numerical solution-space analysis of satisfiability problems. Phys. Rev. E 82(5), 056702–56707. APS (2010)
42. Mei, J., Jiang, Y., Tu, K.: Maximum a posteriori inference in sum-product networks. In: Thirty-Second AAAI Conference on Artificial Intelligence, pp. 1923–1930 (2018)
43. Meilicke, C., Leopold, H., Kuss, E., Stuckenschmidt, H., Reijers, H.A.: Overcoming individual process model matcher weaknesses using ensemble matching. Decis. Support. Syst. **100**, 15–26 (2017)
44. Molina, A., Vergari, A., Stelzner, K., Peharz, R., Subramani, P., Mauro, N.D., Poupart, P., Kersting, K.: Spflow: An easy and extensible library for deep probabilistic learning using sum-product networks. arXiv:1901.03704 (2019)
45. Montanari, A., Parisi, G., Ricci-Tersenghi, F.: Instability of one-step replica-symmetry-broken phase in satisfiability problems. J. Phys. A: Math. Gen. **37**(6), 2073–2079 (2004). IOP Publishing
46. Natarajan, S., Tadepalli, P., Dietterich, T.G., Fern, A.: Learning first-order probabilistic models with combining rules. Ann. Math. Artif. Intell. **54**(1-3), 223–256 (2008)
47. Nath, A., Domingos, P.M.: Learning relational sum-product networks. In: Twenty-Ninth AAAI Conference on Artificial Intelligence, pp 2878–2886 (2015)
48. Ng, K.S., Lloyd, J.W., Uther, W.T.: Probabilistic modelling, inference and learning using logical theories. Ann. Math. Artif. Intell. **54**(1-3), 159–205 (2008)
49. Niu, F., Ré, C., Doan, A., Shavlik, J.: Tuffy: Scaling up statistical inference in markov logic networks using an rdbms. Proc. VLDB Endow. **4**(6), 373–384 (2011)
50. Noessner, J., Niepert, M., Stuckenschmidt, H.: Rockit: Exploiting Parallelism and Symmetry for Map Inference in Statistical Relational Models. In: Twenty-Seventh AAAI Conference on Artificial Intelligence (2013)
51. Papai, T., Singla, P., Kautz, H.: Constraint propagation for efficient inference in markov logic. In: Proceedings of 17th International Conference on Principles and Practice of Constraint Programming (CP 2011), no. 6876 in Lecture Notes in Computer Science (LNCS), pp 691–705 (2011)
52. Park, J.D.: Using weighted max-sat engines to solve mpe. In: Proceedings of the Eighteenth National Conference on Artificial Intelligence, pp. 682–687. AAAI Press, Menlo Park (2002)
53. Parkes, A.J.: Clustering at the phase transition. In: Proceedings of the 14th National Conference on Artificial Intelligence, pp. 340–345. AAAI Press. at the convention center in Providence (1997)
54. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann Publishers Inc., San Francisco (1988)
55. Peharz, R., Gens, R., Pernkopf, F., Domingos, P.: On the latent variable interpretation in sum-product networks. IEEE Trans. Pattern Anal. Mach. Intell. **39**(10), 2030–2044 (2016)
56. Poon, H., Domingos, P.: Sum-Product Networks: a New Deep Architecture. In: 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), pp. 689–690. IEEE (2011)
57. Poon, H., Domingos, P., Sumner, M.: A general method for reducing the complexity of relational inference and its application to mcmc. In: Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, pp 1075–1080. AAAI Press, Chicago (2008)

58. Ravikumar, P., Lafferty, J.: Quadratic programming relaxations for metric labeling and markov random field map estimation. In: Proceedings of the 23rd international conference on Machine learning, pp. 737–744. ACM (2006)
59. Richardson, M., Domingos, P.: Markov logic networks. Mach. Learn. **62**(1-2), 107–136 (2006). Kluwer Academic Publishers
60. Riedel, S.: Improving the accuracy and efficiency of map inference for markov logic. In: UAI, pp 468–475. AUAI Press (2008)
61. Rooshenas, A., Lowd, D.: Learning sum-product networks with direct and indirect variable interactions. In: International Conference on Machine Learning, pp 710–718 (2014)
62. Sarkhel, S., Gogate, V.: Lifting walksat-based local search algorithms for map inference. In: Proceedings of Statistical Relational Artificial Intelligence Workshop at the Twenty-Seventh AAAI Conference on Artificial Intelligence, pp. 64–67. AAAI Press, Bellevue (2013)
63. Sarkhel, S., Venugopal, D., Singla, P., Gogate, V.: Lifted MAP inference for markov logic networks. In: Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, vol. 33, pp. 859–867. JMLR: W & CP, Reykjavik (2014a)
64. Sarkhel, S., Venugopal, D., Singla, P., Gogate, V.G.: An integer polynomial programming based framework for lifted map inference. In: Advances in Neural Information Processing Systems, pp 3302–3310 (2014b)
65. Schoenfisch, J., Meilicke, C., von Stülpnagel, J., Ortmann, J., Stuckenschmidt, H.: Root cause analysis in it infrastructures using ontologies and abduction in markov logic networks. Inf. Syst. **74**, 103–116 (2018)
66. Selman, B., Kautz, H., Cohen, B., et al.: Local search strategies for satisfiability testing. Cliques, coloring, and satisfiability: Second DIMACS implementation challenge **26**, 521–532 (1993)
67. Singla, P., Domingos, P.: Entity resolution with markov logic. In: ICDM, pp 572–582. IEEE Computer Society (2006a)
68. Singla, P., Domingos, P.: Memory-efficient inference in relational domains. In: Proceedings of the Twenty-first National Conference on Artificial Intelligence (AAAI-06), vol. 6, pp 488–493. AAAI Press, Boston (2006b)
69. Skarlatidis, A.: Logical markov random fields (lomrf): an open-source implementation of markov logic networks. https://github.com/anskarl/LoMRF (2012)
70. Slaney, J., Walsh, T.: Backbones in optimization and approximation. In: Proceedings of the 17th International Joint Conference on Artificial Intelligence, vol. 1, pp. 254–259. Morgan Kaufmann Publishers Inc., Seattle (2001)
71. Szeliski, R.: Image alignment and stitching: a tutorial. Found. Trends® Comput. Graph. Vis. **2**(1), 1–104 (2006). Now Publishers Inc.
72. Wainwright, M., Jaakkola, T., Willsky, A.: Tree consistency and bounds on the performance of the max-product algorithm and its generalizations. Stat. Comput. **14**(2), 143–166 (2004). Springer
73. Wainwright, M., Jaakkola, T., Willsky, A.: MAP estimation via agreement on (hyper)trees: Message-passing and linear programming approaches. IEEE Transactions on Information Theory, vol. 51, pp. 3697–3717. IEEE computer society (2005)
74. Weiss, Y., Freeman, W.T.: On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. IEEE Trans. Inf. Theory **47**(2), 736–744 (2001). IEEE computer Society
75. Yanover, C., Meltzer, T., Weiss, Y.: Linear programming relaxations and belief propagation–an empirical study. J. Mach. Learn. Res. **7**, 1887–1907 (2006). JMLR. org
76. Zhang, W.: Phase transitions and backbones of the asymmetric traveling salesman problem. J. Artif. Intell. Res. (JAIR) **21**, 471–497 (2004). AAAI Press