Markov Random Field MAP as Set Partitioning

James Cussens JAMES.CUSSENS@YORK.AC.UK
Department of Computer Science & York Cross-Disciplinary Centre for Systems Analysis
University of York, UK

Abstract

The Markov Random Field (MRF) MAP inference problem is considered from the viewpoint of integer programming (IP). The problem is shown to be a (pure) set partitioning problem (SPP). This allows us to bring existing work on SPP to bear on the MAP problem. Facets (maximally strong linear inequalities) of the closely related set packing (SP) problem are shown to be useful for MRF MAP. These facets include odd hole and odd anti-hole inequalities which are shown to be findable using a zero-half cut generator. Experimental results using CPLEX show that for MRF MAP problems, generating more zero-half cuts than normal typically brings performance improvements. Pre-processing methods to reduce the size of MRF MAP problems are also considered, and some preliminary results on their usefulness presented.

Keywords: maximum a posteriori (MAP); integer programming; Markov random fields; set partitioning; zero-half cuts

1. The MRF MAP problem

A Markov random field (MRF) has a structure defined by a *hypergraph* $H = (X, E)$ where X is a set of random variables $X = \{X_1, \ldots, X_p\}$ and E is a set of subsets (i.e. *hyperedges*) of X. The hyperedges of an MRF are often called 'cliques', but here we do not make the assumption that there is some graph for which the hyperedges are (maximal) cliques, that is, there is no assumption that the hypergraph is *conformal* (Duchet, 1995). An MRF defines a joint probability distribution for X as a normalised product of *potential functions*. Each potential function has an associated hyperedge and maps each instantiation of the variables in that hyperedge to some non-negative real number. We will work with log potential functions $\Psi = \{\psi_j : E_j \in E\}$. Let $x_{\downarrow j}$ denote some joint instantiation of the variables in the hyperedge $E_j \in E$, then ψ_j maps each instantiation $x_{\downarrow j}$ to some value in $\mathbb{R} \cup \{-\infty\}$. For any full joint instantiation x, we have $\log p(x) = \log p(X_1 = x_1, \ldots, X_p = x_p) = -\log Z + \sum_{E_j \in E} \psi_j(x_{\downarrow j})$, where $x_{\downarrow j}$ is the projection of x to hyperedge E_j and Z is a normalising constant. We will assume throughout that there is at least one x with $p(x) > 0$, and thus $\log p(x) \neq -\infty$. The MAP problem for an MRF is to find a full joint instantiation x^* of maximal (log) probability: $x^* = \arg\max_x \sum_{E_j \in E} \psi_j(x_{\downarrow j})$

If there is *evidence*, i.e. some variables are fixed to particular values, then this can be incorporated by setting $\psi_j(x_{\downarrow j}) = -\infty$ whenever $x_{\downarrow j}$ is inconsistent with the evidence. More generally, if we somehow know that $x_{\downarrow j}$ cannot be part of some optimal full joint instantiation then this can be indicated by setting $\psi_j(x_{\downarrow j}) = -\infty$.

2. MRF MAP Integer Programming Encoding

We can encode an MRF MAP problem instance into an integer program (IP) using a variant of what Hurley et al. (2016) call the *tuple encoding*, which as they note "is the only 01LP encoding usually

considered in MRFs." For each variable X_i and each value x_i of X_i a binary IP variable $I(X_i = x_i)$, with zero objective coefficient, is created. For each X_i a constraint is added stating that X_i takes exactly one of its values:

$$\sum_{x_i \in \mathcal{X}_i} I(X_i = x_i) = 1 \qquad \text{(where } \mathcal{X}_i \text{ is the set of possible values of } X_i\text{)} \tag{1}$$

For each hyperedge E_j and each *feasible* joint instantiation $x_{\downarrow j}$ (i.e. $\psi_j(x_{\downarrow j}) \neq -\infty$) of the variables in E_j, a binary variable $I(x_{\downarrow j})$is created, with objective value $\psi_j(x_{\downarrow j})$. For each value x_i of each variable X_i in each hyperedge E_j a constraint is then added to ensure the values of the $I(X_i = x_i)$ and $I(x_{\downarrow j})$ IP variables are consistent:

$$I(X_i = x_i) = \sum_{x_{\downarrow j}(i) = x_i} I(x_{\downarrow j}) \tag{2}$$

where $x_{\downarrow j}(i)$ denotes the component of $x_{\downarrow j}$ that is the value of variable X_i. Note that it is possible to eliminate all $I(X_i = x_i)$ variables using (2) resulting in an IP with only $I(x_{\downarrow j})$ variables.

Constraints (1) and (2) suffice to create an IP for solving the corresponding MRF MAP problem, but it is typically advantageous to add additional equations so that the resulting IP has a tighter linear relaxation. For each pair of hyperedges E_j, $E_{j'}$ whose intersection contains at least 2 random variables and for each instantiation $x_{j \cap j'}$ of the random variables in $E_j \cap E_{j'}$ we add a constraint that E_j and $E_{j'}$ must be consistent for $x_{j \cap j'}$:

$$\sum_{x_{\downarrow j}(j \cap j') = x_{j \cap j'}} I(x_{\downarrow j}) = \sum_{x_{\downarrow j'}(j \cap j') = x_{j \cap j'}} I(x_{\downarrow j'}) \tag{3}$$

Constraints of this sort are given, for example, by Koller and Friedman (2009, §13.5.1) in their presentation, but were not used by Hurley et al. (2016).

3. MRF MAP as set partitioning

Using (1) it is easy to see that each instance of (2) can be replaced with:

$$\sum_{x_{i'} \in \mathcal{X}_i, x_i \neq x_{i'}} I(X_i = x_{i'}) + \sum_{x_{\downarrow j}(i) = x_i} I(x_{\downarrow j}) = 1 \tag{4}$$

So an MRF MAP problem can be formulated as an IP where (1) each variable is binary and (2) each constraint states that a sum of variables equals 1. It is not difficult to see that eliminating $I(X_i = x_i)$ variables and/or adding constraints such as (3) still allows the problem to meet these 2 requirements. Any problem meeting these 2 requirements is called a *set partitioning* problem. Formally, a set partitioning (SPP) problem is of the following form:

$$\text{(SPP)} \qquad \min \{dx : Ax = e,\ x_j = 0 \text{ or } 1,\ \forall j \in N\}$$

where A is an $m \times n$ zero-one matrix, $d \in \mathbb{R}^n$ is an arbitrary objective, $e = (1, \ldots, 1)$ is an m-vector, and $N = \{1, \ldots, n\}$. The advantage of viewing MRF MAP as set partitioning (SPP) is that it allows us to exploit the considerable amount of research done on SPP.

3.1 Facets of the set packing polytope

The set partitioning problem is closely related to both the *set packing* problem:

$$\text{(SP)} \qquad \max \{d'x : Ax \leq e,\ x_j = 0 \text{ or } 1,\ \forall j \in N\}$$

and the *set covering* problem:

$$\text{(SC)} \qquad \min \{d''x : Ax \geq e,\ x_j = 0 \text{ or } 1,\ \forall j \in N\}$$

As Balas and Padberg (1976) show, it is possible, by altering the objective vector, to reformulate any set partitioning problem (SPP) into either as a set packing problem (SP) or a set covering problem (SC). From this it follows that any SP or SC algorithm can be applied to MRF MAP.

Connecting SPP to SP is useful when searching for *cutting planes* for an MRF MAP IP. IP solvers begin by solving the linear relaxation of the IP instance, a problem that is polynomially solvable. In the linear relaxation variables are permitted to take any real value between their lower and upper bounds and so the solution x_{LP} is typically fractional and thus not a solution to the original problem. The objective value for x_{LP} does however provide a useful upper bound on an optimal solution. IP solvers will then typically look for *cutting planes*: inequalities which are valid for the IP but which are violated by x_{LP}. Adding such inequalities and solving the new linear relaxation produces a tighter upper bound and may even result in an integer solution. Generally, the best cutting planes to add are *facets*, the inequalities which jointly define the convex hull of all (integer) feasible solutions. The convex hull of integer feasible solutions for set partitioning, set packing and set covering problems are called the *set partitioning polytope*, *set packing polytope* and *set covering polytope*, respectively.

In this section we will consider facets of the set *packing* (not set *partitioning*) polytope. Rather than reformulate an SPP as an equivalent SP problem, we *relax* a SPP instance into a non-equivalent SP instance by replacing $=$ with $\leq$ in the problem formulation. Note that the SP relaxation has exactly the same variables and constraint matrix A as the SPP instance. Due to the close connection between SPP and SP, facets of this set packing relaxation often provide good cutting planes for the original SPP instance (Balas and Padberg, 1976). In this section we examine these facets.

To find such facets the *intersection graph* G_A for an SP problem with matrix A is crucial. The vertices of G_A are the SP variables. There is an edge between x_i and x_j iff both occur together in some set packing constraint, which means that at most one can have value 1 in any feasible solution. Call a set of vertices in a graph *complete* if each pair of vertices are adjacent. A *clique* is a maximally complete set. *Odd holes* and *odd anti-holes* in G_A define SP inequalities which can be used as cutting planes for SPP. A *hole* is a chordless cycle of length greater than three, an *anti-hole* is the complement of a hole. An *odd* hole/anti-hole is just one with odd length. If G_A contains no odd holes and no odd anti-holes it is *perfect* in which case the SP polytope is integral, and so SP can be solved in polynomial time by solving its linear relaxation.

To see how odd holes define inequalities, consider the SPP formulation of MRF MAP for the hypergraph with hyperedges $\{\{A, B\}, \{B, C\}, \{C, D\}, \{A, D\}\}$ and suppose all variables are binary with values 0 and 1. Abbreviate, for example, $I(A = 0, B = 1)$ to a_0b_1 and all other IP variables for joint instantiations similarly. The intersection graph for the SP relaxation and an odd hole subgraph are shown in Fig 1(i) and Fig 1(ii), respectively.

It is not difficult to see that at most 2 IP variables in the odd hole can have value 1, leading to the valid inequality:

$$a_0b_0 + b_1c_0 + c_1d_0 + a_0d_1 + a_1d_0 \leq 2 \tag{5}$$

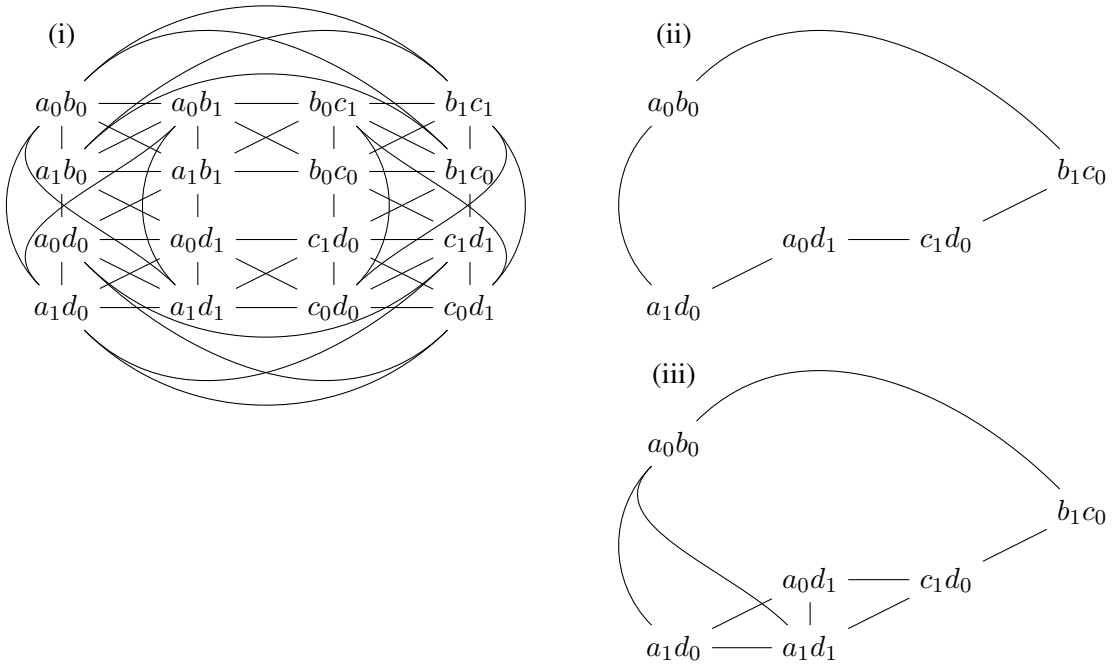

Figure 1: (i): Intersection graph for $\{\{A, B\}, \{B, C\}, \{C, D\}, \{A, D\}\}$. (ii): An odd hole in the intersection graph. (iii): The odd hole 'with lifting' in the intersection graph.

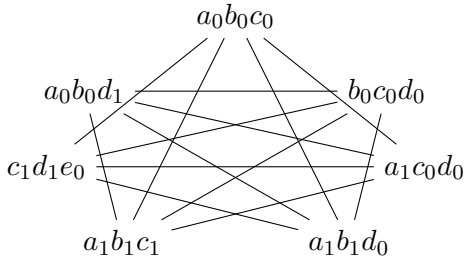

Figure 2: An odd anti-hole in the intersection graph for:
$\{\{A, B, C\}, \{A, B, D\}, \{A, C, D\}, \{B, C, D\}, \{C, D, E\}\}$.

In general, in an odd hole of length n the variables sum to at most $(n-1)/2$. In an odd anti-hole of any length the variables sum to at most 2. If the intersection graph is exactly equal to an odd hole or odd anti-hole then the relevant inequality is a facet for the SP problem (Balas and Padberg, 1976). Fig 2 shows an anti-hole of length 7 for the hypergraph $\{\{A, B, C\}, \{A, B, D\}, \{A, C, D\}, \{B, C, D\}, \{C, D, E\}\}$ with binary variables, from which we get the valid inequality:

$$a_1b_1c_1 + c_1d_1e_0 + a_0b_0d_1 + a_0b_0c_0 + b_0c_0d_0 + a_1c_0d_0 + a_1b_1d_0 \leq 2 \tag{6}$$

As just mentioned, if the 5 variables in the odd hole shown in Fig 1(ii) were the only variables then the given odd hole inequality would be a facet—a maximally strong inequality. However, since there are other variables it can be strengthened by 'lifting': adding in extra variables. There is a large literature on (the various) lifting procedures, see Balas and Padberg (1976) and references therein. Here we settle for an example. The subgraph shown in Fig 1(iii) illustrates how (5) can be lifted to:

$$a_0b_0 + b_1c_0 + c_1d_0 + a_0d_1 + a_1d_0 + a_1d_1 \leq 2 \tag{7}$$

The non-cycle variable a_1d_1 can be lifted into the inequality since it is adjacent to a 3-chain in the cycle. It is, in fact, adjacent to a 4-chain, but a 3-chain is sufficient. This observation concerning 3-chains was made by Alvarez-Valdes et al. (2005) as part of a heuristic approach to lifting odd-cycle inequalities.

In MRF MAP it is often possible to do a lot of lifting using *graph substitution*. Consider the hypergraph $\{\{A, B, E, F\}, \{B, C\}, \{C, D\}, \{A, D\}\}$ and the lifted odd-cycle shown in Fig 1(iii). If a_0b_0 in Fig 1(iii) is replaced with, say, $a_0b_0e_0f_0$ then we have a lifted odd cycle with its associated inequality. However, a much better inequality is possible by removing the vertex $a_0b_0e_0f_0$ and substituting for it the graph which is the clique with nodes $a_0b_0e_0f_0$, $a_0b_0e_0f_1$, $a_0b_0e_1f_0$ and $a_0b_0e_1f_1$. The graph substitution construction is due to Chvátal (1975) and is discussed by Balas and Padberg (1976).

To examine whether facets of the *SP relaxation* of an MRF MAP SPP are also facets of the SPP instance, the convex hull for the MAP problem for the hypergraph $\{\{A, B\}, \{B, C\}, \{C, D\}, \{A, D\}\}$ assuming binary variables, was computed using the lrs algorithm (Avis, 2000) (`http://cgm.cs.mcgill.ca/~avis/C/lrs.html`). Although this problem, as originally stated, has 16 variables the convex hull has only 8 dimensions due to the presence of 8 linearly independent equations in the problem which can be used to reduce the number of variables to 8. lrs found that there are 24 facets for this convex hull in $\mathbb{R}^8$. These are the 8 lower bounds on the 8 variables, 8 inequalities which follow immediately from marginal consistency and then a further 8 inequalities. This last set of 8 inequalities consists of 7 inequalities which are odd hole inequalities for 5 vertices lifted with one additional vertex (like in Fig 1(ii)) and only one inequality $c_0d_0 \leq a_1b_1 + b_0c_0 + a_0d_0$ which is not an odd-hole (or odd anti-hole) inequality. So on this small instance at least, all but one of the 'interesting' facets of the SPP convex hull are lifted odd hole inequalities.

4. Zero-half cuts

Inequalities corresponding to odd holes and odd anti-holes provide inequalities which can be used as cuts to strengthen the linear relaxation of an MRF MAP problem. But how do we find them (quickly)? In this section we show that zero-half cutting plane algorithms can be used.

Given inequalities representing the constraints in the odd hole shown in Fig 1(ii): $a_0b_0 + b_1c_0 \leq 1, b_1c_0 + c_1d_0 \leq 1, a_0d_1 + c_1d_0 \leq 1, a_0d_1 + a_1d_0 \leq 1, a_1d_0 + a_0b_0 \leq 1$, we can derive the following inequality by multiplying each of the above by $1/2$ and adding them: $a_0b_0 + b_1c_0 + c_1d_0 + a_0d_1 + a_1d_0 \leq 5/2$. Since each variable is an integer this inequality can be strengthened by rounding down the RHS to $\lfloor 5/2 \rfloor = 2$ thus producing the odd hole inequality (5).

The odd hole inequality (5) is thus a *zero-half* cut (short for *zero-half Chvátal-Gomory cut*) since it can be derived from other inequalities by multiplying by $1/2$, summing and then rounding down the RHS. It is not difficult to see that *any* odd hole cut is a zero-half cut constructed using the constraints coming from the edges of the cycle.

The anti-hole cut (6) is a zero-half cut using the following set packing inequalities derived from 5 of the 7 cliques in Fig 2: $a_0b_0c_0 + a_1c_0d_0 + a_1b_1c_1 \leq 1, a_0b_0c_0 + a_1c_0d_0 + c_1d_1e_0 \leq 1, b_0c_0d_0 + a_1b_1d_0 + c_1d_1e_0 \leq 1, b_0c_0d_0 + a_1b_1d_0 + a_0b_0d_1 \leq 1, b_0c_0d_0 + a_1b_1c_1 + a_0b_0d_1 \leq 1$. In fact, we can always find 5 cliques in an odd anti-hole which produce a zero-half cut. If an odd anti-hole has 5 vertices $1, 2, 3, 4, 5$ then it has exactly 5 cliques. If the corresponding hole is $1-2-3-4-5-1$ then these cliques are $1-3$, $2-4$, $3-5$, $4-1$ and $5-2$. It is easy to see that the the odd anti-hole inequality can be generated as a zero-half cut from the 5 associated clique inequalities. Consider now an anti-hole with 7 vertices with corresponding hole $1-2-3-4-5-6-7-1$. We can take each of the cliques $x, x+2(\mod 5)$ from the 5 vertices case and extend to $x, x+2(\mod 7), x+4(\mod 7)$ to get the cliques $1-3-5$, $2-4-6$, $3-5-7$, $4-6-1$ and $5-7-2$. Since each vertex appears at least twice the clique inequalities can be used to generate the odd anti-hole inequality as a zero-half cut. This manner of generating the needed 5 clique inequalities for an anti-hole of size $2(k+1)+1$ from those for an anti-hole of size $2k+1$ is always possible. It follows, by induction, that all odd anti-hole inequalities are zero-half cuts.

The connection between odd-hole and odd anti-hole cuts and zero-half cuts suggests an obvious cutting plane approach for MRF MAP: aggressively use a zero-half cut generating algorithm. We do exactly this in the next section.

4.1 Do zero-half cuts help?

This section contains an evaluation of how useful zero-half cuts are. These experiments were performed using CPLEX 12.6 on a desktop PC running Ubuntu Linux with four 3.2GHz CPUs and 7.8 Gb of RAM. All 458 MRF MAP instances from the PIC 2011 Challenge website `http://www.cs.huji.ac.il/project/PASCAL/` were used. A Python script was used which took MRF MAP instances in PIC format and created an IP instance using CPLEX's Python API.

Experiments were conducted with CPLEX parameters set at their default values except that CPLEX was asked to work hard at producing provably optimal solutions (at the possible expense of failing to do so and missing good sub-optimal solutions). Denote this setting as DEF. CPLEX was then run again with the same settings as DEF except for the following 3 non-default parameter settings: `mip.cuts.zerohalfcut.set(2)` `mip.limits.cutpasses.set(9999999)` `mip.limits.cutsfactor.set(90)`. The last two settings effectively remove limits on cut generation, the first demands that CPLEX uses its zero-half cut generator very aggressively. Denote this setting as ALT.

Fig 3 plots the CPLEX solving times (in seconds) for the two different parameter settings on the 320 MRF MAP instances where at least one setting did not time out, i.e. found an optimal solution within 2000 seconds. Many problems are solved quickly for both parameter settings. However these results suggest that generating zero-half cuts aggressively typically provides improved performance on harder instances—in accordance with the theoretical analysis given above. Note that ALT solves 7 instances which timed out with DEF, whereas DEF only solved 2 instances which timed out with ALT.

5. Reducing the size of MRF MAP problems

One problem with encoding MRF MAP as an IP is that in some cases the resulting IP can have very many variables and constraints leading, typically, to slow solving. In this section, we consider some methods for reformulating MRF MAP problems which lead to smaller IP instances. The goal is to

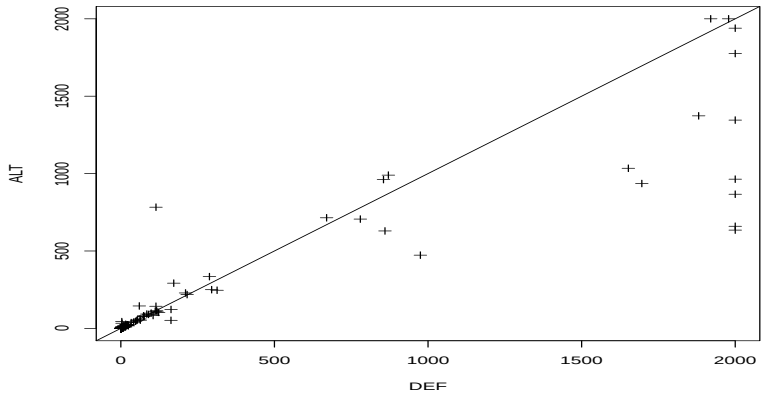

Figure 3: CPLEX solving times (ALT parameters vs DEF parameters) on 320 MRF MAP instances.

reduce the number of $I(x_{\downarrow j})$ IP variables that are needed. We use the number of $I(x_{\downarrow j})$ variables as a measure of the *size* of the problem. We then report on some preliminary experiments examining how helpful such size reduction is.

The simplest, but surprisingly effective, size-reducing measure is to ensure the hypergraph is *reduced* by removing hyperedges contained in other hyperedges: if $E_j \subseteq E_{j'}$ then E_j is *redundant* and removed from E. Its log potential function is added to that of $E_{j'}$:

$$\psi_{j'} \leftarrow \psi_{j'} + \psi_j \tag{8}$$

Another simple size-reducing procedure is to remove any variable X_i that occurs in only one hyperedge E_j. In this case the Markov blanket of X_i is $E_j \backslash \{X_i\}$. For each instantiation of $E_j \backslash \{X_i\}$ we can choose a maximal value of X_i, record this choice, and remove all other values. X_i can then be removed from E_j and H. Once an optimal solution is available for the MAP problem for the MRF with X_i removed, an optimal solution to the original MRF MAP problem can be recovered using the recorded choices for X_i.

Removing a variable in this way can create a redundant hyperedge which can be removed using (8). Continually reducing the hypergraph and removing variables occurring in a single hyperedge is known as *Graham's algorithm*. If (and only if) the hypergraph is *decomposable* (the non-redundant hyperedges are (maximal) cliques of a chordal graph) then Graham's algorithm will eventually remove all variables and hyperedges (Graham, 1979). So, as is well known, MRF MAP is easy for decomposable hypergraphs.

Our third size-reducing method stems from the fact that *any* two hyperedges E_j and $E_{j'}$ can be removed from the MRF and replaced by their union $E_{j''} = E_j \cup E_{j'}$ where $\psi_{j''} \leftarrow \psi_j + \psi_{j'}$. It follows that any variable X_i can be removed from the MRF by (1) replacing the hyperedges which contain it with the single hyperedge $E_{j(i)}$, which is their union, (2) eliminating X_i (now contained only in $E_{j(i)}$) using the procedure given above and (3) removing any newly redundant hyperedges using (8). This variable elimination procedure (Larrosa, 2000) will only reduce the size of the MRF MAP IP in some cases. Here is an example where it does effect a size-reduction. Suppose variables X_1, X_2 and X_3 have the following arities: $r_1 = 10, r_2 = 2, r_3 = 2$. Suppose $E_1 = \{X_1, X_2\}$ and

$E_2 = \{X_1, X_3\}$ are the only hyperedges containing X_1 and that all joint instantiations associated with E_1 and E_2 are feasible, then removing X_1 changes the size of the problem by $4 - 20 - 20 = -36$. Suppose further that the hypergraph contained the edge $E_3 = \{X_2, X_2\}$. E_3 will be removed due to redundancy, so in this case the change in size is $-36 - 4 = -40$. If the removal of a variable X_i by this procedure does not increase the size of the MRF, we will say that X_i *has a small Markov blanket*. Note that Graham's algorithm can be adapted so that not only variables contained in a single hyperedge are removed, but also those with small Markov blankets.

In the example just given the reduction in size was due to the removal of a high arity variable. In our fourth size-reducing method, size is reduced by replacing a set of hyperedges by their union even when no variable can be removed. For example, suppose a hypergraph contained the following hyperedges: $\{X_1, X_2\}$, $\{X_1, X_3\}$, $\{X_1, X_4\}$, $\{X_2, X_3\}$, $\{X_2, X_4\}$, $\{X_3, X_4\}$, where each X_i is binary. Replacing these 6 hyperedges by their union changes the size of the MRF by $1 \times 16 - 6 \times 4 = -8$. If replacing a set of hyperedges by their union does not increase the size of the MRF we say that the hyperedges *have a small union*.

Our fifth and last size-reducing method applies when a variable's Markov blanket is too big to allow its removal (without an increase in size), but where we can still rule out particular joint instantiations $x_{\downarrow j}$. Suppose variable A is contained only in hyperedges $E_1 = \{A, B, C\}$, $E_2 = \{A, B, D\}$ and $E_3 = \{A, E\}$, and all 5 variables are binary. Suppose that $A = 1$ is the only optimal choice for variable A when $B = C = D = E = 0$. Note that it only because B, C, D, E is the Markov blanket for A that it is possible to determine an optimal choice for A without reference to any other variables. Abbreviate $A = 1, B = 0, C = 0$ to $a_1b_0c_0$ and all other joint instantiations similarly. Let $I(a_1b_0c_0) = 1$ indicate that $A = 1, B = 0, C = 0$ is part of an *optimal* full joint instantiation, and similarly for all other joint instantiations. It follows that **if** $B = C = D = E = 0$, then $I(a_1b_0c_0)I(a_1b_0d_0)I(a_1e_0) = 1$. Now in any full joint instantiation exactly one of the 16 possible instantiations of B, C, D, E must hold, and for each of those 16 instantiations an optimal choice for A can be determined. Suppose, for simplicity, that the only optimal choice for A is 0 when either $B = 0, C = 1, D = 1, E = 0$ or $B = 0, C = 0, D = 1, E = 1$ and $A = 1$ is the only optimal choice for all of the other 14 joint instantiations of B, C, D, E. It follows that exactly one of the following 16 terms will equal 1 in any *optimal* full joint instantiation: $I(a_1b_0c_0)I(a_1b_0d_0)I(a_1e_0)$, $I(a_1b_0c_0)I(a_1b_0d_0)I(a_1e_1)$, $I(a_1b_0c_0)I(a_1b_0d_1)I(a_1e_0)$, $I(a_0b_0c_0)I(a_0b_0d_1)I(a_0e_1)$, $I(a_1b_0c_1)I(a_1b_0d_0)I(a_1e_0)$, $I(a_1b_0c_1)I(a_1b_0d_0)I(a_1e_1)$, $I(a_0b_0c_1)I(a_0b_0d_1)I(a_0e_0)$, $I(a_1b_0c_1)I(a_1b_0d_1)I(a_1e_1)$, $I(a_1b_1c_0)I(a_1b_1d_0)I(a_1e_0)$, $I(a_1b_1c_0)I(a_1b_1d_0)I(a_1e_1)$, $I(a_1b_1c_0)I(a_1b_1d_1)I(a_1e_0)$, $I(a_1b_1c_0)I(a_1b_1d_1)I(a_1e_1)$, $I(a_1b_1c_1)I(a_1b_1d_0)I(a_1e_0)$, $I(a_1b_1c_1)I(a_1b_1d_0)I(a_1e_1)$, $I(a_1b_1c_1)I(a_1b_1d_1)I(a_1e_0)$ and $I(a_1b_1c_1)I(a_1b_1d_1)I(a_1e_1)$. For each hyperedge E_1, E_2 and E_3, one of the joint instantiations appearing in these 16 terms must hold in any optimal full joint instantiation (since otherwise the 16 terms would sum to 0). It follows that $a_0b_1c_0$ and $a_0b_1c_1$ for hyperedge E_1 and $a_0b_1d_0$ and $a_0b_1d_1$ for E_2 can never hold in any optimal full joint instantiation. This knowledge can be represented by setting $\psi_1(a_0b_1c_0) = \psi_1(a_0b_1c_1) = \psi_2(a_0b_1d_0) = \psi_2(a_0b_1d_1) = -\infty$.

This method for removing sub-optimal joint instantiations can be profitably applied over several rounds. Continuing our example, suppose that, by applying this method to the Markov blanket of B we deduced that $\psi(a_0b_0c_1)$ could be set to $-\infty$, i.e. $I(a_0b_0c_1) = 0$. It follows that $I(a_0b_0c_1)I(a_0b_0d_1)I(a_0e_0) = 0$ which in turn implies that $I(a_0e_0) = 0$ since this is the only term in in which $I(a_0e_0)$ occurs. So, by exploiting the Markov structure of the MRF we can perform mul-

tiple rounds of fixing some of the $I(x_{\downarrow j})$ variables to 0. In the language of constraint programming this is *domain reduction propagation*.

5.1 Does size reduction help?

In preliminary experimental work we have found that size-reduction can sometimes lead to far quicker solving but in other cases can slow down solving. We have focused on the 21 'Grids' MRF MAP instances from the PIC 2011 challenge. In all 21 cases it was possible to reduce the size of the original MRF MAP instance considerably, even though, for speed, not all possible size reductions were applied. For example, the original grid80x80.f15.wrap instance had 19,200 hyperedges, 6,400 random variables and 64,000 feasible instantiations. After CPLEX's pre-processing this led to an IP with 38,394 constraints and 57,592 IP variables. After applying size-reduction this instance ended up with 3,105 hyperedges, 3,200 random variables and 39,682 feasible instantiations. The resulting IP (after pre-processing by CPLEX) had 40,171 constraints and 42,602 variables. Note that *the number of constraints has increased* despite the reduction in IP variables. This is because the creation of fewer but bigger hyperedges leads to more 'intersection' constraints of type (3).

We then attempted to solve each of the 21 Grids instances using the original and size-reduced MRF MAP instances, in both cases using the ALT CPLEX parameter settings. We used the WCSP-encoded versions of each instance created by Hurley et al. (2016), where the problem becomes one of minimisation. We denote our approach as CPLEX(ZRI) for 'CPLEX with many Zero-half cuts, Reduction and Intersection constraints'. Using CPLEX 12.8.0 and a single core of a 3.60GHz machine with 16 GB of RAM each instance bar two was solved to optimality within 3600 seconds, with optimal objective value given in the OPT column. Solving times are shown in Table 1. For the RED=YES column (where size-reduction was applied) we give the total time taken including the Python implemented, non-optimised size-reduction procedures. CPLEX solving time alone is given in parentheses.

A very clear pattern emerges: size-reduction is beneficial on 'wrap' instances and deleterious on non-wrap instances. 'wrap' indicates that the original grid has been wrapped around to create a torus. It is not clear why size-reduction was more useful for the wrapped instances.

Table 1 also contains results (under HURLEY-BEST) from Hurley et al. (2016) on these instances. Hurley et al. compared 6 solvers on a wide range of MRF MAP instances, including daopt (Otten et al., 2012) a version of which won the PIC 2011 MAP challenge. Using a 3600 seconds timeout, they found that toulbar2 (Favier et al., 2011) performed best on the grid80x80.f15 instance and CPLEX, using 'direct' encoding, did best on the other 20 Grids instances. The objective value achieved by the best solver is given (OBJ) as well as solving time (t). Some care is required when comparing the CPLEX(ZRI) and HURLEY-BEST results since Hurley et al. were using CPLEX 12.6.0.0 and (a single core of) a 2.3GHz machine with 8GB of RAM. Nonetheless, these results provide some evidence that **on these 21 instances** CPLEX(ZRI) compares well to existing state-of-the-art methods.

Hurley et al. (2016) state that "Surprisingly, cplex with the direct encoding was the best on the Grid category (n = 6400, d = 2), benefiting from a large number of *zero-half cuts*." which supports our finding that this category is particularly suitable for a zero-half cutting plane strategy. However, **on other** MRF MAP instances so far examined, CPLEX(ZRI) is beaten by the best solver (often toulbar2) from Hurley et al. (2016). (Go to `https://www.cs.york.ac.uk/aig/sw/cplex_zri/` to see the full results.) Leaving aside easy instances (quickly solved by all

		CPLEX(ZRI)		HURLEY-BEST	
Instance	OPT	RED=NO	RED=YES	OBJ	t
grid20x20.f10	91581	2	7(2)	91581	9
grid20x20.f10.wrap	96412	3	3(0)	96412	10
grid20x20.f15	128075	2	5(0)	128075	10
grid20x20.f15.wrap	136511	2	3(0)	136511	10
grid20x20.f5.wrap	55939	1	3(0)	55939	6
grid40x40.f10	370567	16	39(24)	370567	101
grid40x40.f10.wrap	398635	20	16(1)	398635	457
grid40x40.f15	521289	19	47(27)	521289	149
grid40x40.f15.wrap	562547	27	20(1)	562547	546
grid40x40.f2	122308	3	22(1)	122308	12
grid40x40.f2.wrap	128534	3	14(0)	128534	25
grid40x40.f5	218240	6	19(4)	218240	39
grid40x40.f5.wrap	233621	11	14(0)	233621	200
grid80x80.f10	1555353	1121	1643(1556)	1558819	3600
grid80x80.f10.wrap	1632909	3071	87(12)	1646415	3600
grid80x80.f15	2190818	2150	*3678(3600)	2239504	3600
grid80x80.f15.wrap	2272095	**3600	79(13)	2293410	3600
grid80x80.f2	511424	18	117(13)	511424	121
grid80x80.f2.wrap	514152	20	77(2)	514152	110
grid80x80.f5	917776	71	151(67)	917946	3600
grid80x80.f5.wrap	950425	152	68(4)	951943	3600

Table 1: Comparative results on PIC 'Grids' examples. Times in seconds rounded down. *Objective value = 2232119 at timeout. **Objective value = 2275766 at timeout

solvers) the CPLEX(ZRI) strategy of adding very many zero-half cuts and focusing on optimality does lead to reasonable progress on improving the bound on optimal solutions, but on many hard problems an optimal solution will not have been found within 3600 seconds and the incumbent will typically be far from optimal.

6. Related work

The MRF MAP problem is sufficiently important to have generated a large literature, much of it using IP and/or LP relaxations (Wainwright and Jordan, 2003; Sontag and Jaakkola, 2008; Batra et al., 2011; Sontag et al., 2012; Mezuman et al., 2013; Hurley et al., 2016; Rowland et al., 2017). Weller and Jebara (2013) derive a *nand Markov random field (NMRF)* from a given MRF. The NMRF is the same as the intersection graph used in this paper and elsewhere (Balas and Padberg, 1976). After reparameterising appropriately (and perhaps reducing the problem) the goal is then to find a maximum weight stable set (MWSS) on the NMRF/intersection graph. MWSS is just an alternative name for node packing (NP) and as Balas and Padberg note "one way of solving set packing (and set partitioning) problems, is to solve the associated node packing problem" (Balas and Padberg, 1976).

However, Weller and Jebara go much further than just translating the problem: providing useful decomposition theorems and analysing 'frustrated' cycles in the NMRF.

Sontag et al. (2012) focus on efficiently finding *cycle inequalities* which can be added as cuts to tighten the LP relaxation. In contrast to earlier work (Sontag and Jaakkola, 2008), these inequalities are applied in the dual problem, giving rise to a column generation approach. These cycle inequalities are closely related to the odd hole inequalities discussed in this paper. It would be interesting to compare their algorithm to the Grötschel-Lovasz-Schrijver and Hoffman-Padberg algorithms, and also to look into lifting their inequalities.

Merging several hyperedges, as described in Section 5, creates new IP variables which are products of the original ones. Sherali and Lee (1996) show that doing this creates tighter linear relaxations for set partitioning, providing an motivation additional to size reduction. Favier et al. (2011), in contrast, have a solving approach where splitting hyperedges apart provides considerable benefits.

7. Future work

The current paper is an initial exploration of the value of viewing MRF MAP as set partitioning. Further work is required to better understand when generating many zero-half cuts and/or reducing the problem is beneficial. Also, the 5 reduction procedures given in Section 5 are here only applied during pre-processing. But there is no reason why they should not be applied *during the search*. Larrosa (2000), for example, has shown how doing variable elimination during search can be beneficial.

A more radical option is to exploit the *quasi-integrality* of the polytope defined by the linear relaxation of set partitioning. Call this polytope LSPP. The quasi-integrality of LSPP means that any edge joining two vertices of the set partitioning polytope is also an edge of LSPP. This means that "the set partitioning problem can in principle be solved by a modified version of the simplex method, generating only integer solutions" (Balas and Padberg, 1976). Some impressive results have been obtained by applying this *integral simplex* algorithm to SPP (Zaghrouti et al., 2014). It would be interesting to try for MRF MAP, perhaps exploiting some of the special structure of MRF MAP SPP problems.

ACKNOWLEDGEMENTS

Thanks to: 3 anonymous reviewers for comments on an earlier version; Uri Heinemann for info on the 'Grids' instances; to Simon de Givry for help on the WCSP encoding and Milan Studený for mediating the financial support from his institution.

References

R. Alvarez-Valdes, F. Parreño, and J. Tamarit. A branch-and-cut algorithm for the pallet loading problem. *Computers & Operations Research*, 32(11):3007 – 3029, 2005.

D. Avis. A revised implementation of the reverse search vertex enumeration algorithm. In *Polytopes - Combinatorics and Computation*, pages 177–198. Birkhäuser Basel, 2000.

E. Balas and M. W. Padberg. Set partitioning: A survey. *SIAM Review*, 18(4):710–760, 1976.

D. Batra, S. Nowozin, and P. Kohli. Tighter relaxations for MAP-MRF inference: A local primal-dual gap based separation algorithm. In *Proc. Artificial Intelligence and Statistics*, 2011.

V. Chvátal. On certain polytopes associated with graphs. *Journal of Combinatorial Theory (B)*, 18: 138–154, 1975.

P. Duchet. Hypergraphs. In *Handbook of Combinatorics*, volume 1, pages 381–432. North-Holland, Amsterdam, 1995.

A. Favier, S. de Givry, A. Legarra, and T. Schiex. Pairwise decomposition for combinatorial optimization in graphical models. In *Proc. IJCAI-11*, 2011.

M. H. Graham. On the universal relation. Technical report, University of Toronto, Toronto, Canada, 1979.

B. Hurley, B. O'Sullivan, D. Allouche, G. Katsirelos, T. Schiex, M. Zytnicki, and S. de Givry. Multi-language evaluation of exact solvers in graphical model discrete optimization. *Constraints*, 21 (3):413–434, July 2016.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

J. Larrosa. Boosting search with variable elimination. In *Proc. CP*, pages 291–305, 2000.

E. Mezuman, D. Tarlow, A. Globerson, and Y. Weiss. Tighter linear program relaxations for high order graphical models. In *Proc. UAI 2013*, 2013.

L. Otten, A. Ihler, K. Kask, and R. Dechter. Winning the PASCAL 2011 MAP Challenge with enhanced AND/OR branch-and-bound. In *NIPS Workshop on Discrete Optimization (DiscML)*, 2012.

M. Rowland, A. Pacchiano, and A. Weller. Conditions beyond treewidth for tightness of higher-order LP relaxations. In *Proc. Artificial Intelligence and Statistics (AISTATS)*, 2017.

H. D. Sherali and Y. Lee. Tighter representations for set partitioning. *Discrete Applied Mathematics*, 68:153–167, 1996.

D. Sontag and T. Jaakkola. New outer bounds on the marginal polytope. In *Proc. NIPS'08*, 2008.

D. Sontag, D. K. Choe, and Y. Li. Efficiently searching for frustrated cycles in MAP inference. In *Proc. UAI'12)*, 2012.

M. Wainwright and M. I. Jordan. Graphical models, exponential families and variational inference. Technical Report 649, UC Berkeley, Dept. of Statistics, 2003.

A. Weller and T. Jebara. On MAP inference by MWSS on perfect graphs. In *Proc. UAI'13*, 2013.

A. Zaghrouti, F. Soumis, and I. El Hallaoui. Integral simplex using decomposition for the set partitioning problem. *Operations Research*, 62(2):435–449, 2014.