

Markov Logic as a Modelling Language for Weighted Constraint Satisfaction Problems

Dominik Jain, Paul Maier, and Gregor Wylezich

Technische Universität München, Department of Informatics
Boltzmanstraße 3, 85748 Garching, Germany
{jain,maierpa}@cs.tum.edu, gregor.wylezich@msg-systems.com

Abstract. Many real-world problems, for example resource allocation, can be formalized as soft constraint optimization problems. A fundamental issue is the compact and precise declaration of such problems. We propose Markov logic networks (MLNs), a representation formalism well-known from statistical relational learning, as a simple yet highly expressive modelling framework, for MLNs enable the representation of general principles that abstract away from concrete entities in order to achieve a separation between the model and the data to which it is applied. MLNs provide the full power of first-order logic and combine it with probabilistic semantics, thus allowing a flexible representation of soft constraints. We introduce an automatic conversion of maximum a posteriori (MAP) inference problems in MLNs to weighted constraint satisfaction problems to leverage a large body of available solving methods, and we make our software suite available to the public. We demonstrate the soundness of our approach on a real-world room allocation problem, providing experimental results.

1 Introduction

In soft constraint programming, it is highly desirable to have modelling languages that enable a concise, declarative and, most importantly, general representation of (families of) optimization problems. By abstracting away from concrete entities to which constraints apply and representing instead general principles about (classes of) entities, a model can be applied to arbitrary problem instances with a varying number of entities/objects – simply by repeatedly applying these general principles appropriately. In the subfield of artificial intelligence that has emerged as statistical relational learning [1], numerous representation formalisms that address precisely the issue of describing such general principles as soft constraints in a probabilistic context have recently been proposed. One of the most expressive such formalisms is that of Markov logic networks (MLNs) [2], which essentially combines a first-order logic representation with the semantics of graphical models to define (a family of) probability distributions in relational domains. The use of first-order formulas – supporting universal quantification – allows general statements about a domain to be made, which then apply to any entities that are presently under consideration. The probabilistic semantics come about by associating with each formula a probabilistic parameter that quantifies the hardness

that the constraint represents on the set of possible worlds that is implied by the set of entities for which the model is instantiated. Each formula can thus be viewed as a generalized soft constraint, and there is a clear connection between MLNs and weighted constraint satisfaction problems (WCSPs). Because MLNs are so conceptually simple and yet highly expressive, we propose them as a general modelling and representation language for soft constraint optimization problems, in particular for WCSPs.

As an example, we consider a resource allocation problem: Given a company’s current room allocation, the problem is to find an optimal or near-optimal assignment of its employees to workplaces and rooms. Many of the company’s real-life constraints which, for example, stem from team building, business unit affiliations, etc., can be modeled abstractly by appropriately weighting allocation preferences against the costs of employee relocations. These constraints form the abstract *model*, whereas specific information (on, for example, the concrete set of rooms and the workplaces within them as well as the current allocation) forms the data or *evidence*. This separation allows us to use one and the same model for any of the company’s facilities, which may differ in the number of rooms, employees, teams or business units. A concrete problem instance is formed by combining the model with specific evidence.

Our primary contribution is to introduce MLNs as a highly flexible and expressive modelling language for WCSPs (which abstracts away from problem-specific aspects), describing how soft constraints can be formulated in MLNs and how maximum a posteriori inference in MLNs can be rephrased in the WCSP framework. Furthermore, our translation procedure enables the use of a large body of inference algorithms that are available for WCSPs to be applied to statistical relational models, for which MLNs can already be regarded as a unifying framework. In our experiments, we show the practical soundness of our approach on a real-world problem, also providing preliminary comparisons on the performance of standard inference methods for MLNs and WCSP methods. The software tools we developed for the representation of MLNs and their translation into WCSPs is made available to the public.

2 Related Work

In the field of constraint programming, a number of high-level languages are available, such as OPL [3], Oz [4], Eclipse [5], Zinc [6] and a subset of Zinc, Minizinc [7]. They all follow the general principle of separate problem modeling and high level algorithm specification. Zinc allows preferences as to be modelled as soft constraints. To our knowledge, however, existing implementations do not support this language feature.

Bistarelli et al. developed the soft constraint logic programming (SCLP) framework [8] and several extensions, such as soft concurrent constraint programming [9]. While these support the modelling of preferences, they lack high-level constructs such as universal quantification.

A rich language focussing on local search is Comet [10], which allows to model problems in terms of constraints, constraint combination operators and objective functions. It also provides high-level language features to implement stochastic local search and constraint programming algorithms. Comet is similarly expressive as MLNs, as it allows universal quantification and assigns weights to constraints. In contrast to MLNs, it is not based on first-order logic, and, to our knowledge, does not allow constraint optimization problems to be exported in order to solve them with external tools such as Toulbar2.

3 Markov Logic Networks (MLNs)

An MLN can be seen as a first-order knowledge base with weights attached to each of the formulas, which can be understood as a template for the construction of probabilistic graphical models, namely Markov random fields (MRFs).

Definition 1. *Formally, a Markov logic network L is a set of pairs (F_i, w_i) , where F_i is a formula in first-order logic and w_i is a real number, the weight of formula F_i . Together with a finite set of entities E (constant terms to which the predicates appearing in the formulas can be applied), an MLN can be instantiated to define a ground Markov random field $M_{L,E}$ as follows:*

1. $M_{L,E}$ contains one binary node for each possible grounding of each predicate appearing in the formulas of the Markov logic network L .
2. $M_{L,E}$ contains one feature for each possible grounding of each formula F_i in L .¹ The value of this feature is 1 if the ground formula is true, and 0 otherwise. The weight of the feature is w_i .

The ground MRF's set of variables $X = \{X_1, \dots, X_n\}$ is thus the set of ground atoms that is implicitly defined by the predicates in the MLN and the set of entities E . The Markov logic network specifies a probability distribution over the set of possible worlds \mathcal{X} , i.e. the set of possible assignments of truth values to each of the ground atoms in X , as follows,

$$P_{M_{L,E}}(X = x) = \frac{1}{Z} \exp \left(\sum_i w_i \cdot n_i(x) \right) \quad (1)$$

where $Z := \sum_{x' \in \mathcal{X}} \exp(\sum_i w_i \cdot n_i(x'))$ is a normalizing constant, the sum is over indices of MLN formulas, and $n_i(x)$ is the number of true groundings of the i -th formula in possible world x . This follows the log-linear representation

¹ Note that in standard MLN semantics, each of the conjuncts of a grounded universally quantified formula represents a separate, weighted grounding, i.e. for a universally quantified formula F_i with weight w_i , the ground Markov random field will contain one feature with weight w_i for every assignment of the universally quantified variables in F_i to entities in E . Existential quantification is, of course, also supported, but its use can sometimes be problematic because it results in very large ground formulas (which cannot soundly be split as in the case of universal quantification).

of a Markov random field. The weight of a formula should thus be viewed as a logarithmized factor that either increases (if the weight is positive) or decreases the value $\exp(\sum_i w_i n_i(x)) = \prod_i \exp(w_i)^{n_i(x)}$ that is associated with a possible world x and which is to be viewed relative to the corresponding values of other possible worlds. If two possible worlds differ only in the truth value of a single ground instance of formula F_i , then the weight w_i can be viewed as the log odds of the formula being true, i.e. if $w_i = \log(p/(1-p))$, then the world that satisfies the formula is $p/(1-p)$ times more probable than the one that does not satisfy it. Usually, however, formulas will share variables (ground atoms) and one formula may not be satisfied without dissatisfying another. In many cases, it is sensible to consider, for a particular set of variables, a group of mutually exclusive and exhaustive formulas, such that, for each possible world, exactly one of the formulas from the group holds true. We can then define the weights for formulas within the group in such a way that for each pair of weights w_i, w_j , the ratio of the corresponding factors corresponds to the probability of the respective cases, i.e. that $\exp(w_i)/\exp(w_j)$ corresponds to the ratio of the frequencies with which the cases represented by formulas i and j should occur.

In practical applications, it is advisable to use a typed predicate logic and thus a set of typed entities E and to include in the model corresponding predicate type declarations, avoiding the creation of nonsensical ground atoms. Furthermore, we use the standard extension to MLNs that allows relations to be declared as functional, as this is a common trait that should be considered. For instance, since every employee works in exactly one room, the room is functionally determined by the employee, which can be declared in the predicate declaration as follows: *employeeIn(employee, room!)*. For any functional predicate, we obtain, for every binding of the arguments that are not functionally determined, a set of ground atoms that are mutually exclusive and exhaustive (comprising precisely the completions of the respective binding).

Table 1 shows the full MLN model for our example room allocation problem. The predicates refer to various relations such as employees working within certain rooms (*employeeIn*), a workplace being in a certain room (*workplaceIn*), an employee being required to work alone (*alone*), employees working in teams (*teammate*), rooms being nearby (*nearby*), two employees being required to work closely together (*closeTo*), rooms being assigned to business units (*assignedTo*), employees belonging to business units (*belongsTo*) and the workplaces employees are assigned to before and after relocations (*workplaceBefore*, *workplaceAfter*). The 9 MLN formulas represent the following aspects: (1) An employee required to work alone must not share his/her office with anyone, (2) an employee always works in the room his/her workplace is located in, (3) each workplace is occupied by at most one employee, (4, 5) people required to work close by should not share the same room but should preferably work in rooms that are nearby (6) an employee should preferably use a room that is assigned to her business unit, (7) employees should preferably not be relocated (as it causes costs), (8) a team of employees should preferably work within the same room or (9) should (if that is not possible) work in rooms nearby. The weights that are specified as

Predicate Declarations

| | |
|--|---|
| $workplaceIn(workplace, room!)$ | $assignedTo(room, business\ unit!)$ |
| $belongsTo(employee, business\ unit!)$ | $workplaceBefore(employee, workplace!)$ |
| $workplaceAfter(employee, workplace!)$ | $employeeIn(employee, room!)$ |
| $teammate(employee, employee)$ | $alone(employee)$ |
| $nearby(room, room)$ | $closeTo(employee, employee)$ |

Weighted Formulas

| # | weight | formula |
|---|----------------|---|
| 1 | hard | $employeeIn(e_1, r) \wedge employeeIn(e_2, r) \wedge alone(e_1) \rightarrow (e_1 = e_2)$ |
| 2 | hard | $workplaceAfter(e, p) \wedge workplaceIn(p, r) \rightarrow employeeIn(e, r)$ |
| 3 | hard | $workplaceAfter(e_1, p) \wedge workplaceAfter(e_2, p) \rightarrow (e_1 = e_2)$ |
| 4 | hard | $closeTo(e_1, e_2) \rightarrow \neg employeeIn(e_1, r) \vee \neg employeeIn(e_2, r)$ |
| 5 | $\log(6)$ | $closeTo(e_1, e_2) \wedge employeeIn(e_1, r_1) \wedge employeeIn(e_2, r_2) \wedge \neg(r_1 = r_2) \wedge nearby(r_1, r_2)$ |
| 6 | $\log(13.5)$ | $employeeIn(e, r) \wedge assignedTo(r, b) \rightarrow belongsTo(e, b)$ |
| 7 | $-\log(3.125)$ | $workplaceBefore(e, p_1) \wedge workplaceAfter(e, p_2) \wedge \neg(p_1 = p_2)$ |
| 8 | $\log(10)$ | $employeeIn(e_1, r) \wedge employeeIn(e_2, r) \wedge teammate(e_1, e_2)$ |
| 9 | $\log(6)$ | $employeeIn(e_1, r_1) \wedge employeeIn(e_2, r_2) \wedge teammate(e_1, e_2) \wedge \neg(r_1 = r_2) \wedge nearby(r_1, r_2)$ |

Table 1. Model for room allocation in Markov logic. (Free variables in the formulas are implicitly universally quantified.)

hard in Table 1 are, for practical purposes, substituted by a very large positive real number, thus assigning to any possible worlds that violate any of their groundings a negligible probability value; the soft weights were chosen according to the company’s priorities.

4 Transforming Instances of MLNs into WCSPs

In the following, we show that there is, for any set of entities E , a correspondence between the most likely possible world(s) $\operatorname{argmax}_x P_{ML,E}(X = x)$ of a Markov random field instantiated from an MLN and the optimal solution(s) of appropriately derived WCSPs.

Definition 2. A weighted constraint satisfaction problem (WCSP) is a tuple $\mathcal{R} = \langle Y, D, C \rangle$ [11]:

1. $Y = \{Y_1, \dots, Y_n\}$ is a set of n variables.
2. $D = \{D_1, \dots, D_n\}$ is the collection of the domains of the variables in Y , such that $D_i = \operatorname{dom}(Y_i)$ is the domain of Y_i . For a given variable Y_i we may also denote its domain by D_{Y_i} . We write D_S to denote the Cartesian product $\prod_{Y_i \in S} D_i$ for some subset of the variables $S \subseteq Y$. $\mathcal{Y} := D_Y$ denotes the Cartesian product of all domains and hence represents the set of possible variable assignments.

3. $C = \{c_1, \dots, c_r\}$ is a finite set of r soft constraints. A soft constraint c_i is a function on a sequence of variables V from the set Y (V is called the scope of the constraint) such that c_i maps assignments (of values to the variables in V) to cost values: $c_i : D_V \rightarrow \{0, \dots, \top\}$. If an assignment is mapped to \top , it is considered inconsistent.
4. A solution to \mathcal{R} is a consistent assignment to all variables. An optimal solution $y \in \mathcal{Y}$ minimizes the accumulated cost $\sum_{i=1}^r c_i(y)$ over all constraints (we assume that y is implicitly projected to the actual scope of c_i).

We require a mapping from features of ground Markov random fields to WCSP constraints. In a WCSP, soft constraints are described in terms of costs, whereas the parameters of MLNs can either indicate an increased probability (desirable configuration of variables) or – in the case of negative weights – decreased probability (undesirable configuration). Observe, however, that an MLN retains its semantics if any of its formulas are negated along with their weights, i.e. that the probability distributions over possible worlds implied by an MLN L and an MLN L' derived from L by changing a pair $(F_k, w_k) \in L$ to $(\neg F_k, -w_k) \in L'$ are the same:

$$\begin{aligned}
P_{M_{L',C}}(X = x) &= \frac{\exp\left(\sum_{i,i \neq k} w_i \cdot n_i(x) - w_k \cdot (N_k - n_k(x))\right)}{\sum_{x' \in \mathcal{X}} \exp\left(\sum_{i,i \neq k} w_i \cdot n_i(x') - w_k \cdot (N_k - n_k(x'))\right)} \\
&= \frac{\exp\left(\sum_i w_i \cdot n_i(x) - w_k \cdot N_k\right)}{\sum_{x' \in \mathcal{X}} \exp\left(\sum_i w_i \cdot n_i(x') - w_k \cdot N_k\right)} \\
&= \frac{\exp\left(\sum_i w_i \cdot n_i(x)\right)}{\sum_{x' \in \mathcal{X}} \exp\left(\sum_i w_i \cdot n_i(x')\right)} \cdot \frac{\exp(-w_k \cdot N_k)}{\exp(-w_k \cdot N_k)} = P_{M_{L,C}}(X = x)
\end{aligned} \tag{2}$$

where N_k is the number of groundings of the k -th formula. We can thus transform any MLN into a semantically equivalent MLN that contains only positive weights. Since positive weights indicate probable configurations, it is then straightforward to consider, within a WCSP, costs proportionate to the weight of a formula whenever the formula is unsatisfied. In the standard WCSP framework as defined above, we have the additional requirement that costs be natural numbers, the MLN weights however are now in \mathbb{R}_0^+ . We thus need to scale the weights appropriately, noting that while scaling weights of an MLN with some positive constant λ changes the probability distribution over possible worlds, it does not affect the set of most probable states:

$$\operatorname{argmax}_{x \in \mathcal{X}} \frac{1}{Z} \exp\left(\sum_i w_i \cdot n_i(x)\right) = \operatorname{argmax}_{x \in \mathcal{X}} \sum_i w_i \cdot \lambda \cdot n_i(x) \tag{3}$$

To ensure that even minor relative differences between weights are maintained, we propose $\lambda = T/\delta_{\min}$ as the scaling factor, where the division by the smallest pairwise difference between any two weights δ_{\min} ensures that this smallest difference becomes at least 1, and T is a large positive constant used to reduce precision loss when subsequently rounding to integer values.

Consider now a concrete MLN $L = \{(F_i, w_i)\}$, which is instantiated for a set of domain elements E , yielding the ground Markov random field $M_{L,E}$ with set

of ground formulas $\hat{F} = \{\hat{F}_1, \dots, \hat{F}_N\}$ and corresponding weight vector \hat{w} (with weights already transformed as described above). We seek to derive from $M_{L,E}$ a WCSP $\mathcal{R} = \langle Y, D, C \rangle$, whose set of optimal solutions is to be equivalent to $M_{L,E}$'s set of most probable possible worlds.

Let the set of variables in $M_{L,E}$ be X , and let $\mathcal{X} = \prod_{X_i \in X} \text{dom}(X_i) = \mathbb{B}^n$ be the set of possible worlds. We map subsets $A \subseteq X$ of mutually exclusive and exhaustive variables to a single variable in Y with an integer domain $\{0, \dots, |A| - 1\}$;² all other (boolean) variables simply have direct correspondences in Y . There is thus a direct (non-injective) mapping $S : X \rightarrow Y$ from variables in $M_{L,E}$ to their correspondences in \mathcal{R} as well as a bijective mapping $s : \mathcal{X} \rightarrow \mathcal{Y}$ from states of $M_{L,E}$ to states of \mathcal{R} .

For each feature \hat{F}_i in $M_{L,E}$, we define a constraint c_i in C . If the feature is not satisfied by some assignment $X = x$, the constraint defines a cost of \hat{w}_i , and 0 otherwise. (Because the positive weights in the MLN increase the probability of any possible world that satisfies the corresponding formula, an unsatisfied formula should therefore incur proportionate costs.) In other words, c_i is defined as follows,

$$\begin{aligned} c_i(s(x)) &= \begin{cases} \hat{w}_i & \text{if } \hat{f}_i(x) = 0 \\ 0 & \text{if } \hat{f}_i(x) = 1 \end{cases} \\ &= (1 - \hat{f}_i(x)) \cdot \hat{w}_i \end{aligned} \quad (4)$$

where \hat{f}_i is the feature function associated with \hat{F}_i , and we assume that $s(x)$ is implicitly projected to the actual scope of c_i .

We can solve MAP inference in $M_{L,E}$, i.e. the maximization problem

$$\operatorname{argmax}_{x \in \mathcal{X}} \frac{1}{Z} \exp \left(\sum_{i=1}^N \hat{f}_i(x) \cdot \hat{w}_i \right) = \operatorname{argmax}_{x \in \mathcal{X}} \sum_{i=1}^N \hat{f}_i(x) \cdot \hat{w}_i \quad (5)$$

by calculating the solution of the WCSP \mathcal{R} , since

$$\begin{aligned} \operatorname{argmin}_{y \in \mathcal{Y}} \sum_{i=1}^N c_i(y) &= \operatorname{argmax}_{y \in \mathcal{Y}} \sum_{i=1}^N -c_i(y) = \operatorname{argmax}_{y \in \mathcal{Y}} \sum_{i=1}^N (1 - \hat{f}_i(s^{-1}(y))) \cdot -\hat{w}_i = \\ \operatorname{argmax}_{x \in \mathcal{X}} \sum_{i=1}^N (1 - \hat{f}_i(x)) \cdot -\hat{w}_i &= \operatorname{argmax}_{x \in \mathcal{X}} \sum_{i=1}^N (-\hat{w}_i + \hat{f}_i(x) \cdot \hat{w}_i) = \operatorname{argmax}_{x \in \mathcal{X}} \sum_{i=1}^N \hat{f}_i(x) \cdot \hat{w}_i \end{aligned} \quad (6)$$

MLNs can thus be soundly used to abstractly formulate WCSPs, and WCSP solvers can be used to find the most likely assignment for any instance of an MLN.

² For example, the predicate *workplaceIn* results in a group of mutually exclusive and exhaustive variables for every workplace W . If the set of rooms is $\{R_1, \dots, R_n\}$, then out of the set of ground atoms $A = \{\text{workplaceIn}(W, R_1), \dots, \text{workplaceIn}(W, R_n)\}$, exactly one is required to be true, and we can replace this set of boolean variables by a single variable *workplaceIn*(W) with the domain $\{0, \dots, n - 1\}$ representing the set of rooms.

It is important to note that not only optimal solutions are retained. Even if the WCSP solver can return only an approximate solution, the ordering of solutions is sound and corresponds to the ordering imposed by the original MLN. Let w be the weight vector of the original MLN, which was only (equivalently) transformed into an MLN with only positive weights but where the scaling factor λ has not yet been applied, as this is not an equivalence transformation. We thus have $c_i(s(x)) = (1 - \hat{f}_i(x)) \cdot \hat{w}_i \cdot \lambda$, where \hat{w} is again the weight vector of the instantiated MRF derived from w . For two states x_1 and x_2 of the ground MRF, we obtain:

$$\begin{aligned}
\sum_i^N c_i(s(x_1)) &< \sum_i^N c_i(s(x_2)) \\
-\sum_i^N c_i(s(x_1)) &> -\sum_i^N c_i(s(x_2)) \\
\sum_i^N (\hat{f}_i(x_1) - 1) \cdot \hat{w}_i \cdot \lambda &> \sum_i^N (\hat{f}_i(x_2) - 1) \cdot \hat{w}_i \cdot \lambda \\
\sum_i^N \hat{f}_i(x_1) \cdot \hat{w}_i &> \sum_i^N \hat{f}_i(x_2) \cdot \hat{w}_i \\
\exp\left(\sum_i^N \hat{f}_i(x_1) \cdot \hat{w}_i\right) &> \exp\left(\sum_i^N \hat{f}_i(x_2) \cdot \hat{w}_i\right)
\end{aligned} \tag{7}$$

The ordering of solutions in the WCSP is thus consistent with the probability distribution represented by the MRF that was generated from the original MLN.

Example Translation. We now illustrate the translation process we described above using our room allocation example. Consider formula #7 in Table 1:

$$-\log(3.125) \text{ workplaceBefore}(e, p_1) \wedge \text{workplaceAfter}(e, p_2) \wedge \neg(p_1 = p_2)$$

The first step is to negate the formula in order to obtain a positive weight, i.e.

$$\log(3.125) \neg\text{workplaceBefore}(e, p_1) \vee \neg\text{workplaceAfter}(e, p_2) \vee (p_1 = p_2).$$

Assuming that the set E of entities is partitioned into the set $\{P_1, P_2, P_3\}$ of workplaces and the set $\{E_1, E_2\}$ of employees, the ground Markov random field will contain the ground formulas shown in Table 2 as features. Ground formulas where $(p_1 = p_2)$ evaluates to *True* are omitted, as tautologies do not actually represent constraints.

To create a WCSP $\mathcal{R} = \langle Y, D, C \rangle$, we first generate the set of variables Y and their domains D . Each ground atom represents a boolean variable in the MRF. The predicates *workplaceBefore(employee, workplace!)* and *workplaceAfter(employee, workplace!)*, however, were declared as functional, mapping employees to workplaces. This means that, for each functional predicate, the set of ground

$\log(3.125) \neg \text{workplaceBefore}(E_1, P_1) \vee \neg \text{workplaceAfter}(E_1, P_2) \vee \text{False}$
 $\log(3.125) \neg \text{workplaceBefore}(E_1, P_2) \vee \neg \text{workplaceAfter}(E_1, P_1) \vee \text{False}$
 $\log(3.125) \neg \text{workplaceBefore}(E_1, P_1) \vee \neg \text{workplaceAfter}(E_1, P_3) \vee \text{False}$
 $\log(3.125) \neg \text{workplaceBefore}(E_1, P_3) \vee \neg \text{workplaceAfter}(E_1, P_1) \vee \text{False}$
 $\log(3.125) \neg \text{workplaceBefore}(E_1, P_2) \vee \neg \text{workplaceAfter}(E_1, P_3) \vee \text{False}$
 $\log(3.125) \neg \text{workplaceBefore}(E_1, P_3) \vee \neg \text{workplaceAfter}(E_1, P_2) \vee \text{False}$
 $\log(3.125) \neg \text{workplaceBefore}(E_2, P_1) \vee \neg \text{workplaceAfter}(E_2, P_2) \vee \text{False}$
 $\log(3.125) \neg \text{workplaceBefore}(E_2, P_2) \vee \neg \text{workplaceAfter}(E_2, P_1) \vee \text{False}$
 $\log(3.125) \neg \text{workplaceBefore}(E_2, P_1) \vee \neg \text{workplaceAfter}(E_2, P_3) \vee \text{False}$
 $\log(3.125) \neg \text{workplaceBefore}(E_2, P_3) \vee \neg \text{workplaceAfter}(E_2, P_1) \vee \text{False}$
 $\log(3.125) \neg \text{workplaceBefore}(E_2, P_2) \vee \neg \text{workplaceAfter}(E_2, P_3) \vee \text{False}$
 $\log(3.125) \neg \text{workplaceBefore}(E_2, P_3) \vee \neg \text{workplaceAfter}(E_2, P_2) \vee \text{False}$

Table 2. Ground formulas

atoms for a particular employee are mutually exclusive and exhaustive, e.g. $\forall x \in \mathcal{X} \forall e \exists_1 r. x \models \text{workplaceBefore}(e, p)$. We thus reduce the problem size by simply mapping each such set of ground atoms to a single WCSP variable whose domain is the set of workplaces the employee can occupy (rather than having each ground atom as a separate binary WCSP variable). For example, we define a WCSP variable $Y_1 \equiv \text{workplaceBefore}(E_1)$ representing the set $\{\text{workplaceBefore}(E_1, P_1), \text{workplaceBefore}(E_1, P_2), \text{workplaceBefore}(E_1, P_3)\}$ of MRF variables, with its domain $D_1 = \{0, 1, 2\}$ representing the three workplaces. Similarly, we define $Y_2 \equiv \text{workplaceBefore}(E_2)$, $Y_3 \equiv \text{workplaceAfter}(E_1)$ and $Y_4 \equiv \text{workplaceAfter}(E_2)$.

Considering just the one MLN formula, $Y = \{Y_1, Y_2, Y_3, Y_4\}$ is therefore our set of WCSP variables, and the set of domains is $D = \{D_1, D_2, D_3, D_4\}$ with $D_i = \{0, 1, 2\}, i = 1..4$. We store mappings that associate the integer values with the corresponding workplaces $\{P_1, P_2, P_3\}$. (For example, assuming the mapping $\{0 \mapsto P_1, 1 \mapsto P_2, 2 \mapsto P_3\}$, the semantics of the assignment $Y_1 = 0, Y_2 = 2, Y_3 = 1, Y_4 = 2$ would be that employee E_1 moves from workplace P_1 to P_2 and E_2 stays at P_3 .)

To create the set of constraints C , each of the 12 ground formulas above is translated to a WCSP constraint. Let \hat{F}_1 be the first ground formula in Table 2 and let \hat{f}_1 be its feature function. Remember that we obtain the corresponding WCSP constraint, specifically its cost function, as $c_1(s(x)) = (1 - \hat{f}_1(x)) \cdot \hat{w}_1 \cdot \lambda$. With the above variable mapping and $\hat{w}_1 = \log(3.125)$, the first formula translates to the constraint

| Y_1 | Y_3 | c |
|--------|-------|------------------------------------|
| 0 | 1 | $\lambda \log(3.125) \approx 3798$ |
| others | | 0 |

The constraint is represented as a table, where each table row represents assignment tuples and their associated cost. Every assignment $Y = y$ that renders the original ground formula true (feature $\hat{f}_1(s^{-1}(y)) = 1$) results in costs of 0, while every assignment that renders it false incurs costs of $\lambda \log(3.125)$. $\lambda = T/\delta_{\min} \approx$

1000/0.3 is the previously described scaling factor, where $\delta_{\min} \approx 0.3$ is the minimum difference between distinct weights of our room allocation problem and $T = 1000$ prevents precision loss when rounding to integers. The formula is false only if $\text{workplaceBefore}(E1, P1) = \text{True}$ and $\text{workplaceBefore}(E1, P2) = \text{True}$, which corresponds to the assignment $Y_1 = 0, Y_3 = 1$ indicated above. All other assignments render the formula true, i.e. they incur no costs.

The other ground formulas yield similar constraints. Note that the first six formulas all result in WCSP constraints over the variables Y_1 and Y_3 , which could therefore easily be combined into a single constraint, further reducing the problem size. Moreover, variables that are provided as evidence (i.e. ground atoms whose truth values are provided in the database with which the MLN is combined to yield the ground MRF), can be removed from the ground formulas altogether and therefore need not even appear as WCSP variables.

5 Experimental Results

We created four instances of our room allocation problem, applying the general constraints of our MLN to instances of variable size with varying evidence that indicates the set of entities as well as individual requirements in each case. Table 3 lists the properties of each problem instance as well as the results we obtained on a Windows computer with a 3 GHz CPU and 4 GB of RAM. To solve the WCSP instances we used a branch and bound search with soft local consistency enforcing implemented in Toulbar2 (using its default settings), a suite of soft constraint optimization algorithms³. We compared it against the standard MLN MAP inference algorithm, MaxWalkSAT [12]. We made two runs on the instances, imposing a time limit of $T_1 = 180$ seconds in the first run and $T_2 = 1800$ seconds in the second. The limits were necessary for the comparison with MaxWalkSAT. Also, they represent requirements of the company that provided the example problem. The quality of the solutions found is given in percent, where the initial assignment (before relocation) corresponds to 0% while

³ https://mulcyber.toulouse.inra.fr/frs/?group_id=29 (April 2009)

| | Problem instance | | | |
|--------------------|------------------|---------------|-----------------|-----------------|
| | A | B | C | D |
| # entities | 7, 22, 20, 2 | 11, 34, 29, 2 | 19, 64, 53, 3 | 23, 73, 57, 4 |
| # evidence atoms | 1657 | 3591 | 11863 | 14715 |
| # total atoms | 2237 | 4896 | 16262 | 20187 |
| WCSP size | 9685, 40, 22 | 30863, 58, 34 | 188060, 106, 64 | 249754, 114, 73 |
| sol. quality T_1 | 100 | 100 | 99 | 93 |
| sol. quality T_2 | 100 | 100 | 99.1 | 97 |

Table 3. Four instances of the room allocation problem used in our experiments. The numbers of entities are: rooms, workplaces, employees and business units. The WCSP size is: no. of constraints, no. of variables and the maximum domain size.

an optimal solution corresponds to 100%. Toulbar2 solves instances A and B in T_1 , and finds near-optimal solutions for the larger instances C and D in T_1 and T_2 . Given several hours of time, Toulbar2 also solved these instances optimally. Experiments with the MaxWalkSAT algorithm for MLNs showed mostly comparable performance, yet, of course, its stochastic nature did not guarantee optimality in all cases (even for the smaller instances) and its convergence speed is highly dependent on the choice of the noise probability parameter, which cannot universally be chosen. Therefore, WCSP algorithms are a practical alternative. The WCSP conversion constitutes but a marginal overhead and the results show that it is sensible to represent soft constraint optimization problems as MLNs.

6 Software

We developed a full implementation of Markov logic networks, an application for translating an MLN into a WCSP for any given evidence database (using the standard WCSP format employed by Toulbar2 and other solvers) and implemented the MaxWalkSAT algorithm as well as other standard MLN inference

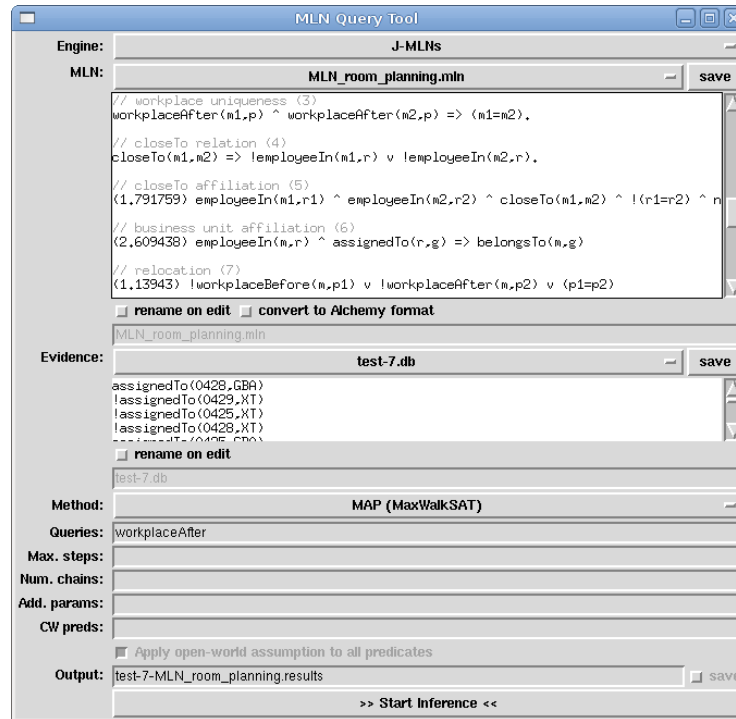


Fig. 1. Graphical inference tool.

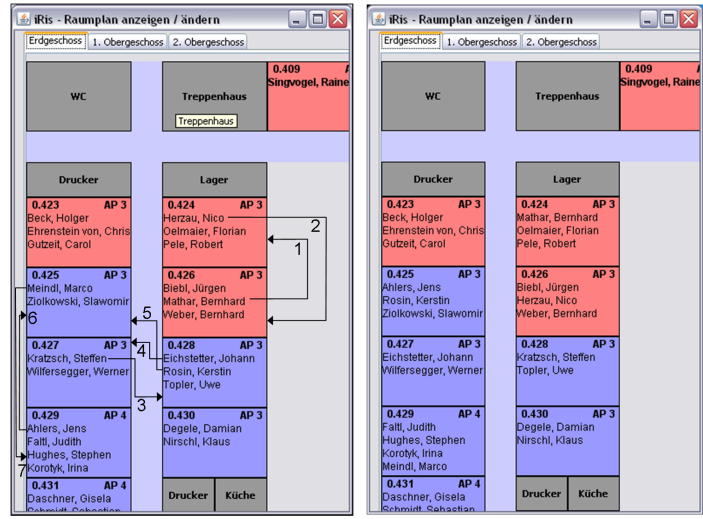


Fig. 2. Room allocation tool developed within the company (GUI text is in German), indicating an allocation before (left) and after optimization (right). The arrows indicate the relocations that are necessary in this particular problem instance to achieve an optimal allocation.

algorithms. Our suite of tools⁴ (implemented in Java and Python) includes a graphical tool (see Figure 1) which allows convenient editing of MLN problem representations as well as of evidence databases and supports the execution of inference algorithms, including, in particular, Toulbar2 inference followed by our automatic translation procedure.

Within a joint project with the company, we developed a specialized graphical room allocation tool, which graphically displays allocations and features both a general constraint editor for the convenient specification of factors for groups of mutually exclusive and exhaustive formulas, and a specialized constraint editor specific to the room allocation problem. Figure 2 shows an exemplary problem instance, indicating how a (sub-optimal) initial allocation was optimized.

7 Conclusion

We presented Markov logic networks as a flexible, expressive and general modeling language for soft constraint optimization problems, in particular for WCSPs. By abstractly representing generalized soft constraints, an MLN models an entire class of WCSPs. We described a transformation procedure, which, for any concrete set of data instances, yields one particular WCSP (from the class of WCSPs being modelled). From our experimental results, we conclude that this procedure

⁴ <https://www9.informatik.tu-muenchen.de/people/maier/mln2wcsp-software>

is sound and can successfully be applied to real-world problems. Furthermore, from the perspective of statistical relational models, our transformation procedure enables a large body of existing WCSP algorithms to be applied for the task of MAP inference.

References

1. Getoor, L., Taskar, B.: Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning). The MIT Press (2007)
2. Richardson, M., Domingos, P.: Markov Logic Networks. *Mach. Learn.* **62** (2006) 107–136
3. Van Hentenryck, P.: The OPL Optimization Programming Language. MIT Press (1999)
4. Van Roy, P., Brand, P., Duchier, D., Haridi, S., Henz, M., Schulte, C.: Logic Programming in the Context of Multiparadigm Programming: the Oz Experience. *Theory and Practice of Logic Programming* **3** (2003) 717–763
5. Apt, K.R., Wallace, M.: Constraint Logic Programming Using Eclipse. Cambridge University Press, New York, NY, USA (2007)
6. Marriott, K., Nethercote, N., Rafeh, R., Stuckey, P.J., Garcia De La Banda, M., Wallace, M.: The Design of the Zinc Modelling Language. *Constraints* **13** (2008) 229–267
7. Nethercote, N., Stuckey, P.J., Becket, R., Brand, S., Duck, G.J., Tack, G.: Minizinc: Towards a Standard CP Modelling Language. In: *Proc. CP-2007*. Volume 4741 of LNCS., Providence, RI, USA, Springer (2007) 529–543
8. Bistarelli, S., Rossi, F.: Semiring-Based Constraint Logic Programming: Syntax and Semantics. *ACM Transactions on Programming Languages and Systems* **23** (2001) 1–29
9. Bistarelli, S., Montanari, U., Rossi, F.: Soft Concurrent Constraint Programming. In: *Proc. ESOP-2002*. (2002) 53–67
10. Michel, L., Hentenryck, P.V.: Comet in Context. In: *Proc. PCK50-2003*, New York, NY, USA, ACM (2003) 95–107
11. Meseguer, P., Rossi, F., Schiex, T.: 9 Soft Constraints. *Foundations of Artificial Intelligence*. In: *Handbook of Constraint Programming*. Elsevier (2006) 281–328
12. Kautz, H., Selman, B., Jiang, Y.: A General Stochastic Approach to Solving Problems with Hard and Soft Constraints. In: *The Satisfiability Problem: Theory and Applications*, American Mathematical Society (1996) 573–586