Graphical Models for Inference and Learning in Computer Vision

Julian McAuley

August, 2011

A thesis submitted for the degree of Doctor of Philosophy
of the Australian National University

Declaration

The work in this thesis is my own except where otherwise stated.

Julian McAuley

Acknowledgements

One can hardly think of a better place than Canberra to do research. King O'Malley, in selecting a location for our nation's capital, proclaimed that 'cold climates have produced the greatest geniuses,' though one can only give so much credit to the weather. I've been lucky to live in a great city, but also to study at a great institution, under a fantastic supervisor. I first worked with Tibério on a summer internship in 2005, then as a research assistant in 2006, as an honours student in 2007, and finally as a graduate student since 2008. They've been six fantastic years, and I can only hope that the next six are as fruitful and as fun.

Many people have come and gone in that time, and have given me supervision and support along the way. I've received indispensable advice from Alex Smola, Bob Williamson, Li Cheng, Mark Reid, Richard Hartley, Scott Sanner, Marconi Barbosa, Wray Buntine, Edwin Bonilla, and Chris Webers, to name but a few. I've also been lucky to collaborate with Teo de Campos, Florent Perronnin, Gabriela Csurka, Rogerio Feris, Arnau Ramisa, Pedro Felzenszwalb, and others whom I've never met in person.

I'm also grateful to my fellow students for their friendship and encouragement: Javen Shi, Choon Hui Teo, Dmitry Kamenetsky, Novi Quadrianto, Jin Yu, Matthew Robards, Xinhua Zhang, Shengbo Guo, Owen Thomas, Quoc Le, Le Song, and many others whom I've met along the way. Especially I'd like to thank James Petterson, Roslyn Lau, and Thomas Mensink for proofreading this thesis; I'd like to thank James a second time for sharing with me his cubicle and his boundless optimism.

Finally, I'd like to thank my friends and family, whose tireless and unconditional support has been invaluable over the last three years, just as it had been for the twenty-three before those.

Abstract

Graphical models are indispensable as tools for inference in computer vision, where highly structured and interdependent output spaces can be described in terms of low-order, *local* relationships. One such problem is that of *graph matching*, where the goal is to localise various parts of an object within an image: although the number of joint configurations of these parts may be very large, the relationships between them can typically be described in terms of simple skeletal structures, which lead to tractable inference.

We study problems of this type from three perspectives: firstly, how can we design graphical models to accurately model matching problems in computer vision? Secondly, how can machine learning be applied to such models, so that our inference algorithms can leverage the characteristics of a certain dataset? Thirdly, how can we exploit the specific energies that arise in this domain to develop faster inference algorithms, beyond the pessimistic worst-case results known for inference in graphical models? Naturally, these three problems are closely related, since both accurate and efficient inference are required by many learning algorithms.

Contents

Chapter 1

Contribution of this Thesis

Broadly, the core contributions of this thesis fall into three categories. Initially, we investigated new graphical models to solve graph matching problems more accurately and efficiently. Secondly, we applied *structured learning* approaches to matching problems from vision, in order to develop algorithms that are more robust to the types of variation in vision datasets. Finally, we started working on inference algorithms for graphical models that exhibit a certain type of 'factorisation', to be described later. The type of factorisation was initially motivated by graphical models for matching problems, however we ultimately applied such models to a broad class of applications beyond graph matching, including some outside of computer vision altogether.

1.1 Publications Included in this Thesis

The following publications constitute the primary contribution of this thesis, and are to be discussed in the following chapters. The contribution of each is briefly described in Section 1.4.

1.1.1 Graphical models for Vision and Matching

Julian J. McAuley, Tibério S. Caetano, and Marconi S. Barbosa. Graph rigidity, cyclic belief propagation and point pattern matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008 (McAuley et al., 2008a).

Julian J. McAuley and Tibério S. Caetano. Fast matching of large point sets under occlusions. *Pattern Recognition*, 2011 (McAuley and Caetano, 2011b).

1.1.2 Structured Learning for Vision

Julian J. McAuley, Tibério S. Caetano, and Alex J. Smola. Robust near-isometric matching via structured learning of graphical models. *Advances in Neural Information Processing Systems*, 2008 (McAuley et al., 2008b).

Julian J. McAuley, Teofilo de Campos, and Tibério S. Caetano. Unified graph matching in euclidean spaces. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010 (McAuley et al., 2010b).

1.1.3 Inference Algorithms

Julian J. McAuley and Tibério S. Caetano. Exploiting within-clique factorizations in junction-tree algorithms. *Artificial Intelligence and Statistics*, 2010 (McAuley and Caetano, 2010a).

Julian J. McAuley and Tibério S. Caetano. Exploiting data-independence for fast belief-propagation. *International Conference on Machine Learning*, 2010 (McAuley and Caetano, 2010b).

Julian J. McAuley and Tibério S. Caetano. Faster algorithms for max-product message-passing. *Journal of Machine Learning Research*, 2011 (McAuley and Caetano, 2011a).

1.2 Other Primary-Author Publications

These publications were produced during the course of this thesis, though they are outside of the main thread of the papers mentioned above. Both are concerned with structured learning for vision, but the specific applications and inference algorithms used are different from those papers in the previous sections.

Julian J. McAuley, Teofilo de Campos, Gabriela Csurka, and Florent Perronnin. Hierarchical image-region labeling via structured learning. In *British Machine Vision Conference*, 2009 (McAuley et al., 2009).

Julian J. McAuley, Arnau Ramisa, Tibério S. Caetano Optimization of robust loss functions for weakly-labeled image taxonomies: an ImageNet case study In

Energy Minimization Methods in Computer Vision and Pattern Recognition, 2011 (McAuley et al., 2011).

1.3 Contributing-Author Publications

Below are the publications to which I made a significant contribution but was not the primary author.

1.3.1 Graphical models for Graph Matching

Tibério S. Caetano and Julian J. McAuley. Faster graphical models for point-pattern matching. *Spatial Vision*, 2009 (Caetano and McAuley, 2009).

1.3.2 Structured Learning for Vision

Tibério S. Caetano, Julian J. McAuley, Li Cheng, Quoc V. Le, and Alex J. Smola. Learning graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009 (Caetano et al., 2009).

Longbin Chen, Julian J. McAuley, Rogerio S. Feris, Tibério S. Caetano, and Matthew Turk. Shape classification through structured learning of matching measures. *IEEE Conference on Computer Vision and Pattern Recognition*, 2009 (Chen et al., 2009).

1.3.3 Inference Algorithms

Pedro F. Felzenszwalb and Julian J. McAuley. Fast inference with min-sum matrix product. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011 (Felzenszwalb and McAuley, 2011).

1.4 Summary of Core Publications

In McAuley et al. (2008a), we introduced a new graphical model for solving near-isometric matching problems. Previously, a model had been proposed with quartic time complexity, via exact inference in a chordal graphical model with maximal cliques of size four. Significantly, the model we introduced is not chordal, though we showed that loopy belief propagation in such a model converges to

the optimal solution, leading to an iterative algorithm that is cubic time per iteration. We later relaxed the iterative requirement in Caetano and McAuley (2009), leading to a strictly cubic time algorithm. More recently, we developed algorithms that are sub-cubic in the *expected* case (McAuley and Caetano, 2011b).

We first applied machine learning to these problems in McAuley et al. (2008b), showing that such graphical models can be used in practical vision scenarios. This extended a previous line of work from Caetano et al. (2009), in which we applied machine learning to the problem of graph matching, but did so using classical matching techniques, rather than performing matching via a graphical model. We continued to apply structured learning to graphical models for matching tasks in Chen et al. (2009) and McAuley et al. (2010b).

Finally, we began to investigate the core algorithms used to perform inference in graphical models. The running time of inference in a chordal graph grows exponentially in the graph's maximal clique size, which fails to leverage the fact that the potentials *within* the cliques may themselves decompose, in which case inference via message passing can be related to basic matrix multiplication operations. In McAuley and Caetano (2010a) we showed that this was indeed the case for the models for graph matching that we had developed. This proved to be a general-purpose technique with many applications beyond graph matching, which we studied in McAuley and Caetano (2010b) and McAuley and Caetano (2011a).

After presenting the necessary technical material in Chapter 3, a more detailed introduction to these publications is given in Chapter 4.

1.5 This Document

In Chapter 2 we present a literature review of graph matching, structured learning, and inference in graphical models. In Chapter 3 we summarise the core technical material common to the above publications. Finally, in Chapter 4 we present the core publications in further detail, briefly summarising the contribution of each.

Chapter 2

Literature Review

2.1 Graphical Models for Vision and Matching

Identifying a correspondence between two sets of point features is a fundamental problem in pattern recognition, with applications in fields as diverse as molecular biology (Nussinov and Wolfson, 1991; Akutsu et al., 2003), computational chemistry (Martin et al., 1993; Finn et al., 1998), and astronomy (Murtaugh, 1992). It is especially important in computer vision, where applications include object detection (Frome et al., 2004); image categorisation, classification, and retrieval (Mori et al., 2001; Belongie et al., 2002); pose reconstruction (Mori and Malik, 2002); fingerprint recognition (van Wamelen et al., 2004), image registration (Fitzgibbon, 2001); and shape matching (Chui and Rangarajan, 2000; Mori et al., 2005), to name but a few.

Broadly speaking, applications of such techniques in vision consist of identifying a correspondence between two or more sets of *interest points*. There are a variety of reasons why one might model a computer vision problem in this setting: in certain applications, one is interested in the correspondence itself, for instance when identifying a match between 'spots' (each representing a protein) in a pair of electrophoresis gels (Rogers and Graham, 2007), or localising specific parts of a complex object in a scene (Felzenszwalb and Schwartz, 2007). In other settings, a correspondence can be used as a proxy to recover the transformation between the points in two images, a critical step in reconstructing a three dimensional view of a scene (Hartley and Zisserman, 2004), or the pose of an object (Sigal and Black, 2006). Finally, and perhaps most simply, if matching can be posed as an energy minimisation problem, then the energy of a match can be used as a similarity measure between two scenes, and can be embedded in a detection or classification

system (Carmichael and Hebert, 2004; Berg et al., 2005; Mikolajczyk et al., 2006; Leordeanu et al., 2007); such a setting also allows matching to be combined with powerful learning techniques (Zhang and Malik, 2003; LeCun et al., 2004; Chen et al., 2009).

The problem of identifying such a correspondence, or 'matching', can be modelled in several different forms, depending on the demands of the application domain in which it is required (Amit and Kong, 1996; Carcassoni and Hancock, 2000; Belongie et al., 2002; Robles-Kelly and Hancock, 2006). Perhaps the simplest is the so-called 'linear assignment' problem, where one wishes to maximise some similarity score between corresponding points, subject only to a 'monogamy' constraint, i.e., a constraint stating that the correspondence should be bijective (Hitchcock, 1941; Burkard et al., 2009). Such models are appealing largely because they admit an efficient solution (Kuhn, 1955; Munkres, 1957; Papadimitriou and Steiglitz, 1982; Jonker and Volgenant, 1987; Huang and Jebara, 2011), though the bijectivity assumption proves to be natural for vision problems, where it encodes the fact that an interest point in one image can correspond to at most one interest point in another. Although a linear assignment formulation captures only first-order relationships between points, and therefore can only be used to model *local* appearance, such first-order models have proved very useful when coupled with powerful image features that describe the local appearance of an interest point (Lowe, 1999; Belongie and Malik, 2000; Belongie et al., 2002; Mikolajczyk and Schmid, 2004).

The quadratic assignment problem extends this model to include second-order affinities (Gold and Rangarajan, 1996; Anstreicher, 2003), encoding for example the property that distances should be preserved by the correspondence, a typical assumption made when matching shapes (Felzenszwalb, 2005). However, with the inclusion of second-order affinities, any hope of a tractable solution vanishes due to reductions from notoriously hard isomorphism problems (Ullman, 1976; LaPaugh and Rivest, 1978; Garey and Johnson, 1979). Nevertheless, due to the expressiveness of such models, many approximations and heuristics have been proposed for the quadratic assignment problem. An incomplete list includes *spectral* methods (Shapiro and Brady, 1992; Carcassoni and Hancock, 2003; Caelli and Kosinov, 2004; Wang and Hancock, 2004; Leordeanu and Hebert, 2005; Cour et al., 2006), *relaxation labelling* and *probabilistic approaches* (Rosenfeld and Kak, 1982; Kittler and Hancock, 1989; Li, 1994; Christmas et al., 1994; Wilson and Hancock, 1997; Hancock and Wilson, 2002; Caetano et al., 2004a), *semidefinite relaxations* (Schellewald, 2004), *replicator equations* (Pelillo, 1999), *tree search*

(Messmer and Bunke, 1998), *graduated assignment* (Gold and Rangarajan, 1996), and *Reproducing Kernel Hilbert Space* (RKHS) methods (van Wyk et al., 2002).

In all of the above cases, matching is modelled essentially as a problem of identifying a correspondence between the nodes and the edges of an *attributed* graph, though the problem of how such graph attributes are obtained is yet to be addressed. In vision problems, the nodes in the graph are typically obtained using an *interest point detector*. Examples of interest points include *corners* (Harris and Stephens, 1988; Smith, 1992; Shi and Tomasi, 1994; Tomasi and Kanade, 2004), *edges* (Canny, 1986; Deriche, 1987), *blobs* (Lindeberg, 1993; Bay et al., 2006) and regions invariant to certain types of transformations (Lindeberg, 1998; Lowe, 2004; Mikolajczyk and Schmid, 2004). Intuitively, the attributes of the nodes in such a graph encode the local appearance of an interest point (Lowe, 1999), or in some sense how a particular node 'sees' the rest of the graph (Belongie and Malik, 2000; Belongie et al., 2002). Edge features can encode properties such as distances between pixels in an image (Fischler and Elschlager, 1973), the intensity or gradient along a path (Kass et al., 1987; Coughlan and Ferreira, 2002), or some topological information from the graph itself (Caetano et al., 2009). In some cases, features of more than two nodes could be used, describing the appearance of the image within a bounded region (Duchenne et al., 2009), or scale-invariant properties about the 'shape' of the graph (Felzenszwalb, 2005; McAuley et al., 2008b).

The idea that edge attributes can encode *distances*, and the assumption that these distances should be preserved by the matching (i.e., the isometry assumption) is particularly natural for many vision problems. This assumption proves to be sufficiently strong that it may again lead to tractable solutions to matching problems; fast algorithms exist for the case of *exact* isometric matching (i.e., no noise) (de Rezende and Lee, 1995; Alt and Guibas, 1999), but achieving an optimal solution for even a small, known amount of noise becomes a computationally difficult (albeit polynomial) problem (Alt et al., 1988). Others seek solutions that are fast under certain conditions, but may be slow or inaccurate in other cases (e.g. graphs such as grids) (Goodrich et al., 1999; van Wamelen et al., 2004). A line of work of particular interest is that of Caelli and Caetano (2005); Caetano and Caelli (2006b); Caetano et al. (2006), which is concerned with algorithms that are *provably optimal and efficient* in the noise-free case, but which give empirically good performance in the case of noise.

The latter line of work draws upon the field of *rigidity theory* (Laman, 1970; Lovász and Yemini, 1982; Connelly, 2005; Jordán and Szabadka, 2009). The idea

of rigidity is related to fault-tolerance in a network, where one wishes to minimise the number of constraints required to enforce certain structural invariants, which in practice translates to minimising the number of edges – or communication channels – in a network (Li et al., 2003). The idea of marrying rigidity theory with inference in graphical models was first suggested by Caetano et al. (2004b), and continued in Caetano and Caelli (2006a,b); Caetano et al. (2006), where new graphs were proposed that are globally rigid when embedded in the plane, but which also have low treewidth, meaning that they can be used to solve isometric matching problems efficiently via inference in a graphical model. We have followed this line of research in a number of subsequent papers (McAuley et al., 2008a; Caetano and McAuley, 2009; McAuley and Caetano, 2010a, 2011b), and more recently we have used this idea in conjunction with higher-order image features and structured learning techniques (McAuley et al., 2008b, 2010b). The inference and learning algorithms explored in these works are the topics of the following sections.

2.2 Structured Learning for Vision

Having phrased matching problems in terms of identifying correspondences between the nodes of two attributed graphs, a natural question is how one can *learn* the graph attributes in question. Perhaps the earliest work along these lines is Pelillo and Refice (1994), in which the authors learn compatibility functions for the relaxation labelling process; this is however a different problem than graph matching, and the 'compatibility functions' have a different meaning. Nevertheless it does provide an initial motivation for learning in the context of matching tasks. Frome et al. (2007) and Blaschko and Lampert (2008) also apply learning to recognition and localisation tasks, albeit using different models to the ones discussed here.

'Structured learning' methods are concerned with learning the parameters for models whose output spaces or loss functions have *interdependent* variables, such as the variables in a graphical model. As we shall see in the following section, such models are ubiquitous in computer vision problems. In the case of the correspondence problems already mentioned, the fact that the assignment should be bijective introduces dependencies between the problem's variables, i.e., it introduces 'structure' into the output space.

Early work on learning in structured output spaces includes Taskar et al.

(2004), which showed examples from character recognition and text classification. In Tsochantaridis et al. (2004, 2005) such techniques were applied to natural language parsing problems. Lacoste-Julien et al. (2006) used structured estimation tools in a quadratic assignment setting for word alignment, which is closely related in terms of methodology to the assignment problems already discussed. We studied structured learning for matching problems in Caetano et al. (2009); McAuley et al. (2008b, 2010b). We applied structured learning to classification problems in McAuley et al. (2009) and McAuley et al. (2011), and in Chen et al. (2009) we did so using graph matching as a proxy for image similarity.

Caetano et al. (2009) confirmed the need for efficient and exact inference in structured learning problems, an issue also discussed in Finley and Joachims (2008). In fact, this issue proved to be so significant that exact algorithms based on linear assignment outperformed approximate quadratic assignment solvers once learning was applied in Caetano et al. (2007, 2009), which motivated us to study exact inference schemes for graph matching. As we discussed earlier, the assumption of *isometry* allows us to include higher-order features in matching problems, while keeping inference efficient and exact (McAuley et al., 2008b, 2010b). Some of the schemes we developed for inference are discussed below.

2.3 Inference Algorithms

Many problems from computer vision can be modelled as inference in a graphical model (Li, 1995). Other than the already discussed examples from graph matching (Caetano et al., 2006), a selection of problems includes segmentation (Kass et al., 1987; Boykov and Jolly, 2001; Szeliski et al., 2008), classification (Wang et al., 2009), restoration (Geman and Geman, 1984; Lan et al., 2006; McAuley et al., 2006), depth estimation (Scharstein and Szeliski, 2002; Tappen and Freeman, 2003; Szeliski et al., 2008; Felzenszwalb and Huttenlocher, 2006), and optical flow (Felzenszwalb and Huttenlocher, 2006).

It is well-known that exact inference in *tree-structured* graphical models can be accomplished efficiently by message-passing operations following a simple protocol making use of the distributive law (Pearl, 1988; Kschischang et al., 2001). It is also well-known that exact inference in *arbitrary* graphical models can be solved by the junction-tree algorithm; its efficiency is determined by the size of the maximal cliques after triangulation, a quantity related to the treewidth of the graph (Aji and McEliece, 2000). Many of the ideas for inference in graphical

models generalise basic dynamic programming operations (Bertele and Brioschi, 1972), which had already seen use in vision applications (Amini et al., 1990).

A selection of introductory texts includes Aji and McEliece (2000); Kschischang et al. (2001); Wainwright and Jordan (2008) (tutorial papers), Roweis and Ghahramani (1999); Smyth (1998); McAuley et al. (2010a) (review articles), Koller and Friedman (2009); Jensen (2001); Bishop (2006) (introductory books), Edwards (2000) (undirected models), Pearl (1988, 2000) (directed models), Cowell et al. (2003) (exact inference), Jordan (1998) (learning and approximate inference) and Lauritzen and Spiegelhalter (1988); Lauritzen (1996) (mathematical theory).

In spite of prohibitive worst-case hardness results based on the treewidth, a fruitful branch of research has involved exploiting strong assumptions about the *structure* of a model that can be used do decrease the running time of inference (Kjærulff, 1998; Ishikawa, 2003; Felzenszwalb and Huttenlocher, 2006; Kumar and Torr, 2006; Kolmogorov and Shioura, 2007; Petersen et al., 2008; Kersting et al., 2009). Well-known examples include submodularity, sparsity, and convexity (Kolmogorov and Zabih, 2002; Felzenszwalb and Huttenlocher, 2006). However, there are numerous lesser-known yet important results: exploiting shared potentials (Kersting et al., 2009); choosing message-passing schemes based on specific inputs (Elidan et al., 2006); exploiting potential functions that are 'truncated' (Veksler, 2007); exploiting topology in bipartite, planar, or grid-like models (Petersen et al., 2008); exploiting nested structure within potential functions (Kjærulff, 1998). A class of models that we study are those whose potential functions 'factorise', meaning that the number of variables in any data-dependent factor is bounded by the treewidth (McAuley and Caetano, 2010a,b, 2011a; Felzenszwalb and McAuley, 2011).

A selection of models from vision that exhibit such types of factorisation includes those of Coughlan and Ferreira (2002); Felzenszwalb (2005); Galley (2006); Sigal and Black (2006); Donner et al. (2007); Tresadern et al. (2009). Typically, pairwise constraints in these models describe some notion of 'elasticity', much like the isometric matching problems mentioned previously (McAuley et al., 2008a). These models 'factorise' in the sense that only pairwise constraints are included, but the 'skeletal' structure of the objects in question is not tree-structured. To make an analogy with a real-world problem, a bridge may be constructed out of pairwise parts (beams), though they would not be connected together in a tree if the bridge is to be structurally sound – instead, they are connected together in a structure such as a truss (National Park Service, 1976). In much the same

way, graphical models representing rigid objects (or objects with rigid parts), will typically not be tree-structured, even though they may be made up of pairwise components. Due to this relationship, the results from global rigidity discussed previously (Connelly, 2005; Jordán and Szabadka, 2009) explain why this class of models is common for vision problems. A selection of models from other areas that exhibit similar types of factorisation includes 'skip-chain' models for natural language processing (Sutton and McCallum, 2006); simultaneous localisation and mapping (Paskin, 2003); and linear programming relaxations (Sontag et al., 2008).

In any such model, belief-propagation reduces to basic matrix multiplication operations, or related operations on tensors (McAuley and Caetano, 2011a). Indeed, classical dynamic programming solutions to inference in graphical models reduce to the same basic operations (Amini et al., 1990). In light of this relationship, a seemingly promising avenue of research involves applying sub-cubic algorithms for matrix-matrix multiplication (Strassen, 1969; Coppersmith and Winograd, 1990; Cohn et al., 2005), and sub-*quadratic* algorithms for matrix-vector multiplication (Williams, 2007), and using these as a component in belief-propagation schemes. Although similar ideas have been suggested in Park and Darwiche (2003), practical applications with small state-spaces are unlikely to benefit from these asymptotic results.

Alternately, the primitive addition and multiplication operations performed by belief-propagation schemes may be replaced by any pair of operations that form a *semiring*, depending on the type of query being performed (Aji and McEliece, 2000). Generally speaking, computing marginal distributions requires operations in the 'sum-product' semiring, which reduces to regular matrix multiplication; computing the most likely configuration of the nodes in a graphical model (*'maximum a posteriori'*, or *'MAP'* inference) requires inference in the 'max-product' semiring, which yields a variant of matrix multiplication known as the 'tropical' or 'distance' product (Simon, 1988; Mikhalkin, 2008; Maclagan and Sturmfels, 2009). Critically, sub-cubic algorithms for matrix multiplication depend on the addition operator being invertible, and therefore no longer apply in the tropical semiring; in fact, no sub-cubic solution is currently known for this problem, and there is weak evidence to suggest that there may be no such solution in the worst-case (Kerr, 1970).

While the existence of a sub-cubic solution remains an open problem, the complexity of tropical matrix multiplication is a topic of interest due to its relationships to existing problems via sub-cubic transformations (Aho et al., 1983;

Williams and Williams, 2010). A selection of such problems includes parsing in context free grammars (Valiant, 1975), metricity detection (Williams and Williams, 2010), and computing distances under the L-infinity norm (McAuley and Caetano, 2011b). Another such problem of interest is the so-called 'all-pairs shortest-paths' problem, which has received particular attention in terms of its *expected-case* performance (Alon et al., 1997; Karger et al., 1993; Chan, 2007; Moffat and Takaoka, 1987; Frieze, 1985), recently culminating in a *quadratic*-time algorithm under certain conditions (Peres et al., 2010).

Unfortunately, the transformations between these problems do not necessarily preserve the assumptions required by their expected-case analysis (Williams and Williams, 2010). For this reason, we studied the expected-case analysis of tropical matrix multiplication in its own right (McAuley and Caetano, 2010a,b, 2011a; Felzenszwalb and McAuley, 2011). Fast expected-case results for this problem lead to asymptotically faster solutions to the 'factorisable' problems already mentioned, including the matching problems discussed previously (McAuley and Caetano, 2010a; Huang and Jebara, 2011).

Chapter 3

Preliminaries

3.1 Graphical Models for Vision and Matching

3.1.1 Assignment Problems

Given two sets of objects $P = \{p_1 \dots p_M\}$ and $Q = \{q_1 \dots q_N\}$, the *linear assignment* problem (Hitchcock, 1941; Burkard et al., 2009) consists of finding an *adjacency matrix* $y \in \{0,1\}^{M \times N}$ such that

$$\hat{y} = \underset{y}{\operatorname{argmax}} \sum_{i=1}^{M} \sum_{i'=1}^{N} c_{ii'} y_{ii'}, \tag{3.1}$$

where $c \in \mathbb{R}^{M \times N}$ is some *compatibility matrix* encoding the 'goodness' of the match between p_i and $q_{i'}$. What makes (eq. 3.1) an assignment problem is the constraint that there is (at most) one non-zero entry in every row and column of y, i.e.,

$$\sum_{i=1}^{M} y_{ii'} \leq 1 \quad \text{and} \quad \sum_{i'=1}^{N} y_{ii'} \leq 1. \tag{3.2}$$

A classical solution to this problem (assuming equality constraints in (eq. 3.2) and that $M = N$) runs in $O(N^3)$ time (Munkres, 1957), though faster solutions exist for certain input distributions (Huang and Jebara, 2011).

Alternately, we can view y in terms of a function $f : \{1 \dots M\} \to \{1 \dots N\}$ such that $y_{ii'} = 1$ if and only if $f(i) = i'$. When expressed in this form, (eq. 3.1) can be rewritten as

$$\hat{f} = \underset{f}{\operatorname{argmax}} \sum_{i \in \operatorname{dom}(f)} c_{if(i)}, \tag{3.3}$$

where the constraints of (eq. 3.2) are equivalent to stating that f should be a *partial bijection*. If equality constraints are used in (eq. 3.2) (assuming that

$M = N$), this is equivalent to stating that f should be a total bijection, in which case it is simply a *permutation*. The formulation of (eq. 3.3) shall prove more natural once we express matching as inference in a graphical model.

Using the function notation defined above, the *quadratic assignment* problem (Gold and Rangarajan, 1996; Anstreicher, 2003) extends the formulation of (eq. 3.1) by adding second-order compatibilities, so that we have

$$\hat{f} = \operatorname*{argmax}_{f} \sum_{i \in \mathrm{dom}(f)} \left[c_{if(i)} + \sum_{j \in \mathrm{dom}(f)} d_{ijf(i)f(j)} \right], \tag{3.4}$$

where $d \in \mathbb{R}^{M \times M \times N \times N}$ is some *pairwise* compatibility function encoding the goodness of the match between the pairs (p_i, p_j) and $(q_{i'}, q_{j'})$. Subject to the assignment constraints of (eq. 3.2) this formulation fails to admit an efficient solution (it is NP-hard, the travelling salesman problem being a special case; see Garey and Johnson, 1979), though as mentioned in Chapter 2 a number of approximation schemes exist, some of which include higher-order affinities (Duchenne et al., 2009).

If $p_i \in P$ and $q_{i'} \in Q$ are assumed to be points in $\mathbb{R}^n$, then the problem of *isometric* matching (de Rezende and Lee, 1995; Alt and Guibas, 1999) is concerned with finding a (total) function f such that the distances between points p_i and p_j in P are preserved by their images $q_{f(i)}$ and $q_{f(j)}$ in Q. A compatibility function that solves this problem would be (for example)

$$d_{ijkl} = -\left|\left| |p_i - p_j| - |q_k - q_l| \right|\right|. \tag{3.5}$$

In other words, there will be a 'zero-cost' solution to (eq. 3.4) if and only if an isometry exists between P and Q (McAuley et al., 2008a).

We use linear and quadratic assignment models like those in (eq. 3.1) and (eq. 3.4) for matching in (Caetano et al., 2009). A number of problems arise, however, when applying structured learning schemes with approximate inference, as discussed in (Finley and Joachims, 2008). For this reason we aim to model matching as inference in a graphical model: by lifting the bijectivity requirement, and including only a certain subset of pairwise constraints, we can develop models that are both tractable and expressive, as discussed below.

3.1.2 Assignment as Inference in a Graphical Model

We can already view (eq. 3.4) as inference in a graphical model, albeit an intractable one (note that we can decompose (eq. 3.2) into a series of pairwise

constraints). To make (eq. 3.4) tractable, we must introduce some *sparsity* on the set of pairwise compatibilities that are included.

To do so, we induce a graph $\mathcal{G} = (P, E)$, where $P = \{p_1 \dots p_M\}$ is the same set described in the previous section. Then in place of (eq. 3.4) we solve

$$\hat{f} = \underset{f}{\operatorname{argmax}} \sum_{(i,j)\in E} e_{ij}(f(i), f(j)) \tag{3.6}$$

(instead of enforcing the constraints of (eq. 3.2), we merely require that f is a function). Here we have used the notation of *potential functions* e, so that $c_{if(i)}$, $c_{jf(j)}$, and $d_{ijf(i)f(j)}$ are absorbed into $e_{ij}(f(i), f(j))$.

Clearly if an isometric match exists between P and Q, then (eq. 3.6) will have as its optimal solution a function $\hat{f}$ with zero cost, no matter what topology E is used. However, (eq. 3.6) can not be said to 'solve' the isometric matching problem unless the converse is true: i.e., we require that *every* zero-cost solution corresponds to an isometric match.

Naturally the *complete* graph $\mathcal{G} = K_M$ would be sufficient, though inference would be intractable: a fundamental result states that (eq. 3.6) can be solved in $O(MN^{k+1})$, where k is the *treewidth* of $\mathcal{G}$ (recall that M is the number of nodes and N is the number of states per node). A graph has treewidth k if it has maximal cliques of size $k+1$ after triangulation, so that the complete graph K_M has treewidth $M-1$ (Aji and McEliece, 2000).

As mentioned in Chapter 2, the idea of 'rigidity' is concerned with minimising the number of constraints required to enforce certain structural invariants in a network. In light of the above result, we wish instead to choose sets of constraints that induce graphs with low treewidth. The essential result of Caetano et al. (2006) is that there exist graphs that are globally rigid when embedded in the plane (and can therefore be used to solve isometric matching problems), but which also have low treewidth, leading to practical inference algorithms; we follow a similar programme in McAuley et al. (2008a); Caetano and McAuley (2009); McAuley and Caetano (2011b). We further address the idea of global rigidity below.

3.1.3 Global Rigidity

If each edge in a graphical model is thought of as enforcing a single constraint, then we would like a network with the lowest possible treewidth that includes sufficiently many constraints to solve certain types of matching problems. What

this means in practice is that if the graphical model is 'rigid' in the sense described below, then *maximum a posteriori* inference in this model shall produce the *same* solution that would be obtained if a fully connected model (which is intractable) could be used.

'Global rigidity' is defined as follows (see Connelly, 2005; Jordán and Szabadka, 2009): a straight-line embedding $\mathcal{G}'$ of the graph $\mathcal{G}$ is said to be *globally rigid* in $\mathbb{R}^n$ if the lengths of the edges in $\mathcal{G}'$ uniquely determine the lengths of the edges of the graph complement of $\mathcal{G}'$. Consequently, if $\mathcal{G}$ has a globally rigid embedding, the energy function of (eq. 3.6) has a solution with zero cost if and only if the quadratic assignment formulation of (eq. 3.4) has *the same* zero-cost solution (using the potential function of (eq. 3.5)) (McAuley et al., 2008a).

If there is no solution with zero cost, the optimal solution to (eq. 3.6) can no longer be guaranteed to match the optimal solution to (eq. 3.4). However, such models may lead to accurate solutions in practice (McAuley et al., 2008a; Caetano and McAuley, 2009; McAuley and Caetano, 2011b), and have the advantage that they can easily be embedded in a structured learning setting, as we shall see in Section 3.2.

In McAuley et al. (2008a), we present a model with treewidth three that satisfies the above property, which naïvely leads to an $\Theta(MN^4)$ solution to the isometric matching problem. However, we show that loopy belief-propagation in such a model converges to the optimal solution in $\Theta(MN^3)$ *per iteration*, extending previous convergence results of Weiss (2000). Later in Caetano and McAuley (2009) we show that the iterative requirement can be dropped, leading to a strongly cubic solution. In McAuley and Caetano (2010a); Felzenszwalb and McAuley (2011); McAuley and Caetano (2011b) we develop solutions that are fast in the expected case, up to $O(MN^2 \log N)$ for certain input distributions.

3.2 Structured Learning for Vision

In addition to solving matching problems under isometries as in (eq. 3.5), we would also like our matches to preserve various other properties of nodes or sets of nodes in P and Q. Examples include the local appearance of points in P and Q, derived from (for example) SIFT (Lowe, 1999) and Shape Context features (Belongie et al., 2002), or other features as discussed in Chapter 2. Other than isometries, second-order features may include properties based on image gradients (Kass et al., 1987), or some topological information derived from $\mathcal{G}$ (McAuley

et al., 2010b). The task of 'trading-off' the importance of these different features, so as to capture the type of variation present in a particular dataset is precisely the goal of structured learning algorithms (Caetano et al., 2009).

We assume that our optimisation problem consists of a linear parameterisation of a feature vector $\Phi(f; P, Q)$ and a parameter vector θ, so that our goal is to choose f that yields the highest score $\langle \Phi(f; P, Q), \theta \rangle$, i.e.,

$$\hat{f} = \underset{f}{\operatorname{argmax}} \langle \Phi(f; P, Q), \theta \rangle . \tag{3.7}$$

Each entry in $\Phi(f; P, Q)$ is some measure of the compatibility between P and Q under the match f. Since we hope for (eq. 3.7) to be tractable, we assume that the compatibility decomposes over the edges E as in (eq. 3.6), i.e.,

$$\Phi(f; P, Q) = \sum_{(i,j) \in E} \langle \Phi_{i,j}(f(i), f(j)), \theta \rangle , \tag{3.8}$$

so that it can be solved via inference in a graphical model (McAuley et al., 2008b).

Given a training sample $\mathcal{P} = \{P^1 \dots P^N\}$, $\mathcal{Q} = \{Q^1 \dots Q^N\}$, and a set of labelled correspondences $f^n : \{1 \dots |P^n|\} \to \{1 \dots |Q^n|\}$, our goal is to choose θ so that the matches provided by the solution to (eq. 3.7) best match the solutions f^n provided by the user. To do so, we must first define a *loss function* $\Delta(f, f^n)$, which gives some measure of the error induced by choosing the function f when the correct solution is f^n. Possibly the most straightforward example is the so-called Hamming loss, which counts the number of assignments where f differs from f^n, i.e.,

$$\Delta^{\text{Hamming}}(f, f^n) = \frac{1}{|f|} \sum_{i \in \text{dom}(f)} \delta(f(i) \neq f^n(i)), \tag{3.9}$$

where δ is an indicator function returning 1 if its argument is true, and 0 otherwise. Given that elements in Q^n typically represent point coordinates, another natural error measure is one based on the distance between the predicted matches and the correct matches, such as the so-called endpoint error

$$\Delta^{\text{endpoint}}(f, f^n) = \frac{1}{|f|} \sum_{i \in \text{dom}(f)} \left\| Q^n_{f(i)} - Q^n_{f^n(i)} \right\| \tag{3.10}$$

(see McAuley et al., 2008a).

Having chosen an appropriate loss function, the goal of the learning algorithm is to minimise the *regularised empirical risk*, defined as

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \left[\underbrace{\frac{1}{N} \sum_{n=1}^{N} \Delta(f, f^n)}_{\text{empirical risk}} + \underbrace{\lambda \Omega(\theta)}_{\text{regulariser}} \right], \tag{3.11}$$

where $\Omega(\theta)$ is some convex function penalising complex θ, and λ is a hyperparameter trading-off the importance of the two terms.

The optimisation problem of (eq. 3.11) is non-convex. More critically, the loss is a piecewise constant function of θ: there are countably many values for the loss but uncountably many values for the parameters, so there are large equivalence classes of parameters that correspond to precisely the same loss. A similar problem occurs when one aims to optimise a 0/1 loss in binary classification; in that case, a typical workaround consists of minimising a surrogate convex loss function that upper-bounds the 0/1 loss, such as the hinge loss, which gives rise to support vector machines. An analogous approach, popularised in Taskar et al. (2004); Tsochantaridis et al. (2005), optimises a convex upper bound on the structured loss of (eq. 3.11). The resulting optimisation problem is

$$[\theta^*, \xi^*] = \underset{\theta,\xi}{\operatorname{argmin}} \left[\frac{1}{N} \sum_{n=1}^{N} \xi_n + \lambda \Omega(\theta) \right] \tag{3.12a}$$

$$\text{s.t. } \langle \Phi(f^n; P^n, Q^n), \theta \rangle - \langle \Phi(f; P^n, Q^n), \theta \rangle \geq \Delta(f, f^n) - \xi_n,\ \xi_n \geq 0 \tag{3.12b}$$

$$\forall n \in \{1 \dots N\},\ f : \{1 \dots |P^n|\} \to \{1 \dots |Q^n|\}$$

The constraints (eq. 3.12b) basically enforce a loss-sensitive margin: θ is learned so that mispredictions f that incur some loss end up with a score $\langle \Phi(f; P^n, Q^n), \theta \rangle$ that is smaller than the score $\langle \Phi(f^n; P^n, Q^n), \theta \rangle$ of the correct prediction f^n by a margin equal to that loss (minus the slack ξ_n). The formulation is a generalisation of support vector machines for the multi-class case (Tsochantaridis et al., 2005; Lacoste-Julien et al., 2006).

There are two options for solving the convex relaxation of (eq. 3.12). One is to explicitly include all $N \times |P|^{|Q|}$ constraints and then solve the resulting quadratic program using one of several existing methods. This may not be feasible if $N \times |P|^{|Q|}$ is too large. In this case, we can use a constraint generation strategy. This consists of iteratively solving the quadratic program by adding at each iteration the constraint corresponding to the most violated f for the current model θ and training instance n. This is done by maximising the violation gap ξ_n, i.e., solving at each iteration the problem

$$\hat{f} = \underset{f}{\operatorname{argmax}} \{ \Delta(f, y^n) + \langle \Phi(f; P^n, Q^n), \theta \rangle \} \tag{3.13}$$

(see Tsochantaridis et al., 2005). Due to the decomposability of the loss functions chosen in (eq. 3.9) and (eq. 3.10), this again takes the same general form as (eq. 3.6), and is therefore tractable for the matching problems that we consider.

We use such models to learn the parameters of matching algorithms in Caetano et al. (2009); McAuley et al. (2008a, 2010b), and we embed such models in a classification system in Chen et al. (2009). We also apply structured learning algorithms to problems in vision (other than matching) in McAuley et al. (2009, 2011).

3.3 Inference Algorithms

As mentioned, inference in a graphical model can be accomplished by message-passing operations following a simple protocol making use of the distributive law (Pearl, 1988; Aji and McEliece, 2000; Kschischang et al., 2001). The fundamental step encountered in message-passing algorithms is defined below. The message from a clique X to an intersecting clique Y (assumed to be maximal cliques in a triangulated graph) is defined as

$$m_{X\to Y}(\mathbf{x}_{X\cap Y}) \propto \sum_{\mathbf{x}_{X\setminus Y}} \left\{ \Phi_X(\mathbf{x}_X) \prod_{Z\in\Gamma(X)\setminus Y} m_{Z\to X}(\mathbf{x}_{X\cap Z}) \right\}, \tag{3.14}$$

where $\Gamma(X)$ is the set of neighbours of the clique X, i.e., the set of maximal cliques that intersect with X. If such messages are computed after X has received messages from all of its neighbours except Y (i.e., $\Gamma(X) \setminus Y$), then this defines precisely the update scheme used by the junction-tree algorithm. The same update scheme is used for loopy belief-propagation, though it is done iteratively in a randomised fashion.

Once a clique X has received messages from all of its neighbours, the 'belief' at that clique is defined as

$$b_X(\mathbf{x}_X) \propto \Phi_X(\mathbf{x}_X) \prod_{Z\in\Gamma(X)} m_{Z\to X}(\mathbf{x}_{X\cap Z}). \tag{3.15}$$

The message-passing scheme of (eq. 3.14) and (eq. 3.15) is known as *sum-product* belief-propagation; it can be shown that under this scheme (subject to appropriate normalisation) the belief at clique X is precisely the *marginal distribution* of $\mathbf{x}_X$ (Aji and McEliece, 2000).

For the setting described in Section 3.2, we are not interested in the marginal probability of $\mathbf{x}_X$, but rather the most likely joint configuration of the variables. This is accomplished by *max-product* belief-propagation, where summation in (eq. 3.14) is replaced by maximisation (taking logarithms yields a *max-sum* formulation as in (eq. 3.6)).

Note that X in (eq. 3.14) is a *maximal clique* in a *triangulated* graph. Given our statements about rigidity mentioned above and in Chapter 2, the maximal cliques must contain at least three nodes (i.e., the treewidth must be at least two) in models for rigid matching, even though the individual factors are pairwise. The simplest possible factorisation assuming that $X = \{i, j, k\}$ is simply

$$\Phi_{i,j,k} = \underbrace{\Phi_{i,j}(x_i, x_j)}_{A} \times \underbrace{\Phi_{i,k}(x_i, x_k)}_{B} \times \underbrace{\Phi_{j,k}(x_j, x_k)}_{C}. \tag{3.16}$$

Treating each of the three factors in (eq. 3.16) as *matrices*, we can see that the marginal $\sum_i \Phi_{i,j,k}$ is just $A^T B \odot C$, where $\odot$ is the Hadamard (elementwise) product, so that the complexity of sum-product message-passing as in (eq. 3.14) is precisely related to the complexity of matrix multiplication. In the case of max-product message-passing, $\max_i \Phi_{i,j,k}$ is just $(A^T \otimes B) \odot C$, where $\otimes$ is matrix multiplication in the max-product semiring. As mentioned in Chapter 2, this operation (also known as 'tropical' matrix multiplication) is closely related to solutions to the all-pairs shortest-path problem (Aho et al., 1983; Williams and Williams, 2010).

We made the observation that graph matching via inference in a graphical model was related to tropical matrix multiplication in McAuley and Caetano (2010a). We showed that a more general class of inference problems can be related to tropical matrix-vector multiplication in McAuley and Caetano (2010b). In McAuley and Caetano (2011a) we extended these ideas and applied them to a number of other problems outside of graph matching. Felzenszwalb and McAuley (2011) further improved upon the results therein.

Chapter 4

Core Publications

4.1 Graphical Models for Vision and Matching

As in the previous chapter we discuss the problem of identifying an isometrically transformed instance of a pattern P within a scene Q. Again we assume that $|P| = M$ and $|Q| = N$.

In **McAuley et al. (2008a)**,[1] we developed a graphical model for isometric point-pattern matching that was asymptotically faster than existing models that made the same guarantees. As mentioned in Chapter 2, we worked on models that are *provably optimal* in the noise-free case, but which lead to empirically good performance in the case of noise. The starting point for our model was that of Caetano et al. (2006), as shown in Figure 4.1 (left); the model we developed is shown in Figure 4.1 (right).

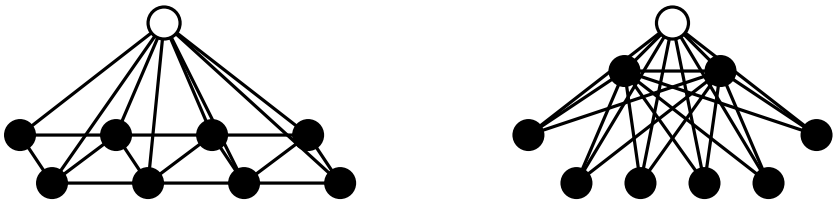
To show that the model from Caetano et al. (2006) is globally rigid, it helps to make an analogy with a GPS network: if the top three nodes in Figure 4.1 (left) represent satellites (whose positions are assumed to be known), and the bottom nodes represent users of a GPS device, then it is sufficient that they know their *distance* from each satellite in order to determine their location (assuming that they lie on a plane). In other words, the lengths of the edges in (a straight-line embedding of) the graph uniquely determine the lengths of the edges in its complement; see Caetano et al. (2006, Lemma 2) for a proof. Although we have no such analogy for the model at the right of Figure 4.1, a proof of its rigidity when embedded in the plane is given in McAuley et al. (2008a, Proposition 5).

Both models have treewidth 3 (i.e., they have maximal cliques of size four after triangulation), leading naïvely to $\Theta(MN^4)$ inference in a model with M

[1]The first reference to each core publication is marked in bold.

model from Caetano et al. (2006) model from McAuley et al. (2008a)

Figure 4.1: Graphical models for isometric matching from Caetano et al. (2006, left) and McAuley et al. (2008a, right).



model from Caetano and McAuley (2009) model from McAuley and Caetano (2011b)

Figure 4.2: Graphical models for isometric matching from Caetano and McAuley (2009, left) and McAuley and Caetano (2011b, right).

nodes and N states per node (corresponding to a pointset of size M within a pointset of size N). However, we propose a loopy belief-propagation scheme in which messages are passed between maximal cliques in the original model *only if they share an edge*; in this way, the graph of cliques that communicate with each other (by passing messages) forms a single loop, and we adapt existing results by Weiss (2000) to show that max-product message-passing will converge to yield the correct *maximum a posteriori* solution under weak assumptions. This correctness result is given in McAuley et al. (2008a, Theorem 8).

We further developed graphical models for isometric point-pattern matching in Caetano and McAuley (2009) and **McAuley and Caetano (2011b)**. The models we developed are shown in Figure 4.2. Note that both models again have treewidth 3. However, the white nodes in each model are binary variables encoding whether or not the pattern P appears reflected in the scene Q, meaning that the number of states in any maximal clique is $2 \cdot N^3$, leading to an $\Theta(MN^3)$ solution in Caetano and McAuley (2009), without requiring iterative belief-propagation as in McAuley et al. (2008a).

The model of McAuley and Caetano (2011b) uses the same potential func-

tions and has the same treewidth as that of Caetano and McAuley (2009), which again naïvely leads to $\Theta(MN^3)$ inference. However, the advantage of the graph topology of McAuley and Caetano (2011b) as shown in Figure 4.2 (right) is that by conditioning on the top three nodes in the graphical model, the MAP solution can be efficiently found using fast search techniques, such as nearest-neighbour queries in a *KD-tree* (Bentley, 1975). In practice this leads to an $O(MN^2 \log N)$ solution to matching (under weak assumptions), and has a lower memory footprint than belief-propagation techniques.

In McAuley and Caetano (2011b) we also discuss how *occlusions* can be handled by models for isometric matching. We propose a model that is able to identify *subsets* of points in P that appear isometrically transformed in Q, a property not satisfied by the previous models. In other words the model is able to identify the *largest common isometric subgraph* shared by P and Q.

Each of the above models have their own advantages and disadvantages depending on the application of interest: the model of McAuley and Caetano (2011b) is the fastest, and can handle occlusions, but it makes assumptions that prevent certain types of image features being used in the model. The models of McAuley et al. (2008a); Caetano and McAuley (2009) are advantageous because they can easily be incorporated into structured learning schemes, especially the latter as it does not rely on the convergence of loopy belief-propagation. The model of McAuley et al. (2008a) proves to be useful once we discuss inference algorithms in Section 4.3, since it includes only pairwise factors, which allows us to exploit relationships to matrix multiplication.

4.2 Structured Learning for Vision

Although structured learning had been applied to matching problems in Caetano et al. (2007, 2009), if the objects being matched are *shapes*, or they are composed of rigid parts, the isometry assumption made above proves to be a natural one. Therefore we considered the problem of applying learning to models for isometric (or near-isometric) matching in **McAuley et al. (2008b)** and **McAuley et al. (2010b)**.

The model of McAuley et al. (2008a) presented above is particularly appealing for shape matching: since it forms a ring it can be embedded on the *boundary* of an object. An example is shown in Figure 4.3. As mentioned in Chapter 3, such a model is no longer guaranteed to correctly solve matching problems once noise

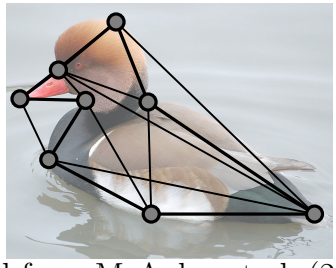


Figure 4.3: The model from McAuley et al. (2008a), embedded on the boundary of an object, which we used in McAuley et al. (2008b).

is introduced, though intuitively the features along the boundary of an object are likely to be the most important. We used features encoding distance and shape information, and found that such features were amongst the most influential in terms of obtaining accurate matching results (McAuley et al., 2008a).

In McAuley et al. (2010b) we applied structured learning in a model similar to that of Figure 4.2 (left), which benefited from the fact that iterative belief-propagation schemes were not required to perform inference. We also introduced *topological* information into the model, based on graph topologies induced from the pointsets P and Q (such as the Delaunay triangulation). Furthermore we showed that topological information can be preserved *exactly*, under the assumption that isometries are also preserved (in other words we show that the problem of subgraph isomorphism can be solved efficiently, subject to the additional constraint that the isomorphism induces an isometry (see McAuley et al., 2010b, Section 2.2)).

In Chen et al. (2009) we embedded models for matching within image classification schemes, where the score of a match between two images is used as a proxy for whether those images contain objects of the same class. Rather than training our algorithm to solve matching tasks, and then using the matching score for classification directly (as had already been attempted using existing models), our model trains the matching scores *so as to produce the most discriminative classifier*. By comparison, a model that is trained to match objects of the same class may 'accidentally' produce good matches for objects of different classes, and will therefore produce scores that are not discriminative.

We also applied structured learning to problems from computer vision in McAuley et al. (2009) and McAuley et al. (2011). In McAuley et al. (2009) we proposed a hierarchical model for image categorisation, so that labels at the

lowest level (i.e., the patch or pixel level) produce a segmentation of the image, while labels at the highest level (i.e., the image level) simply classify the image. The motivation for such a model is that different object classes are easy to classify at different scales, and therefore the scales at which the classification problem is 'easy' should influence the scales at which it is hard.

In McAuley et al. (2011) we applied structured learning to multiclass classification problems in large-scale, weakly-annotated datasets. As it is infeasible to obtain complete labellings of web-scale datasets (e.g. annotating every object that appears in millions of images), structured learning methods in this setting must be able to exploit weakly labelled data (Deng et al., 2009, 2010). We proposed a model where the full labelling is treated as a latent completion of the observed weak label; once the full labelling is obtained, structured learning can proceed as normal. This is similar to the programme for structured learning with latent variables described in (Yu and Joachims, 2009).

4.3 Inference Algorithms

We mentioned in Chapter 3 that message-passing algorithms in certain graphical models reduce to matrix-matrix and matrix-vector multiplication. Matrix-matrix multiplication arises in treewidth 2 graphical models with pairwise factors, in which case message updates factorise according to

$$m_{\{i,j,k\}\to\{j,k,l\}}(x_j, x_k) \propto \underbrace{\Psi_{j,k}(x_j, x_k)}_{A} \times \sum_i \underbrace{\Psi_{i,j}(x_i, x_j)}_{B} \times \underbrace{\Psi_{i,k}(x_i, x_k)}_{C} \quad (4.1)$$

$$\propto A \odot B^T C \quad (4.2)$$

($\Psi_{i,j}$ is assumed to incorporate any local potentials and messages over x_i and x_j, as in (eq. 3.14)). In fact, in *any* graphical model with pairwise factors other than a tree, message-passing can be seen as matrix-matrix multiplication by appropriately merging variables (see McAuley and Caetano, 2011a, Section 5).

The factorisation of (eq. 4.1) is precisely what we observed in McAuley et al. (2008a), as shown in Figure 4.1 (right): pairwise factors within maximal cliques of size three. As we noted in Section 2.3, this type of factorisation is particularly common in graphical models for computer vision, where high-treewidth models decompose into pairwise terms. Note that if each variable has N states (i.e., there are N interest points in the scene Q), then (eq. 4.1) is naïvely an $\Theta(N^3)$ operation. Since automatic interest point detectors typically generate scenes with

hundreds or even thousands of points, developing sub-cubic solutions to (eq. 4.1) is naturally a problem of interest.

In principle one could apply sub-cubic algorithms for matrix multiplication like those of (Strassen, 1969; Coppersmith and Winograd, 1990; Cohn et al., 2005) to (eq. 4.1), though this is unlikely to be beneficial for realistically sized state-spaces. Instead, in **McAuley and Caetano (2010a)** we studied the *max-product* variant of matrix multiplication (or equivalently 'tropical' matrix multiplication), meaning that summation in (eq. 4.1) is replaced by maximisation, so that the primitive operation we address becomes

$$A_{j,k} \times \max_i \left\{ B^T_{j,i} \times C_{i,k} \right\} \tag{4.3}$$

(one can equivalently replace multiplication by addition, or maximisation by minimisation); this is precisely the operation that arises when performing MAP inference. Although finding a truly sub-cubic algorithm for tropical matrix multiplication remains an open problem (Williams and Williams, 2010), we investigate the *expected-case* performance under certain input distributions.

The core contribution of McAuley and Caetano (2010a) is an expected-case $O(\sqrt{N})$ procedure for the *max-product inner-product* operation, i.e.,

$$max_i \left\{ a_i \times b_i \right\} \tag{4.4}$$

(see McAuley and Caetano, 2011a, Algorithm 2). The proposed algorithm assumes that the rank statistics of a and b in (eq. 4.4) are known, and exhibits $O(\sqrt{N})$ performance when their rank statistics of a are *independent* of those of b (see McAuley and Caetano, 2011a, Theorem 2). Better or worse performance may be obtained (between $\Theta(1)$ and $\Theta(N)$) depending on the dependence structure of the rank statistics of a and b (see McAuley and Caetano, 2011a, Section 7.2). Importantly, the factorisation of (eq. 4.1) allows us to compute all of the required rank statistics in $O(N^2 \log N)$ (computing the rank statistics for each row/column of a matrix takes $O(N \log N)$, and there are N rows/columns), meaning that the N^2 'inner-products' required by (eq. 4.1) can be computed in $O(N^2\sqrt{N})$ using the proposed algorithm (see McAuley and Caetano, 2011a, Algorithm 4).

In **McAuley and Caetano (2010b)** we studied message-passing schemes that factorise according to

$$m_{\{i,j\} \rightarrow \{j,k\}}(x_j) \propto \underbrace{\Psi_j(x_j|y)}_{a} \times \sum_i \underbrace{\Psi_{i,j}(x_i, x_j)}_{B} \times \underbrace{\Psi_i(x_i|y)}_{c} \tag{4.5}$$

$$\propto a \odot B^T c, \tag{4.6}$$

where y is assumed to be some problem-dependent *observation*, so that $\Psi_{i,j}(x_i, x_j)$ in (eq. 4.5) is a problem-*independent* prior. This can be viewed as an instance of matrix-vector multiplication, where the matrix can be pre-processed *offline*, since it is independent of the observation. This is precisely the type of problem studied in Williams (2007), where it is shown that matrix-vector multiplication has a sub-*quadratic* solution under the same conditions.

Again, we studied (eq. 4.5) in a max-product setting, where 'pre-processing' the matrix B amounts to computing its rank statistics, yielding an $O(N\sqrt{N})$ solution to (eq. 4.5) (see McAuley and Caetano, 2011a, Algorithm 3). This is precisely the type of factorisation that appears in a number of pairwise models from vision, where an energy minimisation problem is defined in terms of node and edge energies, but the edge energies merely encode priors.

McAuley and Caetano (2011a) is an extension of the work presented in McAuley and Caetano (2010a,b). A number of applications of tropical matrix multiplication are considered, including the all-pairs shortest-paths problem, L^∞ distance computation, and linear programming relaxations (Sontag et al., 2008). In Felzenszwalb and McAuley (2011) we further consider the problem of tropical matrix multiplication, developing an algorithm that solves (eq. 4.1) in expected-case $O(N^2 \log N)$, albeit under stronger conditions than the algorithm of McAuley and Caetano (2010a); specifically, it assumes that the *entries* in the matrices are independent samples from a uniform distribution, whereas McAuley and Caetano (2010a) only assume that the rank statistics are independent. In spite of the stronger assumptions, the method's performance appears to match its expected-case analysis in a variety of applications.

Bibliography

Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *Data Structures and Algorithms*. Addison-Wesley, 1983. (Cited on pages 11 and 20.)

Srinivas M. Aji and Robert J. McEliece. The generalized distributive law. *IEEE Transactions on Information Theory*, 46(2):325–343, 2000. (Cited on pages 9, 10, 11, 15, and 19.)

Tatsuya Akutsu, Kyotetsu Kanaya, Akira Ohyama, and Asao Fujiyama. Point matching under non-uniform distortions. *Discrete Applied Mathematics*, 127 (1):5–21, 2003. (Cited on page 5.)

Noga Alon, Zvi Galil, and Oded Margalit. On the exponent of the all pairs shortest path problem. *Journal of Computer and System Sciences*, 54(2):255–262, 1997. (Cited on page 12.)

Helmut Alt and Leonidas Guibas. Discrete geometric shapes: Matching, interpolation, and approximation. In *Handbook of Computational Geometry*, pages 121–153. Elsevier, 1999. (Cited on pages 7 and 14.)

Helmut Alt, Kurt Mehlhorn, Hubert Wagener, and Emo Welzl. Congruence, similarity and symmetries of geometric objects. *Discrete Computational Geometry*, 3(3):237–256, 1988. (Cited on page 7.)

Amir A. Amini, Terry E. Weymouth, and Ramesh Jain. Using dynamic programming for solving variational problems in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(9):855–867, 1990. (Cited on pages 10 and 11.)

Yali Amit and Augustine Kong. Graphical templates for model registration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(3):225–236, 1996. (Cited on page 6.)

Kurt M. Anstreicher. Recent advances in the solution of quadratic assignment problems. *Mathematical Programming*, 97:27–42, 2003. (Cited on pages 6 and 14.)

Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded Up Robust Features. In *European Conference on Computer Vision*, 2006. (Cited on page 7.)

Serge Belongie and Jitendra Malik. Matching with shape contexts. In *IEEE Workshop on Contentbased Access of Image and Video Libraries*, pages 20–26, 2000. (Cited on pages 6 and 7.)

Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(24):509–521, 2002. (Cited on pages 5, 6, 7, and 16.)

Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975. (Cited on page 23.)

Alexander C. Berg, Tamara L. Berg, and Jitendra Malik. Shape matching and object recognition using low distortion correspondence. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005. (Cited on page 6.)

Umberto Bertele and Francesco Brioschi. *Nonserial Dynamic Programming*. Academic Press, 1972. (Cited on page 10.)

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, August 2006. (Cited on page 10.)

Matthew B. Blaschko and Christoph H. Lampert. Learning to localize objects with structured output regression. In *European Conference on Computer Vision*, 2008. (Cited on page 8.)

Yuri Boykov and Marie-Pierre Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *International Conference on Computer Vision*, 2001. (Cited on page 9.)

Rainer Burkard, Mauro Dell'Amico, and Silvano Martello. *Assignment Problems*. Society for Industrial and Applied Mathematics, 2009. (Cited on pages 6 and 13.)

Terry Caelli and Tibério S. Caetano. Graphical models for graph matching: Approximate models and optimal algorithms. *Pattern Recognition Letters*, 26 (3):339–346, 2005. (Cited on page 7.)

Terry Caelli and Serhiy Kosinov. An eigenspace projection clustering method for inexact graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(4):515–519, 2004. (Cited on page 6.)

Tibério S. Caetano and Terry Caelli. Approximating the problem, not the solution: an alternative view of point set matching. *Pattern Recognition*, 39: 552–561, 2006a. (Cited on page 8.)

Tibério S. Caetano and Terry Caelli. A unified formulation of invariant point pattern matching. In *International Conference on Pattern Recognition*, pages 121–124, 2006b. (Cited on pages 7 and 8.)

Tibério S. Caetano and Julian J. McAuley. Faster graphical models for point-pattern matching. *Spatial Vision*, 22(5):443–453, 2009. (Cited on pages 3, 4, 8, 15, 16, 22, and 23.)

Tibério S. Caetano, Terry Caelli, and Dante A. C. Barone. Graphical models for graph matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2004a. (Cited on page 6.)

Tibério S. Caetano, Terry Caelli, and Dante A. C. Barone. An optimal probabilistic graphical model for point set matching. In *SSPR & SPR*, pages 162–170, Lisbon, 2004b. (Cited on page 8.)

Tibério S. Caetano, Terry Caelli, Dale Schuurmans, and Dante A. C. Barone. Graphical models and point pattern matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1646–1663, 2006. (Cited on pages 7, 8, 9, 15, 21, and 22.)

Tibério S. Caetano, Li Cheng, Quoc V. Le, and Alex J. Smola. Learning graph matching. In *International Conference on Computer Vision*, 2007. (Cited on pages 9 and 23.)

Tibério S. Caetano, Julian J. McAuley, Li Cheng, Quoc V. Le, and Alex J. Smola. Learning graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(6):1048–1058, 2009. (Cited on pages 3, 4, 7, 9, 14, 17, 19, and 23.)

John F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986. (Cited on page 7.)

Marco Carcassoni and Edwin R. Hancock. Point pattern matching with robust spectral correspondence. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2000. (Cited on page 6.)

Marco Carcassoni and Edwin R. Hancock. Spectral correspondence for point pattern matching. *Pattern Recognition*, 36:193–204, 2003. (Cited on page 6.)

Owen Carmichael and Martial Hebert. Shape-based recognition of wiry objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26 (12):1537–1552, 2004. (Cited on page 6.)

Timothy M. Chan. More algorithms for all-pairs shortest paths in weighted graphs. In *ACM Symposium on Theory of Computing*, pages 590–598, 2007. (Cited on page 12.)

Longbin Chen, Julian J. McAuley, Rogerio S. Feris, Tibério S. Caetano, and Matthew Turk. Shape classification through structured learning of matching measures. *IEEE Conference on Computer Vision and Pattern Recognition*, 2009. (Cited on pages 3, 4, 6, 9, 19, and 24.)

William J. Christmas, Josef Kittler, and Maria Petrou. Structural matching in computer vision using probabilistic relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):749–764, 1994. (Cited on page 6.)

Haili Chui and Anand Rangarajan. A new algorithm for non-rigid point matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2000. (Cited on page 5.)

Henry Cohn, Robert Kleinberg, Balazs Szegedy, and Chris Umans. Group-theoretic algorithms for matrix multiplication. In *Symposium on Foundations of Computer Science*, 2005. (Cited on pages 11 and 26.)

Robert Connelly. Generic global rigidity. *Discrete Computational Geometry*, 33 (4):549–563, 2005. (Cited on pages 7, 11, and 16.)

Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, 1990. (Cited on pages 11 and 26.)

James M. Coughlan and Sabino J. Ferreira. Finding deformable shapes using loopy belief propagation. In *European Conference on Computer Vision*, 2002. (Cited on pages 7 and 10.)

Timothee Cour, Praveen Srinivasan, and Jianbo Shi. Balanced graph matching. In *Advances in Neural Information Processing Systems*, 2006. (Cited on page 6.)

Robert G. Cowell, Philip A. Dawid, Steffen L. Lauritzen, and David J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer, 2003. (Cited on page 10.)

Pedro Jussieu de Rezende and Der-Tsai Lee. Point set pattern matching in d-dimensions. *Algorithmica*, 13:387–404, 1995. (Cited on pages 7 and 14.)

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009. (Cited on page 25.)

Jia Deng, Alexander C. Berg, Kai Li, and Li Fei-Fei. What does classifying more than 10,000 image categories tell us? In *European Conference on Computer Vision*, 2010. (Cited on page 25.)

Rachid Deriche. Using Canny's criteria to derive a recursively implemented optimal edge detector. *International Journal of Computer Vision*, 1(2):167–187, 1987. (Cited on page 7.)

René Donner, Georg Langs, and Horst Bischof. Sparse MRF appearance models for fast anatomical structure localisation. In *British Machine Vision Conference*, 2007. (Cited on page 10.)

Olivier Duchenne, Francis Bach, Inso Kweon, and Jean Ponce. A tensor-based algorithm for high-order graph matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009. (Cited on pages 7 and 14.)

David I. Edwards. *Introduction to Graphical Modelling*. Springer, 2000. (Cited on page 10.)

Gal Elidan, Ian McGraw, and Daphne Koller. Residual belief propagation: informed scheduling for asynchronous message passing. In *Uncertainty in Artificial Intelligence*, 2006. (Cited on page 10.)

Pedro F. Felzenszwalb. Representation and detection of deformable shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(2):208–220, 2005. (Cited on pages 6, 7, and 10.)

Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient belief propagation for early vision. *International Journal of Computer Vision*, 70(1):41–54, 2006. (Cited on pages 9 and 10.)

Pedro F. Felzenszwalb and Julian J. McAuley. Fast inference with min-sum matrix product. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011. to appear. (Cited on pages 3, 10, 12, 16, 20, and 27.)

Pedro F. Felzenszwalb and Joshua D. Schwartz. Hierarchical matching of deformable shapes. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007. (Cited on page 5.)

Thomas Finley and Thorsten Joachims. Training structural svms when exact inference is intractable. In *International Conference on Machine Learning*, 2008. (Cited on pages 9 and 14.)

Paul W. Finn, Lydia E. Kavraki, Jean-Claude Latombe, Rajeev Motwani, Christian R. Shelton, Suresh Venkatasubramanian, and Andrew Yao. Rapid: Randomized pharmacophore identification for drug design. *Computational Geometry*, 10:263–272, 1998. (Cited on page 5.)

Martin A. Fischler and Robert A. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computers*, 22(1):67–92, 1973. (Cited on page 7.)

Andrew Fitzgibbon. Robust registration of 2D and 3D point sets. In *British Machine Vision Conference*, 2001. (Cited on page 5.)

Alan M. Frieze. The shortest-path problem for graphs with random arc-lengths. *Discrete Applied Mathematics*, 10(1):57–77, 1985. (Cited on page 12.)

Andrea Frome, Daniel Huber, Ravi Kolluri, Thomas Bulow, and Jitendra Malik. Recognizing objects in range data using regional point descriptors. In *European Conference on Computer Vision*, 2004. (Cited on page 5.)

Andrea Frome, Yoram Singer, Fei Sha, and Jitendra Malik. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *International Conference on Computer Vision*, 2007. (Cited on page 8.)

Michel Galley. A skip-chain conditional random field for ranking meeting utterances by importance. In *Empirical Methods in Natural Language Processing*, 2006. (Cited on page 10.)

Michael R. Garey and David S. Johnson. *Computers and Intractability*. W. H. Freeman and Co., 1979. (Cited on pages 6 and 14.)

Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distribution and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984. (Cited on page 9.)

Steven Gold and Anand Rangarajan. A graduated assignment algorithm for graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(4):377–388, 1996. (Cited on pages 6, 7, and 14.)

Michael T. Goodrich, Joseph S. B. Mitchell, and Mark W. Orletsky. Approximate geometric pattern matching under rigid motions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(4):371–379, 1999. (Cited on page 7.)

Edwin R. Hancock and Richard C. Wilson. Graph-based methods for vision: A Yorkist manifesto. *SSPR & SPR*, 2002. (Cited on page 6.)

Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, 1988. (Cited on page 7.)

Richard I. Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004. (Cited on page 5.)

Frank L. Hitchcock. The distribution of a product from several sources to numerous localities. *Journal of Mathematical Psychology*, 20:224–230, 1941. (Cited on pages 6 and 13.)

Bert Huang and Tony Jebara. Fast b-matching via sufficient selection belief propagation. In *Artificial Intelligence and Statistics*, 2011. (Cited on pages 6, 12, and 13.)

Hiroshi Ishikawa. Exact optimization for markov random fields with convex priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10): 1333–1336, 2003. (Cited on page 10.)

Finn V. Jensen. *Bayesian Networks and Decision Graphs*. Springer, 2001. (Cited on page 10.)

Roy Jonker and Ton A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38:325–340, 1987. (Cited on page 6.)

Michael Jordan, editor. *Learning in Graphical Models*. MIT Press, 1998. (Cited on page 10.)

Tibor Jordán and Zoltán Szabadka. Operations preserving the global rigidity of graphs and frameworks in the plane. *Computational Geometry*, 42(6-7): 511–521, 2009. (Cited on pages 7, 11, and 16.)

David R. Karger, Daphne Koller, and Steven J. Phillips. Finding the hidden path: time bounds for all-pairs shortest paths. *SIAM Journal of Computing*, 22(6):1199–1217, 1993. (Cited on page 12.)

Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1987. (Cited on pages 7, 9, and 16.)

Leslie R. Kerr. The effect of algebraic structure on the computational complexity of matrix multiplication. *PhD Thesis*, 1970. (Cited on page 11.)

Kristian Kersting, Babak Ahmadi, and Sriraam Natarajan. Counting belief propagation. In *Uncertainty in Artificial Intelligence*, 2009. (Cited on page 10.)

Josef V. Kittler and Edwin R. Hancock. Combining evidence in probabilistic relaxation. *International Journal of Pattern Recognition and Artificial Intelligence*, 3:29–51, 1989. (Cited on page 6.)

Uffe Kjærulff. Inference in bayesian networks using nested junction trees. In *Proceedings of the NATO Advanced Study Institute on Learning in graphical models*, 1998. (Cited on page 10.)

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009. (Cited on page 10.)

Vladimir Kolmogorov and Akiyoshi Shioura. New algorithms for the dual of the convex cost network flow problem with application to computer vision. Technical report, University College London, 2007. (Cited on page 10.)

Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimized via graph cuts? In *European Conference on Computer Vision*, 2002. (Cited on page 10.)

Frank R. Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001. (Cited on pages 9, 10, and 19.)

Harold W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955. (Cited on page 6.)

M. Pawan Kumar and Philip Torr. Fast memory-efficient generalized belief propagation. In *European Conference on Computer Vision*, 2006. (Cited on page 10.)

Simon Lacoste-Julien, Ben Taskar, Dan Klein, and Michael I. Jordan. Word alignment via quadratic assignment. In *NAACL HLT*, 2006. (Cited on pages 9 and 18.)

Gerard Laman. On graphs and rigidity of plane skeletal structures. *Journal of Engineering Mathematics*, 4:331–340, 1970. (Cited on page 7.)

Xiang-Yang Lan, Stefan Roth, Daniel P. Huttenlocher, and Michael J. Black. Efficient belief propagation with learned higher-order Markov random fields. In *European Conference on Computer Vision*, 2006. (Cited on page 9.)

Andrea S. LaPaugh and Ronald L. Rivest. The subgraph homeomorphism problem. In *ACM Symposium on Theory of Computing*, 1978. (Cited on page 6.)

Steffen L. Lauritzen. *Graphical Models*. Oxford University Press, 1996. (Cited on page 10.)

Steffen L. Lauritzen and David J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Series B*, 50:157–224, 1988. (Cited on page 10.)

Yann LeCun, Fu Jie Huang, and Leon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. *IEEE Conference on Computer Vision and Pattern Recognition*, 2004. (Cited on page 6.)

Marius Leordeanu and Martial Hebert. A spectral technique for correspondence problems using pairwise constraints. In *International Conference on Computer Vision*, 2005. (Cited on page 6.)

Marius Leordeanu, Martial Hebert, and Rahul Sukthankar. Beyond local appearance: Category recognition from pairwise interactions of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007. (Cited on page 6.)

Stan Z. Li. A Markov random field model for object matching under contextual constraints. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1994. (Cited on page 6.)

Stan Z. Li. *Markov Random Field Modeling in Computer Vision*. Springer, 1995. (Cited on page 9.)

Xiang-Yang Li, Peng-Jun Wan, Yu Wang, and Chih-Wei Yi. Fault tolerance deployment and topology control in wireless networks. In *ACM Symposium on Mobile Ad Hoc Networking and Computing*, 2003. (Cited on page 8.)

Tony Lindeberg. Detecting salient blob-like image structures and their scales with a scale-space primal sketch: A method for focus-of-attention. *International Journal of Computer Vision*, 11(3):283–318, 1993. (Cited on page 7.)

Tony Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):77–116, 1998. (Cited on page 7.)

László Lovász and Yechiam Yemini. On generic rigidity in the plane. *SIAM Journal of Algebraic Discrete Methods*, 3(1):91–98, 1982. (Cited on page 7.)

David G. Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision*, 1999. (Cited on pages 6, 7, and 16.)

David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. (Cited on page 7.)

Diane Maclagan and Bernd Sturmfels. Introduction to tropical geometry. Book in preparation, 2009. (Cited on page 11.)

Yvonne C. Martin, Mark G. Bures, Elizabeth A. Danaher, Jerry DeLazzer, Isabella Lico, and Patricia A. Pavlik. A fast new approach to pharmacophore mapping and its application to dopaminergic and bezodiazepine agonists. *Journal of Computer-Aided Molecular Design*, 7(1):83–102, 1993. (Cited on page 5.)

Julian J. McAuley and Tibério S. Caetano. Exploiting within-clique factorizations in junction-tree algorithms. *Artificial Intelligence and Statistics*, 2010a. (Cited on pages 2, 4, 8, 10, 12, 16, 20, 26, and 27.)

Julian J. McAuley and Tibério S. Caetano. Exploiting data-independence for fast belief-propagation. *International Conference on Machine Learning*, 2010b. (Cited on pages 2, 4, 10, 12, 20, 26, and 27.)

Julian J. McAuley and Tibério S. Caetano. Faster algorithms for max-product message-passing. *Journal of Machine Learning Research*, 12:1349–1388, 2011a. (Cited on pages 2, 4, 10, 11, 12, 20, 25, 26, and 27.)

Julian J. McAuley and Tibério S. Caetano. Fast matching of large point sets under occlusions. *Pattern Recognition*, 2011b. to appear. (Cited on pages 1, 4, 8, 12, 15, 16, 22, and 23.)

Julian J. McAuley, Tibério S. Caetano, Alex J. Smola, and Matthias O. Franz. Learning high-order MRF priors of color images. In *International Conference on Machine Learning*, 2006. (Cited on page 9.)

Julian J. McAuley, Tibério S. Caetano, and M. S. Barbosa. Graph rigidity, cyclic belief propagation and point pattern matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):2047–2054, 2008a. (Cited on pages 1, 3, 8, 10, 14, 15, 16, 17, 19, 21, 22, 23, 24, and 25.)

Julian J. McAuley, Tibério S. Caetano, and Alex J. Smola. Robust near-isometric matching via structured learning of graphical models. *Advances in Neural Information Processing Systems*, 2008b. (Cited on pages 2, 4, 7, 8, 9, 17, 23, and 24.)

Julian J. McAuley, T. de Campos, G. Csurka, and F. Perronnin. Hierarchical image-region labeling via structured learning. In *British Machine Vision Conference*, 2009. (Cited on pages 2, 9, 19, and 24.)

Julian J. McAuley, Tibério S. Caetano, and Wray Buntine. Graphical models. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning*, volume 1, pages 471–479. Springer, 2010a. (Cited on page 10.)

Julian J. McAuley, Teofilo de Campos, and Tibério S. Caetano. Unified graph matching in euclidean spaces. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010b. (Cited on pages 2, 4, 8, 9, 16, 19, 23, and 24.)

Julian J. McAuley, A. Ramisa, and Tibério S. Caetano. Optimization of robust loss functions for weakly-labeled image taxonomies: An imagenet case study. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, 2011. (Cited on pages 3, 9, 19, 24, and 25.)

Bruno T. Messmer and Horst Bunke. A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):493–503, 1998. (Cited on page 7.)

Grigory Mikhalkin. Tropical geometry. Book in preparation, 2008. (Cited on page 11.)

Krystian Mikolajczyk and Cordelia Schmid. Scale and affine invariant interest point detectors. *IJCV*, 60(1):63–86, October 2004. (Cited on pages 6 and 7.)

Krystian Mikolajczyk, Bastian Leibe, and Bernt Schiele. Multiple object class detection with a generative model. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006. (Cited on page 6.)

Alistair Moffat and Tadao Takaoka. An all pairs shortest path algorithm with expected time $O(n^2 \log n)$. *SIAM Journal of Computing*, 16(6):1023–1031, 1987. (Cited on page 12.)

Greg Mori and Jitendra Malik. Estimating human body configurations using shape context matching. In *European Conference on Computer Vision*, 2002. (Cited on page 5.)

Greg Mori, Serge Belongie, and Jitendra Malik. Shape contexts enable efficient retrieval of similar shapes. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2001. (Cited on page 5.)

Greg Mori, Serge Belongie, and Jitendra Malik. Efficient shape matching using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(11):1832–1837, 2005. (Cited on page 5.)

Jarnes Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1), 1957. (Cited on pages 6 and 13.)

Fionn Murtaugh. A new approach to point-pattern matching. *Astronomical Society of the Pacific*, 104(674):301–307, 1992. (Cited on page 5.)

National Park Service. Trusses: A study by the historical American engineering record, 1976. (Cited on page 10.)

Ruth Nussinov and Haim J. Wolfson. Efficient detection of three-dimensional structural motifs in biological macromolecules by computer vision techniques. *Proceedings of the National Academy of Sciences*, 88:10495–10499, 1991. (Cited on page 5.)

Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, 1982. (Cited on page 6.)

James D. Park and Adnan Darwiche. A differential semantics for jointree algorithms. In *Advances in Neural Information Processing Systems*, 2003. (Cited on page 11.)

Mark A. Paskin. Thin junction tree filters for simultaneous localization and mapping. In *International Joint Conference on Artificial Intelligence*, 2003. (Cited on page 11.)

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988. (Cited on pages 9, 10, and 19.)

Judea Pearl. *Causality*. Cambridge University Press, 2000. (Cited on page 10.)

Marcello Pelillo. Replicator equations, maximal cliques, and graph isomorphism. *Neural Computation*, 11:1933–1955, 1999. (Cited on page 6.)

Marcello Pelillo and Mario Refice. Learning compatibility coefficients for relaxation labeling processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9):933–945, 1994. (Cited on page 8.)

Yuval Peres, Dmitry Sotnikov, Benny Sudakov, and Uri Zwick. All-pairs shortest paths in $O(n^2)$ time with high probability. In *Symposium on Foundations of Computer Science*, 2010. (Cited on page 12.)

Kersten Petersen, Janis Fehr, and Hans Burkhardt. Fast generalized belief propagation for MAP estimation on 2D and 3D grid-like Markov random fields. In *DAGM*, 2008. (Cited on page 10.)

Antonio Robles-Kelly and Edwin R. Hancock. Point pattern matching via spectral geometry. In *SSPR & SPR*, 2006. (Cited on page 6.)

Mike Rogers and Jim Graham. Robust and accurate registration of 2-D electrophoresis gels using point-matching. *IEEE Transactions on Image Processing*, 16(3):624–635, 2007. (Cited on page 5.)

Azriel Rosenfeld and Avinash C. Kak. *Digital Picture Processing*. Academic Press, 1982. (Cited on page 6.)

Sam Roweis and Zoubin Ghahramani. A unifying review of linear gaussian models. *Neural Computation*, 11:305–345, 1999. (Cited on page 10.)

Daniel Scharstein and Richard S. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1):7–42, 2002. (Cited on page 9.)

Christian Schellewald. *Convex mathematical programs for relational matching of object views*. PhD thesis, University of Mannhein, 2004. (Cited on page 6.)

Larry S. Shapiro and Michael Brady. Feature-based correspondence - an eigenvector approach. *Image and Vision Computing*, 10(5):283–288, 1992. (Cited on page 6.)

Jianbo Shi and Carlo Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1994. (Cited on page 7.)

Leonid Sigal and Michael J. Black. Predicting 3D people from 2D pictures. In *Conference on Articulated Motion and Deformable Objects*, 2006. (Cited on pages 5 and 10.)

Imre Simon. Recognizable sets with multiplicities in the tropical semiring. In *Mathematical Foundations of Computer Science*, 1988. (Cited on page 11.)

Stephen M. Smith. A new class of corner finder. In *British Machine Vision Conference*, pages 139–148, 1992. (Cited on page 7.)

Padhraic Smyth. Belief networks, hidden Markov models, and Markov random fields: a unifying view. *Pattern Recognition Letters*, 18:1261–1268, 1998. (Cited on page 10.)

David Sontag, Talya Meltzer, Amir Globerson, Tommi Jaakkola, and Yair Weiss. Tightening LP relaxations for MAP using message passing. In *Uncertainty in Artificial Intelligence*, 2008. (Cited on pages 11 and 27.)

Volker Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 14(3):354–356, 1969. (Cited on pages 11 and 26.)

Charles Sutton and Andrew McCallum. Introduction to conditional random fields for relational learning. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2006. (Cited on page 11.)

Richard Szeliski, Ramin Zabih, Daniel Scharstein, Olga Veksler, Vladimir Kolmogorov, Assem Agarwala, Marshall Tappen, and Carsten Rother. A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(6):1068–1080, 2008. (Cited on page 9.)

Marshall F. Tappen and William T. Freeman. Comparison of graph cuts with belief propagation for stereo, using identical MRF parameters. In *International Conference on Computer Vision*, 2003. (Cited on page 9.)

Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin Markov networks. In *Advances in Neural Information Processing Systems*, 2004. (Cited on pages 8 and 18.)

Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. *Pattern Recognition*, 37(1):165–168, 2004. (Cited on page 7.)

Philip A. Tresadern, Harish Bhaskar, Steve A. Adeshina, Chris J. Taylor, and Tim F. Cootes. Combining local and global shape models for deformable object matching. In *British Machine Vision Conference*, 2009. (Cited on page 10.)

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector learning for interdependent and structured output spaces. In *International Conference on Machine Learning*, 2004. (Cited on page 9.)

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005. (Cited on pages 9 and 18.)

Jeffrey D. Ullman. An algorithm for subgraph isomorphism. *Journal of the ACM*, 23(1):31–42, 1976. (Cited on page 6.)

Leslie G. Valiant. General context-free recognition in less than cubic time. *Journal of Computer and Systems Sciences*, 10:308–314, 1975. (Cited on page 12.)

Paul B. van Wamelen, Zi Li, and Sitharama S. Iyengar. A fast expected time algorithm for the 2-D point pattern matching problem. *Pattern Recognition*, 37(8):1699–1711, 2004. (Cited on pages 5 and 7.)

Michaël A. van Wyk, Tariq S. Durrani, and Barend J. van Wyk. A RKHS interpolator-based graph matching algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):988–995, 2002. (Cited on page 7.)

Olga Veksler. Graph cut based optimization for MRFs with truncated convex priors. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007. (Cited on page 10.)

Martin J. Wainwright and Michael I. Jordan. *Graphical Models, Exponential Families, and Variational Inference*, volume 1 of *Foundations and Trends in Machine Learning*. Now Publishers, 2008. (Cited on page 10.)

Chong Wang, David Blei, and Li Fei-Fei. Simultaneous image classification and annotation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009. (Cited on page 9.)

Hongfang Wang and Edwin R. Hancock. A Kernel view of spectral point pattern matching. In *SSPR & SPR*, pages 361–369, 2004. (Cited on page 6.)

Yair Weiss. Correctness of local probability propagation in graphical models with loops. *Neural Computation*, 12:1–41, 2000. (Cited on pages 16 and 22.)

Ryan Williams. Matrix-vector multiplication in sub-quadratic time (some preprocessing required). In *Symposium on Discrete Algorithms*, 2007. (Cited on pages 11 and 27.)

Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix, and triangle problems. In *Symposium on Foundations of Computer Science*, 2010. (Cited on pages 12, 20, and 26.)

Richard C. Wilson and Edwin R. Hancock. Structural matching by discrete relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):634–648, 1997. (Cited on page 6.)

Chun-Nam J. Yu and Thorsten Joachims. Learning structural SVMs with latent variables. In *International Conference on Machine Learning*, 2009. (Cited on page 25.)

Hao Zhang and Jitendra Malik. Learning a discriminative classifier using shape context distances. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2003. (Cited on page 6.)

Graph Rigidity, Cyclic Belief Propagation, and Point Pattern Matching

Julian J. McAuley, Tibério S. Caetano, and Marconi S. Barbosa

Abstract—A recent paper [1] proposed a provably optimal polynomial time method for performing near-isometric point pattern matching by means of exact probabilistic inference in a chordal graphical model. Its fundamental result is that the chordal graph in question is shown to be *globally rigid*, implying that exact inference provides the *same* matching solution as exact inference in a complete graphical model. This implies that the algorithm is optimal when there is no noise in the point patterns. In this paper, we present a new graph that is also globally rigid but has an advantage over the graph proposed in [1]: Its maximal clique size is smaller, rendering inference significantly more efficient. However, this graph is not chordal, and thus, standard Junction Tree algorithms cannot be directly applied. Nevertheless, we show that loopy belief propagation in such a graph converges to the optimal solution. This allows us to retain the optimality guarantee in the noiseless case, while substantially reducing both memory requirements and processing time. Our experimental results show that the accuracy of the proposed solution is indistinguishable from that in [1] when there is noise in the point patterns.

Index Terms—Point pattern matching, graph matching, graphical models, belief propagation, global rigidity, chordal graphs.

1 Introduction

POINT pattern matching is a fundamental problem in pattern recognition and has been modeled in several different forms depending on the demands of the application domain in which it is required [2], [3], [4], [5]. A classic formulation that is realistic in many practical scenarios is that of near-isometric point pattern matching, in which we are given both a "template" $(\mathcal{T})$ and a "scene" $(\mathcal{S})$ point pattern, and it is assumed that $\mathcal{S}$ contains an instance of $\mathcal{T}$ (say, $\mathcal{T}'$) apart from an isometric transformation and possibly some small jitter in the point coordinates. The goal is to identify $\mathcal{T}'$ in $\mathcal{S}$ and find which points in $\mathcal{T}$ correspond to which points in $\mathcal{T}'$.

Recently, a method was introduced that solves this problem efficiently by means of exact belief propagation in a certain graphical model [1].[1] The approach is appealing because it is optimal not only in that it consists of exact inference in a graph with small maximal clique size $(= 4$ for matching in $\mathbb{R}^2)$ but also in that the *graph itself* is optimal. There it is shown that the maximum a posteriori (MAP) solution in the sparse and tractable graphical model where inference is performed is actually the *same* MAP solution that would be obtained if a fully connected model (which is intractable) could be used. This is due to the so-called *global rigidity* of the chordal graph in question: When the graph is embedded in the plane, the lengths of its edges uniquely determine the lengths of the absent edges (i.e., the edges of the graph complement) [9]. The computational complexity of the optimal point pattern matching algorithm is then shown to be $O(nm^4)$ (both in terms of processing time and memory requirements), where n is the number of points in the template point pattern and m is the number of points in the scene point pattern (usually with $m > n$ in applications). This reflects precisely the computational complexity of the Junction Tree algorithm in a chordal graph with $O(n)$ nodes, $O(m)$ states per node, and maximal cliques of size 4. The authors present experiments which give evidence that the method substantially improves on well-known matching techniques, including Graduated Assignment [10].

In this paper, we show how the same optimality proof can be obtained with an algorithm that runs in $O(nm^3)$ time per iteration. In addition, memory requirements are precisely decreased by a factor of m. We are able to achieve this by identifying a new graph, which is globally rigid but has a *smaller* maximal clique size: three. The main problem we face is that our graph is *not* chordal, so in order to enforce the running intersection property for applying the Junction Tree algorithm, the graph should first be triangulated; this would not be interesting in our case, since the resulting triangulated graph would have larger maximal clique size. Instead, we show that belief propagation in this graph *converges to the optimal solution*, although not necessarily in a single iteration. In practice, we find that convergence occurs after a small number of iterations, thus improving the runtime by an order of magnitude. We compare the performance of our model to that in [1] with synthetic and real point sets derived from images and show that in fact comparable accuracy is obtained while substantial speedups are observed.

2 Background

We consider point matching problems in $\mathbb{R}^2$. The problem we study is that of near-isometric point pattern matching (as defined above), i.e., one assumes that a near-isometric instance $(\mathcal{T}')$ of the template $(\mathcal{T})$ is somewhere "hidden" in the scene $(\mathcal{S})$. By "near-isometric," it is meant that the relative distances of points in $\mathcal{T}$ are approximately preserved in $\mathcal{T}'$. For simplicity of exposition, we assume that $\mathcal{T}$, $\mathcal{T}'$, and $\mathcal{S}$ are ordered sets (their elements are indexed). Our aim is to find a map $x : \mathcal{T} \mapsto \mathcal{S}$ with image $\mathcal{T}'$ that best preserves the relative distances of the points in $\mathcal{T}$ and $\mathcal{T}'$, i.e.,

$$x^* = \underset{x}{\operatorname{argmin}} \|D(\mathcal{T}) - D(x(\mathcal{T}))\|_2^2, \tag{1}$$

where $D(\mathcal{T})$ is the matrix whose (i,j)th entry is the euclidean distance between points indexed by i and j in $\mathcal{T}$ and $\|\cdot\|_2$ is the Frobenius norm. Note that finding x^* is inherently a combinatorial optimization problem since $\mathcal{T}'$ is itself a subset of $\mathcal{S}$, the scene point pattern. In [1], a generic point in $\mathcal{T}$ is modeled as a random variable (X_i) and a generic point in $\mathcal{S}$ is modeled as a possible realization of the random variable (x_i). As a result, a joint realization of all the random variables corresponds to a match between the template and the scene point patterns. A graphical model (see [11] and [12]) is then defined on this set of random variables whose edges are set according to the topology of a so-called 3-tree graph (any 3-tree that spans $\mathcal{T}$). A 3-tree is a graph obtained by starting with the complete graph on three vertices, K_3, and then adding new vertices that are connected only to those

1. See also [6] for further treatment of shape matching in the Bayesian setting; see [7] and [8] for further treatment using tree-structured models.

• *The authors are with the Statistical Machine Learning group, NICTA, Locked Bag 8001 Canberra ACT 2601 Australia, and the Research School of Information Sciences and Engineering, Australian National University, ACT 0200.*
E-mail: {julian.mcauley, Tiberio.Caetano}@nicta.com.au, marconi@ifsc.usp.br.

Manuscript received 29 Sept. 2007; revised 16 Jan. 2008; accepted 29 Apr. 2008; published online 14 May 2008.
Recommended for acceptance by M. Pelillo.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2007-09-0655.
Digital Object Identifier no. 10.1109/TPAMI.2008.124.

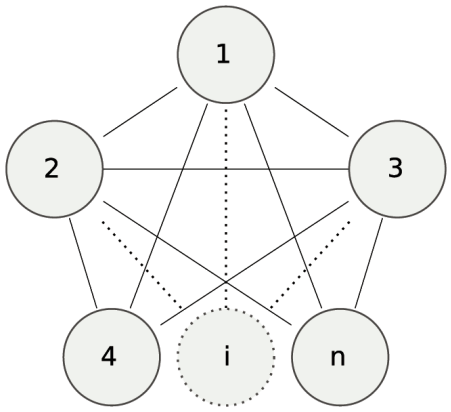


Fig. 1. An example of a 3-tree. Each node corresponds to a point in the template set.

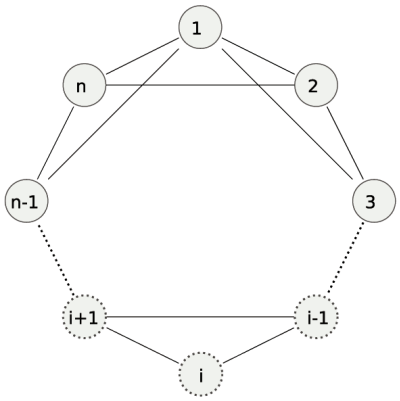


Fig. 2. The general form of the graph we consider, with n nodes.

same three vertices.[2] Fig. 1 shows an example of a 3-tree; it should be noted that the points in the template set correspond to the *nodes* in this graph, whereas the points in the scene correspond to their *assignments*. The reasons claimed in [1] for introducing 3-trees as a graph topology for the probabilistic graphical model are that 1) 3-trees are globally rigid in the plane and 2) 3-trees are chordal[3] graphs. This implies 1) that the 3-tree model is "optimal" (in a way that will be made clear in Section 3 in the context of the new graph we propose) and 2) that 3-trees have a Junction Tree with fixed maximal clique size $(= 4)$; as a result, it is possible to perform exact inference in polynomial time [1].

Potential functions are defined on pairs of neighboring nodes and are large if the difference between the distance of neighboring nodes in the template and the distance between the nodes they map to in the scene is small (and small if this difference is large). This favors isometric matchings. More precisely,

$$\psi_{ij}(X_i = x_i, X_j = x_j) = f\big(d(X_i, X_j) - d(x_i, x_j)\big), \qquad (2)$$

where $f(\cdot)$ is typically some unimodal function peaked at zero (e.g., a zero-mean Gaussian function) and $d(\cdot,\cdot)$ is the euclidean distance between the corresponding points (for simplicity of notation, we do not disambiguate between random variables and template points or realizations and scene points). For the case of *exact* matching, i.e., when there exists an x^* such that the minimal value in (1) is zero, then $f(\cdot) = \delta(\cdot)$ (where $\delta(\cdot)$ is just the indicator function $1_{\{0\}}(\cdot)$). The potential function of a maximal clique (Ψ) is then simply defined as the product of the potential functions over its six $(= C_2^4)$ edges (which will be maximal when every factor is maximal). It should be noted that the potential function of each edge is included in no more than *one* of the cliques containing that edge.

For the case of exact matching (i.e., no jitter), it is shown in [1] that running the Junction Tree algorithm on the 3-tree graphical model with $f(\cdot) = \delta(\cdot)$ will actually find a MAP assignment which coincides with x^*, i.e., such that $\|D(\mathcal{T}) - D(x^*(\mathcal{T}))\|_2^2 = 0$. This is due to the "graph rigidity" result, which tells us that equality of the lengths of the edges in the 3-tree and the edges induced by the matching in $\mathcal{T}'$ is sufficient to ensure the equality of the lengths of all pairs of points in $\mathcal{T}$ and $\mathcal{T}'$. This will be made technically precise in Section 3, when we prove an analogous result for another graph.[4]

3 An Improved Graph

Here, we introduce another globally rigid graph that has the advantage of having a *smaller maximal clique size*. Although the graph is *not* chordal, we will show that exact inference *is* tractable and that we will indeed benefit from the decrease in the maximal clique size. As a result, we will be able to obtain optimality guarantees like those from [1].

Our graph is constructed using Algorithm 1.

Algorithm 1. Graph generation for $\mathcal{G}$.

1. Create a cycle graph by traversing all the nodes in $\mathcal{T}$ (in any order)
2. Connect all nodes whose distance in this cycle graph is two (i.e., connect each node to its neighbor's neighbor)

This algorithm will produce a graph like the one shown in Fig. 2. We will denote by $\mathcal{G}$ the set of graphs that can be generated by Algorithm 1. $G = (V, E)$ will denote a generic graph in $\mathcal{G}$.

In order to present our results, we need to start with the definition of a globally rigid graph.

Definition 1. *A planar graph embedding G is said to be globally rigid in $\mathbb{R}^2$ if the lengths of the edges uniquely determine the lengths of the edges of the graph complement of G.*

Therefore, our statements are really about *graph embeddings* in $\mathbb{R}^2$, but for simplicity of presentation, we will simply refer to these embeddings as "graphs."

This means that there are no degrees of freedom for the absent edges in the graph: They must all have specified and fixed lengths. To proceed, we need a simple definition and some simple technical lemmas.

Definition 2. *A set of points is said to be in a general position in $\mathbb{R}^2$ if no three points lie in a straight line.*

Lemma 3. *Given a set of points in general position in $\mathbb{R}^2$, if the distances from a point P to two other fixed points are determined, then P can be in precisely two different positions.*

Proof. Consider two circles, each centered at one of the two reference points with radii equal to the given distances to point P. These circles intersect at precisely two points (since the three points are not collinear). This proves the statement. □

2. Technically, connecting new vertices to the three nodes of the original K_3 graph is not required: It suffices to connect new vertices to any existent 3-clique.

3. A chordal graph is one in which every cycle of length greater than three has a chord. A chord of a cycle is an edge not belonging to the cycle but which connects two nodes in the cycle (i.e., a "shortcut" in a cycle).

4. It is worth noting that a 2-tree would be sufficient if we only wished to allow for translations and rotations, and a 1-tree would be sufficient if we only wished to allow for translations [13]. Since we wish to handle all isometries, these models are not further considered.

The following lemma follows directly from Lemma 1 in [1] and is stated without proof:

Lemma 4. *Given a set of points in general position in $\mathbb{R}^2$, if the distances from a point P to three other fixed points are determined, then the position of P is uniquely determined.*

We can now present a proposition.

Proposition 5. *Any graph $G \in \mathcal{G}$ arising from Algorithm 1 is globally rigid in the plane if the nodes are in general position in the plane.*

Proof. Define a reference frame S, where points 1, 2, and n have specific coordinates (we say that the points are "determined"). We will show that all points then have determined positions in S and therefore have determined relative distances, which by definition implies that the graph is globally rigid.

We proceed by contradition: Assume there exists at least one undetermined point in the graph. Then, we must have an undetermined point i such that $i-1$ and $i-2$ are determined (since points 1 and 2 are determined). By virtue of Lemma 4, points $i+1$ and $i+2$ must then be also undetermined (otherwise, point i would have determined distances from three determined points and as a result would be determined).

Let us now assume that *only* points i, $i+1$, and $i+2$ are undetermined. Then, the only possible realizations for points i, $i+1$, and $i+2$ are their reflections with respect to the straight line that passes through points $i-1$ and $i+3$, since these are the only possible realizations that maintain the rigidity of the triangles $(i-1,i,i+1)$, $(i,i+1,i+2)$, and $(i+1,i+2,i+3)$, since $i-1$ and $i+3$ are assumed fixed. However, since $i+4$ and $i-2$ are also fixed by assumption, this would break the rigidity of triangles $(i+2,i+3,i+4)$ and $(i,i-1,i-2)$. Therefore, $i+3$ cannot be determined. This can then be considered as the base case in an induction argument that goes as follows: Assume *only* $i,\ldots,i+p$ are undetermined. Then, by reflecting these points over the line that joins $i-1$ and $i+p+1$ (which are fixed by assumption), we obtain the only other possible realization consistent with the rigidity of the triangles who have all their vertices in $i-1,\ldots,i+p+1$. However, this realization is inconsistent with the rigidity of triangles $(I+p,i+p+1,i+p+2)$ and $(i,i-1,i-2)$; therefore, $i+p+1$ must not be determined and, by induction, any point j such that $j>i+2$ must not be determined, which contradicts the assumption that n is determined. As a result, the assumption that there is at least one undetermined point in the graph is false. This implies that the graph has all points determined in S, and therefore, all relative distances are determined, and by definition, the graph is globally rigid. This proves the statement. □

Although we have shown that graphs $G \in \mathcal{G}$ are globally rigid, notice that they are *not* chordal. For the graph in Fig. 2, the cycles $(1,3,5,\ldots,n-1,1)$ and $(2,4,6,\ldots,n,2)$ have no chord. Moreover, triangulating this graph in order to make it chordal will necessarily increase (to at least 4) the maximal clique size (which is not sufficient for our purposes since we arrive at the case of [1]).

Instead, consider the clique graph formed by $G \in \mathcal{G}$. If there are n nodes, the clique graph will have cliques $(1,2,3),(2,3,4),\ldots,(n-2,n-1,n),(n-1,n,1),(n,1,2)$. This clique graph forms a cycle, which is depicted in Fig. 3.[5]

5. Note that if we connected *every* clique whose nodes intersected, the clique graph would have the same topology as the graph in Fig. 2; here, we have only formed enough connections so that the intersection of any two cliques is shared by the cliques on at least one path between them (similar to the running intersection property for Junction Trees).

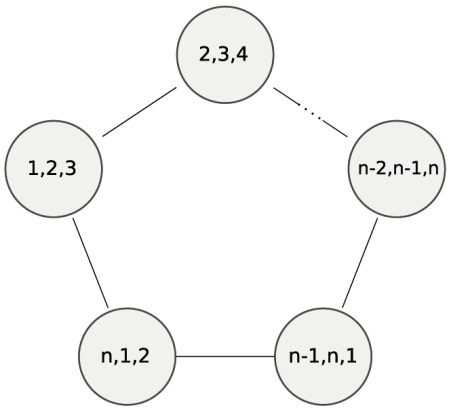


Fig. 3. The clique graph obtained from the graph in Fig. 2.

We now draw on results first obtained by Weiss [14] and confirmed elsewhere [15], [16]. There it is shown that, for graphical models with a single cycle, belief propagation converges to the optimal MAP assignment, although the computed marginals may be incorrect. Note that, for our purposes, this is precisely what is needed: We are after the most likely joint realization of the set of random variables, which corresponds to the best match between the template and the scene point patterns. Max-product belief propagation [17] in a cycle graph like the one shown in Fig. 3 amounts to computing the following messages iteratively:

$$m_{i\mapsto i+1}(U_i\cap U_{i+1}) = \max_{U_i\setminus U_{i+1}} \Psi(U_i) m_{i-1\mapsto i}(U_i\cap U_{i-1}), \qquad (3)$$

where U_i is the set of singleton variables in clique node i, $\Psi(U_i)$ is the potential function for clique node i, and $m_{i\mapsto i+1}$ is the message that passed from clique node i to clique node $i+1$. Upon reaching the convergence monitoring threshold, the optimal assignment for singleton variable j in clique node i is then computed by $\operatorname{argmax}_{U_i\setminus j} \Psi(U_i) m_{i-1\mapsto i}(U_i\cap U_{i-1}) m_{i+1\mapsto i}(U_i\cap U_{i+1})$.

Unfortunately, the above result is only shown in [14] when the *graph itself* forms a cycle, whereas we only have that the *clique graph* forms a cycle. However, it is possible to show that the result still holds in our case by considering a new graphical model in which the *cliques themselves* form the nodes, whose cliques are now just the edges in the clique graph. The result in [14] can now be used to prove that belief propagation in *this* graph converges to the optimal MAP assignment, which (by appropriately choosing potential functions for the new graph) implies that belief propagation should converge to the optimal solution in the *original* clique graph also.

To demonstrate this, we need not only to show that belief propagation in the new model converges to the optimal assignment, but also that belief propagation in the new model is *equivalent* to belief propagation in the original model.

Proposition 6. *The original clique graph (Fig. 3) can be transformed into a model containing only pairwise potentials, whose optimal MAP assignment is the same as the original model's.*

Proof. Consider a "node" $C_1=(X_1,X_2,X_3)$ (in the original clique graph) whose neighbors share exactly two of its nodes (for instance, $C_2=(X_2,X_3,X_4)$). Where the domain for each node in the original clique graph was simply $\{1,2,\ldots,|\mathcal{S}|\}$, the domain for each "node" in our new graph simply becomes $\{1,2,\ldots,|\mathcal{S}|\}^3$.

In this setting, it is no longer possible to ensure that the assignment chosen for each "node" is consistent with the assignment to its neighbor, that is, for an assignment (x_1,x_2,x_3)

to C_1 and (x'_2, x'_3, x_4) to C_2, we cannot guarantee that $x_2 = x'_2$ or $x_3 = x'_3$. Instead, we will simply define the potential functions on this new graph in such a way that the optimal MAP assignment implicitly ensures this equality. Specifically, we shall define the potential functions as follows: For two cliques $C_I = (I_1, I_2, I_3)$ and $C_J = (J_1, J_2, J_3)$ in the original graph (which share two nodes, say, (I_2, I_3) and (J_1, J_2)), define the *pairwise* potential for the clique $(\Psi'_{I,J})$ in the new graph as

$$\Psi'_{I,J}(i_{(123)}, j_{(123)}) = \begin{cases} \Psi_I(i_1, i_2, i_3), & \text{if } (i_2, i_3) = (j_1, j_2), \\ \rho, & \text{otherwise}, \end{cases} \tag{4}$$

where Ψ_I is simply the clique potential for the Ith clique in the *original* graph; $i_{(123)} \in \text{domain}(I_1) \times \text{domain}(I_2) \times \text{domain}(I_3)$ (sim for $j_{(123)}$). That is, we are setting the pairwise potential to simply be the original potential of *one* of the cliques if the assignments are compatible and ρ otherwise. If we were able to set $\rho = 0$, we would guarantee that the optimal MAP assignment was exactly the optimal MAP assignment in the original graph—however, this is not possible, since the result of [15] only holds when the potential functions have finite dynamic range. Hence, we must simply choose ρ sufficiently small so that the optimal MAP assignment cannot possibly contain an incompatible match—it is clear that this is possible, for example, $\rho = \left(\prod_C \max_{\mathbf{x}_C} \Psi_C(\mathbf{x}_C)\right)^{-1}$ will do (if the potentials are scaled to be at least one).

The result of [14] now implies that belief propagation in *this* graph will converge to the optimal MAP assignment, which we have shown is equal to the optimal MAP assignment in the *original* graph.[6] □

Proposition 7. *The messages passed in the new model are equivalent to the messages passed in the original model, except for repetition along one axis.*

Proof. We use induction on the number of iterations. First, we must show that the outgoing messages are the same during the first iteration (during which the incoming messages are not included). We will denote by $m^i_{(X_1,X_2,X_3) \mapsto (X_2,X_3,X_4)}$ the message from (X_1, X_2, X_3) to (X_2, X_3, X_4) during the ith iteration:

$$m^1_{(X_1,X_2,X_3) \mapsto (X_2,X_3,X_4)}(x_2, x_3) = \max_{X_1} \Psi_{(X_1,X_2,X_3)}(x_1, x_2, x_3), \tag{5}$$

$$\begin{aligned} &m^1_{(X_{(123)},X_{(234)}) \mapsto (X_{(234)},X_{(345)})}(x_{(234)}) \\ &\quad = \max_{X_{(123)}} \Psi'_{X_{(123)},X_{(234)}}(x_{(123)}, x_{(234)}) \\ &\quad = 1 \times \max_{X_1} \Psi_{(X_1,X_2,X_3)}(x_1, x_2, x_3) \\ &\quad = m^1_{(X_1,X_2,X_3) \mapsto (X_2,X_3,X_4)}(x_2, x_3). \end{aligned} \tag{6}$$

This result only holds due to the fact that ρ will never be chosen when maximizing along any axis. We now have that the messages are equal during the first iteration (the only difference being that the message for the new model is repeated along one axis).[7] Next, suppose that during the $(n-1)$st iteration, the messages (for both models) are equal to $\kappa(x_1, x_2)$. Then, for the nth iteration, we have

$$\begin{aligned} &m^n_{(X_1,X_2,X_3) \mapsto (X_2,X_3,X_4)}(x_2, x_3) \\ &\quad = \max_{X_1} \left\{ \Psi_{(X_1,X_2,X_3)}(x_1, x_2, x_3) \kappa(x_1, x_2) \right\}, \end{aligned} \tag{7}$$

$$\begin{aligned} &m^n_{(X_{(123)},X_{(234)}) \mapsto (X_{(234)},X_{(345)})}(x_{(234)}) \\ &\quad = \max_{X_{(123)}} \left\{ \Psi'_{X_{(123)},X_{(234)}}(x_{(123)}, x_{(234)}) \kappa(x_1, x_2) \right\} \\ &\quad = 1 \times \max_{X_1} \left\{ \Psi_{(X_1,X_2,X_3)}(x_1, x_2, x_3) \kappa(x_1, x_2) \right\} \\ &\quad = m^n_{(X_1,X_2,X_3) \mapsto (X_2,X_3,X_4)}(x_2, x_3). \end{aligned} \tag{8}$$

Hence, the two message passing schemes are equivalent by induction. □

We can now state our main result.

Theorem 8. *Let $G \in \mathcal{G}$ be a graph generated according to the procedure described in Algorithm 1. Assume that there is a perfect isometric instance of $\mathcal{T}$ within the scene point pattern $\mathcal{S}$.[8] Then, the MAP assignment x^* obtained by running belief propagation over the clique graph derived from G is such that $\|D(\mathcal{T}) - D(x^*(\mathcal{T}))\|_2^2 = 0$.*

Proof. For the exact matching case, we simply set $f(\cdot) = \delta(\cdot)$ in (2). Now, for a graph $G \in \mathcal{G}$ given by Algorithm 1, the clique graph will simply be a cycle, as shown in Fig. 3, and following Propositions 6 and 7, as well as the already mentioned result in [14], belief propagation will find the correct MAP assignment x^*, i.e.,

$$\begin{aligned} x^* &= \operatorname*{argmax}_x P_G(X = x) \\ &= \operatorname*{argmax}_x \prod_{i,j:(i,j) \in E} \delta\big(d(X_i, X_j) - d(x_i, x_j)\big), \end{aligned} \tag{9}$$

where P_G is the probability distribution for the graphical model induced by the graph G. Now, we need to show that x^* *also* maximizes the criterion which ensures isometry, i.e., we need to show that the above implies that

$$\begin{aligned} x^* &= \operatorname*{argmax}_x P_{\text{complete}}(X = x) \\ &= \operatorname*{argmax}_x \prod_{i,j} \delta\big(d(X_i, X_j) - d(x_i, x_j)\big), \end{aligned} \tag{10}$$

where P_{complete} is the probability distribution of the graphical model induced by the complete graph. Note that x^* must be such that the lengths of the edges in E are precisely equal to the lengths of the edges in $E_{\mathcal{T}'}$ (i.e., the edges induced in $\mathcal{S}$ from E by the map $X = x^*$). By the global rigidity of G, the lengths of $\bar{E}$ must then be also precisely equal to the lengths of $\bar{E}_{\mathcal{T}'}$. This implies that $\prod_{i,j:(i,j) \in \bar{E}} \delta(d(X_i, X_j) - d(x^*_i, x^*_j)) = 1$. Since (10) can be expanded as

$$\begin{aligned} x^* = \operatorname*{argmax}_x \Bigg\{ &\prod_{i,j:(i,j) \in E} \delta\big(d(X_i, X_j) - d(x_i, x_j)\big) \\ &\prod_{i,j:(i,j) \in \bar{E}} \delta\big(d(X_i, X_j) - d(x_i, x_j)\big) \Bigg\}, \end{aligned} \tag{11}$$

it becomes clear that x^* will also maximize (10). This proves the statement. □

6. To reiterate, the fact that the dynamic range is finite ensures that belief propagation converges to a unique fixed point (for graphs with a single loop) [15]; from this, the results in [14] and [16] guarantee optimality of the MAP assignment.

7. To be completely precise, the message for the new model is actually a function of only a *single* variable—$X_{(234)}$. By "repeated along one axis," we mean that for any given $(x_2, x_3, x_4) \in \text{domain}(X_2) \times \text{domain}(X_3) \times \text{domain}(X_4)$, the message at this point is independent of x_4, which therefore has no effect when maximizing.

8. In addition, we require that there is only a *single* isometric instance of $\mathcal{T}$ in $\mathcal{S}$ in order to ensure that the optimal MAP assignment is unique.

4 EXPERIMENTS

We have set up a series of experiments comparing the proposed model to that of [1]. Here, we compare graphs of the type shown in Fig. 2 to graphs of the type shown in Fig. 1.

The parameters used in our experiments are given as follows:

ϵ. This parameter controls the noise level used in our model. Here, we apply Gaussian noise to each of the points in $\mathcal{T}$ (with standard deviation ϵ in each axis). We have run our experiments on a range of noise levels between 0 and $4/256$ (where the original points in $\mathcal{T}$ are chosen randomly between 0 and 1). Note that this is the same as the setting used in [1].

Potential functions $\psi_{ij}(X_i = x_i, X_j = x_j) = f(d(X_i, X_j) - d(x_i, x_j))$. As in [1], we use a Gaussian function, i.e., $\exp\left(\frac{(d(X_i,X_j)-d(x_i,x_j))^2}{2\sigma^2}\right)$. The parameter σ is fixed beforehand as $\sigma = 0.4$ for the synthetic data and $\sigma = 150$ for the real-world data (as is done in [1]). While this potential function does *not* enforce that the chosen mapping is injective, it ensures that noninjective mappings are discouraged by having a low potential.

Dynamic range. As mentioned in Section 2, the potential function $\Psi(\mathbf{x})$ is simply the product of $\psi_{ij}(X_i = x_i, X_j = x_j)$ for all edges (i, j) in $\mathbf{x}$ (here, each maximal clique $\mathbf{x}$ contains three edges). The *dynamic range* of a function is simply defined as its maximum value divided by its minimum value (i.e., $\frac{\max_{\mathbf{x}} \Psi(\mathbf{x})}{\min_{\mathbf{x}} \Psi(\mathbf{x})}$). In order to prove convergence of our model, it is necessary that the dynamic range of our potential function is finite [15]. Therefore, rather than using $\Psi(\mathbf{x})$ directly, we use $\Psi'(\mathbf{x}) = (1/d) + (1 - 1/d)\Psi(\mathbf{x})$. This ensures that the dynamic range of our model is no larger than d and that $\Psi' \to \Psi$ as $d \to \infty$. In practice, we found that varying this parameter did not have a significant effect on convergence time. Hence, we simply fixed a large finite value $(d = 1{,}000)$ throughout.

MSE-cutoff. In order to determine the point at which belief propagation has converged, we compute the marginal distribution of every clique and compare it to the marginal distribution after the previous iteration. Belief propagation is terminated when this mean-squared error is less than a certain cutoff value for every clique in the graph. When choosing the mode of the marginal distributions after convergence, if two values differ by less than the square root of this cutoff, both of them are considered as possible MAP estimates (although this was rarely an issue when the cutoff was sufficiently small). We found that as $|\mathcal{S}|$ increased, the mean squared error between iterations tended to be smaller and therefore that smaller cutoff values should be used in these instances. Indeed, although the number of viable matches increases as $|\mathcal{S}|$ increases, the distributions increase in sparsity at an even faster rate—hence, the distributions tend to be *less peaked* on the average and changes are likely to have less effect on the mean squared error. Hence, we decreased the cutoff values by a factor of 10 when $|\mathcal{S}| \geq 30$.[9]

The clique graph in which messages are passed by our belief propagation algorithms is exactly that shown in Fig. 3. It is worth noting, however, that we also tried running belief propagation using a clique graph in which messages were passed between *all* intersecting cliques; we found that this made no difference to the performance of the algorithm[10] and we have therefore restricted our experiments to the clique graph in Fig. 3 with respect to its optimality guarantees.

9. Note that this is not a parameter in [1], in which only a single iteration is ever required.

10. Apart from one slight difference, including the additional edges appears to provide convergence in fewer iterations. However, since the number of messages being passed is doubled, the overall runtime for both clique graphs was ultimately similar.

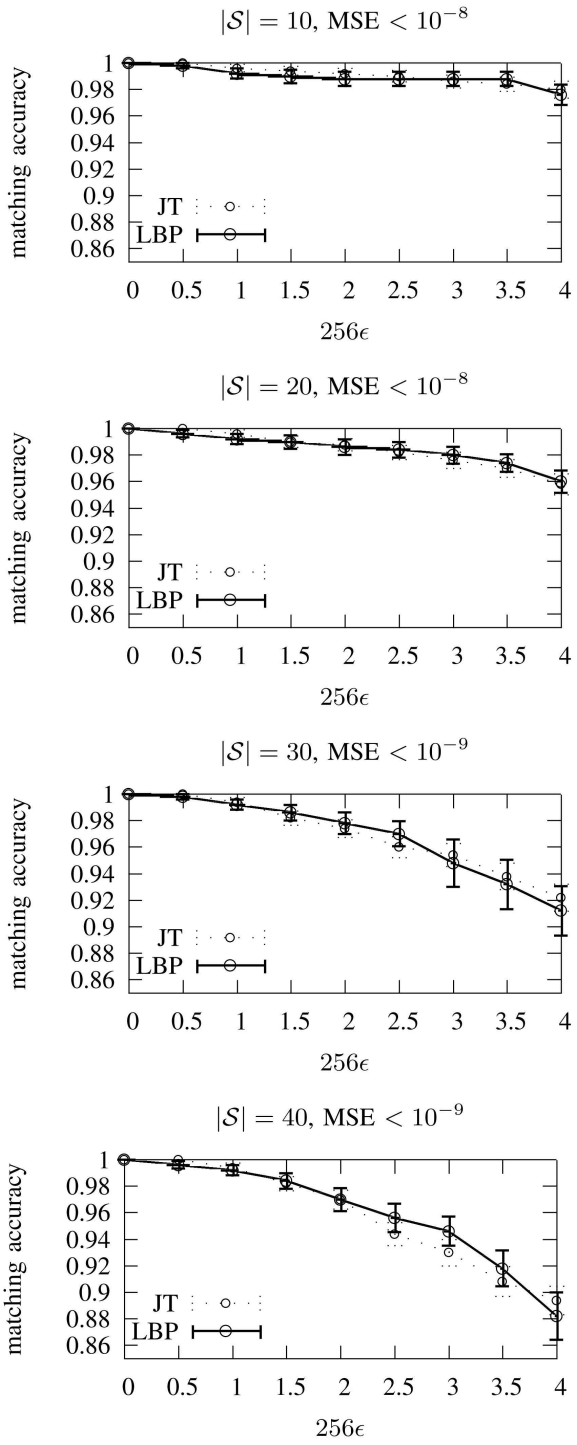


Fig. 4. Matching accuracy (proportion of correct matches) of our model against that in [1]. The performance of our model is statistically indistinguishable from that in [1] for all noise levels. The error bars indicate the average and standard error of 50 experiments.

For the sake of runtime comparison, we implemented the proposed model as well as that of [1] using the *Elefant*[11] belief propagation libraries in Python. However, to ensure that the results presented are consistent with those of [1], we simply used code that the authors provided when reporting the matching accuracy of their model. Our implementation computes messages (3) in a random order for all sites during each iteration.

Fig. 4 compares the matching accuracy (proportion of correct matches) of our model with that in [1] for $|\mathcal{S}| = 10$, 20, 30, and 40

11. http://elefant.developer.nicta.com.au/.

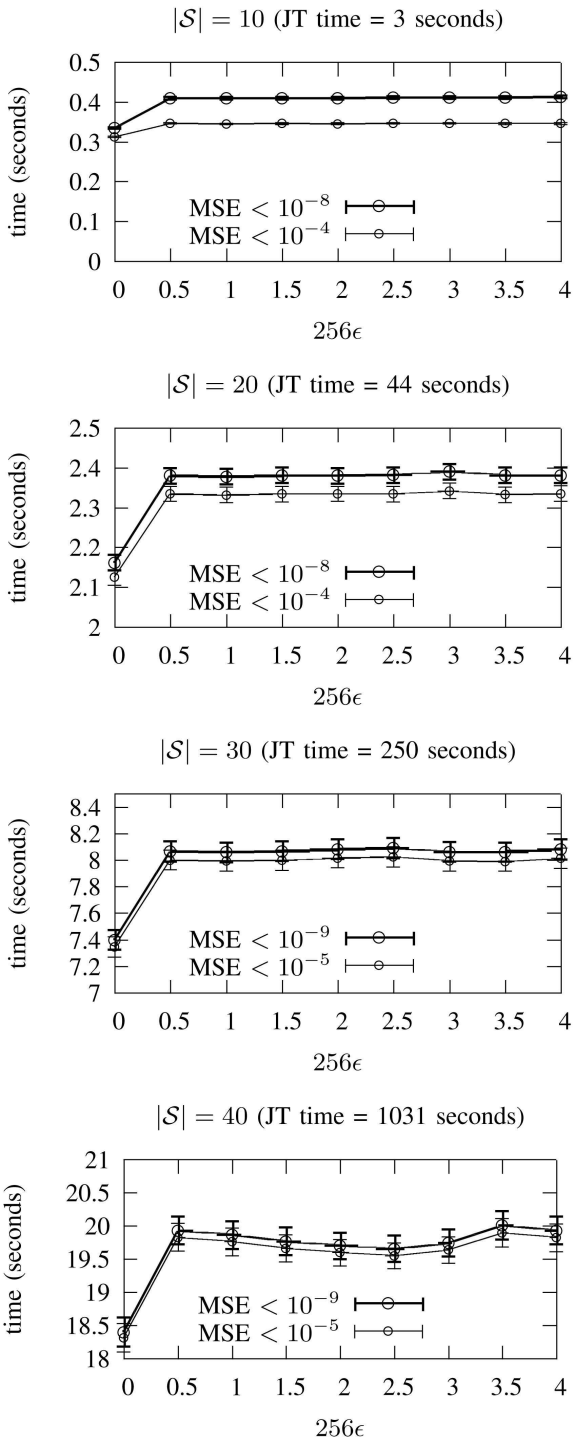


Fig. 5. Running-time of our model as the jitter varies, for different MSE-cutoffs. Speed-ups are almost exactly one order of magnitude.

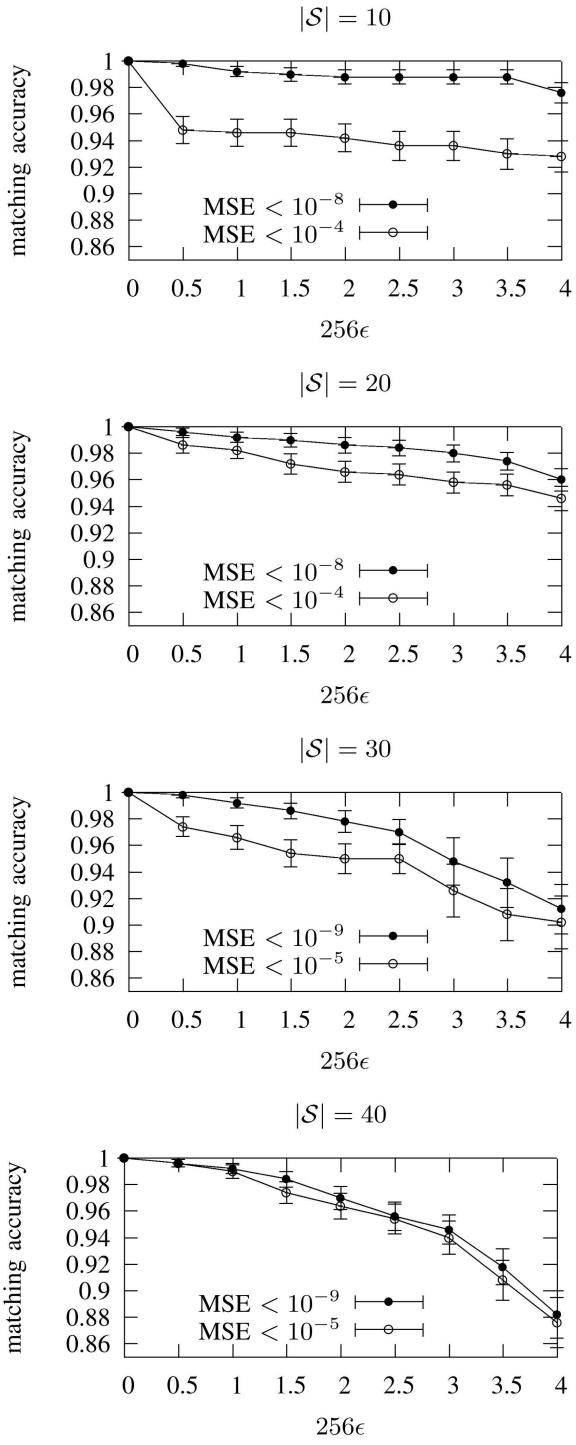


Fig. 6. Matching accuracy of our model as the MSE-cutoff varies. This figure suggests that the higher cutoff value should be sufficient when matching larger point sets.

(here, we fix $|\mathcal{T}| = 10$). The performance of our algorithm is indistinguishable from that of the Junction Tree algorithm.

Figs. 5 and 6 show the runtime and matching accuracy (respectively) of our model, as we vary the mean-squared error cutoff. Obviously, it is necessary to use a sufficiently low cutoff in order to ensure that our model has converged, but choosing too small a value may adversely effect its runtime. We found that the mean-squared error varied largely during the first few iterations, and we therefore enforced a minimum number of iterations (here, we chose at least 5) in order to ensure that belief propagation was not terminated prematurely. Fig. 5 reveals that the runtime is not significantly altered when increasing the MSE-cutoff—revealing that the model has almost always reached the lower cutoff value after five iterations (in which case, we should expect a speedup of precisely $|\mathcal{S}|/5$). Furthermore, decreasing the MSE-cutoff does not significantly improve the matching accuracy for larger point sets (Fig. 6), so choosing the lower cutoff does little harm if runtime is a major concern. Alternately, the Junction Tree model (which only requires a single iteration), took (for $S = 10$ to 40) 3, 44, 250, and 1,031 seconds, respectively. These models differ only in the topology of the network (see Section 3) and the size of the messages being passed; our method easily achieves an order of magnitude improvement for large networks.[12]

12. In fact, the speedup appears to be *more* than an order of magnitude for the large graphs, which is likely a side effect of the large memory requirements of the Junction Tree algorithm.

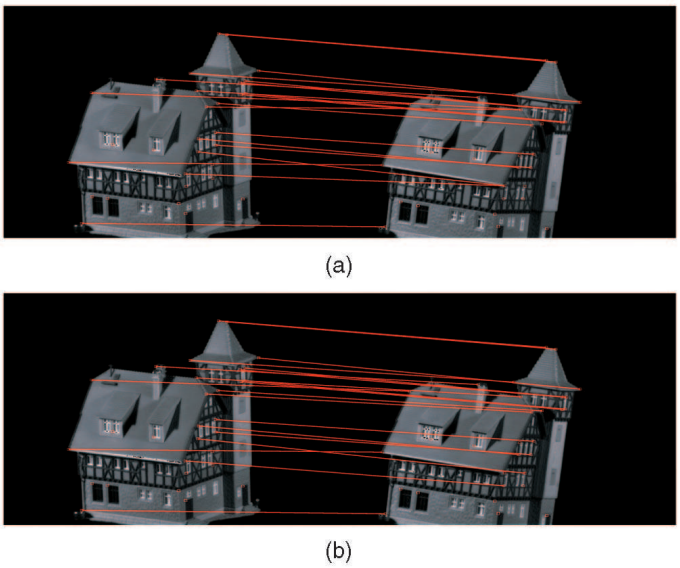


Fig. 7. (a) Points matched using the Junction Tree algorithm (the points in the left frame were matched to the corresponding points in the right frame); 16 points are correctly matched by this algorithm. (b) Points matched using our algorithm; 17 points are correctly matched.

Finally, we present matching results using data from the CMU house sequence.[13] In this data set, 30 points corresponding to certain features of the house are available over 111 frames. Fig. 7 shows the 71st and the last (111th) frames from this data set. Overlayed on these images are the 30 significant points, together with the matches generated by the Junction Tree algorithm and our own (matching the first 20 points); in this instance, the Junction Tree algorithm correctly matched 16 points and ours matched 17. Fig. 8 shows how accurately points between frames are matched as the baseline (separation between frames) varies. We also vary the number of points in the template set ($|\mathcal{T}|$) from 15 to 30. Our model seems to outperform the Junction Tree model for small baselines, whereas, for large baselines and larger point sets, the Junction Tree model seems to be the best. It is however difficult to draw conclusions from both models in these cases, since they are designed for the near-isometric case, which is violated for larger baselines.

5 Conclusions

We have shown that the near-isometric point pattern matching problem can be solved much more efficiently than what is currently reported as the state of the art, while maintaining the same optimality guarantees for the noiseless case and comparable accuracy for the noisy case. This was achieved by identifying a new type of graph with the same global rigidity property of previous graphs, in which exact inference is far more efficient. Although exact inference is not *directly* possible by means of the Junction Tree algorithm since the graph is not chordal, what we managed to show is that loopy belief propagation in such a graph does converge to the optimal solution in a sufficiently small number of iterations. In the end, the advantage of the smaller clique size of our model dominates the disadvantage caused by the need for more than a single iteration.

13. http://vasc.ri.cmu.edu/idb/html/motion/house/index.html.

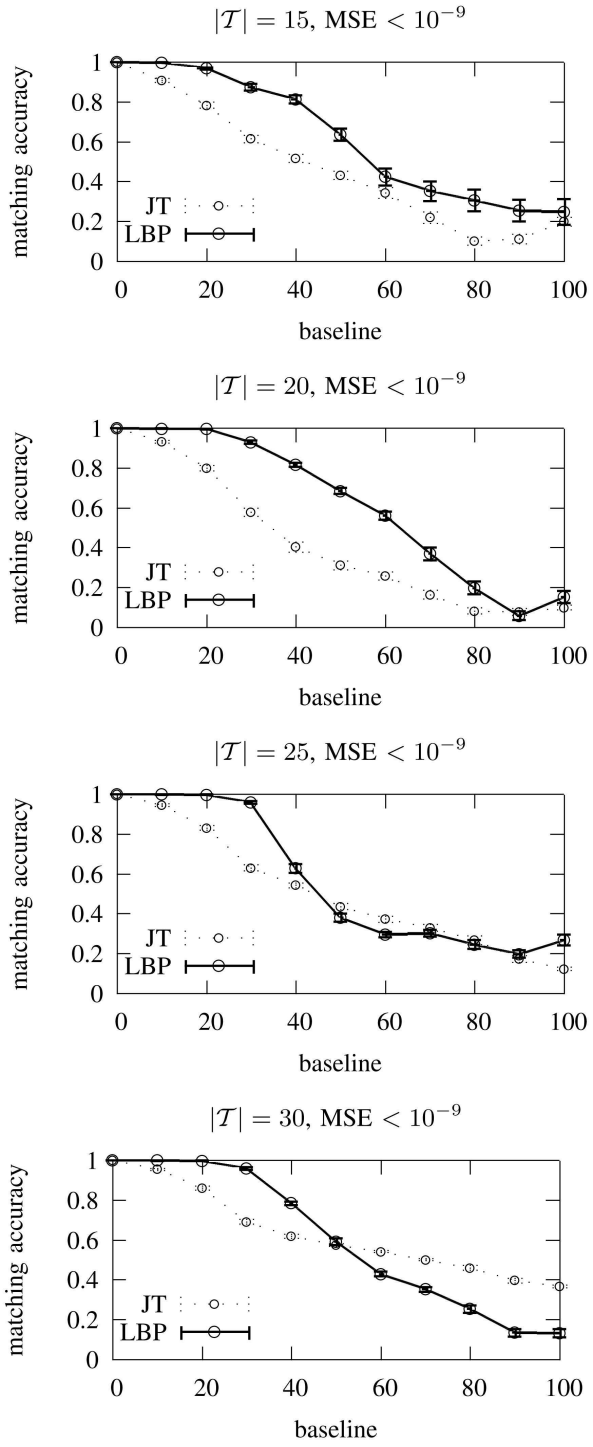


Fig. 8. Matching accuracy of our model using the "house" data set, as the baseline (separation between frames) varies.

References

[1] T.S. Caetano, T. Caelli, D. Schuurmans, and D.A.C. Barone, "Graphical Models and Point Pattern Matching," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, no. 10, pp. 1646-1663, Oct. 2006.

[2] Y. Amit and A. Kong, "Graphical Templates for Model Registration," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 18, no. 3, pp. 225-236, Mar. 1996.

[3] M. Carcassoni and E.R. Hancock, "Point Pattern Matching with Robust Spectral Correspondence," *Computer Vision and Pattern Recognition*, vol. 1, p. 1649, 2000.

[4] S. Belongie, J. Malik, and J. Puzicha, "Shape Matching and Object Recognition Using Shape Contexts," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 4, pp. 509-522, Apr. 2002.

[5] A. Robles-Kelly and E.R. Hancock, "Point Pattern Matching via Spectral Geometry," *Syntactical and Structural Pattern Recognition, Statistical Pattern Recognition*, pp. 459-467, 2006.

[6] A. Rangarajan, J. Coughlan, and A.L. Yuille, "A Bayesian Network Framework for Relational Shape Matching," *Proc. Int'l Conf. Computer Vision (ICCV '03)*, p. 671, 2003.
[7] P.F. Felzenszwalb, "Representation and Detection of Deformable Shapes," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, no. 2, pp. 208-220, Feb. 2005.
[8] D. Crandall, P. Felzenszwalb, and D. Huttenlocher, "Spatial Priors for Part-Based Recognition Using Statistical Models," *Proc. Int'l Conf. Computer Vision and Pattern Recognition (CVPR '05)*, vol. 1, pp. 10-17, 2005.
[9] R. Connelly, "Generic Global Rigidity," *Discrete and Computational Geometry*, vol. 33, no. 4, pp. 549-563, 2005.
[10] S. Gold and A. Rangarajan, "A Graduated Assignment Algorithm for Graph Matching," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 18, no. 4, pp. 377-388, Apr. 1996.
[11] S.L. Lauritzen, *Graphical Models (Oxford Statistical Science Series)*. Oxford Univ. Press, July 1996.
[12] C.M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, Aug. 2006.
[13] T.S. Caetano and T. Caelli, "A Unified Formulation of Invariant Point Pattern Matching," *Proc. Int'l Conf. Pattern Recognition (ICPR '06)*, pp. 121-124, 2006.
[14] Y. Weiss, "Correctness of Local Probability Propagation in Graphical Models with Loops," *Neural Computation*, vol. 12, pp. 1-41, 2000.
[15] A.T. Ihler, J.W. Fisher, and A.S. Willsky, "Message Errors in Belief Propagation," *Advances in Neural Information Processing Systems*, pp. 609-616, 2005.
[16] Y. Weiss and W.T. Freeman, "On the Optimality of Solutions of the Max-Product Belief-Propagation Algorithm in Arbitrary Graphs," *IEEE Trans. Information Theory*, vol. 47, 2001.
[17] J.S. Yedidia, W.T. Freeman, and Y. Weiss, "Generalized Belief Propagation," *Advances in Neural Information Processing Systems*, pp. 689-695, 2000.

Robust Near-Isometric Matching via Structured Learning of Graphical Models

Julian J. McAuley
NICTA/ANU
julian.mcauley
@nicta.com.au

Tibério S. Caetano
NICTA/ANU
tiberio.caetano
@nicta.com.au

Alexander J. Smola
Yahoo! Research*
alex@smola.org

Abstract

Models for near-rigid shape matching are typically based on distance-related features, in order to infer matches that are consistent with the isometric assumption. However, real shapes from image datasets, even when expected to be related by "almost isometric" transformations, are actually subject not only to noise but also, to some limited degree, to variations in appearance and scale. In this paper, we introduce a graphical model that parameterises appearance, distance, and angle features and we learn all of the involved parameters via structured prediction. The outcome is a model for near-rigid shape matching which is *robust* in the sense that it is able to capture the possibly limited but still important scale and appearance variations. Our experimental results reveal substantial improvements upon recent successful models, while maintaining similar running times.

1 Introduction

Matching shapes in images has many applications, including image retrieval, alignment, and registration [1, 2, 3, 4]. Typically, matching is approached by selecting features for a set of landmark points in both images; a correspondence between the two is then chosen such that some distance measure between these features is minimised. A great deal of attention has been devoted to defining complex features which are robust to changes in rotation, scale etc. [5, 6].[1]

An important class of matching problems is that of *near-isometric* shape matching. In this setting, it is assumed that shapes are defined up to an isometric transformation (allowing for some noise), and therefore distance features are typically used to encode the shape. Recent work has shown how the isometric constraint can be exploited by a particular type of graphical model whose topology encodes the necessary properties for obtaining optimal matches in polynomial time [11].

Another line of work has focused on *structured learning* to optimize graph matching scores, however no explicit exploitation of the geometrical constraints involved in shape modeling are made [12].

In this paper, we combine the best of these two approaches into a single model. We produce an exact, efficient model to solve near-isometric shape matching problems using not only isometry-invariant features, but also appearance and scale-invariant features. By doing so we can learn the relative importances of variations in appearance and scale with regard to variations in shape *per se*. Therefore, even knowing that we are in a near-isometric setting, we will capture the eventual variations in appearance and scale into our matching criterion in order to produce a *robust* near-isometric matcher. In terms of learning, we introduce a two-stage structured learning approach to address the speed and memory efficiency of this model.

*Alexander J. Smola was with NICTA at the time of this work.

[1] We restrict our attention to this type of approach, i.e. that of matching landmarks between images. Some notable approaches deviate from this norm – see (for example) [7, 8, 9, 10].

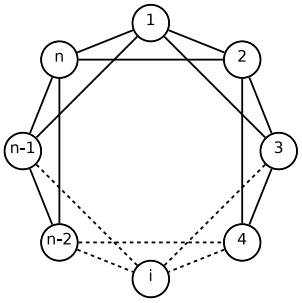


Figure 1: The graphical model introduced in [11].

2 Background

2.1 Shape Matching

'Shape matching' can mean many different things, depending on the precise type of query one is interested in. Here we study the case of identifying an instance of a template shape ($\mathcal{S} \subseteq \mathcal{T}$) in a target scene ($\mathcal{U}$) [1].[2] We assume that we *know* $\mathcal{S}$, i.e. the points in the template that we want to query in the scene. Typically both $\mathcal{T}$ and $\mathcal{U}$ correspond to a set of 'landmark' points, taken from a pair of images (common approaches include [6, 13, 14]).

For each point $t \in \mathcal{T}$ and $u \in \mathcal{U}$, a certain set of *unary features* are extracted (here denoted by $\phi(t)$, $\phi(u)$), which contain local information about the image at that point [5, 6]. If $y : \mathcal{S} \to \mathcal{U}$ is a generic mapping representing a potential match, the goal is then to find a mapping $\hat{y}$ which minimises the aggregate distance between corresponding features, i.e.

$$\hat{y} = f(\mathcal{S},\mathcal{U}) = \operatorname*{argmin}_{y} \sum_{i=1}^{|\mathcal{S}|} c_1(s_i, y(s_i)), \text{ where } c_1(s_i, y(s_i)) = \|\phi(s_i) - \phi(y(s_i))\|_2^2. \quad (1)$$

(here $\|\cdot\|_2$ denotes the L_2 norm). For injective y eq. (1) is a linear assignment problem, efficiently solvable in cubic time. In addition to unary or first-order features, pairwise or second-order features can be induced from the locations of the unary features. In this case eq. (1) would be generalised to minimise an aggregate distance between pairwise features. This however induces an NP-hard problem (quadratic assignment). Discriminative structured learning has recently been applied to models of both linear and quadratic assignment in [12].

2.2 Graphical Models

In isometric matching settings, one may suspect that it may not be necessary to include all pairwise relations in quadratic assignment. In fact a recent paper [11] has shown that if only the distances as encoded by the graphical model depicted in figure 1 are taken into account (nodes represent points in $\mathcal{S}$ and states represent points in $\mathcal{U}$), exact probabilistic inference in such a model can solve the isometric problem optimally. That is, an energy function of the following form is minimised:[3]

$$\sum_{i=1}^{|\mathcal{S}|} c_2(s_i, s_{i+1}, y(s_i), y(s_{i+1})) + c_2(s_i, s_{i+2}, y(s_i), y(s_{i+2})). \quad (2)$$

In [11], it is shown that loopy belief propagation using this model converges to the optimal assignment, and that the number of iterations required before convergence is small in practice.

We will extend this model by adding a unary term, $c_1(s_i, y(s_i))$ (as in (eq. 1)), and a third-order term, $c_3(s_i, s_{i+1}, s_{i+2}, y(s_i), y(s_{i+1}), y(s_{i+2}))$. Note that the graph topology remains the same.

[2] Here $\mathcal{T}$ is the set of *all points* in the template scene, whereas $\mathcal{S}$ corresponds to those points in which we are interested. It is also important to note that we treat $\mathcal{S}$ as an *ordered* object in our setting.

[3] s_{i+1} should be interpreted as $s_{(i+1) \mod |\mathcal{S}|}$ (i.e. the points form a loop).

2.3 Discriminative Structured Learning

In practice, feature vectors may be very high-dimensional, and which components are 'important' will depend on the specific properties of the shapes being matched. Therefore, we introduce a parameter, θ, which controls the relative importances of the various feature components. Note that θ is parameterising the matching criterion itself. Hence our minimisation problem becomes

$$\hat{y} = f(\mathcal{S},\mathcal{U};\theta) = \underset{y}{\operatorname{argmax}} \langle h(\mathcal{S},\mathcal{U},y),\theta\rangle \tag{3}$$

$$\text{where } h(\mathcal{S},\mathcal{U},y) = -\sum_{i=1}^{|\mathcal{S}|} \Phi(s_i, s_{i+1}, s_{i+2}, y(s_i), y(s_{i+1}), y(s_{i+2})). \tag{4}$$

(y is a mapping from $\mathcal{S}$ to $\mathcal{U}$, Φ is a third-order feature vector – our specific choice is shown in section 3).[4] In order to measure the performance of a particular weight vector, we use a *loss function*, $\Delta(\hat{y}, y^i)$, which represents the cost incurred by choosing the assignment $\hat{y}$ when the correct assignment is y^i (our specific choice of loss function is described in section 4). To avoid overfitting, we also desire that θ is sufficiently 'smooth'. Typically, one uses the squared L_2 norm, $\|\theta\|_2^2$, to penalise non-smooth choices of θ [15].

Learning in this setting now becomes a matter of choosing θ such that the empirical risk (average loss on all training instances) is minimised, but which is also sufficiently 'smooth' (to prevent overfitting). Specifically, if we have a set of training pairs, $\left\{\mathcal{S}^1 \dots \mathcal{S}^N\right\}$, $\left\{\mathcal{U}^1 \dots \mathcal{U}^N\right\}$, with labelled matches $\left\{y^1 \dots y^N\right\}$, then we wish to minimise

$$\underbrace{\frac{1}{N}\sum_{i=1}^{N} \Delta(f(\mathcal{S}^i,\mathcal{U}^i;\theta), y^i)}_{\text{empirical risk}} + \underbrace{\frac{\lambda}{2}\|\theta\|_2^2}_{\text{regulariser}} . \tag{5}$$

Here λ (the regularisation constant) controls the relative importance of minimising the empirical risk against the regulariser. In our case, we simply choose λ such that the empirical risk on our validation set is minimised.

Solving (eq. 5) exactly is an extremely difficult problem and in practice is not feasible, since the loss is piecewise constant on the parameter θ. Here we capitalise on recent advances in large-margin structured estimation [15], which consist of obtaining convex relaxations of this problem. Without going into the details of the solution (see, for example, [15, 16]), it can be shown that a convex relaxation of this problem can be obtained, which is given by

$$\min_{\theta} \frac{1}{N}\sum_{i=1}^{N} \xi_i + \frac{\lambda}{2}\|\theta\|_2^2 \tag{6a}$$

subject to

$$\langle h(\mathcal{S}^i,\mathcal{U}^i,y^i) - h(\mathcal{S}^i,\mathcal{U}^i,y),\theta\rangle \geq \Delta(y,y^i) - \xi_i$$

$$\text{for all } i \text{ and } y \in \mathcal{Y} \tag{6b}$$

(where $\mathcal{Y}$ is the space of all possible mappings). It can be shown that for the solution of the above problem, we have that $\xi_i^* \geq \Delta(f(\mathcal{S}^i,\mathcal{U}^i;\theta), y^i)$. This means that we end up minimising an upper bound on the loss, instead of the loss itself.

Solving (6) requires only that we are able, for any value of θ, to find

$$\underset{y}{\operatorname{argmax}} \left(\langle h(\mathcal{S}^i,\mathcal{U}^i,y),\theta\rangle + \Delta(y,y^i)\right). \tag{7}$$

In other words, for each value of θ, we are able to identify the mapping which is consistent with the model (eq. 3), yet incurs a high loss. This process is known as 'column generation' [15, 16]. As we will define our loss as a sum over the nodes, solving (eq. 7) is no more difficult than solving (eq. 3).

[4] We have expressed (eq. 3) as a maximisation problem as a matter of convention; this is achieved simply by negating the cost function in (eq. 4).

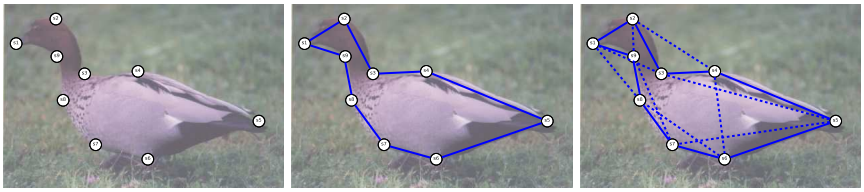


Figure 2: Left: the (ordered) set of points in our template shape ($\mathcal{S}$). Centre: connections between immediate neighbours. Right: connections between neighbour's neighbours (our graphical model).

3 Our Model

Although the model of [11] solves isometric matching problems optimally, it provides no guarantees for *near*-isometric problems, as it only considers those compatibilities which form cliques in our graphical model. However, we are often only interested in the boundary of the object: if we look at the instance of the model depicted in figure 2, it seems to capture exactly the important dependencies; adding additional dependencies between distant points (such as the duck's tail and head) would be unlikely to contribute to this model.

With this in mind, we introduce three new features (for brevity we use the shorthand $y_i = y(s_i)$):

$\Phi_1(s_1, s_2, y_1, y_2) = (d_1(s_1, s_2) - d_1(y_1, y_2))^2$, where $d_1(a, b)$ is the Euclidean distance between a and b, scaled according to the width of the target scene.

$\Phi_2(s_1, s_2, s_3, y_1, y_2, y_3) = (d_2(s_1, s_2, s_3) - d_2(y_1, y_2, y_3))^2$, where $d_2(a, b, c)$ is the Euclidean distance between a and b scaled by the average of the distances between a, b, and c.

$\Phi_3(s_1, s_2, s_3, y_1, y_2, y_3) = (\angle(s_1, s_2, s_3) - \angle(y_1, y_2, y_3))^2$, where $\angle(a, b, c)$ is the angle between a and c, w.r.t. b.[5]

We also include the unary features $\Phi_0(s_1, y_1) = (\phi(s_1) - \phi(y_1))^2$ (i.e. the pointwise squared difference between $\phi(s_1)$ and $\phi(y_1)$). Φ_1 is exactly the feature used in [11], and is invariant to isometric transformations (rotation, reflection, and translation); Φ_2 and Φ_3 capture triangle similarity, and are thus also invariant to scale. In the context of (eq. 4), we have

$$\Phi(s_1, s_2, s_3, y_1, y_2, y_3) := \big[\Phi_0(s_1, y_1), \Phi_1(s_1, s_2, y_1, y_2) + \Phi_1(s_1, s_3, y_1, y_3), \\ \Phi_2(s_1, s_2, s_3, y_1, y_2, y_3) + \Phi_2(s_1, s_3, s_2, y_1, y_3, y_2), \Phi_3(s_1, s_2, s_3, y_1, y_2, y_3)\big]. \quad (8)$$

In practice, landmark detectors often identify several hundred points [6, 17], which is clearly impractical for an $O(|\mathcal{S}||\mathcal{U}|^3)$ method ($|\mathcal{U}|$ is the number of landmarks in the target scene). To address this, we adopt a two stage learning approach: in the first stage, we learn only unary compatibilities, exactly as is done in [12]. During the second stage of learning, we collapse the first-order feature vector into a single term, namely

$$\Phi'_0(s_1, y_1) = \langle \theta_0, \Phi_0(s_1, y_1) \rangle \quad (9)$$

(θ_0 is the weight vector learned during the first stage). We now perform learning for the third-order model, *but consider only the p 'most likely' matches for each node*, where the likelihood is simply determined using $\Phi'_0(s_1, y_1)$. This reduces the performance and memory requirements to $O(|\mathcal{S}|p^3)$. A consequence of using this approach is that we must now tune *two* regularisation constants; this is not an issue in practice, as learning can be performed quickly using this approach.[6]

[5] Using features of such different scales can be an issue for regularisation – in practice we adjusted these features to have roughly the same scale. For full details, our implementation is available at (*not included for blind review*).

[6] In fact, even in those cases where a single stage approach was tractable (such as the experiment in section 4.1), we found that the two stage approach worked better. Typically, we required much less regularity during the second stage, possibly because the higher order features are heterogeneous.

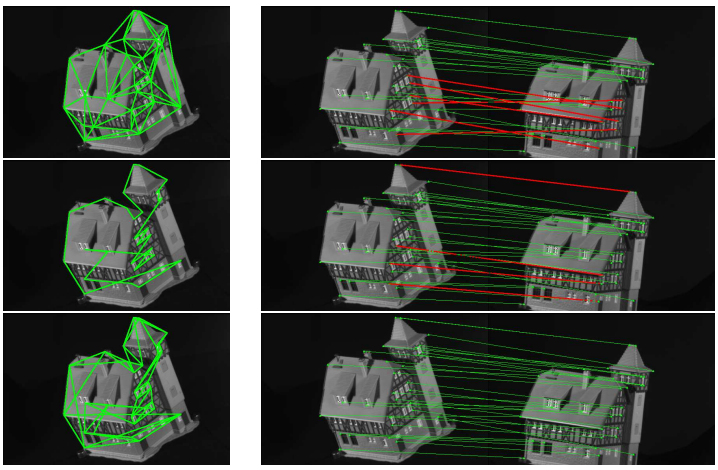


Figure 3: Left: The adjacency structure of the graph (top); the boundary of our 'shape' (centre); the topology of our graphical model (bottom). Right: Example matches using linear assignment (top, 6/30 mismatches), quadratic assignment (centre, 4/30 mismatches), and the proposed model (bottom, no mismatches). The images shown are the 12^{th} and 102^{nd} frames in our sequence. Correct matches are shown in green, incorrect matches in red. All matches are reported *after* learning.

4 Experiments

4.1 House Data

In our first experiment, we compare our method to those of [11] and [12]. Both papers report the performance of their methods on the CMU 'house' sequence – a sequence of 111 frames of a toy house, with 30 landmarks identified in each frame.[7] As in [12], we compute the Shape Context features for each of the 30 points [5].

In addition to the unary model of [12], a model based on *quadratic* assignment is also presented, in which pairwise features are determined using the adjacency structure of the graphs. Specifically, if a pair of points (p_1, p_2) in the template scene is to be matched to (q_1, q_2) in the target, there is a feature which is 1 if there is an edge between p_1 and p_2 in the template, *and* an edge between q_1 and q_2 in the target (and 0 otherwise). We also use such a feature for this experiment, however our model only considers matchings for which (p_1, p_2) forms an edge in our graphical model (see figure 3, bottom left). The adjacency structure of the graphs is determined using the Delaunay triangulation, (figure 3, top left).

As in [11], we compare pairs of images with a fixed baseline (separation between frames). For our loss function, $\Delta(\hat{y}, y^i)$, we used the normalised Hamming loss, i.e. the proportion of mismatches. Figure 4 shows our performance on this dataset, as the baseline increases. On the left we show the performance without learning, for which our model exhibits the best performance by a substantial margin.[8]

Our method is also the best performing after learning – in fact, we achieve almost zero error for all but the largest baselines (at which point our model assumptions become increasingly violated, and we have less training data). In figure 5, we see that the running time of our method is similar to the quadratic assignment method of [12]. To improve the running time, we also show our results with $p = 10$, i.e. for each point in the template scene, we only consider the 10 'most likely' matches, using the weights from the first stage of learning. This reduces the running time by more than an order of

[7] `http://vasc.ri.cmu.edu/idb/html/motion/house/index.html`

[8] Interestingly, the quadratic method of [12] performs *worse* than their unary method; this is likely because the *relative* scale of the unary and quadratic features is badly tuned before learning, and is indeed similar to what the authors report. Furthermore, the results we present for the method of [12] after learning are much *better* than what the authors report – in that paper, the unary features are scaled using a pointwise exponent ($-\exp(-|\phi_a - \phi_b|^2)$), whereas we found that scaling the features linearly ($|\phi_a - \phi_b|^2$) worked better.

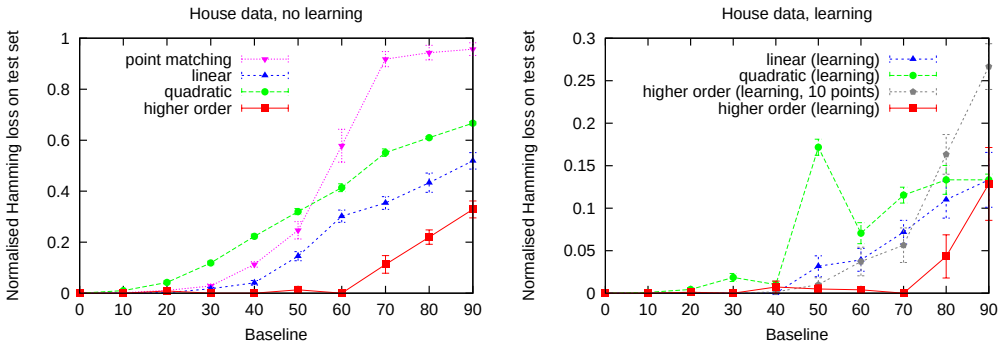


Figure 4: Comparison of our technique against that of [11] ('point matching'), and [12] ('linear', 'quadratic'). The performance before learning is shown on the left, the performance after learning is shown on the right. Our method exhibits the best performance both before and after learning (*note the different scales of the two plots*). Error bars indicate standard error.

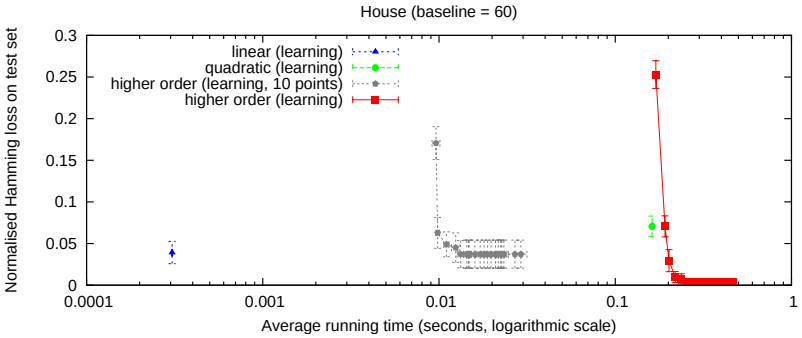


Figure 5: The running time and performance of our method, compared to those of [12] (note that the method of [11] has running time identical to our method). Our method is run from 1 to 20 iterations of belief propagation, although the method appears to converge in fewer than 5 iterations.

magnitude, bringing it closer to that of linear assignment; even this model achieves approximately zero error up to a baseline of 50.

Finally, figure 6 (left) shows the weight vector of our model, for a baseline of 60. The first 60 weights are for the Shape Context features (determined during the first stage of learning), and the final 5 show the weights from our second stage of learning (the weights correspond to the first-order features, distances, adjacencies, scaled distances, and angles, respectively – see section 3). We can provide some explanation of the learned weights: the Shape Context features are separated into 5 radial, and 12 angular bins – the fact that there are peaks around the 16^{th} and 24^{th}, features indicates that some particular radial bins are more important than the others; the fact that several consecutive bins have low weight indicates that some radial bins are unimportant (etc.). It is much more difficult to reason about the second stage of learning, as the features have different scales, and cannot be compared directly – however, it appears that all of the higher-order features are important to our model.

4.2 Bikes Data

For our second experiment, we used images of bicycles from the Caltech 256 Dataset [18]. Bicycles are reasonably rigid objects, meaning that matching based on their shape is logical. Although the images in this dataset are fairly well aligned, they are subject to reflections as well as some scaling and shear. For each image in the dataset, we detected landmarks automatically, and six points on the frame were hand-labelled (see figure 7). Only shapes in which these interest points were not occluded were used, and we only included images that had a background; in total, we labelled 44

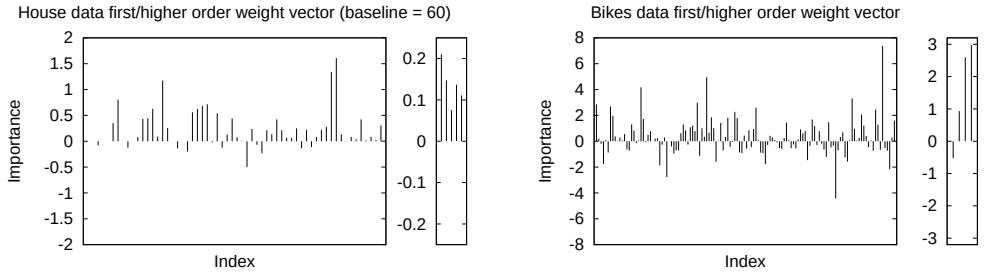


Figure 6: Left: The weight vector of our method after learning, for the 'house' data. The first 60 weights are for the Shape Context features from the first stage of of learning; the final 5 weights are for the second stage of learning. Right: The same plot, for the 'bikes' data.

Figure 7: Top: A selection of our training images. Bottom: An example match from our test set. Left: The template image (with the shape outlined in green, and landmark points marked in blue). Centre: The target image, and the match (in red) using unary features with the affine invariant/SIFT model of [17] after learning (endpoint error = 0.27). Right: the match using our model after learning (endpoint error = 0.04).

images. The first image was used as the 'template', the other 43 were used as targets. Thus we are learning to match bicycles similar to the chosen template.

Initially, we used the SIFT landmarks and features as described in [6]. Since this approach typically identifies several hundred landmarks, we set $p = 20$ for this experiment (i.e. we consider the 20 most likely points). Since we cannot hope to get exact matches, we use the endpoint error instead of the normalised Hamming loss, i.e. we reward points which are close to the correct match.[9] Table 1 reveals that the performance of this method is quite poor, even with the higher-order model, and furthermore reveals no benefit from learning. This may be explained by the fact that although the SIFT features are invariant to scale and rotation, they are not invariant to reflection.

In [17], the authors report that the SIFT features *can* provide good matches in such cases, as long as landmarks are chosen which are locally invariant to affine transformations. They give a method for identifying affine-invariant feature points, whose SIFT features are then computed.[10] We achieve much better performance using this method, and also observe a significant improvement after learning. Figure 7 shows an example match using both the unary and higher-order techniques.

Finally, figure 6 (right) shows the weights learned for this model. Interestingly, the first-order term during the second stage of learning has almost zero weight. This must not be misinterpreted: during the second stage, the response of each of the 20 candidate points is so similar that the first-order features are simply unable to convey any new information – yet they are still very useful in determining the 20 candidate points.

[9] Here the endpoint error is just the average Euclidean distance from the correct label, scaled according to the width of the image.

[10] We used publicly available implementations of both methods.

Table 1: Performance on the 'bikes' dataset. The endpoint error is reported, with standard errors in parentheses (note that the second-last column, 'higher-order' uses the weights from the *first* stage of learning, but not the second).

Detector/descriptor		**unary**	**+ learning**	**higher-order**	**+ learning**
SIFT [6]	Training:	0.335 (0.038)	0.319 (0.034)	0.234 (0.047)	0.182 (0.031)
	Validation:	0.343 (0.027)	0.329 (0.019)	0.236 (0.031)	0.257 (0.033)
	Testing:	0.351 (0.024)	0.312 (0.015)	0.302 (0.045)	0.311 (0.039)
Affine invariant/SIFT [17]	Training:	0.322 (0.018)	0.280 (0.016)	0.233 (0.042)	0.244 (0.042)
	Validation:	0.337 (0.015)	0.298 (0.019)	0.245 (0.028)	0.229 (0.032)
	Testing:	0.332 (0.024)	0.339 (0.028)	0.277 (0.035)	**0.231 (0.034)**

5 Conclusion

We have presented a model for near-isometric shape matching which is robust to typical additional variations of the shape. This is achieved by performing structured learning in a graphical model that encodes features with several different types of invariances, so that we can directly learn a "compound invariance" instead of taking for granted the exclusive assumption of isometric invariance. Our experiments revealed that structured learning with a principled graphical model that encodes both the rigid shape as well as non-isometric variations gives substantial improvements, while still maintaining competitive performance in terms of running time.

Acknowledgements: We thank Marconi Barbosa and James Petterson for proofreading. NICTA is funded by the Australian Government's *Backing Australia's Ability* initiative, and the Australian Research Council's *ICT Centre of Excellence* program.

References

[1] Belongie, S., Malik, J., Puzicha, J.: Shape matching and object recognition using shape contexts. PAMI **24** (2002) 509–522

[2] Mori, G., Belongie, S., Malik, J.: Shape contexts enable efficient retrieval of similar shapes. In: CVPR. (2001) 723–730

[3] Mori, G., Malik, J.: Estimating human body configurations using shape context matching. In: ECCV. (2002) 666–680

[4] Frome, A., Huber, D., Kolluri, R., Bulow, T., Malik, J.: Recognizing objects in range data using regional point descriptors. In: ECCV. (2004)

[5] Belongie, S., Malik, J.: Matching with shape contexts. In: CBAIVL00. (2000) 20–26

[6] Lowe, D.G.: Object recognition from local scale-invariant features. In: ICCV. (1999) 1150–1157

[7] Felzenszwalb, P.F., Huttenlocher, D.P.: Pictorial structures for object recognition. IJCV **61** (2005) 55–79

[8] Felzenszwalb, P.F., Schwartz, J.D.: Hierarchical matching of deformable shapes. In: CVPR. (2007)

[9] LeCun, Y., Huang, F.J., Bottou, L.: Learning methods for generic object recognition with invariance to pose and lighting. CVPR (2004) 97–104

[10] Carmichael, O., Hebert, M.: Shape-based recognition of wiry objects. PAMI **26** (2004) 1537–1552

[11] McAuley, J.J., Caetano, T.S., Barbosa, M.S.: Graph rigidity, cyclic belief propagation and point pattern matching. PAMI **30** (2008) 2047–2054

[12] Caetano, T., Cheng, L., Le, Q., Smola, A.: Learning graph matching. In: ICCV. (2007) 1–8

[13] Canny, J.: A computational approach to edge detection. In: RCV. (1987) 184–203

[14] Smith, S.: A new class of corner finder. In: BMVC. (1992) 139–148

[15] Tsochantaridis, I., Hofmann, T., Joachims, T., Altun, Y.: Support vector machine learning for interdependent and structured output spaces. In: ICML. (2004)

[16] Teo, C., Le, Q., Smola, A., Vishwanathan, S.: A scalable modular convex solver for regularized risk minimization. In: KDD. (2007)

[17] Mikolajczyk, K., Schmid, C.: Scale and affine invariant interest point detectors. **60** (2004) 63–86

[18] Griffin, G., Holub, A., Perona, P.: Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology (2007)

Hierarchical Image-Region Labeling via Structured Learning

Julian McAuley[1,2]
julian.mcauley@nicta.com.au
Teofilo de Campos[1,3]
t.decampos@st-annes.oxon.org
Gabriela Csurka[1]
gabriela.csurka@xrce.xerox.com
Florent Perronnin[1]
florent.perronnin@xrce.xerox.com

[1] Xerox Research Centre Europe
Meylan, France
[2] Australian National University/NICTA
Canberra, Australia
[3] University of Surrey
Guildford, United Kingdom

(all authors were at Xerox at the time of this work)

Abstract

We present a graphical model which encodes a series of hierarchical constraints for classifying image regions at multiple scales. We show that inference in this model can be performed efficiently and exactly, rendering it amenable to structured learning. Rather than using feature vectors derived from images themselves, our model is parametrised using the outputs of a series of first-order classifiers. Thus our model learns which classifiers are useful at different scales, and also the relationships between classifiers at different scales. We present promising results on the VOC2007 and VOC2008 datasets.

1 Introduction

When classifying and segmenting images, some categories may be possible to identify based on global properties of the image, whereas others will depend on highly local information; many approaches deal with this problem by extracting features at multiple scales. However, segmentation based on local information can be highly noisy unless smoothness constraints are enforced. These two facts present a problem from a learning perspective: while it is possible to learn a first-order classifier (i.e., a classifier based on *local* information) which incorporates information from multiple scales, learning a classifier which enforces smoothness constraints is an example of *structured learning*, which appears to have made very little progress in this area due to the NP-hardness of many smoothness-enforcing algorithms.

In this paper, we will present a tree-structured graphical model that can be used to enforce smoothness constraints, while incorporating image features extracted at multiple scales. The tractability of this model will render it easily amenable to structured learning, which we believe is novel in this kind of segmentation scenario. We will define a loss based on approximate segmentations of an image (such as bounding boxes), allowing us to exploit the large amount of information in datasets such as VOC2007 and VOC2008 [7, 8]. We will use visual features from [6], which appear to exhibit state-of-the-art performance in classification problems, and show that their patch-level classification performance can be improved by structured learning.

The main contributions in our paper are as follows: firstly, in contrast to many patch-level segmentation schemes, inference in our model is efficient and exact, allowing us to circumvent the problems encountered when performing structured learning with approximate algorithms. Secondly, instead of parametrising our model using image features directly, our model is parametrised using the outputs of a series of first-order classifiers, and can therefore easily extend and combine existing first-order classification approaches.

2 Related literature

The idea of partitioning an image into regions at multiple scales is certainly not novel [11, 16]. However, these papers are solely concerned with global categorisation, whereas we also consider the problem of local region labeling. Furthermore, we shall not use the features of the image regions themselves, but rather our features are probability scores generated by existing first-order models, such as those from [6].

The approach of using tree-structured graphical models to incorporate global data into local segmentation problems (or simply to circumvent the problems associated with grid-structured models) is also not new; see for example [12]. However, to our knowledge, our approach is the first to apply structured-learning in this scenario, in order to improve the classification results of first-order classifiers. Similarly, others use information at an image-level to guide segmentation at the patch-level [23]. Other papers using similar approaches (though for different applications) include [9, 20, 25].

Many papers use lattice-structured Markov Random Fields (MRFs) to deal with the problem of patch-level segmentation. The unary and pairwise terms in such models may be similar to ours, i.e., the pairwise energy typically denotes a smoothness constraint. Energy minimization in such a model is NP-hard in the general case, so approximate forms of inference must be used [30]. There are many examples in this framework, though some important papers include [4] (α-*expansion,* $\alpha\beta$-*swap*), [22] (*normalised cuts*), and [17] (*log-cut*). [26] presents a comparative study of these ideas. We avoid such models as the need to perform approximate inference is of major concern from a learning perspective.[1] Other recent papers which apply learning to the problem of image segmentation/localisation include [2, 27].

Other authors also use grid structured models, but restrict their potential functions such that the energy minimization problem can be solved exactly. Examples include boolean-submodular functions [3, 15], *lattice*-submodular functions [14], and convex functions [13]. However, the energy functions we wish to use certainly do not fall into any of these categories.

3 Our model

The nodes ($\mathscr{X}$) in our graphical model ($\mathscr{M}$) are similar to the regions used in [11, 16] (though in those papers, they do not form a graph); these nodes are depicted in Figure 1, and will be indexed by $x_{l,(i,j)}$ where l is the node's level in the graphical model, and (i,j) is its grid position. In fact, a similar graphical model has been suggested for binary segmentation in [18].

[1] Some papers make an exception to this rule, such as [5] and [21]. Recently, some theoretical results have been obtained regarding the performance of structured learning in such cases [10].

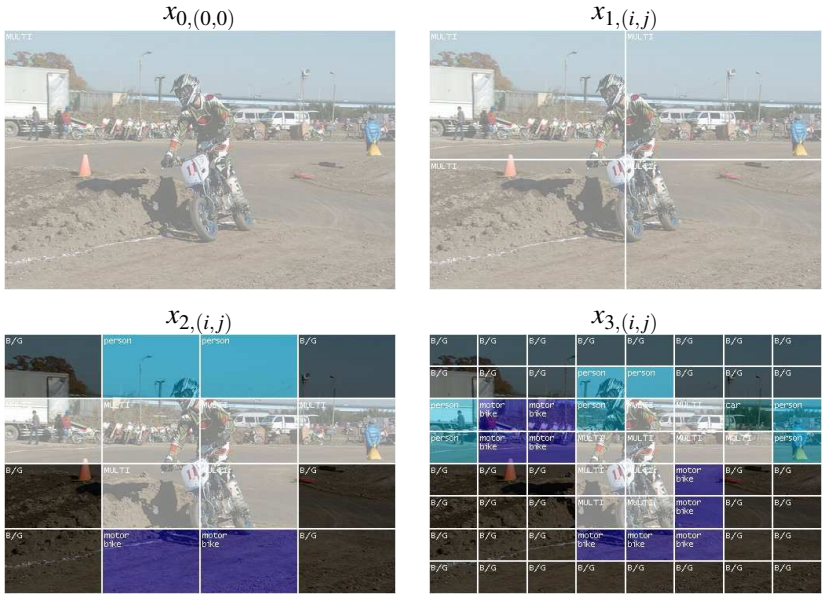


Figure 1: Nodes on the first four levels of our model. Nodes are indexed by $x_{l,(i,j)}$ where l is the node's level in the graphical model, and (i, j) is its grid position.

In words, a node on level k is connected to a node on level $k+1$, if and only if the regions corresponding to these nodes overlap. More formally,

$$x_{k,(i,j)} \text{ is connected to } \left\{x_{k+1,(2i,2j)}, x_{k+1,(2i,2j+1)}, x_{k+1,(2i+1,2j)}, x_{k+1,(2i+1,2j+1)}\right\} \quad (1)$$

(equivalently, $x_{k,(i,j)}$ is connected to $x_{k-1,(i/2,j/2)}$). Note that the graphical model is *undirected*. It is worth noting that there are no connections *between* nodes at a given level, and as such we are not enforcing neighbourhood constraints *per se*. Neighbourhood constraints will only be enforced indirectly, due to the fact that neighbors are connected via their parents.[2] Such a formulation is preferable because it results in a *tractable* graphical model (it forms a quadtree), whereas enforcing neighbourhood constraints directly typically requires that we resort to approximate forms of belief propagation.

The number of levels in this graph will depend on the size of the images in question, as well as how densely image patches are sampled. In practice, our graph is built to the maximal depth such that every region contains at least one patch, meaning that the 'regions' on the bottom level are essentially 'patches'. Details are given in Section 5.

3.1 Modeling hierarchical constraints in $\mathscr{M}$

When used in a classification scenario, this model will ensure that nodes on higher levels (i.e., nodes with smaller values of l) will always be assigned to at least as generic a class as nodes on lower levels; this is precisely analogous to the notion of *inheritance* in object-oriented programming. The simplest inheritance diagram (denoted $\mathscr{H}$) that we may use is depicted in Figure 2.

[2] In other words, we assume that two neighbors are *conditionally independent*, given their parents.

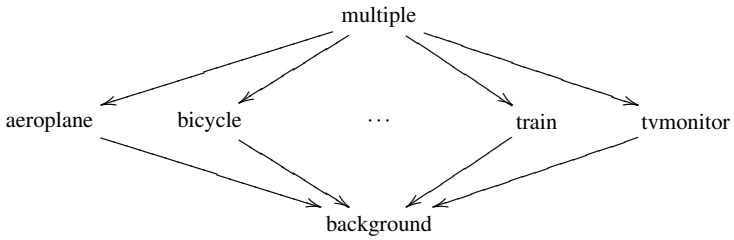


Figure 2: The simplest possible hierarchy; the most general possible assignment simply states that an image region may contain multiple classes; the least general states that a region does not contain any class ($\downarrow$ denotes 'more general than'). Class labels are taken from [7, 8], though any set of labels could be used.

Note that in $\mathscr{H}$, the class 'background' is actually a child of all specific classes; this is done because our training data may only approximately segment the image (i.e., using a bounding box). Thus an object that is classified as 'cat' on a higher level may be separated into 'cat' and 'background' on the level below. In principle, we could add additional classes to the hierarchy, representing 'clusters' of specific classes – for instance, tables and chairs may only appear in indoor scenes, whereas sheep and horses may only appear outdoors. The problem of learning such a visual hierarchy has been addressed in [24] (among others), and incorporating such hierarchies is certainly an avenue for future work.

These requirements will be enforced as *hard* constraints in $\mathscr{M}$ (i.e., assignments which disobey these constraints will have cost ∞). We will use the notation $a \prec b$ to indicate that a class a is *less specific* than b.

4 Probability maximization in $\mathscr{M}$

4.1 Unary potentials

In the most general case, the unary potentials (i.e., first order probabilities) in $\mathscr{M}$ are defined as follows (note that we have suppressed the subscript of the region x for brevity):

$$E(x;y) = \left\langle \Phi^1(x,y), \theta^{\text{nodes}} \right\rangle, \tag{2}$$

where $y \in \mathscr{H}$ is the class label to which the region x is to be assigned, and θ^{nodes} parametrises $\Phi^1(x,y)$ – the *joint feature map* of the node x and its assignment y. The optimal value of θ^{nodes} will be determined by our learning scheme, described in section 4.5.

In our specific case, we wish to ensure that the class label given to x was given a high probability according to some first-order classifier (such as that defined in [6]). Specifically, suppose that such a classifier returns a vector of probabilities P_x; then our joint feature map may be

$$\Phi^1(x,y) = \underbrace{(0,\ldots,P_{x,y},\ldots,0)}_{\text{0 everywhere except the } y^{\text{th}} \text{ entry}}. \tag{3}$$

In words, if x is assigned the class y, then the probability given by the first-order model should be high (weighted by θ^{nodes}_y).[3]

[3] To simplify, $E(x;y) = P_{x,y}\theta_y$. The formulation in (eq. 3) is used simply to express this as a linear function of

There are two straightforward generalisations of this model which may be desirable: firstly, we may wish to learn a separate parametrisation for each level; in this case, we would have a copy of $\Phi^1(x,y)$ for each image level, and use an indicator function to 'select' the current level. The second generalisation would be to parametrise *multiple* first-order classifiers, which return probabilities $P_x^1 \cdots P_x^C$ (for instance, features based on histograms of orientations; features based on RGB statistics, etc.). In this case, our joint feature map will simply be a concatenation of the individual feature maps defined by each classifier (i.e., it will be nonzero in exactly C locations). We will use both of these generalisations in our experiments (see Section 5).

4.2 Pairwise potentials

Similarly, we define pairwise potentials for nodes x_k and x_{k+1} (suppressing the remainder of the index):

$$E(x_k, x_{k+1}; y_k, y_{k+1}) = \left\langle \Phi^2(x_k, x_{k+1}; y_k, y_{k+1}), \theta^{\text{edges}} \right\rangle. \tag{4}$$

This time the joint feature map Φ^2 should express two properties: firstly, the constraints of our hierarchy should be strictly enforced; secondly, nodes assigned to the same class on different levels should have similar probabilities (again using the probabilities P_{x_k} and $P_{x_{k+1}}$ returned by our first-order classifier).

To achieve these goals, we define the indicator function H as

$$H(y_k, y_{k+1}) = \begin{cases} \infty & \text{if } y_k \succ y_{k+1}, \\ 0 & \text{if } y_k \prec y_{k+1}, \\ 1 & \text{otherwise } (y_k = y_{k+1}). \end{cases} \tag{5}$$

Note that this indicator function enforces precisely the hierarchical constraints that we desire. It also specifies that there is no cost associated to assigning a child node to a more specific class – thus we are only parametrising the cost when both class labels are the same. Our joint feature map now takes the form

$$\Phi^2(x_k, x_{k+1}; y_k, y_{k+1}) = -H(y_k, y_{k+1}) \left| P_{x_k} - P_{x_{k+1}} \right|^2, \tag{6}$$

where $|p|$ is the *elementwise* absolute value of p. Again we may make the same extensions to this model as outlined in Section 4.1.

4.3 The potential function

The complete maximization function is now defined as

$$g_\theta(\mathscr{X}) = \operatorname*{argmax}_{\mathscr{Y}} \sum_{x \in \mathscr{M}} \left\langle (0, \ldots, P_{x,y(x)}, \ldots, 0), \theta^{\text{nodes}} \right\rangle + \sum_{x_k, x_{k+1} \in \mathscr{M}} \left\langle H(y_k(x_k), y_{k+1}(x_{k+1})) \left| P_{x_k} - P_{x_{k+1}} \right|^2, \theta^{\text{edges}} \right\rangle, \tag{7}$$

where θ is simply the concatenation of our two parameter vectors $(\theta^{\text{nodes}}; \theta^{\text{edges}})$, and $y(x)$ is the assignment given to x under $\mathscr{Y}$ (the full set of labels). As the nodes in $\mathscr{M}$ form a tree, this energy can be maximized via *max-sum belief propagation* (see, for example [1]). The running time of this procedure is in $O(|\mathscr{M}||\mathscr{H}|^2)$, where $|\mathscr{M}|$ is the number of nodes and $|\mathscr{H}|$ is the number of classes.

θ^{nodes}. Also, we use $\log(P_{x,y})$ in practice, since we are maximizing a sum rather than a product.

4.4 Loss function

Our loss function specifies 'how bad' a given assignment $\mathscr{X}$ is compared to the correct assignment $\mathscr{Y}$. We desire that our loss function should decompose as a sum over nodes and edges in $\mathscr{M}$ (for reasons shown in Section 4.5).

Firstly, we must specify how our training labels $\mathscr{Y}$ are produced using existing datasets. One option is simply to assign the class 'multiple' to all regions with which multiple bounding boxes intersect, to assign 'background' to all regions with which *no* bounding boxes intersect, and to assign a specific class label to all others. Specifically, we will define a loss function of the form

$$\Delta(\mathscr{X},\mathscr{Y}) = \sum_{i=1}^{|\mathscr{X}|} \delta(x_i, y_i). \tag{8}$$

One such loss is the *Hamming loss*, which simply takes the value 0 when the region is correctly assigned, and $1/|\mathscr{M}|$ otherwise, where $|\mathscr{M}|$ is the number of regions (or more formally $\delta(x_i,y_i) = \frac{1}{|\mathscr{M}|}(1 - I_{\{x_i\}}(y_i))$.[4] In practice, we scale the loss so that each level of our graphical model makes an equal contribution (i.e., a mistake on level k makes four times the contribution as a mistake on level $k+1$).

4.5 Structured learning in $\mathscr{M}$

Structured learning can now be done in the framework described in [29]. Given a training set $\mathscr{Y}^1 \cdots \mathscr{Y}^N$, our goal is to solve

$$\operatorname*{argmin}_{\theta} \Big[\underbrace{\frac{1}{N}\sum_{n=1}^{N} \Delta(g_\theta(\mathscr{X}^n), \mathscr{Y}^n)}_{\text{empirical risk}} + \underbrace{\lambda \|\theta\|^2}_{\text{regularisation term}} \Big]. \tag{9}$$

Without going into the details of the structured learning algorithm itself (see [29]), we require that in addition to being able to solve (eq. 7), we can also solve

$$\begin{aligned} g_\theta(\mathscr{X}) = \operatorname*{argmax}_{\mathscr{Y}} & \sum_{x\in\mathscr{M}} \left\langle (0,\ldots,P_{x,y(x)},\ldots,0), \theta^{\text{nodes}} \right\rangle + \\ & \sum_{x_k,x_{k+1}\in\mathscr{M}} \left\langle H(y_k(x_k), y_{k+1}(x_{k+1})) \left| P_{x_k} - P_{x_{k+1}} \right|^2, \theta^{\text{edges}} \right\rangle + \Delta(\mathscr{X},\mathscr{Y}), \end{aligned} \tag{10}$$

i.e., for a given value of θ, we can find an assignment which is consistent with our model (eq. 7), yet incurs a high loss. This procedure is known as 'column-generation', and can easily be solved in this scenario, as long as $\Delta(\mathscr{X},\mathscr{Y})$ decomposes as a sum over the nodes and edges in our model (which is certainly true of the Hamming loss).

5 Experiments

We used images from the VOC2007 and VOC2008 datasets to evaluate our model. Specifically, we used VOC2008 for training and validation, and VOC2007 for testing (as testing

[4] When multiple classes are observed in a single region (i.e., when the correct label is 'multiple'), no penalty is incurred if one of these specific classes is chosen.

Correct labeling, using bounding-boxes from VOC2007 ($1-\Delta=1$):

Baseline (with image prior), using no second-order features ($1-\Delta=0.566$):

Non-learning using second-order features, but assigning equal weight to all ($1-\Delta=0.551$):

Learning of all features ($1-\Delta=0.770$):

Colour-code for labels observed in these images:

Figure 3: An example match comparing our technique to non-learning methods. The top sequence contains the 'correct' labeling, as extracted from the VOC2007 dataset (Image 000127; the correct labeling incurs zero loss by definition). The second sequence uses only first order features (i.e., the most likely assignment is chosen for each region independently); the image-level classifier is used at the top level, the mid-level classifier is used at the second and third levels, and the patch-level classifier is used at the bottom level; the fact that many of regions at the bottom level are incorrectly labeled demonstrates the need for consistency constraints (see supplementary material for further analysis). The third sequence shows our method without learning (i.e., assigning equal weight to all features); the low quality of this match demonstrates that the weights are poorly tuned without learning. Finally, the fourth sequence shows the performance of our method, which appears to address both of these issues. A key for the labels used is also shown.

	Level 0	Level 1	Level 2	Level 3
Baseline (see [6])	0.342	0.214	0.232	0.163
Baseline with image prior (see [6])	0.342	0.217	0.242	0.169
Non-learning	**0.426**	0.272	0.137	0.112
Learning	0.413	**0.307**	**0.349**	**0.444**

Table 1: Performance on each level of the graph (on the test set).

data for VOC2008 is not available). This presents a learning scenario in which there is a realistic difference between the training and test sets. The training, validation, and test sets contain 2113, 2227, and 4952 images respectively. The validation set is used to choose the optimal value of the regularisation constant λ.

We extracted SIFT-like features for our model on uniform grids at 5 different scales [19]. We used the methodology of [6] based on Fisher vectors to extract a signature for each region. A patch is considered to belong to a region if its centre belongs to that region, and its overlap with the region is at least 25%. We used three different first-order classifiers, based on sparse logistic regression: one which has been trained to classify the entire collection of features in an image (the 'image-level' classifier), one which has been trained on bounding-boxes (the 'mid-level' classifier), and one which has been trained on individual patches (the 'patch-level' classifier). The baseline to which we compare our method is one which simply selects the highest score using these individual classifiers (i.e., no consistency information is to be used). This baseline is similar to what is reported in [6], though it is important to stress that their method was not optimised to minimize the same loss that is presented here. We also report the performance using the image prior defined in [6], which rejects labelings at the patch level which are inconsistent with the probability scores at the image level.

Classification scores for the classes 'background' and 'multiple' were extracted automatically from the first-order scores: the probability of belonging to the background is 1 minus the highest probability of belonging to any other class; the probability of belonging to multiple classes is the twice the product of the two highest probabilities, capped at 1 (so that if two classes have probabilities greater than 0.5, the product will be greater than 0.5 also).

Structured learning was performed using the 'Bundle Methods for Risk Minimization' code of [28]. This solver requires only that we specify our feature representation $\Phi(\mathscr{X},\mathscr{Y})$ (eq. 3, 6), our loss $\Delta(\mathscr{X},\mathscr{Y})$ (eq. 8), and a column-generation procedure (eq. 10).

A performance comparison between the learning and non-learning versions of our approach, as well as the baseline is shown in Table 2.[5] Figure 3 shows an example match from our test set. Table 1 shows the contribution to the loss made by each level of the graphical model. Finally, Figure 4 shows the weight vector learned by our method. Note that our model exhibits a substantial improvement over the baseline, and non-learning approaches.[6]

Additional results are given in our supplementary material (including 3×3 branching in our tree, different weightings for the class 'multiple', and an analysis of our failure cases).

[5] The non-learning version of the approach just sets $\theta = (1,1,\ldots,1,1)$, though any constant value will do.

[6] Comparison with other methods is certainly difficult, as our loss is neither equivalent to a classification nor a segmentation error. Note however that in Table 1, we achieve an improvement at *both* the segmentation and classification levels. Also, due to the class 'multiple', our loss is *not* equivalent to the criteria normally used to measure performance on the VOC2007 and VOC2008 datasets.

	Training	Validation	Testing
Baseline (see [6])	0.272 (0.004)	0.273 (0.004)	0.233 (0.003)
Baseline with image prior (see [6])	0.275 (0.005)	0.276 (0.004)	0.238 (0.003)
Non-learning	0.235 (0.006)	0.224 (0.005)	0.233 (0.004)
Learning	**0.460** (**0.006**)	**0.456** (**0.006**)	**0.374** (**0.004**)

Table 2: Performance of our method during training, validation, and testing (for the optimal value of λ), compared to non-learning methods. The value reported is simply the proportion of correctly labeled regions, with each level contributing equally (i.e., one minus the loss). Values in parentheses indicate the standard error (across all images).

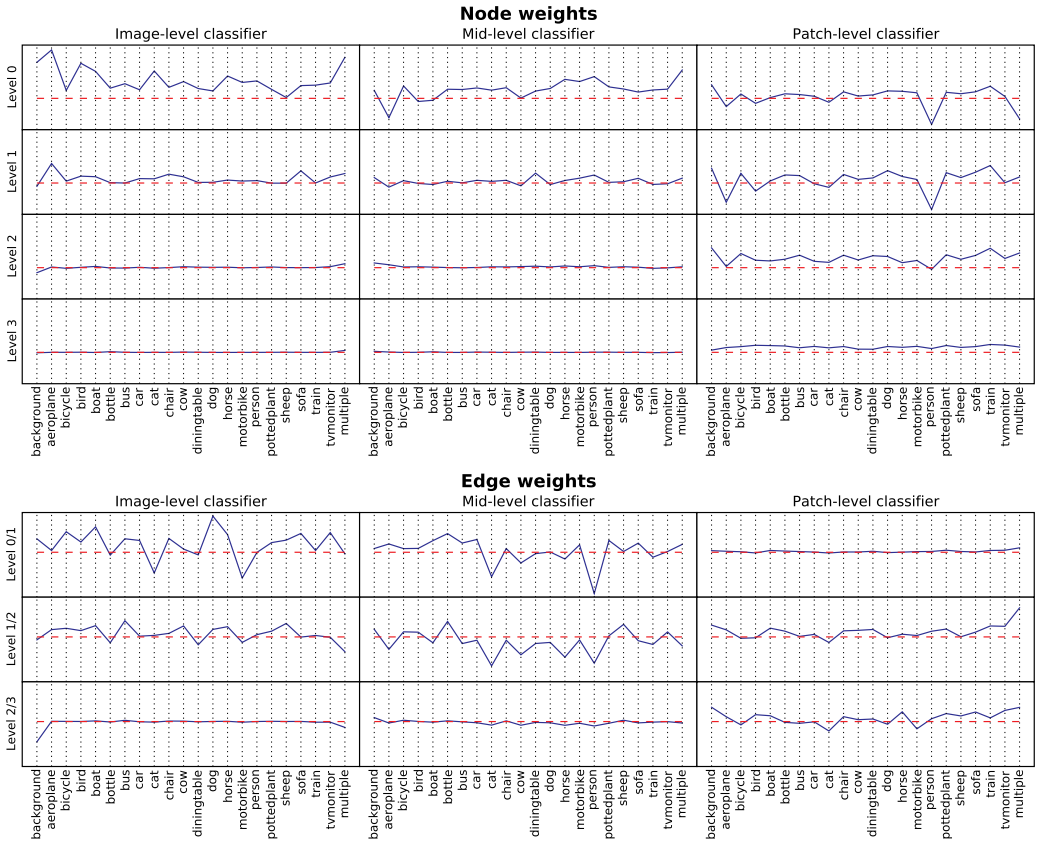


Figure 4: The complete weight-vector for our model. A separate vector of 22 ($= |\mathscr{H}|$) weights is learned for each image level, and for each type of classifier (the dashed line corresponds to zero). This vector has several interesting properties: firstly, the image-level classifier is given higher importance at the top levels, whereas the patch-level classifier is given higher importance at the bottom levels (which is consistent with our expectations). Also, there is a lot of variance between weights of different classes (especially 'multiple', 'background', and 'person'), indicating that certain classes are easier to identify at different scales. Finally, the edge weights have both large positive and negative scores: for instance, the high weight given to 'dog' for the image-level classifier at level 0/1 indicates that the features between these two levels should be very similar, whereas the *negative* weight given to 'cat' (at the same level) indicates that the features should be very *different*.

6 Conclusion

We have presented a graphical model which efficiently performs patch-level, region-level, and image-level labeling simultaneously. This model is useful in that it allows us to encode smoothness constraints for patch-level classification, while still incorporating the important information at higher levels. We have shown how to apply structured learning in this model, which has traditionally been a problem for models incorporating smoothness constraints. We have shown that our model improves in performance over existing models which use only first-order information. Since our model is parametrised using the probability scores from first-order approaches, it should be seen as complimentary to existing first-order techniques.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n^{o} 216529.

References

[1] S.M. Aji and R.J. McEliece. The generalized distributive law. *IEEE Trans. on Information Theory*, 46(2):325–343, 2000.

[2] Matthew B. Blaschko and Christoph H. Lampert. Learning to localize objects with structured output regression. In *ECCV*, 2008.

[3] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. on PAMI*, 26(9):1124–1137, 2004.

[4] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. on PAMI*, 23(11):1222–1239, 2001.

[5] T. Cour, N. Gogin, and J. Shi. Learning spectral graph segmentation. In *AISTATS*, 2005.

[6] G. Csurka and F. Perronnin. A simple high performance approach to semantic segmentation. In *BMVC*, 2008.

[7] M. Everingham, L. Van Gool, C.K.I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html, 2007.

[8] M. Everingham, L. Van Gool, C.K.I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2008 (VOC2008) Results. http://www.pascal-network.org/challenges/VOC/voc2008/workshop/index.html, 2008.

[9] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*, 2008.

[10] T. Finley and T. Joachims. Training structural SVMs when exact inference is intractable. In *ICML*, 2008.

[11] K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *ICCV*, 2005.

[12] X. He, R.S. Zemel, and M.Á. Carreira-Perpiñán. Multiscale conditional random fields for image labeling. *CVPR*, 2004.

[13] H. Ishikawa. Exact optimization for markov random fields with convex priors. *IEEE Trans. on PAMI*, 25(10):1333–1336, 2003.

[14] P. Kohli, A. Shekhovtsov, C. Rother, V. Kolmogorov, and P. Torr. On partial optimality in multi-label MRFs. In *ICML*, 2008.

[15] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? In *ECCV*, 2002.

[16] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[17] V. Lempitsky, C. Rother, and A. Blake. Logcut: Efficient graph cut optimization for markov random fields. In *ICCV*, 2007.

[18] J. Li, R.M. Gray, and R.A. Olshen. Multiresolution image classification by hierarchical modeling with two-dimensional hidden markov models. *IEEE Trans. on Information Theory*, 46(5):1826–1841, 2000.

[19] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[20] D. Ramanan and D.A. Forsyth. Finding and tracking people from the bottom up. *CVPR*, 2003.

[21] D. Scharstein and C. Pal. Learning conditional random fields for stereo. *CVPR*, 2007.

[22] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. on PAMI*, 22(8): 888–905, 2000.

[23] J. Shotton, J. Winn, C. Rother, and A. Criminisi. TextonBoost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *IJCV*, 81(1):2–23, 2009.

[24] J. Sivic, B.C. Russell, A. Zisserman, W.T. Freeman, and A.A. Efros. Unsupervised discovery of visual object class hierarchies. In *CVPR*, 2008.

[25] B. Stenger, A. Thayananthan, P. H. S. Torr, and R. Cipolla. Model-based hand tracking using a hierarchical bayesian filter. *IEEE Trans. on PAMI*, 28(9):1372–1384, 2006.

[26] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *IEEE Trans. on PAMI*, 30(6):1068–1080, 2008.

[27] Martin Szummer, Pushmeet Kohli, and Derek Hoiem. Learning crfs using graph cuts. In *ECCV*, 2008.

[28] C.H. Teo, Q. Le, A.J. Smola, and S.V.N. Vishwanathan. A scalable modular convex solver for regularized risk minimization. In *KDD*, 2007.

[29] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Predicting Structured Data*, pages 823–830, 2004.

[30] J.S. Yedidia, W.T. Freeman, and Y. Weiss. Generalized belief propagation. In *NIPS*, 2000.

Exploiting Within-Clique Factorizations in Junction-Tree Algorithms

Julian J. McAuley
NICTA - Australian National University

Tibério S. Caetano
NICTA - Australian National University

Abstract

We show that the expected computational complexity of the Junction-Tree Algorithm for *maximum a posteriori* inference in graphical models *can be improved*. Our results apply whenever the potentials over maximal cliques of the triangulated graph are factored over subcliques. This is common in many real applications, as we illustrate with several examples. The new algorithms are easily implemented, and experiments show substantial speed-ups over the classical Junction-Tree Algorithm. This enlarges the class of models for which exact inference is efficient.

1 INTRODUCTION

It is well-known that exact inference in *tree-structured* graphical models can be accomplished efficiently by message-passing operations following a simple protocol making use of the distributive law (Aji and McEliece, 2000). It is also well-known that exact inference in *arbitrary* graphical models can be solved by the Junction-Tree Algorithm; its efficiency is determined by the size of the maximal cliques after triangulation, a quantity related to the treewidth of the graph.

Figure 1 illustrates an attempt to apply the Junction-Tree Algorithm to some graphical models containing cycles. If the graphs are not chordal ((a) and (b)), they need to be triangulated, or made chordal (red edges in (c) and (d)). Their clique-graphs are then guaranteed to be *Junction-Trees*, and the distributive law can be applied with the same protocol used for trees (see Aji and McEliece (2000) for an excellent tutorial on exact inference in arbitrary graphs). Although the models in this example contain only pairwise factors, triangulation has increased the size of their maximal cliques, making exact inference substantially more expensive. Hence approximate solutions in the original graph (such as Loopy Belief-Propagation, or inference in a Loopy Factor-Graph) are often preferred over an exact solution via the Junction-Tree Algorithm.

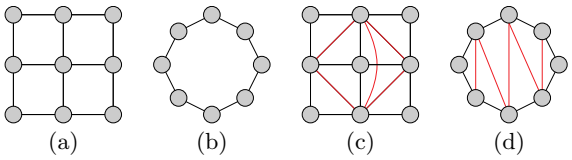


Figure 1: The models at left ((a) and (b)) can be triangulated ((c) and (d)) so that the Junction-Tree Algorithm can be applied. Despite the fact that the new models have larger maximal cliques, the corresponding potentials are still factored over pairs of nodes.

In this paper, we exploit the fact that the maximal cliques (after triangulation) often have potentials that factor over subcliques, as illustrated in Figure 1. We will show that whenever this is the case, the expected computational complexity of exact inference *can be improved* (both the asymptotic upper bound and the actual runtime). This will increase the class of problems for which exact inference is tractable.

This is not to be confused with optimizations produced by *Factor Graphs* (Kschischang et al., 2001). If applied to the above examples, the resulting Factor Graphs would contain cycles and would therefore produce inexact solutions in general. Instead, we work at the level of *Junction-Trees* arising from triangulated graphs, enabling us to leverage within-clique factorizations while performing *exact* inference.

A core operation encountered in the Junction-Tree Algorithm is that of finding the index that chooses the largest product amongst two lists of length N:

$$\hat{i} = \underset{i \in \{1 \dots N\}}{\operatorname{argmax}} \left\{ \mathbf{v}_a[i] \times \mathbf{v}_b[i] \right\}. \tag{1}$$

Our results stem from the realisation that while (eq. 1) appears to be a *linear* time operation, it can be decreased to $O(\sqrt{N})$ (in the expected case) if we know the permutations that sort $\mathbf{v}_a$ and $\mathbf{v}_b$.

Appearing in Proceedings of the 13^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2010, Chia Laguna Resort, Sardinia, Italy. Volume 9 of JMLR: W&CP 9.

1.1 SUMMARY OF RESULTS

A selection of the results to be presented in the remainder of this paper can be summarized as follows.

We are able to lower the asymptotic expected running time of the Junction-Tree Algorithm for *any* graphical model whose clique-potentials factorise into lower-order terms; we always obtain the same solution as the traditional Junction-Tree Algorithm, i.e., no approximations are used. For cliques composed of pairwise factors, we achieve an expected speed-up over the existing approach of *at least* $\Omega(\sqrt{N})$ (assuming N states per node); for cliques composed of K-ary factors, the expected speed-up becomes $\Omega(\frac{1}{K}N^{\frac{1}{K}})$ (Ω denotes an *asymptotic lower-bound*).

As an example, we can exactly compute the *maximum a posteriori* (MAP) states of a ring-structured model (see Fig. 1(b)) with M nodes in $O(MN^2\sqrt{N})$; in contrast, Loopy Belief-Propagation takes $\Theta(MN^2)$ *per iteration*, and the exact Junction-Tree Algorithm takes $\Theta(MN^3)$ by triangulating the graph (Θ denotes an *asymptotically tight bound*).

The expected-case improvement is achieved when the conditional densities of different factors (with respect to their shared variables) have independent order-statistics; if their order-statistics are positively correlated, we typically obtain better performance than the expected case; if they are negatively correlated, we may obtain worse performance, though our algorithm is never asymptotically more expensive than the traditional Junction-Tree Algorithm.

Our results do not apply for every semiring $S(+, \cdot)$, but only to those whose 'addition' operation defines an order; we also assume that under this ordering, our 'multiplication' operator satisfies

$$a < b \wedge c < d \Rightarrow a \cdot c < b \cdot d. \tag{2}$$

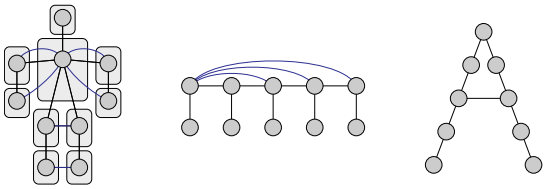
Thus our results certainly apply for the *max-product* and *min-product* semirings (as well as *max-sum* and *min-sum*), but not for *sum-product*. Consequently, our approach is useful for computing MAP-states, but cannot be used to compute marginal distributions. We also assume that the domain of each node is *discrete*.

2 BACKGROUND

In belief-propagation algorithms, the message from a clique X to an intersecting clique Y is defined by

$$m_{X\to Y}(\mathbf{x}_{X\cap Y}) = \max_{\mathbf{x}_{X\setminus Y}} \{\Phi_X(\mathbf{x}_X) \prod_{Z\in\Gamma(X)\setminus Y} m_{Z\to X}(\mathbf{x}_{X\cap Z})\} \tag{3}$$

(where $\Gamma(X)$ returns the neighbours of the clique X). If such messages are computed after Y has received messages from all of its neighbours except X (i.e., $\Gamma(X) \setminus Y$), then this defines precisely the update scheme used by the Junction-Tree Algorithm. The same update scheme is used for Loopy Belief-Propagation, though it is done iteratively in a randomized fashion. MAP-states are computed in a similar fashion, except that the messages from *all* neighbours are included in (eq. 3).



(a) (b) (c)

Figure 2: (a) A model for pose reconstruction from Sigal and Black (2006); (b) A 'skip-chain CRF' from Galley (2006); (c) A model for deformable matching from Coughlan and Ferreira (2002). Although the (triangulated) models have cliques of size three, their potentials factorize into pairwise terms.

Often, the clique-potential $\Phi_X(\mathbf{x}_X)$ will be decomposable into several smaller factors, i.e.,

$$\Phi_X(\mathbf{x}_X) = \prod_{F\subset X} \Phi_F(\mathbf{x}_F). \tag{4}$$

Some simple motivating examples are shown in Figure 2: a model for pose estimation from Sigal and Black (2006), a 'skip-chain CRF' from Galley (2006), and a model for shape matching from Coughlan and Ferreira (2002). In each case, the triangulated model has third-order cliques, but the potentials are only pairwise. Other examples have already been shown in Figure 1; analogous cases are ubiquitous in many real applications (to be shown in Section 4, Table 1).

The optimizations we suggest shall apply to general problems of the form

$$m_M(\mathbf{x}_M) = \max_{\mathbf{x}_{X\setminus M}} \prod_{F\subset X} \Phi_F(\mathbf{x}_F), \tag{5}$$

of which (eq. 3) is a special case (where the messages are considered to be factors). Computing the solution in the naïve way (i.e., evaluating $\prod_{F\subset X} \Phi_F(\mathbf{x}_F)$ for every value of $\mathbf{x}_X$) takes $\Theta(N^{|X|})$, where N is the number of states per node, and $|X|$ is the size of the clique X (we assume that for a given $\mathbf{x}_X$, computing $\prod_{F\subset X} \Phi_F(\mathbf{x}_F)$ takes constant time, as our optimisations shall not modify this cost). There is some loosely related work that applies to the *sum-product* version of this problem, based on arithmetic circuits (Park and Darwiche, 2003), an idea closely related to Strassen's sub-cubic method for matrix-multiplication.

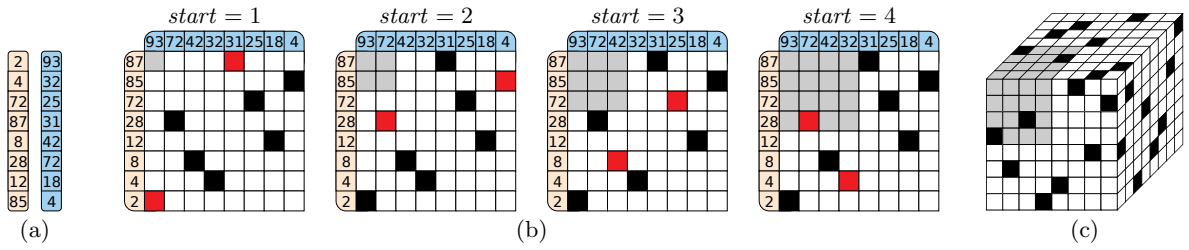


Figure 3: (a) The lists $\mathbf{v}_a$ and $\mathbf{v}_b$ before sorting. (b) Black squares show corresponding elements in the sorted lists ($\mathbf{v}_a[p_a[i]]$ and $\mathbf{v}_b[p_b[i]]$); red squares indicate the elements currently being read ($\mathbf{v}_a[p_a[start]]$ and $\mathbf{v}_b[p_b[start]]$). We can imagine expanding a gray box of size $start \times start$ until it contains an entry; note that the maximum is found during the first step. (c) In the version for three lists, we expand a gray box within a *cube*.

3 OUR APPROACH

To specify an efficient solution to (eq. 3), we first consider the simplest factorization: a clique of size three containing pairwise factors. Here we must compute

$$m_{i,j}(x_i, x_j) = \max_{x_k} \Phi_{i,j}(x_i, x_j)\Phi_{i,k}(x_i, x_k)\Phi_{j,k}(x_j, x_k). \tag{6}$$

For a particular value of $(x_i, x_j) = (a, b)$, we must solve

$$m_{i,j}(a, b) = \Phi_{i,j}(a, b) \times \max_{x_k} \underbrace{\Phi_{i,k}(a, x_k)}_{\mathbf{v}_a} \times \underbrace{\Phi_{j,k}(b, x_k)}_{\mathbf{v}_b}, \tag{7}$$

which we note is in precisely the form shown in (eq. 1).

This is sometimes referred to as 'funny' matrix multiplication, as (eq. 7) is equivalent to regular matrix multiplication with summation replaced by maximization. It is known to have a sub-cubic worst-case solution (Alon et al., 1997); our approach does not improve the worst-case complexity, but gives far better *expected-case* performance than existing solutions.

As we have previously suggested, it will be possible to solve (eq. 7) efficiently if $\mathbf{v}_a$ and $\mathbf{v}_b$ are already sorted. We note that $\mathbf{v}_a$ will be reused for every value of x_j, and likewise $\mathbf{v}_b$ will be reused for every value of x_i. Sorting every row of $\Phi_{i,k}$ and $\Phi_{j,k}$ can be done in $\Theta(N^2 \log N)$ (for $2N$ rows of length N).

The following elementary lemma is the key observation required in order to solve (eq. 7) efficiently:

Lemma 1. *If the p^{th} largest element of $\mathbf{v}_a$ has the same index as the q^{th} largest element of $\mathbf{v}_b$, then we only need to search through the p largest values of $\mathbf{v}_a$, **and** the q largest values of $\mathbf{v}_b$; any values smaller than these cannot possibly contain the largest solution.*

This observation is used to construct Algorithm 1. Here we iterate through the indices starting from the largest values of $\mathbf{v}_a$ and $\mathbf{v}_b$, and stopping once both indices are 'behind' the maximum value found so far (which we then know is the maximum). This algorithm is demonstrated pictorially in Figure 3.

Algorithm 1 Find i that maximizes $\mathbf{v}_a[i] \times \mathbf{v}_b[i]$

Input: two vectors $\mathbf{v}_a$ and $\mathbf{v}_b$, and permutation functions p_a and p_b that sort them in decreasing order (so that $\mathbf{v}_a[p_a[1]]$ is the largest element in $\mathbf{v}_a$)

1: **Initialize:** $start = 1$, $end_a = p_a^{-1}[p_b[1]]$, $end_b = p_b^{-1}[p_a[1]]$ {if $end_b = k$, the largest entry in $\mathbf{v}_a$ has the same index as the k^{th} largest entry in $\mathbf{v}_b$}
2: $best = p_a[1]$, $max = \mathbf{v}_a[best] \times \mathbf{v}_b[best]$
3: **if** $\mathbf{v}_a[p_b[1]] \times \mathbf{v}_b[p_b[1]] > max$ **then**
4: $best = p_b[1]$, $max = \mathbf{v}_a[best] \times \mathbf{v}_b[best]$
5: **end if**
6: **while** $start < end_a$ {we do not check the stopping criterion for end_b; this creates some redundancy, which a more complex implementation avoids} **do**
7: $start = start + 1$
8: **if** $\mathbf{v}_a[p_a[start]] \times \mathbf{v}_b[p_a[start]] > max$ **then**
9: $best = p_a[start]$
10: $max = \mathbf{v}_a[best] \times \mathbf{v}_b[best]$
11: **end if**
12: **if** $p_b^{-1}[p_a[start]] < end_b$ **then**
13: $end_b = p_b^{-1}[p_a[start]]$
14: **end if**
15: {repeat Lines 8–14, interchanging a and b}
16: **end while** {this takes *expected time* $O(\sqrt{N})$}
17: **Return:** *best*

A prescription of how Algorithm 1 can be used to solve (eq. 7) is given in Algorithm 2. Determining precisely the running time of Algorithm 1 (and therefore Algorithm 2) is not trivial, and will be explored in Section 3.2. We note that if the expected-case running time of Algorithm 1 is $O(f(N))$, then the time taken to solve Algorithm 2 shall be $O(N^2(\log N + f(N)))$. We will discuss the running time in Section 3.2, though for the moment we simply state the following theorem:

Theorem 2. *The **expected** running time of Algorithm 1 is $O(\sqrt{N})$, yielding a speed-up of at least $\Omega(\sqrt{N})$ in cliques containing pairwise factors.*

We can extend Algorithms 1 and 2 to cases where there are several overlapping terms in the factors. For in-

stance, Algorithm 2 can be adapted to solve

$$m_{i,j}(x_i, x_j) = \max_{x_k, x_m} \Phi_{i,j}(x_i, x_j) \times \Phi_{i,k,m}(x_i, x_k, x_m) \times \Phi_{j,k,m}(x_j, x_k, x_m), \quad (8)$$

and similar variants containing three factors. Here both x_k and x_m are shared by $\Phi_{i,k,m}$ and $\Phi_{j,k,m}$. As the number of shared terms increases, so does the improvement to the running time. While (eq. 8) would take $\Theta(N^4)$ to solve using the naïve algorithm, it takes only $O(N^3)$ to solve using Algorithm 2. In general, if we have S shared terms, we create a new variable whose domain is their product space; the running time is then $O(N^2\sqrt{N^S})$, yielding a speed-up of $\Omega(\sqrt{N^S})$. over the naïve solution.

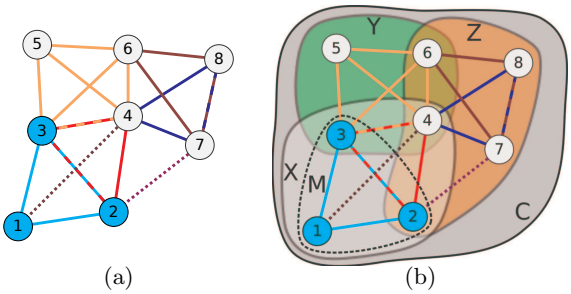
Algorithm 2 Compute the max-marginal of a 3-clique containing pairwise factors, using Algorithm 1

Input: a potential function $\Phi_{i,j,k}(x_i, x_j, x_k)$ with factors $\Phi_{i,j,k}(a, b, c) = \Phi_{i,j}(a, b) \times \Phi_{i,k}(a, c) \times \Phi_{j,k}(b, c)$ whose max-marginal $m_{i,j}$ we wish to compute

1: **for** $n \in \{1 \dots N\}$ **do**
2: compute $\mathbf{P}^i[n]$ by sorting $\Phi_{i,k}(n, x_k)$ {takes $\Theta(N \log N)$}
3: compute $\mathbf{P}^j[n]$ by sorting $\Phi_{j,k}(n, x_k)$ {$\mathbf{P}^i$ and $\mathbf{P}^j$ are $N \times N$ arrays, each row of which is a permutation; $\Phi_{i,k}(n, x_k)$ and $\Phi_{j,k}(n, x_k)$ are functions over x_k, since n is constant in this expression}
4: **end for** {this loop takes $\Theta(N^2 \log N)$}
5: **for** $(a, b) \in \{1 \dots N\}^2$ **do**
6: $(\mathbf{v}_a, \mathbf{v}_b) = (\Phi_{i,k}(a, x_k), \Phi_{j,k}(b, x_k))$
7: $(p_a, p_b) = (\mathbf{P}^i[a], \mathbf{P}^j[b])$
8: $best = Algorithm1\ (\mathbf{v}_a, \mathbf{v}_b, p_a, p_b)$ $\{O(\sqrt{N})\}$
9: $m_{i,j}(a, b) = \Phi_{i,j}(a, b)\Phi_{i,k}(a, best)\Phi_{j,k}(b, best)$
10: **end for** {this loop takes $O(N^2\sqrt{N})$}
11: **Return:** $m_{i,j}$

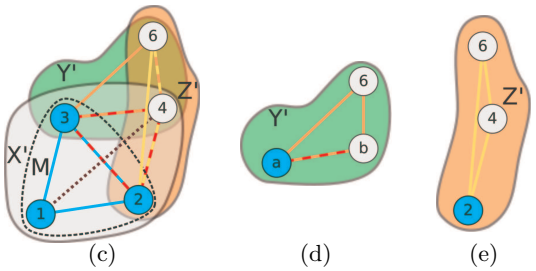
3.1 AN EXTENSION TO CLIQUES WITH ARBITRARY DECOMPOSITIONS

By similar reasoning, we can apply our algorithm in cases where there are more than three factors. We begin with the simplest case, in which our factors can be separated into three *groups* (which we note is always the case for pairwise factors). An illustrative example of such a clique is given in Figure 4(a), which we shall call G (assumed to be maximal in some triangulated graph). Each of the factors in this clique have been labeled using differently coloured edges, and the max-marginal we wish to compute has been labeled using coloured nodes. It is possible to split this graph into three groups X, Y, and Z, such that every factor is contained within a single group, along with the max-marginal we wish to compute.



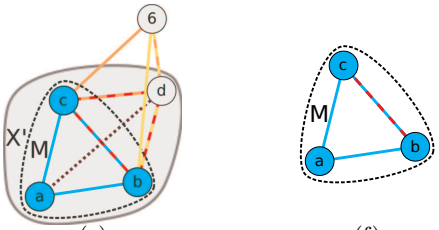
(a) (b)

(a) We begin with a set of factors (each coloured clique is a factor; dashed lines indicate overlapping factors, while dotted lines indicate pairwise factors), which are assumed to belong to some clique in our model; we wish to compute the max-marginal with respect to one of those factors (indicated using coloured nodes); (b) The factors are split into three groups, such that every factor is entirely contained within one of them (Algorithm 3, line 1).



(c) (d) (e)

(c) Any nodes contained in only one of the groups are marginalised (Algorithm 3, lines 2, 3, and 4); the problem is now very similar to that described in Algorithm 2, except that *nodes* have been replaced by *groups*; note that this essentially introduces maximal factors in Y' and Z'; (d) For every value $(a, b) \in \text{dom}(x_3, x_4)$, $\Psi^Y(a, b, x_6)$ is sorted (Algorithm 3, lines 5–7); (e) For every value $(a, b) \in \text{dom}(x_2, x_4)$, $\Psi^Z(a, b, x_6)$ is sorted (Algorithm 3, lines 8–10).



(e) (f)

(e) For every $\mathbf{n} \in \text{dom}(X')$, we choose the best value of x_6 by Algorithm 1 (Algorithm 3, lines 11–16); (f) The result is marginalised with respect to M (Algorithm 3, line 17).

Figure 4: Algorithm 3, explained pictorially. In this case, the most computationally intensive step is the marginalisation of Z (in step (c)), which takes $\Theta(N^5)$. However, the algorithm can actually be applied *recursively* to the group Z, resulting in an overall running time of $O(N^4\sqrt{N})$, for a max-marginal that would have taken $\Theta(N^8)$ to compute using the naïve solution.

Algorithm 3 Compute the max-marginal of G with respect to M, where G is split into three groups

Input: potentials $\Phi_G(\mathbf{x}) = \Phi_X(\mathbf{x}_X)\Phi_Y(\mathbf{x}_Y)\Phi_Z(\mathbf{x}_Z)$, where $M \subseteq X$ (see Fig. 4)

1: **Define:** $X' = ((Y \cup Z) \cap X) \cup M$; $Y' = (X \cup Z) \cap Y$; $Z' = (X \cup Y) \cap Z$ {X' contains the variables in X that are shared by at least one other group; alternately, the variables in $X \setminus X'$ appear only in X (sim. for Y' and Z')}
2: compute $\Psi^X(\mathbf{x}_{X'}) = \max_{X \setminus X'} \Phi_X(\mathbf{x}_X)$ {we are marginalising over those variables in X that do not appear in any of the other groups (or in M); this takes $\Theta(N^{|X|})$ if done by brute force, but may also be done recursively}
3: compute $\Psi^Y(\mathbf{x}_{Y'}) = \max_{Y \setminus Y'} \Phi_Y(\mathbf{x}_Y)$ {$\Theta(N^{|Y|})$}
4: compute $\Psi^Z(\mathbf{x}_{Z'}) = \max_{Z \setminus Z'} \Phi_Z(\mathbf{x}_Z)$ {$\Theta(N^{|Z|})$}
5: **for** $\mathbf{n} \in \mathrm{dom}(X \cap Y)$ **do**
6: compute $\mathbf{P}^Y[\mathbf{n}]$ by sorting $\Psi^Y(\mathbf{n}; \mathbf{x}_{Y' \setminus X})$ {$\Psi^Y(\mathbf{n}; \mathbf{x}_{Y' \setminus X})$ is free over $\mathbf{x}_{Y' \setminus X}$; $\mathbf{P}^Y[\mathbf{n}]$ stores the $|Y' \setminus X|$-dimensional indices that sort it}
7: **end for** {this loop takes $\Theta(|Y' \setminus X| N^{|Y'|} \log N)$}
8: **for** $\mathbf{n} \in \mathrm{dom}(X \cap Z)$ **do**
9: compute $\mathbf{P}^Z[\mathbf{n}]$ by sorting $\Psi^Z(\mathbf{n}; \mathbf{x}_{Z' \setminus X})$
10: **end for** {this loop takes $\Theta(|Z' \setminus X| N^{|Z'|} \log N)$}
11: **for** $\mathbf{n} \in \mathrm{dom}(X')$ **do**
12: $(\mathbf{v}_a, \mathbf{v}_b) = (\Psi^Y(\mathbf{n}|_{Y'}; \mathbf{x}_{Y' \setminus X'}), \Psi^Z(\mathbf{n}|_{Z'}; \mathbf{x}_{Z' \setminus X'}))$ {$\mathbf{n}|_{Y'}$ is the 'restriction' of the vector $\mathbf{n}$ to those indices in Y'; hence $\Psi^Y(\mathbf{n}|_{Y'}; \mathbf{x}_{Y' \setminus X'})$ is free in $\mathbf{x}_{Y' \setminus X'}$, while $\mathbf{n}|_{Y'}$ is fixed}
13: $(p_a, p_b) = \left(\mathbf{P}^Y[\mathbf{n}|_{Y'}], \mathbf{P}^Z[\mathbf{n}|_{Z'}]\right)$
14: *best* = *Algorithm1* $(\mathbf{v}_a, \mathbf{v}_b, p_a, p_b)$
15: $m_X(\mathbf{n}) = \Psi^X(\mathbf{n})\Psi^Y(best; \mathbf{n}|_{Y'})\Psi^Z(best; \mathbf{n}|_{Z'})$
16: **end for** {this loop takes $O(N^{|X'|}\sqrt{N^{|(Y' \cap Z') \setminus X'|}})$}
17: $m_M(\mathbf{x}_M) = Naive(m_X, M)$ {i.e., we are using the naïve algorithm to marginalise $m_X(\mathbf{x}_X)$ with respect to M; this takes $\Theta(N^{|X|})$}

The marginalisation steps of Algorithm 3 (Lines 2, 3, and 4) may further decompose into smaller groups, in which case Algorithm 3 can be applied recursively. For instance, the graph in Figure 5(a) shows the marginalisation step from Algorithm 3, Line 4 (see Fig. 4(c)). Since this marginalisation step is the asymptotically dominant step in the algorithm, applying Algorithm 3 recursively lowers the asymptotic complexity.

Naturally, there are cases for which a decomposition into three terms is not possible, such as

$$m_{i,j,k}(x_i, x_j, x_k) = \max_{x_m} \Phi_{i,j,k}(x_i, x_j, x_k) \times \Phi_{i,j,m}(x_i, x_j, x_m)\Phi_{i,k,m}(x_i, x_k, x_m)\Phi_{j,k,m}(x_j, x_k, x_m) \tag{9}$$

(i.e., a clique of size four with third-order factors). However, if the model contains factors of size K, it must always be possible to split it into $K + 1$ groups (e.g. four in the case of (eq. 9)).

Our optimizations can be applied in these cases simply by adapting Algorithm 1 to solve problems of the form

$$\hat{i} = \underset{i \in \{1 \dots N\}}{\mathrm{argmax}} \{\mathbf{v}_1[i] \times \mathbf{v}_2[i] \times \cdots \times \mathbf{v}_K[i]\}. \tag{10}$$

Figure 3(c) demonstrates how such an algorithm behaves in practice: if we have K lists, the cube in Figure 3(c) becomes a K-dimensional hypercube. Pseudocode is not shown, though it is similar to Algorithm 1. Again, we shall discuss the running time of this extension in Section 3.2. For the moment, we state the following theorem:

Theorem 3. *Algorithm 1 generalises to K lists with an expected running time of $O(KN^{\frac{K-1}{K}})$, yielding a speed-up of $\Omega(\frac{1}{K}N^{\frac{1}{K}})$ in cliques containing K-ary factors (it can be adapted to be $O(\min(N, KN^{\frac{K-1}{K}}))$, if we carefully avoid rereading entries).*

Using this extension, we can extend Algorithm 3 to allow for any number of *groups* (pseudocode is not shown; all statements about the groups Y and Z simply become statements about K groups $\{G_1 \dots G_K\}$). The one remaining case that has not been considered is when the sequences $\mathbf{v}_1 \cdots \mathbf{v}_K$ are functions of different (but overlapping) variables; this can be trivially circumvented by 'padding' each of them to be functions of the same variables, and by carefully applying recursion.

As a final comment we note that we have not provided an algorithm for choosing how to split the variables into $(K + 1)$-groups, and we note that different splits may result in better performance. However, even if the split is chosen in a naïve way, we will still get the performance increases mentioned.

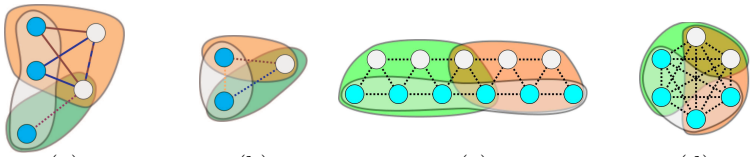
3.2 EXPECTED-CASE COMPLEXITY

In this section we shall determine the expected-case running time of Algorithm 1. Algorithm 1 traverses $\mathbf{v}_a$ and $\mathbf{v}_b$ until it reaches the smallest value of m for which there is some $j \leq m$ such that $m \geq p_b^{-1}[p_a[j]]$. Extensions of Algorithm 1 aim to find the smallest m for which

$$\max(i, p_1[i], \dots, p_{K-1}[i]) \leq m. \tag{11}$$

If M is a random variable representing this smallest value of m, then we wish to find $E(M)$. Simple analysis reveals that the probability of choosing a single permutation p such that $\max(i, p[i]) > m$ is

$$P(M > m) = \frac{(N-m)!(N-m)!}{(N-2m)!N!}. \tag{12}$$



	(a)	(b)	(c)	(d)	(e)
Graph:					{The complete graph K_M, with pairwise terms}
Naïve solution:	$\Theta(N^5)$	$\Theta(N^3)$	$\Theta(N^{11})$	$\Theta(N^6)$	$\Theta(N^M)$
Algorithm 3:	$O(N^3\sqrt{N})$	$O(N^2\sqrt{N})$	$O(N^6\sqrt{N})$	$O(N^5)$	$O(N^{5M/6})$
Speed-up:	$\Omega(N\sqrt{N})$	$\Omega(\sqrt{N})$	$\Omega(N^4\sqrt{N})$	$\Omega(N)$	$\Omega(N^{M/6})$

Figure 5: Some example graphs whose max-marginals are to be computed with respect to the coloured nodes, using the three regions shown. Factors are indicated using differently coloured edges, while dotted edges always indicate pairwise factors. (a) is the region Z from Figure 4 (recursion is applied *again* to achieve this result); (b) is the graph used to motivate Algorithm 2; (c) shows a query in a graph with regular structure; (d) shows a complete graph with six nodes; (e) generalises this to a clique with M nodes.

This is precisely $1 - F(m)$, where $F(m)$ is the cumulative density function of M. It is immediately clear that $1 \leq M \leq \lfloor N/2 \rfloor + 1$, which defines the best and worst-case performance of Algorithm 1.

Using the identity $E(X) = \sum_{x=1}^{\infty} P(X \geq x)$, we can write down a formula for the expected value of M

$$E(M) = \sum_{m=0}^{\lfloor N/2 \rfloor} \frac{(N-m)!(N-m)!}{(N-2m)!N!}, \qquad (13)$$

which reflects the expected running time of Algorithm 1. Unfortunately, the corresponding expectation when we have $K-1$ permutations is not trivial to compute. It is possible to write down a formula that generalizes (eq. 12), though the expression is complicated and not very informative; hence we shall instead rely on the upper bounds mentioned in Theorems 2 and 3. Proofs of these bounds are given in Appendix A.

4 EXISTING APPLICATIONS

Our results are immediately compatible with several applications that rely on inference in graphical models. As we have mentioned, our results apply to *any model whose cliques decompose into lower-order terms.*

Often, potentials are defined only on *nodes* and *edges* of a model. A D^{th}-order Markov model has a tree-width of D, but in some cases contains only pairwise relationships. Similarly 'skip-chain CRFs' (Sutton and McCallum, 2006; Galley, 2006), and Junction-Trees used in SLAM applications (Paskin, 2003) often contain only pairwise terms. In each case, if the tree-width is D, Algorithm 3 takes $O(MN^D\sqrt{N})$ (for a model with M nodes and N states per node), yielding a speed-up of $\Omega(\sqrt{N})$.

Models for shape matching often exhibit similar properties (Sigal and Black, 2006). Third-order cliques factorise into second order terms, resulting in a speed-up of $\Omega(\sqrt{N})$. Another similar model for shape matching is that of Felzenszwalb (2005); this model again contains third-order cliques, though it includes a 'geometric' term constraining all three variables. However, the third-order term is *independent of the input data*, meaning that each of its rows can be sorted *offline*. Here we have an instance of Algorithm 1 with three lists, yielding a speed-up of $\Omega(N^{\frac{1}{3}})$.

In Coughlan and Ferreira (2002), deformable shape-matching is solved approximately using Loopy Belief-Propagation. Their model has only second-order cliques, meaning that inference takes $\Theta(MN^2)$ *per iteration*. Although we cannot improve upon this result, we note that we can typically do *exact* inference in a single iteration in $O(MN^2\sqrt{N})$; thus our model has the same running time as $O(\sqrt{N})$ iterations of the original version. This result applies to all models containing a single loop.

In McAuley et al. (2008), a model is presented for graph-matching using Loopy Belief-Propagation; the maximal cliques for D-dimensional matching have size $(D+1)$, meaning that inference takes $\Theta(MN^{D+1})$ *per iteration* (it is shown to converge to the correct solution); we improve this to $O(MN^D\sqrt{N})$.

Belief-propagation can be used to compute *LP-relaxations* in pairwise graphical models. In Sontag et al. (2008), LP-relaxations are computed for pairwise models by constructing several third-order 'clusters', which compute pairwise messages for each of their edges.

Table 1 summarizes these results. Running times reflect the *expected case*, assuming that *max-product belief-propagation is used in a discrete model.* Some of the referenced articles may suggest variants of the algorithm (e.g. Gaussian models, or approximate schemes); we believe that our approach may revive the exact, discrete version as a tractable option in such cases.

Table 1: Some existing work to which our results can potentially be applied (M nodes, N states per node).

REFERENCE	APPLICATION	RUNNING TIME	OUR APPROACH
McAuley et al. (2008)	D-d graph-matching	$\Theta(MN^{D+1})$, iterative	$O(MN^D\sqrt{N})$, iterative
Sutton and McCallum (2006)	Width-D skip-chain	$O(MN^{D+1})$	$O(MN^D\sqrt{N})$
Paskin (2003) (discrete case)	SLAM, width D	$O(MN^{D+1})$	$O(MN^D\sqrt{N})$
Felzenszwalb (2005)	Deformable matching	$\Theta(MN^3)$	$\Theta(MN^{\frac{8}{3}})$ + offline steps
Coughlan and Ferreira (2002)	Deformable matching	$\Theta(MN^2)$, iterative	$O(MN^2\sqrt{N})$
Sigal and Black (2006)	Pose reconstruction	$\Theta(MN^3)$	$O(MN^2\sqrt{N})$
Sontag et al. (2008)	LP with M clusters	$\Theta(MN^3)$	$O(MN^2\sqrt{N})$

5 EXPERIMENTS

5.1 PERFORMANCE AND BOUNDS

For our first experiment, we compare the performance of Algorithm 1 (and extensions) to the naïve solution. This is a core subroutine of each of the other algorithms, meaning that determining its performance shall give us an indication of the improvements we expect to obtain in real graphical models.

For each experiment, we generate N i.i.d. samples from $[0, 1)$ to obtain the lists $v_1 \dots v_K$. N is the domain size; this may refer to a single node, or a *group* of nodes; thus large values of N may appear even for binary-valued models. K is the number of lists in (eq. 10); we can observe this number of lists only if we are working in cliques of size $K+1$, and then only if the factors are of size K; therefore smaller values of K are probably more realistic in practice (indeed, all but one of the applications in Section 4 had $K = 2$).

The performance of our algorithm is shown in Figure 6 (left), for $K \in \{2, 3, 4\}$. The performance reported is just the number of elements read from the lists. This is compared to N itself, which is the number of elements read by the naïve version. The upper-bounds from Section A are also reported, while the expected performance (i.e., (eq. 13)) is reported for $K = 2$ (we are not aware of an efficiently computable generalisation of (eq. 13) for $K > 2$).

5.2 CORRELATED VARIABLES

The expected case running time of our algorithm was obtained under the assumption that the lists had independent order-statistics, as was the case for the previous experiment. We suggested that we will typically obtain worse performance in the case of negatively correlated variables, and better performance in the case of positively correlated variables; we will assess these claims in this experiment.

We report the performance for two lists (i.e., for Algorithm 1), whose values are sampled from a 2-dimensional Gaussian, with covariance matrix

$$\Sigma = \begin{bmatrix} 1 & c \\ c & 1 \end{bmatrix}, \tag{14}$$

meaning that the two lists are correlated with correlation coefficient c. Performance is shown in Figure 6 (centre) for different values of c.

5.3 2-D GRAPH MATCHING

Naturally, Algorithm 3 has additional overhead compared to the naïve solution, meaning that it will not be beneficial for small N. We reproduce the model from McAuley et al. (2008), which performs 2-dimensional graph matching, using a loopy graph with cliques of size three, containing only second order potentials (as described in Section 4); the $\Theta(NM^3)$ performance of their method is reportedly state-of-the-art.

We perform matching between a *template* graph with M nodes, and a *target* graph with N nodes, which requires a graphical model with M nodes and N states per node (see McAuley et al., 2008). We fix $M = 5$ and vary N. Performance is shown in Figure 6 (right). The running times appear to be comparable after only $N \simeq 6$, meaning that our algorithm has a speed-up over the solution of McAuley et al. (2008) of about $\frac{2}{5}\sqrt{N}$; thus it is significantly faster than the state-of-the-art solution, even for small values of N. Plots of $t = \frac{N^3}{4000}$ and $t/\frac{2\sqrt{N}}{5}$ are overlayed on Figure 6 (right) to estimate the runtime in seconds as a function of N.

6 CONCLUSION

We have presented a series of approaches that allow us to improve the performance of the Junction-Tree Algorithm for models that factorize into terms smaller than their maximal cliques. We are able to improve the expected computational complexity in models whose cliques factorize, no matter the size or number of factors. Our results increase the class of models for which exact inference remains a tractable option.

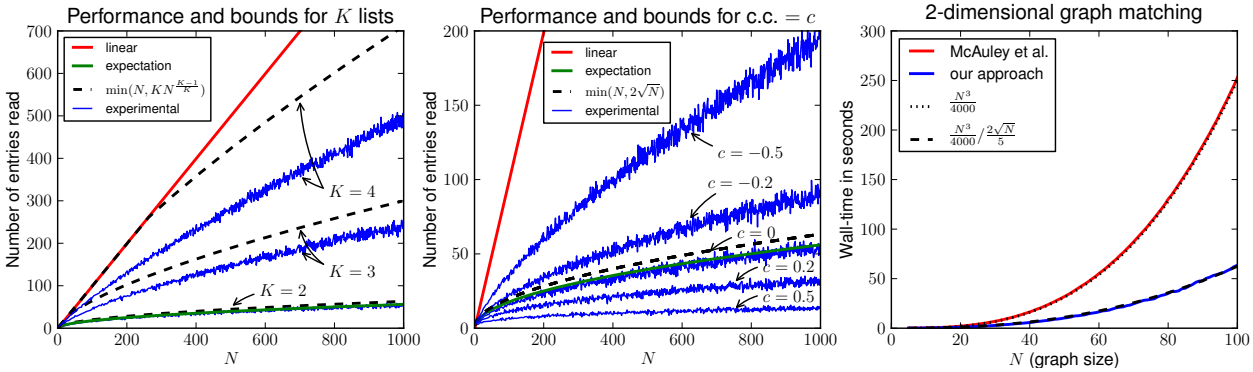


Figure 6: Left: Performance of our algorithm over 100 trials; the dotted lines show the bounds from Section A. Centre: Performance of our algorithm for different correlation coefficients. Right: The running time of our method on a graph matching experiment over 10 trials.

References

Aji, S. M. and McEliece, R. J. (2000). The generalized distributive law. *IEEE Trans. on Information Theory*, 46(2):325–343.

Alon, N., Galil, Z., and Margalit, O. (1997). On the exponent of the all pairs shortest path problem. *J. Comput. Syst. Sci.*, 54(2):255–262.

Coughlan, J. M. and Ferreira, S. J. (2002). Finding deformable shapes using loopy belief propagation. In *ECCV*.

Felzenszwalb, P. F. (2005). Representation and detection of deformable shapes. *IEEE Trans. on PAMI*, 27(2):208–220.

Galley, M. (2006). A skip-chain conditional random field for ranking meeting utterances by importance. In *EMNLP*.

Kschischang, F. R., Frey, B. J., and Loeliger, H. A. (2001). Factor graphs and the sum-product algorithm. *IEEE Trans. on Information Theory*, 47(2):498–519.

McAuley, J. J., Caetano, T. S., and Barbosa, M. S. (2008). Graph rigidity, cyclic belief propagation and point pattern matching. *IEEE Trans. on PAMI*, 30(11):2047–2054.

Park, J. D. and Darwiche, A. (2003). A differential semantics for jointree algorithms. In *NIPS*.

Paskin, M. A. (2003). Thin junction tree filters for simultaneous localization and mapping. In *IJCAI*.

Sigal, L. and Black, M. J. (2006). Predicting 3d people from 2d pictures. In *AMDO*.

Sontag, D., Meltzer, T., Globerson, A., Jaakkola, T., and Weiss, Y. (2008). Tightening LP relaxations for MAP using message passing. In *UAI*.

Sutton, C. and McCallum, A. (2006). *An Introduction to Conditional Random Fields for Relational Learning*.

A BOUNDS AND PROOFS

Proof of Theorem 3 (sketch). We wish to determine the expected value of the smallest m satisfying (eq. 11). It can be shown that replacing the *permutations* in (eq. 11) with *random samples* of the values from 1 to N gives an *upper bound* on the expected value. This allows us to compute an upper bound on (eq. 12):

$$P(M > m) \leq \left(1 - \frac{m}{N}\right)^m, \tag{15}$$

and the corresponding version for K lists:

$$P^K(M > m) \leq \left(1 - \frac{m^{K-1}}{N^{K-1}}\right)^m. \tag{16}$$

In order to claim that the $E(M)$ is $O(f(N,K))$, (for K lists with N elements) it is sufficient to show that

$$\sum_{m=0}^{\infty} \left(1 - \frac{f(N,K)^{K-1}}{N^{K-1}}\right)^m \in O(f(N,K)). \tag{17}$$

Evaluating this geometric progression, we see that $f(N,K) = N^{\frac{K-1}{K}}$ is a suitable choice. At each step, K entries are read, resulting in the $O(KN^{\frac{K-1}{K}})$ time reported. □

Theorem 2 is trivially proved as a special case of Theorem 3. To summarize, the expected running time of Algorithm 1 (for which we have $K = 2$ lists) is $O(\sqrt{N})$. This algorithm can be extended to handle K lists, with running time $O(KN^{\frac{K-1}{K}})$.

Acknowledgements

We would like to thank Pedro Felzenszwalb and Johnicholas Hines for comments on the first draft. NICTA is funded by the Australian Government's *Backing Australia's Ability* initiative, and the Australian Research Council's *ICT Centre of Excellence* program.

Unified Graph Matching in Euclidean Spaces

Julian J. McAuley
NICTA/ANU
Canberra, ACT 0200
julian.mcauley@nicta.com.au

Teófilo de Campos*
University of Surrey
Guildford, GU2 7XH, UK
t.decampos@st-annes.oxon.org

Tibério S. Caetano
NICTA/ANU
Canberra, ACT 0200
tiberio.caetano@nicta.com.au

Abstract

Graph matching is a classical problem in pattern recognition with many applications, particularly when the graphs are embedded in Euclidean spaces, as is often the case for computer vision. There are several variants of the matching problem, concerned with isometries, isomorphisms, homeomorphisms, and node attributes; different approaches exist for each variant. We show how structured estimation methods from machine learning can be used to combine such variants into a single version of graph matching. In this paradigm, the extent to which our datasets reveal isometries, isomorphisms, homeomorphisms, and other properties is automatically accounted for in the learning process so that any such specific qualification of graph matching loses meaning. We present experiments with real computer vision data showing the leverage of this unified formulation.

1. Introduction

Graphs are typically used as high-level models for images, meaning that identifying a correspondence between their nodes (popularly called a 'matching') is a fundamental problem in computer vision. Applications range from object and character recognition [1, 2], to 3-D scene reconstruction [3]. In this paper we are strictly concerned with graphs embedded in Euclidean spaces, i.e., graphs whose nodes encode point coordinates.

Many types of graphs have been considered in the graph matching literature, depending on how one models a given matching problem. Perhaps the simplest setting arises when the graphs are represented as sets of nodes and edges and the goal is to find an *isomorphism* between them [4] (or more generally, a *subgraph* isomorphism, to allow for outlying nodes). Also of interest is the *homeomorphism* problem, which allows for edges to be 'split' by additional nodes, so long as the topology is preserved [5]. For graphs embedded in Euclidean spaces, one is additionally interested in the notion of *isometry*: two graphs are isometric if the corresponding distances between nodes are preserved [6]. Figure 1 illustrates the aspects of isometry, isomorphism and homeomorphism. In addition to these notions, the concept of *attributed* nodes and edges gives rise to the notion of attributed graph matching, i.e., one is interested in comparing specific abstract features encoded by nodes and/or edges of the graph (e.g. nodes can encode colors of image regions, edges can encode adjacency of objects, etc.).

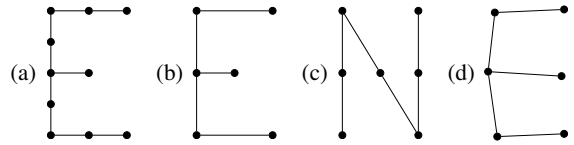


Figure 1. Examples of the transformations typically observed in computer vision scenarios. Isometric: $b \subseteq a$, $b \subseteq c$. Isomorphic: $b \subseteq d$, $c \subseteq a$, $d \subseteq b$. Homeomorphic: $b \subseteq a$, $b \subseteq d$, $c \subseteq a$, $d \subseteq a$, $d \subseteq b$. The notation $x \subseteq y$ allows for 'outliers' in y.

In the literature, these different types of formulations for graph matching usually come with different types of models and algorithms. The purpose of the present paper is to combine these hard notions of different types of graph matching problem into a single concept. Specifically, we focus on combining the properties of isometry, isomorphism, homeomorphism and node attributes into a single model which can not only be solved efficiently and exactly for graphs embedded in Euclidean spaces but is also provably optimal in the absence of noise. We further compare our unified model with recently proposed ones, which make specific assumptions about the type of matching problem at hand, and we find that ours is able to better leverage the fact that real data does not come exactly as isometries, homeomorphisms or isomorphisms: it is simply real data.

1.1. Related Work

We shall be interested in comparing the results of our model with some well-known models that make specific as-

*Julian McAuley and Teófilo de Campos were at Xerox Research Centre Europe at the time of this work.

sumptions about the type of graph matching problem being solved. The first are methods based exclusively on *node attributes*. These methods do not consider the relational aspects between nodes; such approaches typically consist of minimizing a first-order objective function, i.e., points in the template should be mapped to similar points in the target, but relationships between points are not considered. Examples include 'shape-context' matching [2], 'inner distance' matching [7], and the 'longest common subsequence' algorithm [8].

Other approaches consider the *distances* between nodes in the graphs, in order to solve the problem of *isometric matching* [6].

Finally, by expressing the matching problem using a quadratic assignment objective, it is trivial to model topological constraints, though the resulting optimization problem is in general NP-complete. As such, a great body of work has been focused on producing efficient and accurate approximations. Examples include 'spectral methods' [9, 10], 'probabilistic approaches' [11, 12], and 'graduated assignment' [13].

2. The Model

2.1. Isometry

Suppose we have a 'template' graph $\mathcal{G}$ with nodes and edges (V, E), which we wish to identify in a 'target' graph $\mathcal{G}' = (V', E')$ (we allow that $|V'| \geq |V|$, so that there may be outliers in G'). The isometric matching problem consists of finding a mapping $g : V \to V'$ which preserves the distances between pairs of nodes in $\mathcal{G}$, i.e.,

$$g = \underset{f:V\to V'}{\operatorname{argmin}} \sum_{(p_1,p_2)\in V^2} \Big|d(p_1, p_2) - d(f(p_1), f(p_2))\Big|, \quad (1)$$

where $d(\cdot, \cdot)$ is our distance metric.[1] Note that the 'topologies' of our graphs, defined by E and E', are ignored by this objective function.

It is shown in [14, 15] that in the case of exact isometric matching, one need not consider *all* edges in V^2, but only a subset of edges that constitute a 'globally rigid' graph: by definition, preserving the distances of the edges in such a subgraph implies that the distances between all edges in the *complete* graph will be preserved. Thus, for a globally rigid subgraph $\mathcal{R} = (V, E^{\mathcal{R}})$ of $\mathcal{G}$, we need to solve

$$g = \underset{f:V\to V'}{\operatorname{argmin}} \sum_{(p_1,p_2)\in E^{\mathcal{R}}} \Big|d(p_1, p_2) - d(f(p_1), f(p_2))\Big|. \quad (2)$$

If the graph $\mathcal{R}$ has small maximal cliques, (eq. 2) can be modelled as inference in a tractable graphical model (whose nodes and assignments correspond to points in V and V' respectively); [14] reports running times and memory requirements of $O(|V||V'|^{n+1})$, where n is the number of dimensions. We use a similar topology which replaces the 'ring' structure from [14] with a 'junction-tree', which has the advantage that only a single iteration is required for exact inference [15]. The topology of this graph is shown in Figure 2 (for $n = 2$), and it is rigid by Theorem 2.1.

Theorem 2.1. *The graph in Figure 2 is globally rigid when embedded in* $\mathbb{R}^2$.

Proof. The claim is trivially true for $k \leq 3$ since all edges are included. For $k > 3$, assume that the positions of $V_1 \dots V_k$ are determined. If the distances from V_{k-1} and V_k to V_{k+1} are known, then V_{k+1} may occupy precisely two positions. The two positions result in an opposite sign for $\det\begin{bmatrix} V_{k-1} - V_{k+1} \\ V_k - V_{k+1} \end{bmatrix}$ (note that the points V_k are 2-D coordinates, so that this is a 2×2 matrix). If the correct sign is determined by the variable M, then the position of V_{k+1} is uniquely determined. The proof follows by induction. □

Theorem 2.1 generalizes to $\mathbb{R}^n$ by increasing the clique size of the graph to $n + 1$.

2.2. Isomorphism/Homeomorphism

To model isomorphisms and homeomorphisms, the optimization problem we would like to solve becomes

$$g = \underset{f:V\to V'}{\operatorname{argmin}} \sum_{(p_1,p_2)\in E^{\mathcal{R}}} \Big|d(p_1, p_2) - d(f(p_1), f(p_2))\Big| + \sum_{(p_1,p_2)\in E} \Big|h(p_1, p_2) - h(f(p_1), f(p_2))\Big|. \quad (3)$$

Note that in the second term, the function $h(\cdot, \cdot)$ is used. If we want to solve the subgraph *isomorphism* problem, h simply indicates the presence or absence of an edge in E or E' (i.e., $h(f(p_1), f(p_2)) = I_{E'}((f(p_1), f(p_2)))$). Alternately, in the case of the subgraph *homeomorphism* problem, h is just the distance d in $\mathcal{G}$, but becomes the *shortest-path distance* between $f(p_1)$ and $f(p_2)$ in $\mathcal{G}'$. We observe a value of zero in (eq. 3) exactly when the mapping is isometric (by Theorem 2.1), and isomorphic/homeomorphic (as every topological constraint has been included).

This problem can also be solved using a tractable graphical model, so long as E is a subset of $E^{\mathcal{R}}$. To this end, we make the straightforward observation that we can augment $\mathcal{R}$ by 'copying' some of its nodes, as in Figure 3 (producing $\mathcal{G}^\sharp = (V^\sharp, E^{\mathcal{R}^\sharp})$). The resulting graph remains globally rigid, and obtains the correct solution due to the following Theorem:

Theorem 2.2. *If additional replicates of some nodes are added to the model, each of the replicates will map to*

[1] For the case of *exact* matching, we could use the indicator function $1 - I_{|d(p_1,p_2)|}(d(f(p_1), f(p_2)))$; we instead use the difference to allow for some 'noise' in the point coordinates.

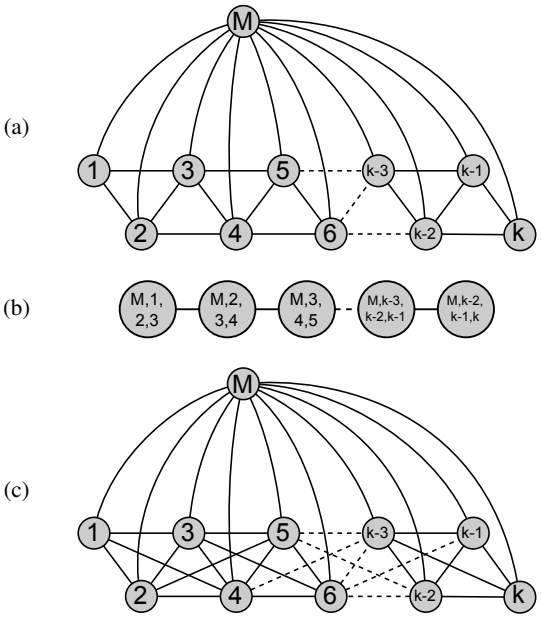


Figure 2. (a) The graphical model used in our experiments (top), and (b) its clique-graph. M is a boolean variable indicating whether or not the template pattern appears reflected in the target. Although the graph has maximal cliques of size four, the running time is cubic in $|V'|$ (for the case of 2-D isometric matching), since the node M is boolean. The model in (a) handles isometric transformations, however we could obtain scale invariance (for example) by inserting an additional constraint, as shown in (c); the size of the maximal cliques grows with the number of parameters in the transformation we wish to handle.

the same solution (assuming that an isometric isomorphism/homeomorphism exists).

Proof. Assume the image of $V_1 \dots V_k$ corresponds to an isometric isomorphism/homeomorphism under the mapping f (and thus we can determine M). This mapping will have zero cost according to our potential function. Suppose V_{k+1} is a copy of a previous node V_j. Given the values of V_{k-1}, V_k, and M, we know from Theorem 2.1 that there is *at most one* value for $f(V_{k+1})$ which incurs zero cost. Since $f(V_{k+1}) = f(V_j)$ incurs zero cost, the minimum cost solution for the model $M, V_1 \dots V_{k+1}$ must have $f(V_{k+1}) = f(V_j)$. Again, the proof follows by induction (the 'base case' is trivial, since there are no replicates). □

In order to add all of the edges in E, we will increase the size of $V^\sharp$ to $O(|V| + |E|)$.[2] At this point we simply solve

$$g = \underset{f:V\to V'}{\operatorname{argmin}} \sum_{(p_1,p_2)\in E^{\mathcal{R}^\sharp}} \Big|d(p_1,p_2) - d(f(p_1),f(p_2))\Big| + \Big|h(p_1,p_2) - h(f(p_1),f(p_2))\Big|, \quad (4)$$

[2]Finding the most efficient construction of this graph is a difficult problem, however finding a construction no larger than $|V| + 2|E|$ is trivial.

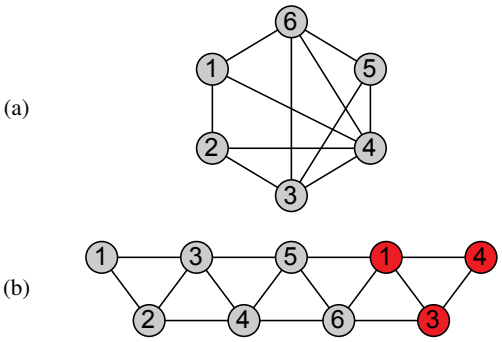


Figure 3. The topology of this graph (top) is not captured by the graphical model from Figure 2. By repeating some of the nodes in the graphical model (highlighted in red), we can include all of the topological constraints; the augmented model will still recover the correct solution (the variable M is suppressed for readability).

which can be done in $O((|V| + |E|)|V'|^{n+1})$ time and space. Assuming the topology E is sparse, this will typically be no worse than $O(|V||V'|^{n+1})$ in practice.

It is worth briefly mentioning that this model can be applied under transformations besides isometries, for instance to the problem of subgraph isomorphism/homeomorphism under perspective projection or affine transformation; this idea is demonstrated in Figure 2. Following the analysis of [16], the number of free parameters in the transformation determines the maximal clique size required in our model. Our experiments are concerned with isometries in 2-D images, which require maximal cliques of size three, resulting in an asymptotic complexity of $O((|V| + |E|)|V'|^3)$. Essentially, we are making a trade-off between the problem classes we wish to handle exactly, versus the computational complexity we can afford. Our third-order model is 'optimal' in the case of isometric transformations, and we shall show in Section 4 that it provides accurate results even as our model assumptions become violated (e.g. under affine transformations and high noise). The low tree-width of our model allows inference to be done more quickly than existing quadratic-assignment solvers, a result that could not be obtained with a more complex model.

3. Parametrization and Structured Learning

We have shown that in the case of *exact* matching, it is sufficient to attribute only the edges, using the two 'features' in (eq. 4). Of course, in the case of *near*-exact matching, we may achieve much better performance if we incorporate first-order features such as Shape-Context or SIFT [2, 17]. Since our graphical model contains 3-cliques, we can also include third-order features, ensuring (for example) preservation of angles and similarity of triangles.

In the most general setting, we can parametrize $\mathcal{G}^\sharp$ as

follows:

$$g = \underset{f:V\to V'}{\operatorname{argmin}} \sum_{p_1\in\mathcal{G}^\sharp} \left\langle \underbrace{\Phi^1(p_1, f(p_1))}_{\text{node features}}, \theta^{\text{nodes}} \right\rangle + \sum_{(p_1,p_2)\in\mathcal{G}^\sharp} \left\langle \underbrace{\Phi^2(p_1, p_2, f(p_1), f(p_2))}_{\text{edge features}}, \theta^{\text{edges}} \right\rangle + \sum_{(p_1,p_2,p_3)\in G^\sharp} \left\langle \underbrace{\Phi^3(p_1, p_2, p_3, f(p_1), f(p_2), f(p_3))}_{\text{triangle features}}, \theta^{\text{tri}} \right\rangle. \quad (5)$$

In order to apply *Structured Learning* [18], we have two requirements: the model must be parametrized linearly (which is satisfied by $\Theta = \left(\theta^{\text{nodes}}; \theta^{\text{edges}}; \theta^{\text{tri}}\right)$), and the so-called 'column-generation' procedure must be tractable. To satisfy the latter requirement, we specify a loss function $\Delta(\hat{g}, g)$ (the cost of choosing the mapping $\hat{g}$ when the correct mapping is g), which decomposes over the nodes in our model; the simplest example is the (normalized) Hamming loss:

$$\Delta(\hat{g}, g) = 1 - \frac{1}{|V'|} \sum_{p\in V'} I_{\{g(p)\}}(\hat{g}(p)). \quad (6)$$

In the majority of our experiments, nodes are attributed using shape-context features. Edges are attributed using distances (the left-hand-side of (eq. 4)), and topological features (the right-hand-side (eq. 4)); we use *both* the indicator (for isomorphisms) *and* the shortest-path distance (for homeomorphisms) – thus our model is in effect *learning* the extent to which the mapping is isomorphic or homeomorphic. 3-cliques are parametrized by preservation of angles, triangle similarity, and an 'occlusion' feature indicating whether two or more nodes in a clique map to the same point.

The goal of structured learning is merely to find the parameter vector Θ which minimizes our regularized risk:

$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmin}} \underbrace{\frac{1}{N} \sum_{i=1}^{N} \Delta(\hat{g}^i, g^i)}_{\text{empirical risk}} + \underbrace{\frac{\lambda}{2} \|\Theta\|_2^2}_{L_2 \text{ regularizer}}, \quad (7)$$

where $g^1 \dots g^N$ is our training set, and λ is a regularization constant. The 'Bundle Methods for Risk Minimization' (BMRM) solver of [19] is used to minimize (a convex upper bound on) this risk. See [18] for more details.

4. Experiments

We replicate several existing graph-matching experiments [20] and [6] for which topological information was provided, and add new experiments on real and synthetic point sets. The techniques that we shall compare can be summarized as follows:

Linear Assignment (from [20]) can be optimized exactly, but only includes node attributes (i.e., no topological information can be included).

Quadratic Assignment (from [20]) can include topological information, but optimization can only be done approximately (due to NP-completeness).

Isometric Matching (from [6]) can include *some* topological information, and can be optimized exactly;[3] is provably optimal for the isometric point-pattern matching problem.

Unified Matching (the proposed approach) can include *all* topological information, and can be optimized exactly; is provably optimal for the isometric isomorphism/homeomorphism problem (thus unifying the above approaches).

4.1. Proof of Concept

For our first experiment, we created a simple point-pattern which includes topological constraints (shown in Figure 5(a), center). In the target scene, several copies of this pattern were randomly (isometrically) transformed, and noise was applied to their topologies (while ensuring that exactly one isomorphic instance existed in the target scene).[4] This represents a problem that cannot be solved by first-order approaches, as there would be many spurious copies of the template shape if the topological information were ignored. Finally, uniform noise was applied to the point coordinates in the target scene.

Note that in this experiment, we use only those features described in Section 2.2 (i.e., there are no node features). Thus we confirm our claim that the problem can be solved using only these features.

In Figure 5(a), we compare our model to quadratic assignment. Although both methods solve the problem close to exactly when there is only a single copy of the template scene in the target ('no outliers'), we found that quadratic assignment was no better than random as soon as outliers were introduced (i.e., $\Delta \simeq 1$; each outlying graph contributes 19 nodes to the scene). Alternately, the proposed method is able to identify the correct instance even in the case of several outliers, and is able to deal with very large graphs due to its modest computational complexity. Note that our model is provably optimal only in the case of zero noise, though it appears to perform quite accurately under reasonable noise levels.

[3] Rather, it is shown in [14] to *converge* to the optimal solution. If the algorithm is not run until convergence, the results may be suboptimal.

[4] This was done only to simplify the experiment – it is of course possible for our method to find *every* isomorphic instance if more than one exists.

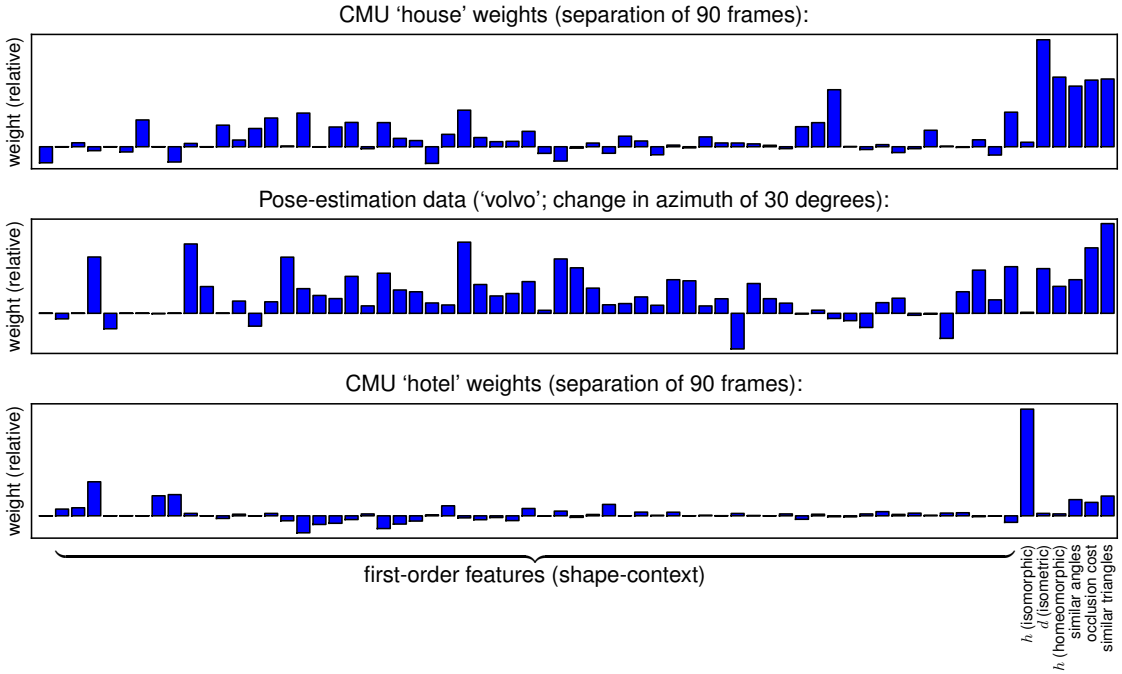


Figure 4. The weight vectors (Θ) learned from some of our experiments. Note that in the 'house' experiment the weight for the 'homeomorphic' feature is very high, whereas the weight for the 'isomorphic' feature is very low; the opposite effect is observed for the 'hotel' experiment. In effect, our algorithm has learned whether isomorphisms or homeomorphisms better capture these datasets.

4.2. Synthetic Data

In [20], the performance of linear and quadratic assignment (with learning) is reported on a series of silhouette images from [21]. The point sets in question are subject to increasing amounts of various distortions (shear, rotation, noise). In Figure 6(a) we report the performance from [20] against our method on the 'shear' dataset; the 'noise' dataset is not shown, as it is similar to our own synthetic dataset from the previous experiment. The 'rotation' dataset is also not shown, as our method will provably achieve zero error on the entire dataset (due to its rotational invariance). The topological structure of the graph was simply determined using the outline of the silhouette (see Figure 6(a)).

The 'balanced graph-matching' algorithm of [10] (which to the best of our knowledge is the state-of-the-art non-learning approach) is also reported for comparison, for those experiments where results were available from [20].

For each of our experiments involving learning, we split our data into training, validiation, and test sets, each of which contains one third of the data. Learning is performed on the *training* set for different values of the hyperparameter $\lambda \in \{0.001, 0.01, 0.1, 1, 10\}$. We always report the performance on the *test* set, for the value of λ that gave the best performance on the *validation* set.

Our learning approach is fully-supervised, i.e., we assume that the correspondences are fully labeled. We find that even a small number of such correspondences can be enough to obtain a substantial improvement over non-learning. Alternately, in cases where fully supervised data cannot be obtained, we note that unsupervised instances of this problem are likely to require the fully supervised version as a component [22].

4.3. 3-D Models from Different Viewpoints

In this experiment, our sequences represent different views of 3-D models. We consider several sequences: the CMU 'house' and 'hotel' sequences (used in [6,20]), as well as two new sequences from [24], which are advantageous in that their changes in rotation/azimuth are known.

Weight vectors from three of our experiments are shown in Figure 4. Unlike the previous experiments, the 'topology' of each point-pattern was *automatically generated* for each image: by forming a spanning tree (so that the graph is connected), and by connecting K-nearest-neighbors (examples are shown in Figure 6). Consequently, there is substantial noise in the topological features. Nevertheless, we observe high weights for the isomorphic and homeomorphic features in these datasets, indicating that these features are useful. Consequently, the unified model achieves better performance when the separation between frames is large. Note that in cases where the transformations are non-isometric, the model simply *learns* not to use the isometric features.

For the CMU 'house' sequence, we also present a timing plot (Figure 5(b), right). To improve the running time of our

model, we used the same approach as [6], where for each node we only consider the P matches with the lowest linear cost, thus reducing the running time to $O((|V| + |E|)P^3)$. This parameter trades off accuracy against running time, though we found that even for small values of P, the impact on accuracy was minor; in this experiment we used $P = 15$, where $|V'| = 30$. The running time of balanced graph-matching [10] is also shown, though this is based on a Matlab implementation, while the others were written in C++. Our implementation is available online [25].

4.4. Video Sequence

In this experiment, we used the Harris-Hessian detector of [23] to identify keypoints in the 'claire' video sequence from [25], and extracted SIFT features for each keypoint [17]. Next, we selected 15 of these keypoints in each frame corresponding to important points-of-interest (such as the eyes/ears). Our goal is exactly the same as in the previous experiments, though we are now dealing with a very large number of outliers (typically, several hundred keypoints were identified), as well as occlusions in both the template and the target scenes (since the keypoint detector may not capture some points-of-interest). Performance on this dataset is shown in Figure 5(c); note that we do not use the Hamming loss in this experiment, but rather the endpoint loss (i.e., the average distance from the 'correct' keypoint) – this is a more sensible choice in the presence of many outliers. Quadratic assignment was not used, as it was prohibitively expensive on graphs of this size.

5. Conclusion

In this paper, we have presented a unified formulation for graph matching in Euclidean spaces, which accounts for isometries, isomorphisms, homeomorphisms, and node attributes. By means of structured estimation, the model is able to automatically learn the extent to which each of these properties are present in the training data instead of relying on *a priori* assumptions about the specific 'type' of graph matching problem under consideration. Inference in this model is efficient and exact since it comprises a junction tree of small tree-width. This method is provably optimal when there is no noise in the point coordinates and attributes, while we have shown that it remains highly accurate under reasonable noise levels.

Acknowledgements

The research leading to these results has received funding from the EC FP7 grant 216529 (PinView), as well as EPSRC/UK grant EP/F069626/1, ACASVA. NICTA is funded by the Australian Government's *Backing Australia's Ability* initiative, and the Australian Research Council's *ICT Centre of Excellence* program.

References

[1] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79, 2005.

[2] J. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. on PAMI*, 24(4):509–522, 2002.

[3] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[4] J. Ullman. An algorithm for subgraph isomorphism. *Journal of the ACM*, 23(1):31–42, 1976.

[5] A. S. LaPaugh and R. L. Rivest. The subgraph homeomorphism problem. In *ACM Symposium on Theory of Computing*, 1978.

[6] J. J. McAuley, T. S. Caetano, and A. J. Smola. Robust near-isometric matching via structured learning of graphical models. *NIPS*, 2008.

[7] H. Ling and D. Jacobs. Shape classification using the inner-distance. *IEEE Trans. on PAMI*, 29(2):286–299, 2007.

[8] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[9] L. Shapiro and J. Brady. Feature-based correspondence - an eigenvector approach. *Image and Vision Computing*, 10:283–288, 1992.

[10] T. Cour, P. Srinivasan, and J. Shi. Balanced graph matching. In *NIPS*, 2006.

[11] R. C. Wilson and E. R. Hancock. Structural matching by discrete relaxation. *IEEE Trans. on PAMI*, 19(6):634–648, 1997.

[12] J. V. Kittler and E. R. Hancock. Combining evidence in probabilistic relaxation. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 3:29–51, 1989.

[13] S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *IEEE Trans. PAMI*, 18(4):377–388, 1996.

[14] J. J. McAuley, T. S. Caetano, and M. S. Barbosa. Graph rigidity, cyclic belief propagation and point pattern matching. *IEEE Trans. on PAMI*, 30(11):2047–2054, 2008.

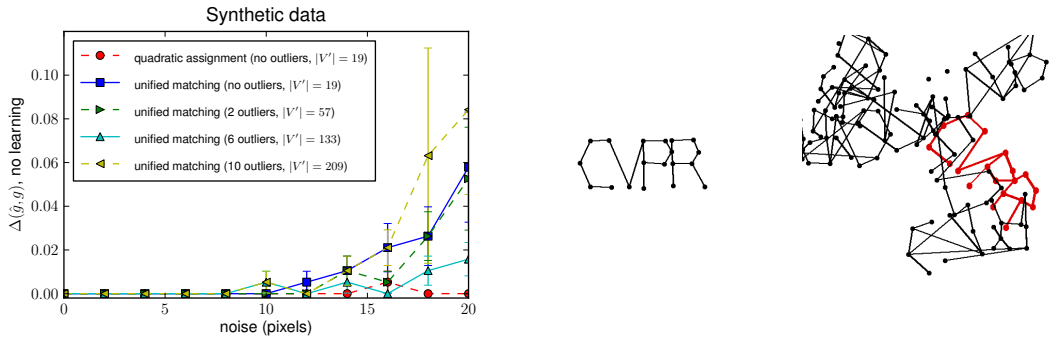
[15] T. S. Caetano and J. J. McAuley. Faster graphical models for point-pattern matching. *Spatial Vision*, 2009.

[16] T. S. Caetano and T. Caelli. A unified formulation of invariant point pattern matching. In *ICPR*, 2006.

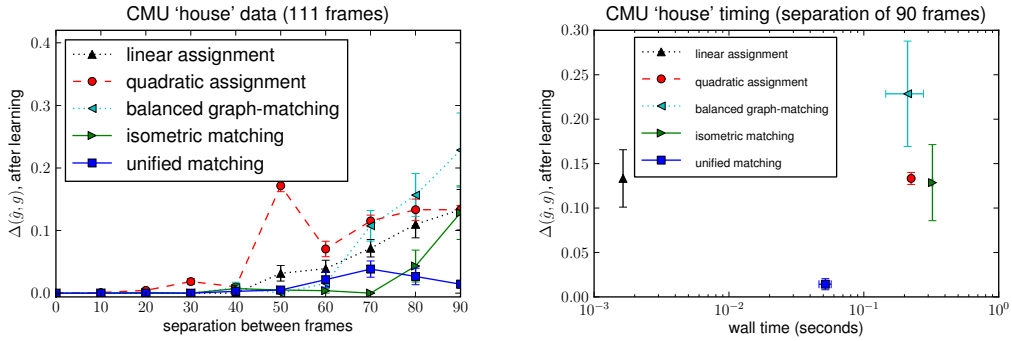
[17] D. G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999.

[18] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *ICML*, 2004.

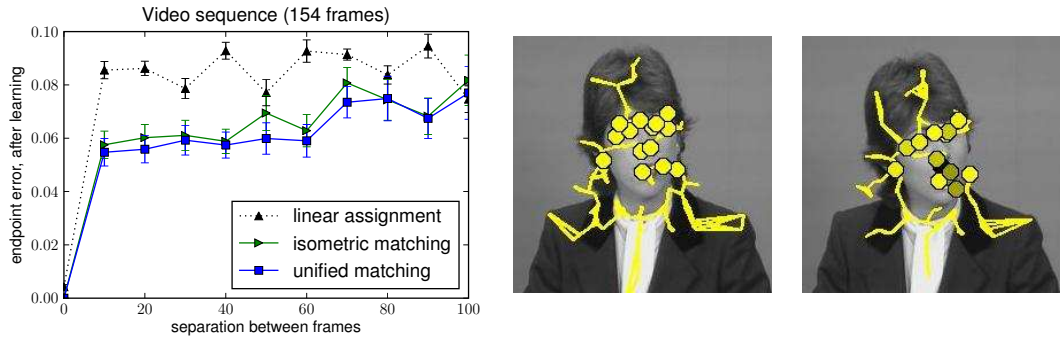
[19] C. H. Teo, Q. Le, A. J. Smola, and S. V. N. Vishwanathan. A scalable modular convex solver for regularized risk minimization. In *KDD*, 2007.



(a) Performance on the 'proof of concept' dataset. The template graph is shown at center, while the target graph is shown at right; the correct match (which was identified by our method) is highlighted in red.



(b) Performance and timing on the CMU 'house' dataset. Note the logarithmic scale in the timing plot.



(c) Performance on SAMPL 'claire' dataset. The template graph is shown at center (yellow nodes indicate points-of-interest), while its mapping in the target is shown at right. Intensity corresponds to error: yellow indicates a correct match, while black indicates an error of 40 pixels.

Figure 5. Performance, timing, and example matches. Note that the x-axis of each plot essentially captures the 'difficulty' of the matching task: as the separation between frames increases, the transformations become less and less rigid. Images have been cropped for visibility.

[20] T. S. Caetano, J. J. McAuley, L. Cheng, Q. V. Le, and A. J. Smola. Learning graph matching. *IEEE Trans. on PAMI*, 31(6):1048–1058, 2009.

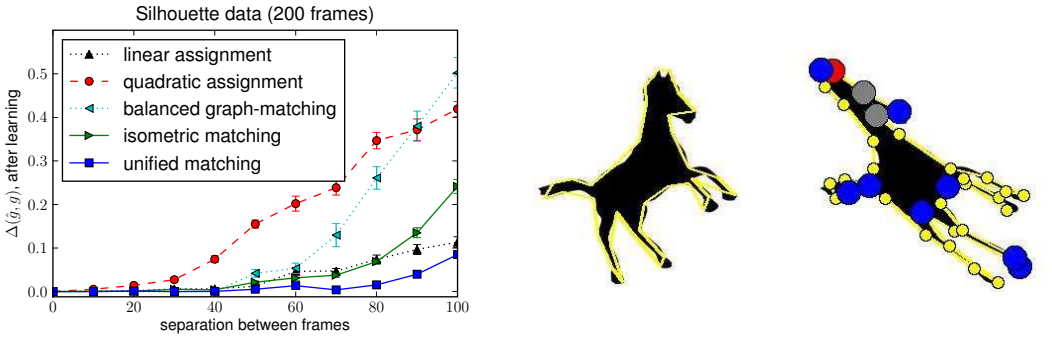
[21] A. M. Bronstein, M. M. Bronstein, A. M. Bruckstein, and R. Kimmel. Analysis of two-dimensional non-rigid shapes. *International Journal of Computer Vision*, 2007.

[22] C. J. Yu and T. Joachims. Learning structural SVMs with latent variables. In *ICML*, 2009.

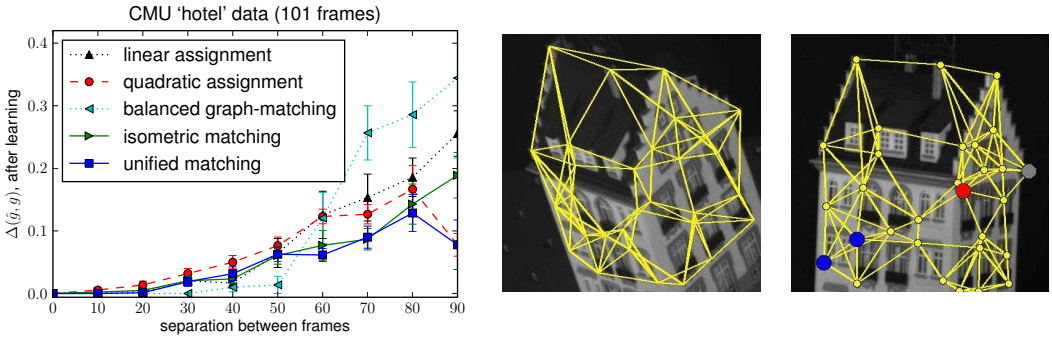
[23] K. Mikolajczyk, B. Leibe, and B. Schiele. Multiple object class detection with a generative model. In *CVPR*, 2006.

[24] F. Vikstén, P. Forssén, B. Johansson, and A. Moe. Comparison of local image descriptors for full 6 degree-of-freedom pose estimation. In *ICRA*, 2009.

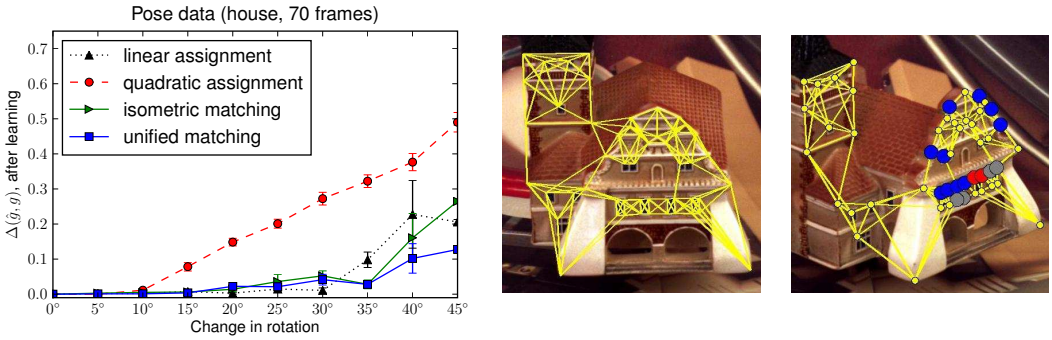
[25] Implementation of our method:
`http://users.cecs.anu.edu.au/~julianm/`
Dataset sources:
`http://sampl.ece.ohio-state.edu/database.htm`
`http://tosca.cs.technion.ac.il`
`http://vasc.ri.cmu.edu/idb/html/motion/`
`http://www.cvl.isy.liu.se/research/objrec/posedb/`.



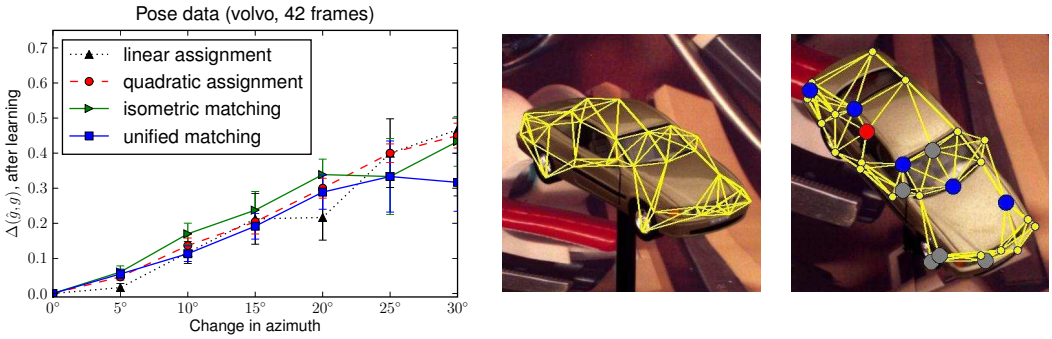
(a) Performance on the silhouette dataset from [20, 21].



(b) Performance on the CMU 'hotel' dataset.



(c) Performance on the pose-estimation dataset ('house').



(d) Performance on the pose-estimation dataset ('volvo').

Figure 6. Additional results (see Figure 5). Template graphs are shown at center with their topologies. Target graphs are shown at right, comparing accuracy with and without topological features. The color of each node indicates whether the correct match was identified by both methods (yellow), neither method (gray), only *without* topological features (red), or only *with* topological features (blue).

Exploiting Data-Independence for Fast Belief-Propagation

Julian J. McAuley JULIAN.MCAULEY@NICTA.COM.AU
Tibério S. Caetano TIBERIO.CAETANO@NICTA.COM.AU
NICTA and Australian National University, Canberra ACT 0200 Australia

Abstract

Maximum a posteriori (MAP) inference in graphical models requires that we maximize the sum of two terms: a *data-dependent* term, encoding the conditional likelihood of a certain labeling given an observation, and a *data-independent* term, encoding some prior on labelings. Often, data-dependent factors contain fewer latent variables than data-independent factors – for instance, many grid and tree-structured models contain only first-order conditionals despite having pairwise priors. In this paper, we note that MAP-inference in such models can be made substantially faster by appropriately preprocessing their data-independent terms. Our main result is to show that message-passing in any such pairwise model has an expected-case exponent of only 1.5 on the number of states per node, leading to significant improvements over existing quadratic-time solutions.

1. Introduction

MAP-inference in a graphical model $\mathcal{G}$ consists of solving an optimization problem of the form

$$\hat{\mathbf{y}}(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{y}} \sum_{C \in \mathcal{C}} \Phi_C(\mathbf{y}_C|\mathbf{x}_C), \tag{1}$$

where $\mathcal{C}$ is the set of maximal cliques in $\mathcal{G}$. This problem is often solved via *message-passing* algorithms such as the junction-tree algorithm, loopy belief-propagation, or inference in a factor graph (Aji & McEliece, 2000; Kschischang et al., 2001).

Computing messages between two intersecting cliques A and B in general involves solving a problem of the form

$$m_{A\to B}(\mathbf{y}_{A\cap B}) = \max_{\mathbf{y}_{A\setminus B}} \Phi_A(\mathbf{y}_A|\mathbf{x}_A) \sum_{D\in\Gamma(A)\setminus\{B\}} m_{D\to A}(\mathbf{y}_{A\cap D}), \tag{2}$$

where $\Gamma(A)$ is the set of cliques that intersect with A. If the nodes of our model have N states, solving (eq. 2) appears to require $\Theta(N^{|A|})$ operations, since there are $N^{|A|}$ possible values of $\mathbf{y}_A$.

Alternately, (eq. 1) can be expressed in the form

$$\hat{\mathbf{y}}(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{y}} \underbrace{\sum_{F\in\mathcal{F}} \Phi_F(\mathbf{y}_F|\mathbf{x}_F)}_{\text{data dependent}} + \underbrace{\sum_{C\in\mathcal{C}} \Phi_C(\mathbf{y}_C)}_{\text{data independent}}, \tag{3}$$

where each $F \in \mathcal{F}$ is a subset of some $C \in \mathcal{C}$.

In this paper, we show that much faster algorithms can be developed whenever the model's *data-dependent* factors contain fewer latent variables than its *data-independent* factors, or equivalently when every $F \in \mathcal{F}$ is a *proper subset* of some $C \in \mathcal{C}$ in (eq. 3). Although our results apply to general models of this form, we shall mainly be concerned with the most common case, in which we have pairwise models with data-independent priors, or problems of the form

$$\hat{\mathbf{y}}(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{y}} \underbrace{\sum_{i\in\mathcal{N}} \Phi_i(y_i|x_i)}_{\text{node potential}} + \lambda \underbrace{\sum_{(i,j)\in\mathcal{E}} \Phi_{i,j}(y_i,y_j)}_{\text{edge potential}}. \tag{4}$$

This encompasses a wide variety of models, including grid-structured models for optical flow and stereo disparity as well as chain and tree-structured models for text or speech. Examples are shown in Figure 1. In all of these examples, we give a solution to (eq. 2) with an *expected-case running time* of only $O(N^{1.5})$, while to our knowledge, the best currently known solution is the naïve $\Theta(N^2)$ version. Our result is achieved by preprocessing the data-independent part of the model *offline*, simply by sorting the rows and columns of $\Phi_{i,j}$.

As our optimizations apply directly to the message passing equations themselves, they can be applied

Appearing in *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010.

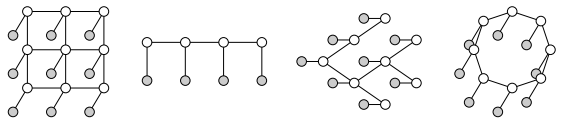


Figure 1. Some graphical models to which our results apply: *cliques containing observations have fewer latent variables than purely latent cliques.* Gray nodes correspond to the observation, white nodes to the labeling. In other words, cliques containing a gray node encode the *data likelihood*, whereas cliques containing only white nodes encode *priors*. We focus on the case where the gray nodes have degree one (i.e., they are connected to only one white node).

to many variants of belief-propagation, such as the junction-tree algorithm, loopy belief-propagation, and factor graphs. In particular, in models where belief-propagation is known to produce the correct solution, i.e., trees (and junction trees in general), our optimizations result in the asymptotically fastest solution for exact inference.

1.1. Related Work

There has been previous work on speeding-up message-passing algorithms by exploiting some type of structure in the graphical model. For example, Kersting et al. (2009) study the case where different cliques share the same potential function. In Felzenszwalb & Huttenlocher (2006), fast message-passing algorithms are provided for the case in which the potential of a 2-clique is only dependent on the *difference* of the latent variables (which is common in some computer vision applications); they also show how the algorithm can be made faster if the graphical model is a bipartite graph. In Kumar & Torr (2006), the authors provide faster algorithms for the case in which the potentials are *truncated*, whereas in Petersen et al. (2008) the authors offer speedups for models that are specifically grid-like.

The latter work is perhaps the most similar in spirit to ours, as it exploits the fact that certain factors can be *sorted* in order to reduce the search space of a certain maximization problem. In practice, this leads to linear speedups over a $\Theta(N^4)$ algorithm. We too shall rely on sorting to reduce the search space of a maximization problem, but additionally we exploit *data-independence* to reduce a $\Theta(N^2)$ algorithm to $O(N^{1.5})$.

Notably, our assumption of *data-independence in the prior* differs substantially from those above; it is arguably a much weaker assumption since it is in fact satisfied by most graphical models of practical interest.

2. Our Approach

We consider an (undirected) pairwise graphical model $\mathcal{G}(\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ is the set of nodes, and $\mathcal{E}$ the set of edges, which factorizes according to (eq. 4). In such a model, computing a message between two neighboring cliques $A = (i, j)$ and $B = (i, k)$ is equivalent in complexity to solving

$$m_{A\to B}(y_i) = \Psi_i(y_i) + \max_{y_j} \Psi_j(y_j) + \Phi_{i,j}(y_i, y_j), \quad (5)$$

where $\Psi_i(y_i)$ is the sum of $\Phi(y_i|x_i)$ and any first-order messages over y_i (similarly for $\Psi_j(y_j)$). The general form of (eq. 5) encompasses many variants of belief-propagation. In all such cases, solving (eq. 5) appears to be a $\Theta(N^2)$ operation, since this is the number of possible arguments to $\Phi_{i,j}$.

For a specific value of $y_i = q$, solving (eq. 5) amounts to solving

$$m_{A\to B}(q) = \Psi_i(q) + \max_{y_j} \underbrace{\Psi_j(y_j)}_{\mathbf{v}_a} + \underbrace{\Phi_{i,j}(q, y_j)}_{\mathbf{v}_b}, \quad (6)$$

which appears to have linear time complexity, as it is equivalent to solving

$$\max_i \{\mathbf{v}_a[i] + \mathbf{v}_b[i]\}. \quad (7)$$

However, we note that solving (eq. 7) is only $O(\sqrt{N})$ *if we know the permutations that sort* $\mathbf{v}_a$ *and* $\mathbf{v}_b$. Our algorithm for solving (eq. 7) is given in Algorithm 1; the execution of this algorithm is explained in Figure 2. For now, we simply state the following theorem regarding the algorithm's running time:

Theorem 1. *The* ***expected*** *running time of Algorithm 1 is* $O(\sqrt{N})$. *This yields a speedup of at least* $\Omega(\sqrt{N})$ *in models containing pairwise priors and unary data-likelihoods.*

We have recently employed an algorithm of this type in McAuley & Caetano (2010), where we showed that fast algorithms can be developed for inference in graphical models whose maximal cliques are larger than their factors. Further discussion of this theorem can be found in Section 3; complete proofs are given in McAuley & Caetano (2009).

An apparent issue is that the cost of sorting every row of $\Phi_{i,j}$ is $\Theta(N^2 \log N)$ (i.e., more expensive than the naïve solution). However we make the observation that this cost can be circumvented *so long as only the data-independent part of the potential is maximal* (i.e., the prior, such as in (eq. 6)). In such cases, the data-independent part of the model can be sorted *offline*.

Algorithm 1 Find i that maximizes $\mathbf{v}_a[i] + \mathbf{v}_b[i]$

Require: permutation functions p_a and p_b that sort $\mathbf{v}_a$ and $\mathbf{v}_b$ in decreasing order

1: **Initialize:** $start \leftarrow 1$
2: $end_a \leftarrow p_a^{-1}[p_b[1]]$ {end_a is the index of the element in $\mathbf{v}_a$ corresponding to the largest element in $\mathbf{v}_b$; see the red line in Figure 2}
3: $end_b \leftarrow p_b^{-1}[p_a[1]]$
4: $best \leftarrow \text{argmax}_{i \in \{p_a[1], p_b[1]\}} \{\mathbf{v}_a[i] + \mathbf{v}_b[i]\}$
5: $max \leftarrow \mathbf{v}_a[best] + \mathbf{v}_b[best]$
6: **while** $start < end_a \wedge start < end_b$ **do**
7: {consider the indices $p_a[start]$ and $p_b[start]$}
8: $start \leftarrow start + 1$
9: **if** $\mathbf{v}_a[p_a[start]] + \mathbf{v}_b[p_a[start]] > max$ **then**
10: $best \leftarrow p_a[start]$
11: $max \leftarrow \mathbf{v}_a[best] + \mathbf{v}_b[best]$
12: **end if**
13: **if** $p_b^{-1}[p_a[start]] < end_b$ **then**
14: $end_b \leftarrow p_b^{-1}[p_a[start]]$
15: **end if**
16: {repeat Lines 9–15, interchanging a and b}
17: **end while**
18: **while** $start < end_a$ **do**
19: {we have considered all candidate values in p_b, but some may remain in p_a}
20: $start \leftarrow start + 1$
21: **if** $\mathbf{v}_a[p_a[start]] + \mathbf{v}_b[p_a[start]] > max$ **then**
22: $best \leftarrow p_a[start]$
23: $max \leftarrow \mathbf{v}_a[best] + \mathbf{v}_b[best]$
24: **end if**
25: **end while**
26: {repeat Lines 18–25, interchanging a and b}
27: **return** $best$ {this takes *expected time* $O(\sqrt{N})$}

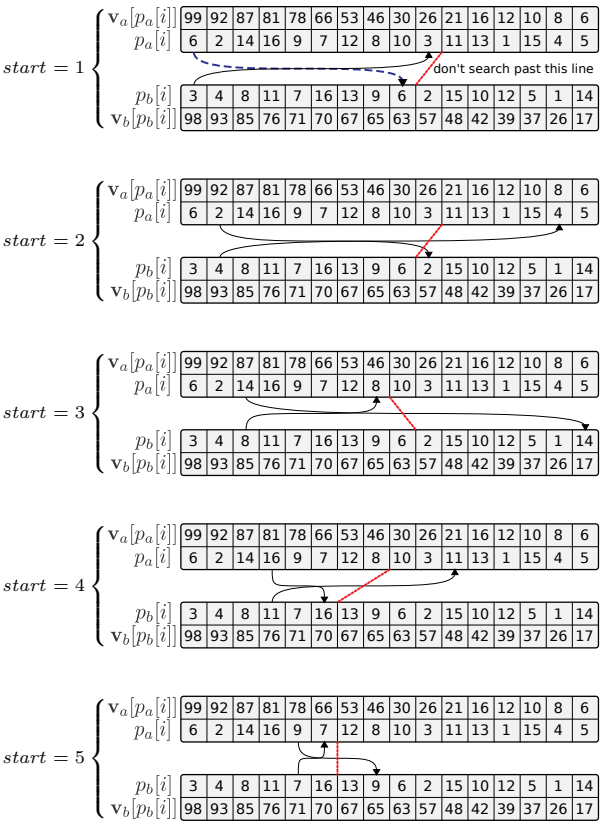


Figure 2. Algorithm 1 explained (best viewed in color): the two arrows connect $\mathbf{v}_a[p_a[start]]$ to $\mathbf{v}_b[p_a[start]]$, and $\mathbf{v}_a[p_b[start]]$ to $\mathbf{v}_b[p_b[start]]$; the red line connects end_a and end_b, which is updated each time an arrowhead lies to its left; we only consider those arrows that whose tail lies to the left of the red line – all others can be ignored; a dashed arrow is shown when a new maximum is found.

Once $\Phi_{i,j}$ has been sorted, (eq. 6) can be solved via Algorithm 2.[1]

Often the prior is *homogeneous* (every edge uses the same prior), meaning that $\Phi_{i,j}$ can be sorted *online*, so long as $|\mathcal{E}| \in \Omega(\log N)$ (i.e., the number of messages that must be computed is asymptotically larger than $\log N$). Similarly, when using iterative inference schemes (such as loopy belief-propagation), the sorting step takes place only during the first iteration; if inference is run for $\Omega(\log N)$ iterations, then speed improvements can still be obtained with online sorting.

3. Runtime Analysis

In this section, we compute the expected-case running time of Algorithm 2 under the assumption that the rows and columns of the data-independent terms have been sorted offline. First note that if Algorithm 1 solves (eq. 7) in $O(f(N))$, then Algorithm 2 must take $O(N \log(N) + N f(N))$; thus we need only compute $f(N)$. We shall demonstrate that $f(N) \in O(\sqrt{N})$ as stated in Theorem 1.

We consider the *expected-case* running time, under the assumption that the order statistics of $\mathbf{v}_a$ and $\mathbf{v}_b$ are independent. It is worth mentioning that we are limited to expected-case analysis, as there is provably no deterministic solution that is sub-linear in N: otherwise we could solve max-sum matrix multiplication (or 'funny matrix multiplication') in $O(N^{2.5})$, though it is known to have no deterministic sub-cubic solution (assuming that only addition and comparison operators are used) (Kerr, 1970; Alon et al., 1997).

The running time of Algorithm 1 depends on the permutation matrix that transforms the sorted values of

[1] C++ implementations of our algorithms are available at http://users.cecs.anu.edu.au/~julianm/

Algorithm 2 Solve (eq. 6) using Algorithm 1

Require: a set of permutation functions $\mathbf{p}$ such that p_i sorts the i^{th} row of $\Phi_{i,j}$ in decreasing order.

compute the permutation function p_a by sorting Ψ_j {takes $\Theta(N \log N)$}
for $q \in \{1 \dots N\}$ **do**
  $(\mathbf{v}_a, \mathbf{v}_b) \leftarrow (\Psi_j, \Phi_{i,j}(q, y_j | x_i, x_j))$
  $r \leftarrow$ *Algorithm1* $(\mathbf{v}_a, \mathbf{v}_b, p_a, p_q)$ $\{O(\sqrt{N})\}$
  $m_{A \to B}(q) \leftarrow \Psi_i(q) + \Psi_j(r) + \Phi_{i,j}(q, r | x_i, x_j)$
end for {*expected-case* $O(N\sqrt{N})$}
return $m_{A \to B}$

$\mathbf{v}_a$ into the sorted values of $\mathbf{v}_b$, which in turn depends only on their order statistics.

Figure 3 gives examples of different permutation matrices, and the number of addition operations that each induces. Here we see that our method is $\Theta(1)$ when the two vectors have the same order statistics, and $\Theta(N)$ when the two vectors have 'opposite' order statistics. Our analysis considers the case that the order statistics of $\mathbf{v}_a$ and $\mathbf{v}_b$ are independent (i.e., every permutation matrix is equally likely). Further details and proofs are given in McAuley & Caetano (2009).

As stated in Figure 3, we can compute the number of additions as follows: starting from the top-left of the permutation matrix, find the smallest gray square that contains an entry; if this square has width M, we perform fewer than $2M$ additions. For a randomly chosen permutation matrix, simple inspection reveals that the probability that $M > m$ is given by

$$P_N(M > m) = \frac{(N-m)!(N-m)!}{(N-2m)!N!}, \tag{8}$$

and thus, using the identity $E(X) = \sum_{x=1}^{\infty} P(X \geq x)$, we can show that the expected value of M is

$$E_N(M) = \sum_{m=0}^{\lfloor N/2 \rfloor} \frac{(N-m)!(N-m)!}{(N-2m)!N!}, \tag{9}$$

which can be shown to be $\Omega(\log N)$ and $O(\sqrt{N})$ (Theorem 1). Thus we can solve (eq. 2) in expected-case $\Theta(NE_N(M))$ which is $O(N^{1.5})$. We shall verify these statements experimentally in Section 5.1.

3.1. Correlated Data

As shown in Figure 3, our algorithm will perform better or worse than the expected-case depending on the particular permutations that sort the data. If the data are positively correlated we will tend to observe permutations that lie close to the main diagonal (the left side of Figure 3); if the data are negatively correlated we will tend to observe permutations that lie close to the off-diagonal (the right side of Figure 3). In such cases, we shall observe better or worse performance (respectively) than the expected-case.

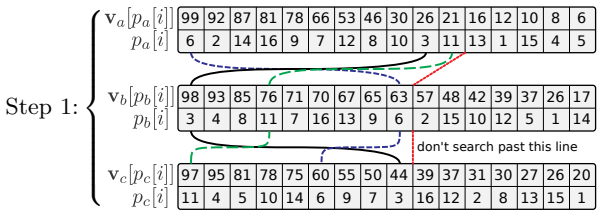


Figure 4. Algorithm 1 can be generalized to handle any number of lists. For K lists (corresponding to K^{th}-order priors), it has running time $O(KN^{\frac{K-1}{K}})$.

4. Generalizations

As we have suggested, our results apply not only to pairwise models, but to *any models whose data-dependent cliques have fewer latent variables than the data-independent cliques*; in this section we shall state our main result in this general case.

We have shown that Algorithm 1 solves (eq. 7) in $O(\sqrt{N})$. By similar reasoning, it can be shown that this algorithm can be adapted to solve

$$\max_i \{\mathbf{v}_1[i] \times \mathbf{v}_2[i] \times \dots \times \mathbf{v}_K[i]\} \tag{10}$$

in sub-linear time. As with our solution to (eq. 7), we can substantially reduce the search space if we know the permutations that sort the lists (see Figure 4). The running time of our algorithm is given by the following theorem (a proof is given in (McAuley & Caetano, 2009)):

Theorem 2. *Algorithm 1 generalizes to K lists with an expected running time of $O(KN^{\frac{K-1}{K}})$ (it can be adapted to be $O(\min(N, KN^{\frac{K-1}{K}}))$, if we carefully avoid rereading entries).*

This generalization can be applied as follows: if the *data-independent* factors are of dimension K (within cliques containing more than K terms), we can obtain an expected speedup of $\Omega(\frac{1}{K}N^{\frac{1}{K}})$; setting $K = 2$ recovers precisely the $\Omega(\sqrt{N})$ speedup for pairwise priors discussed so far in this paper.

An example application to which this generalization can be applied is that of Felzenszwalb (2005). This model contains a third-order geometric prior, while the *data-dependent* factors are only pairwise. Our method

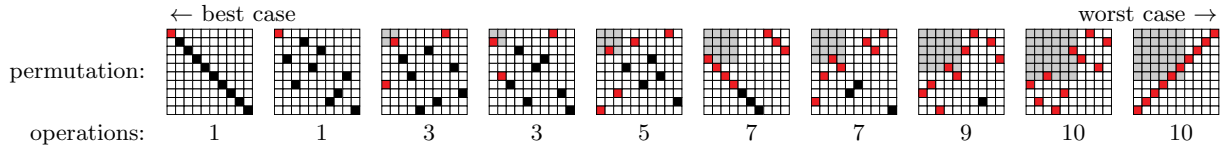


Figure 3. Different permutation matrices and their resulting cost (in terms of additions performed). Each permutation matrix transforms the *sorted* values of one list into the sorted values of the other, i.e., it transforms $\mathbf{v}_a$ as sorted by p_a into $\mathbf{v}_b$ as sorted by p_b. For instance, if there is an entry in the first row and fifth column, this indicates that $p_a[1] = p_b[5]$ (equivalently that $p_b^{-1}[p_a[1]] = 5$, or $p_a^{-1}[p_b[5]] = 1$), meaning that the largest value of $\mathbf{v}_a$ has the same index as the fifth largest value of $\mathbf{v}_b$. The red squares show the entries that must be read before the algorithm terminates (each corresponding to one addition). In reference to Algorithm 1, an entry in row number *start* corresponds to computing $\mathbf{v}_a[p_a[start]] + \mathbf{v}_b[p_a[start]]$; similarly, the entry in column number *start* corresponds to computing $\mathbf{v}_a[p_b[start]] + \mathbf{v}_b[p_b[start]]$. A simple method to determine the number of additions is as follows: starting from the top-left of the permutation matrix, find the smallest gray square that contains an entry; if this square has width M, we shall read fewer than $2M$ entries. Note that the width of this gray square is precisely the value of *start* when the algorithm terminates.

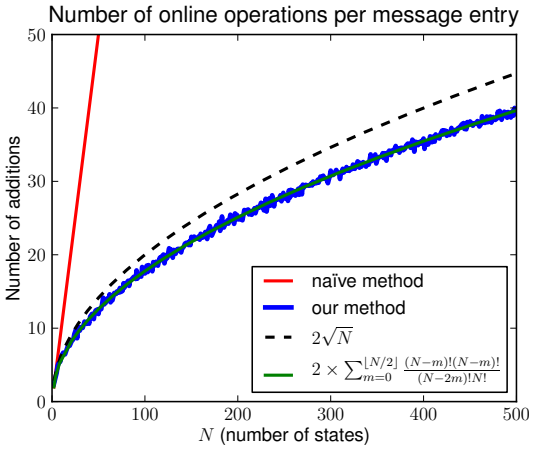


Figure 5. The number of addition operations required to compute each entry of a message (average of 10 trials). The naïve solution requires $\Theta(N)$ operations, whereas our method requires $O(\sqrt{N})$ in the expected-case. The exact expectation is also shown.

allows us to pass messages in this model in $O(N^{\frac{8}{3}})$, i.e., $\Omega(N^{\frac{1}{3}})$ faster than the standard cubic solution.

As the pairwise case is by far the most common, and as it gives the largest speedup, we shall focus on this case in our experiments.

5. Experiments

5.1. Number of Addition Operations

Figure 5 shows the number of addition operations required to solve to (eq. 7); multiplying by $N+1$ gives the number of operations required to compute (eq. 5), i.e., the entire message. $\mathbf{v}_a$ and $\mathbf{v}_b$ are chosen by sampling uniformly from $[0, 1)^N$, and the average of 10 trials is shown. The value reported is precisely the value of $2 \times start$ when Algorithm 1 terminates. This confirms that the expected value given in (eq. 9) matches the experimental performance, and also verifies that the expectation is upper-bounded by $2 \times \sqrt{N}$.

Due to the computational overhead of our solution, we expect the running time of our algorithm to differ from the value shown in Figure 5 by a multiplicative constant, except in cases where $\mathbf{v}_a$ and $\mathbf{v}_b$ are highly (positively or negatively) correlated.

5.2. Inference in Pairwise Models

In each of the following experiments we perform belief-propagation in models of the form given in (eq. 4). Thus each model is completely specified by defining the node potentials $\Phi_i(y_i|x_i)$, the edge potentials $\Phi_{i,j}(y_i, y_j)$, and the topology $(\mathcal{N}, \mathcal{E})$ of the graph.

Furthermore we assume that the edge potentials are *homogeneous*, i.e., that the potential for each edge is the same, or rather that they have the same order statistics (for example, they may differ by a multiplicative or additive constant). This means that the sorting can be done *online* without affecting the asymptotic complexity. When subject to heterogeneous potentials we need merely sort them *offline*; the online cost shall be similar to what we report here.

5.2.1. Chain-Structured Models

In this section, we consider *chain-structured* graphs. Here we have nodes $\mathcal{N} = \{1 \dots Q\}$, and edges $\mathcal{E} = \{(1, 2), (2, 3) \dots (Q-1, Q)\}$. The max-sum algorithm is known to compute the maximum-likelihood solution exactly for tree-structured models.

Figure 6 (top) shows the performance of our method on a model with *random* potentials, i.e., $\Phi_i(y_i|x_i) = U[0, 1)$, $\Phi_{i,i+1}(y_i, y_{i+1}) = U[0, 1)$, where $U[0, 1)$ is the

uniform distribution. Fitted curves are superimposed onto the running time, confirming that the performance of the standard solution grows quadratically with the number of states, while ours grows at a rate of $N^{1.5}$. The residual error r shows how closely the fitted curve approximates the running time; in the case of random potentials, both curves have similar constants.

Figure 6 (bottom) shows the performance of our method on a 'text-denoising' experiment. In this experiment random errors are introduced into a body of text, which the model aims to correct. Here we have $\Phi_i(y_i|x_i) = \lambda(1 - I_{\{x_i\}}(y_i))$, i.e., a constant cost λ is incurred in the event that x_i and y_i are not equal. $\Phi_{i,i+1}(y_i, y_{i+1})$ returns the frequency of the pair (y_i, y_{i+1}) in a training corpus. This experiment was performed on text from each language in the Leipzig Multilingual Corpora (Quasthoff et al., 2006). The first 100,000 characters were used to construct the pairwise statistics of $\Phi_{i,i+1}$, and the next 2,500 characters were used for the denoising experiment.

Although our algorithm results in a faster solution for all languages, we observe a higher constant of complexity than that obtained for random data. This suggests that the pairwise priors in different languages are not independent of the messages; the higher residual error may also suggest that different languages have different order-statistics.

5.2.2. Grid-Structured Models

Similarly, we can apply our method to *grid-structured* models. Here we resort to loopy belief-propagation to approximate the MAP solution, though indeed the same analysis applies in the case of factor graphs (Kschischang et al., 2001). We construct a 50×50 grid model and perform loopy belief-propagation using a random message-passing schedule for five iterations. In these experiments our nodes are $\mathcal{N} = \{1 \dots M\}^2$, and our edges connect the 4-neighbors (similar to the grid shown in Figure 1 (left)).

Figure 7 (top) shows the performance of our method on a grid with random potentials (similar to the experiment in Section 5.2.1). Figure 7 (bottom) shows the performance of our method on an optical flow task (Lucas & Kanade, 1981). Here the states encode *flow vectors*: for a node with N states, the flow vector is assumed to take integer coordinates in the square $[-\sqrt{N}/2, \sqrt{N}/2)^2$ (so that there are N possible flow vectors). For the unary potential we have

$$\Phi_{(i,j)}(y|x) = \left\| Im_1[i,j] - Im_2[(i,j) + f(y)] \right\|, \quad (11)$$

where $Im_1[a,b]$ and $Im_2[a,b]$ return the gray-level of the pixel at (a,b) in the first and second images (respectively), and $f(y)$ returns the flow vector encoded by y. The pairwise potentials simply encode the Euclidean distance between two flow vectors.

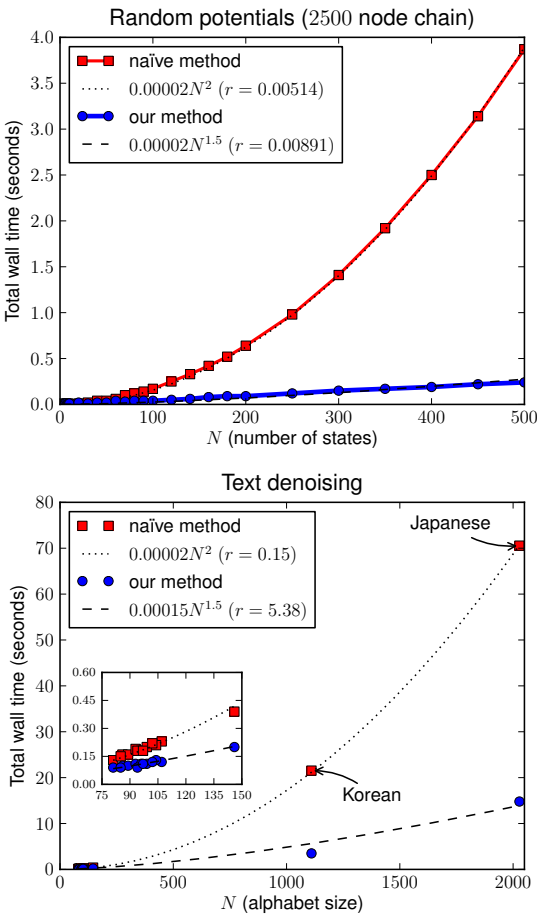


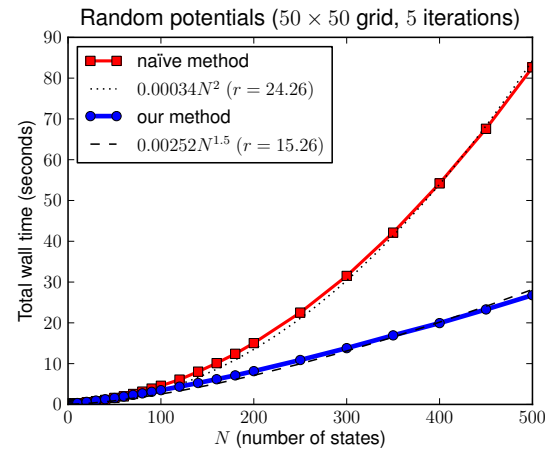
Figure 6. Running time of inference in chain-structured models: random potentials (top), and text denoising (bottom). Fitted curves confirm that the exponent of our method is indeed 1.5 (r denotes the residual error, i.e., the 'goodness' of the fitted curve).

Our fitted curves in Figure 7 show $O(N^{1.5})$ performance for both random data and for optical flow.

5.2.3. Failure Cases

In our previous experiments on text denoising and optical flow we observed running times similar to those for random potentials, indicating that there is no prevalent dependence structure between the order statistics of the messages and the potentials.

In certain applications the order statistics of these terms are highly dependent. The most straightforward example is that of *concave* potentials (or *convex* potentials in a min-sum formulation). For instance, in a



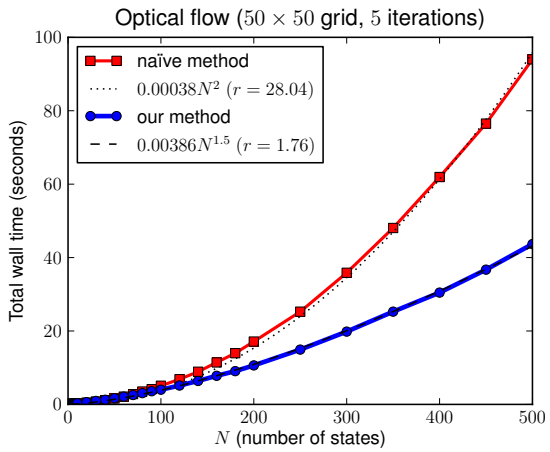
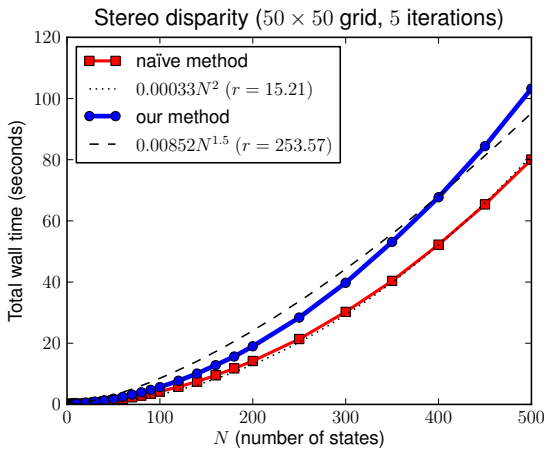


Figure 7. Running time of inference in grid-structured models: random potentials (top), and optical flow (bottom).



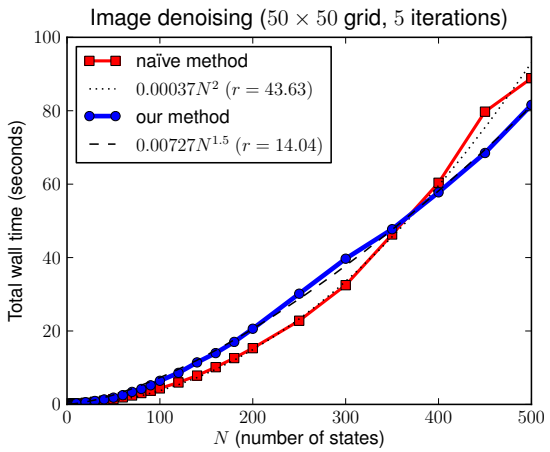


Figure 8. Two experiments whose potentials and messages have highly dependent order statistics: stereo disparity (top), and image denoising (bottom).

stereo disparity experiment, the unary potentials encode that the output should be 'close to' a certain value; the pairwise potentials encode that neighboring nodes should take similar values (Sun et al., 2003).

Whenever both $\mathbf{v}_a$ and $\mathbf{v}_b$ are concave in (eq. 7), the permutation matrix that transforms the sorted values of $\mathbf{v}_a$ to the sorted values of $\mathbf{v}_b$ is block-off-diagonal (see the sixth permutation in Figure 3). In such cases, our algorithm only decreases the number of addition operations by a multiplicative constant, and may in fact be slower due to its computational overhead. This is precisely the behavior shown in Figure 8 (top), in the case of stereo disparity.

It should be noted that there exist algorithms specifically designed for this class of potential functions (Kolmogorov & Shioura, 2007; Felzenszwalb & Huttenlocher, 2006), which are preferable in such instances.

We similarly perform an experiment on image denoising, where the unary potentials are again convex functions of the input (see Lan et al., 2006). Instead of using a pairwise potential that merely encodes smoothness, we extract the pairwise statistics from image data (similar to our experiment on text denoising); thus the potentials are no longer concave. We see in Figure 8 (bottom) that even if a small number of entries exhibit some 'randomness' in their order statistics, we begin to gain a modest speed improvement over the naïve solution (though indeed, the improvements are negligible compared to those shown in previous experiments).

6. Discussion and Future Work

At the core of our work is an $O(\sqrt{N})$ solution to (eq. 7); this solution has many applications beyond those covered in this paper. As suggested in Section 3, our analysis leads to an $O(N^{2.5})$ expected-time solution to 'funny matrix multiplication' – the analogue

of regular matrix multiplication where summation is replaced by maximization.

It can be shown that a sub-cubic solution to funny matrix multiplication has a variety of applications beyond those discussed here. For instance, it allows us to solve the all-pairs shortest path problem in $O(N^{2.5})$ (Aho et al., 1983).

We have also applied similar techniques to a different class of graphical models, by exploiting the fact that the *data-dependent* factors in triangulated graphical models often contain fewer terms than their maximal cliques. In such cases, exact inference in a junction-tree is equivalent to a generalized version of funny matrix multiplication. This leads to faster solutions to a number of computer-vision problems in which large maximal cliques factor into pairwise terms (McAuley & Caetano, 2010).

7. Conclusion

We have presented an algorithm for message passing in models whose data-dependent factors contain fewer latent variables than their data-independent factors. We find this to be useful in models with pairwise priors, as it allows us to do message passing in only $O(N^{1.5})$ for models with N states, thus substantially improving upon the standard quadratic-time solution. In practice, we find that in spite of the computational overhead of our model, speed improvements are gained even for modest values of N, resulting in substantial speedups in a variety of real-world applications.

Acknowledgements

We would like to thank Pedro Felzenszwalb, Johnicholas Hines, and David Sontag for comments on initial versions of this paper. NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program

References

Aho, Alfred V., Hopcroft, John E., and Ullman, Jeffrey D. *Data Structures and Algorithms*. Addison-Wesley, 1983.

Aji, Srinivas M. and McEliece, Robert J. The generalized distributive law. *IEEE Trans. on Information Theory*, 46(2):325–343, 2000.

Alon, Noga, Galil, Zvi, and Margalit, Oded. On the exponent of the all pairs shortest path problem. *Journal of Computer and System Sciences*, 54(2):255–262, 1997.

Felzenszwalb, Pedro F. Representation and detection of deformable shapes. *IEEE Trans. on PAMI*, 27(2): 208–220, 2005.

Felzenszwalb, Pedro F. and Huttenlocher, Daniel P. Efficient belief propagation for early vision. *IJCV*, 70(1):41–54, 2006.

Kerr, Leslie R. The effect of algebraic structure on the computational complexity of matrix multiplication. *PhD Thesis*, 1970.

Kersting, Kristian, Ahmadi, Babak, and Natarajan, Sriraam. Counting belief propagation. In *UAI*, 2009.

Kolmogorov, Vladimir and Shioura, Akiyoshi. New algorithms for the dual of the convex cost network flow problem with application to computer vision. Technical report, University College London, 2007.

Kschischang, Frank R., Frey, Brendan J., and Loeliger, Hans-Andrea. Factor graphs and the sum-product algorithm. *IEEE Trans. on Information Theory*, 47 (2):498–519, 2001.

Kumar, M. Pawan and Torr, Philip. Fast memory-efficient generalized belief propagation. In *ECCV*, 2006.

Lan, Xiang-Yang, Roth, Stefan, Huttenlocher, Daniel P., and Black, Michael J. Efficient belief propagation with learned higher-order markov random fields. In *ECCV*, 2006.

Lucas, Bruce D. and Kanade, Takeo. An iterative image registration technique with an application to stereo vision. In *IJCAI*, 1981.

McAuley, Julian J. and Caetano, Tibério S. Faster Algorithms for Max-Product Message-Passing *CoRR*, abs/0910.3301, 2009.

McAuley, Julian J. and Caetano, Tibério S. Exploiting within-clique factorizations in junction-tree algorithms. *AISTATS*, 2010.

Petersen, K., Fehr, J., and Burkhardt, H. Fast generalized belief propagation for MAP estimation on 2D and 3D grid-like markov random fields. In *DAGM*, 2008.

Quasthoff, U., Richter, M., and Biemann, C. Corpus portal for search in monolingual corpora. In *Language Resources and Evaluation*, 2006.

Sun, Jian, Zheng, Nan-Ning, and Shum, Heung-Yeung. Stereo matching using belief propagation. *IEEE Trans. on PAMI*, 25(7):787–800, 2003.

Journal of Machine Learning Research 12 (2011) 1349-1388 Submitted 4/10; Revised 1/11; Published 4/11

Faster Algorithms for Max-Product Message-Passing*

Julian J. McAuley† JULIAN.MCAULEY@NICTA.COM.AU
Tibério S. Caetano† TIBERIO.CAETANO@NICTA.COM.AU
Statistical Machine Learning Group
NICTA
Locked Bag 8001
Canberra ACT 2601, Australia

Editor: Tommi Jaakkola

Abstract

Maximum A Posteriori inference in graphical models is often solved via message-passing algorithms, such as the junction-tree algorithm or loopy belief-propagation. The exact solution to this problem is well-known to be exponential in the size of the maximal cliques of the triangulated model, while approximate inference is typically exponential in the size of the model's factors. In this paper, we take advantage of the fact that many models have maximal cliques that are larger than their constituent factors, and also of the fact that many factors consist only of latent variables (i.e., they do not depend on an observation). This is a common case in a wide variety of applications that deal with grid-, tree-, and ring-structured models. In such cases, we are able to decrease the exponent of complexity for message-passing by 0.5 for both exact *and* approximate inference. We demonstrate that message-passing operations in such models are equivalent to some variant of matrix multiplication in the tropical semiring, for which we offer an $O(N^{2.5})$ *expected-case* solution.

Keywords: graphical models, belief-propagation, tropical matrix multiplication

1. Introduction

It is well-known that exact inference in *tree-structured* graphical models can be accomplished efficiently by message-passing operations following a simple protocol making use of the distributive law (Aji and McEliece, 2000; Kschischang et al., 2001). It is also well-known that exact inference in *arbitrary* graphical models can be solved by the junction-tree algorithm; its efficiency is determined by the size of the maximal cliques after triangulation, a quantity related to the tree-width of the graph.

Figure 1 illustrates an attempt to apply the junction-tree algorithm to some graphical models containing cycles. If the graphs are not chordal ((a) and (b)), they need to be triangulated, or made chordal (red edges in (c) and (d)). Their clique-graphs are then guaranteed to be *junction-trees*, and the distributive law can be applied with the same protocol used for trees; see Aji and McEliece (2000) for a beautiful tutorial on exact inference in arbitrary graphs. Although the models in these

∗. Preliminary versions of this work appeared in The 27th International Conference on Machine Learning (ICML 2010), and the 13th International Conference on Artificial Intelligence and Statistics (AISTATS 2010), The NIPS 2009 Workshop on Learning with Orderings, The NIPS 2009 Workshop on Discrete Optimization in Machine Learning, and in Learning and Intelligent Optimization (LION 4).

†. Also at Research School of Information Sciences and Engineering, Australian National University, Canberra ACT 0200, Australia.

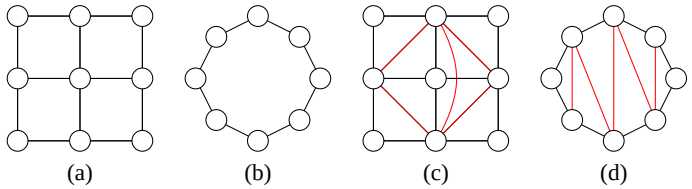


Figure 1: The models at left ((a) and (b)) can be triangulated ((c) and (d)) so that the junction-tree algorithm can be applied. Despite the fact that the new models have larger maximal cliques, the corresponding potentials are still factored over pairs of nodes only. Our algorithms exploit this fact.

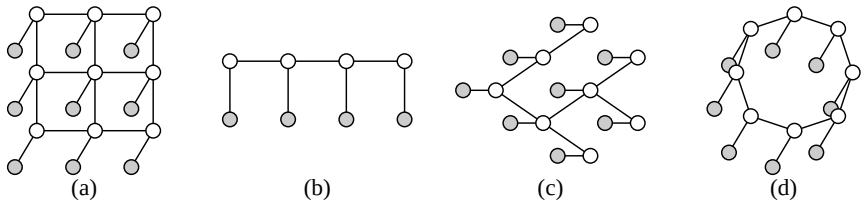


Figure 2: Some graphical models to which our results apply: *factors conditioned upon observations have fewer latent variables than purely latent factors*. White nodes correspond to latent variables, gray nodes to an observation. In other words, factors containing a gray node encode the *data likelihood*, whereas factors containing only white nodes encode *priors*. Expressed more simply, the 'node potentials' depend upon the observation, while the 'edge potentials' do not.

examples contain only pairwise factors, triangulation has increased the size of their maximal cliques, making exact inference substantially more expensive. Hence approximate solutions in the original graph (such as loopy belief-propagation, or inference in a loopy factor-graph) are often preferred over an exact solution via the junction-tree algorithm.

Even when the model's factors are the same size as its maximal cliques, neither exact nor approximate inference algorithms take advantage of the fact that many factors consist only of *latent* variables. In many models, those factors that are conditioned upon the observation contain fewer latent variables than the purely latent factors. Examples are shown in Figure 2. This encompasses a wide variety of models, including grid-structured models for optical flow and stereo disparity as well as chain and tree-structured models for text or speech.

In this paper, we exploit the fact that the maximal cliques (after triangulation) often have potentials that factor over subcliques, as illustrated in Figure 1. We will show that whenever this is the case, the expected computational complexity of message-passing between such cliques *can be improved* (both the asymptotic upper-bound and the actual runtime).

Additionally, we will show that this result can be applied in cliques *whose factors that are conditioned upon an observation* contain fewer latent variables than those factors consisting purely

of latent variables; the 'purely latent' factors can be pre-processed *offline*, allowing us to achieve the same benefits as described in the previous paragraph.

We show that these properties reveal themselves in a wide variety of real applications.

A core operation encountered in the junction-tree algorithm is that of computing the inner-product of two vectors $\mathbf{v}_a$ and $\mathbf{v}_b$. In the max-product semiring (used for MAP inference), the 'inner-product' becomes

$$\max_{i \in \{1 \ldots N\}} \{\mathbf{v}_a[i] \times \mathbf{v}_b[i]\}. \tag{1}$$

Our results stem from the realization that while (Equation 1) appears to be a *linear* time operation, it can be decreased to $O(\sqrt{N})$ (in the expected case) if we know the permutations that sort $\mathbf{v}_a$ and $\mathbf{v}_b$ (i.e., the order statistics of $\mathbf{v}_a$ and $\mathbf{v}_b$). These permutations can be obtained efficiently when the model factorizes as described above.

Preliminary versions of this work have appeared in McAuley and Caetano (2009), McAuley and Caetano (2010a), and McAuley and Caetano (2010b).

1.1 Summary of Results

A selection of the results to be presented in the remainder of this paper can be summarized as follows:

- Our speedups apply to the operation of *passing a single message*. As a result, our method can be used regardless of the message-passing protocol.
- We are able to lower the asymptotic expected running time of max-product message-passing for *any* discrete graphical model whose cliques factorize into lower-order terms.
- The results obtained are exactly those that would be obtained by the traditional version of the algorithm, that is, no approximations are used.
- Our algorithm also applies whenever factors that are conditioned upon an observation contain fewer latent variables than those factors that are not conditioned upon an observation, as in Figure 2 (in which case certain computations can be taken offline).
- For pairwise models satisfying the above properties, we obtain an expected speed-up of *at least* $\Omega(\sqrt{N})$ (assuming N states per node; Ω denotes an *asymptotic lower-bound*). For example, in models with third-order cliques containing pairwise terms, message-passing is reduced from $\Theta(N^3)$ to $O(N^2\sqrt{N})$, as in Figure 1(d). For pairwise models whose edge potential is not conditioned upon an observation, message-passing is reduced from $\Theta(N^2)$ to $O(N\sqrt{N})$, as in Figure 2.
- For cliques composed of K-ary factors, the expected speed-up generalizes to at least $\Omega(\frac{1}{K}N^{\frac{1}{K}})$, though it is *never asymptotically slower* than the original solution.
- The expected-case improvement is derived under the assumption that the order statistics of different factors are *independent*.
- If the different factors have 'similar' order statistics, the performance will be better than the expected case.

- If the different factors have 'opposite' order statistics, the performance will be worse than the expected case, but is never asymptotically more expensive than the traditional version of the algorithm.

Our results do not apply for every semiring $(\oplus, \otimes)$, but only to those whose 'addition' operation defines an order (for example, min or max); we also assume that under this ordering, our 'multiplication' operator $\otimes$ satisfies

$$a < b \wedge c < d \;\; \Rightarrow \;\; a \otimes c < b \otimes d. \tag{2}$$

Thus our results certainly apply to the *max-sum* and *min-sum* ('tropical') semirings (as well as *max-product* and *min-product*, assuming non-negative potentials), but not for *sum-product* (for example). Consequently, our approach is useful for computing MAP-states, but cannot be used to compute marginal distributions. We also assume that the domain of each node is *discrete*.

We shall initially present our algorithm in terms of *pairwise* graphical models such as those shown in Figure 2. In such models message-passing is precisely equivalent to matrix-vector multiplication over our chosen semiring. Later we shall apply our results to models such as those in Figure 1, wherein message-passing becomes some variant of matrix multiplication. Finally we shall explore other applications besides message-passing that make use of tropical matrix multiplication as a subroutine, such all-pairs shortest-path problems.

1.2 Related Work

There has been previous work on speeding-up message-passing algorithms by exploiting different types of structure in certain graphical models. For example, Kersting et al. (2009) study the case where different cliques share the same potential function. In Felzenszwalb and Huttenlocher (2006), fast message-passing algorithms are provided for cases in which the potential of a 2-clique is only dependent on the *difference* of the latent variables (which is common in some computer vision applications); they also show how the algorithm can be made faster if the graphical model is a bipartite graph. In Kumar and Torr (2006), the authors provide faster algorithms for the case in which the potentials are *truncated*, whereas in Petersen et al. (2008) the authors offer speed-ups for models that are specifically grid-like.

The latter work is perhaps the most similar in spirit to ours, as it exploits the fact that certain factors can be *sorted* in order to reduce the search space of a certain maximization problem.

Another course of research aims at speeding-up message-passing algorithms by using 'informed' scheduling routines, which may result in faster convergence than the random schedules typically used in loopy belief-propagation and inference in factor graphs (Elidan et al., 2006). This branch of research is orthogonal to our own in the sense that our methods can be applied independently of the choice of message passing protocol.

Another closely related paper is that of Park and Darwiche (2003). This work can be seen to compliment ours in the sense that it exploits essentially the same type of factorization that we study, though it applies to *sum-product* versions of the algorithm, rather than the *max-product* version that we shall study. Kjærulff (1998) also exploits factorization within cliques of junction-trees, albeit a different type of factorization than that studied here.

In Section 4, we shall see that our algorithm is closely related to a well-studied problem known as 'tropical matrix multiplication' (Kerr, 1970). The worst-case complexity of this problem has been studied in relation to the all-pairs shortest-path problem (Alon et al., 1997; Karger et al., 1993).

Example	description
$A;B$	capital letters refer to sets of nodes (or similarly, cliques);
$A\cup B;A\cap B;A\setminus B$	standard set operators are used ($A\setminus B$ denotes set difference);
$\mathrm{dom}(A)$	the domain of a set; this is just the Cartesian product of the domains of each element in the set;
$\mathbf{P}$	bold capital letters refer to arrays;
$\mathbf{x}$	bold lower-case letters refer to vectors;
$\mathbf{x}[a]$	vectors are indexed using square brackets;
$\mathbf{P}[n]$	similarly, square brackets are used to index a *row* of a 2-d array,
$\mathbf{P}[\mathbf{n}]$	or a row of an $(\lvert\mathbf{n}\rvert+1)$-dimensional array;
$\mathbf{P}^X;\mathbf{v}^a$	superscripts are just labels, that is, $\mathbf{P}^X$ is an array, $\mathbf{v}^a$ is a vector;
$\mathbf{v}_a$	*constant* subscripts are also labels, that is, if a is a constant, then $\mathbf{v}_a$ is a constant vector;
$x_i;\mathbf{x}_A$	*variable* subscripts define variables; the subscript defines the domain of the variable;
$\mathbf{n}\vert_X$	if $\mathbf{n}$ is a constant vector, then $\mathbf{n}\vert_X$ is the *restriction* of that vector to those indices corresponding to variables in X (assuming that X is an ordered set);
$\Phi_A;\Phi_A(\mathbf{x}_A)$	a function over the variables in a set A; the argument $\mathbf{x}_A$ will be suppressed if clear, given that 'functions' are essentially arrays for our purposes;
$\Phi_{i,j}(x_i,x_j)$	a function over a pair of variables (x_i,x_j);
$\Phi_A(\mathbf{n}\vert_B;\mathbf{x}_{A\setminus B})$	if one argument to a function is constant (here $\mathbf{n}\vert_B$), then it becomes a function over fewer variables (in this case, only $\mathbf{x}_{A\setminus B}$ is free);

Table 1: Notation

2. Background

The notation we shall use is briefly defined in Table 1. We shall assume throughout that the *max-product* semiring is being used, though our analysis is almost identical for any suitable choice.

MAP-inference in a graphical model $\mathcal{G}$ consists of solving an optimization problem of the form

$$\hat{\mathbf{x}} = \operatorname*{argmax}_{\mathbf{x}} \prod_{C\in\mathcal{C}} \Phi_C(\mathbf{x}_C),$$

where $\mathcal{C}$ is the set of maximal cliques in $\mathcal{G}$. This problem is often solved via *message-passing* algorithms such as the junction-tree algorithm, loopy belief-propagation, or inference in a factor-graph (Aji and McEliece, 2000; Weiss, 2000; Kschischang et al., 2001).

Often, the clique-potentials $\Phi_C(\mathbf{x}_C)$ shall be decomposable into several smaller factors, that is,

$$\Phi_C(\mathbf{x}_C) = \prod_{F\subseteq C} \Phi_F(\mathbf{x}_F).$$

Some simple motivating examples are shown in Figure 3: a model for pose estimation from Sigal and Black (2006), a 'skip-chain CRF' from Galley (2006), and a model for shape-matching from Coughlan and Ferreira (2002). In each case, the triangulated model has third-order cliques, but the potentials are only pairwise. Other examples have already been shown in Figure 1; analogous cases are ubiquitous in many real applications.

It will often be more convenient to express our objective function as being conditioned upon some *observation*, $\mathbf{y}$. Thus our optimization problem becomes

$$\hat{\mathbf{x}}(\mathbf{y}) = \operatorname*{argmax}_{\mathbf{x}} \prod_{C \in \mathcal{C}} \Phi_C(\mathbf{x}_C|\mathbf{y}) \tag{3}$$

(for simplicity when we discuss 'cliques' we are referring to sets of *latent* variables).

Further factorization may be possible if we express (Equation 3) in terms of those factors that depend upon the observation $\mathbf{y}$, and those that do not:

$$\hat{\mathbf{x}}(\mathbf{y}) = \operatorname*{argmax}_{\mathbf{x}} \prod_{C \in \mathcal{C}} \Big\{ \underbrace{\prod_{F \subseteq C} \Phi_F(\mathbf{x}_F)}_{\text{data-independent}} \times \underbrace{\prod_{Q \subseteq C} \Phi_Q(\mathbf{x}_Q|\mathbf{y})}_{\text{data-dependent}} \Big\},$$

We shall say that those factors that are not conditioned on the observation are 'data-independent'.

Our results shall apply to message-passing equations in those cliques C where for each data-independent factor F we have $F \subset C$, *or* for each data-dependent factor Q we have $Q \subset C$, that is, when all F or all Q in C are *proper* subsets of C. In such cases we say that the clique C is *factorizable*.

The fundamental step encountered in message-passing algorithms is defined below. The message from a clique X to an intersecting clique Y (both sets of *latent* variables) is defined by

$$m_{X \to Y}(\mathbf{x}_{X \cap Y}) = \max_{\mathbf{x}_{X \setminus Y}} \left\{ \Phi_X(\mathbf{x}_X) \prod_{Z \in \Gamma(X) \setminus Y} m_{Z \to X}(\mathbf{x}_{X \cap Z}) \right\} \tag{4}$$

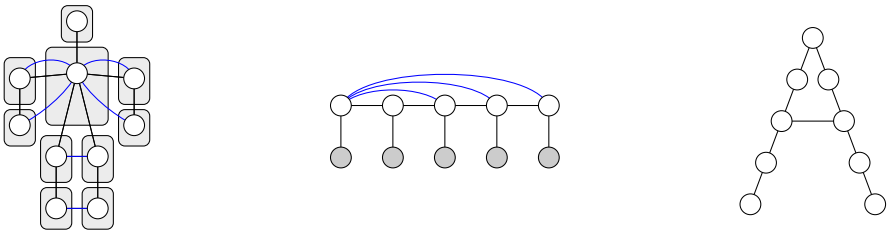
(where $\Gamma(X)$ is the set of neighbors of the clique X, that is, the set of cliques that intersect with X). If such messages are computed after X has received messages from all of its neighbors except Y (i.e., $\Gamma(X) \setminus Y$), then this defines precisely the update scheme used by the junction-tree algorithm. The same update scheme is used for loopy belief-propagation, though it is done iteratively in a randomized fashion.

After all messages have been passed, the MAP-state for a set of latent variables M (assumed to be a subset of a single clique X) is computed using

$$m_M(\mathbf{x}_M) = \max_{\mathbf{x}_{X \setminus M}} \left\{ \Phi_X(\mathbf{x}_X) \prod_{Z \in \Gamma(X)} m_{Z \to X}(\mathbf{x}_{X \cap Z}) \right\}. \tag{5}$$

For cliques that are *factorizable* (according to our previous definition), both (Equation 4) and (Equation 5) take the form

$$m_M(\mathbf{x}_M) = \max_{\mathbf{x}_{X \setminus M}} \left\{ \prod_{F \subseteq X} \Phi_F(\mathbf{x}_F) \prod_{Q \subseteq X} \Phi_Q(\mathbf{x}_Q|\mathbf{y}) \right\}. \tag{6}$$



(a) (b) (c)

Figure 3: (a) A model for pose reconstruction from Sigal and Black (2006); (b) A 'skip-chain CRF' from Galley (2006); (c) A model for deformable matching from Coughlan and Ferreira (2002). Although the (triangulated) models have cliques of size three, their potentials factorize into pairwise terms.

Note that we always have $Z \cap X \subset X$ for messages $Z \to X$, meaning that the presence of the messages has no effect on the 'factorizability' of (Equation 6).

Algorithm 1 gives the traditional solution to this problem, which does not exploit the factorization of $\Phi_X(\mathbf{x}_X)$. This algorithm runs in $\Theta(N^{|X|})$, where N is the number of states per node, and $|X|$ is the size of the clique X (for a given $\mathbf{x}_X$, we treat computing $\prod_{F \subset X} \Phi_F(\mathbf{x}_F)$ as a constant time operation, as our optimizations shall not modify this cost).

In the following sections, we shall consider the two types of factorizability separately: first, in Section 3, we shall consider cliques X whose messages take the form

$$m_M(\mathbf{x}_M) = \max_{\mathbf{x}_{X \setminus M}} \left\{ \Phi_X(\mathbf{x}_X) \prod_{Q \subset X} \Phi_Q(\mathbf{x}_Q|\mathbf{y}) \right\}.$$

We say that such cliques are *conditionally factorizable* (since all conditional terms factorize); examples are shown in Figure 2. Next, in Section 4, we consider cliques whose messages take the form

$$m_M(\mathbf{x}_M) = \max_{\mathbf{x}_{X \setminus M}} \prod_{F \subset X} \Phi_F(\mathbf{x}_F).$$

We say that such cliques are *latently factorizable* (since terms containing only latent variables factorize); examples are shown in Figure 1.

3. Optimizing Algorithm 1: Conditionally Factorizable Models

In order to specify a more efficient version of Algorithm 1, we begin by considering the simplest nontrivial *conditionally factorizable* model: a pairwise model in which each latent variable depends upon the observation, that is,

$$\hat{\mathbf{x}}(\mathbf{y}) = \operatorname*{argmax}_{\mathbf{x}} \underbrace{\prod_{i \in \mathcal{N}} \Phi_i(x_i|y)}_{\text{node potential}} \times \underbrace{\prod_{(i,j) \in \mathcal{E}} \Phi_{i,j}(x_i, x_j)}_{\text{edge potential}}. \tag{7}$$

This is the type of model depicted in Figure 2 and encompasses a large class of grid- and tree-structured models. Using our previous definitions, we say that the node potentials are 'data-dependent', whereas the edge potentials are 'data-independent'.

Algorithm 1 Brute-force computation of max-marginals

Input: a clique X whose max-marginal $m_M(\mathbf{x}_M)$ (where $M \subset X$) we wish to compute; assume that each node in X has domain $\{1 \dots N\}$

for $\mathbf{m} \in \mathrm{dom}(M)$ {i.e., $\{1 \dots N\}^{|M|}$} **do**
  $max := -\infty$
  for $\mathbf{z} \in \mathrm{dom}(X \setminus M)$ **do**
    if $\prod_{F \subset X} \Phi_F(\mathbf{m}|_F; \mathbf{z}|_F) > max$ **then**
      $max := \prod_{F \subset X} \Phi_F(\mathbf{m}|_F; \mathbf{z}|_F)$
    end if
  end for {this loop takes $\Theta(N^{|X \setminus M|})$}
  $m_M(\mathbf{m}) := max$
end for {this loop takes $\Theta(N^{|X|})$}
Return: m_M

Message-passing in models of the type shown in (Equation 7) takes the form

$$m_{A \to B}(x_i) = \Phi_i(x_i|y) \times \max_{x_j} \Phi_j(x_j|y) \times \Phi_{i,j}(x_i, x_j) \tag{8}$$

(where $A = \{i, j\}$ and $B = \{i, k\}$). Note once again that in (Equation 8) we are not concerned solely with exact inference via the junction-tree algorithm. In many models, such as grids and rings, (Equation 7) shall be solved *approximately* by means of either loopy belief-propagation, or inference in a factor-graph, which consists of solving (Equation 8) according to protocols other than the optimal junction-tree protocol.

It is useful to consider $\Phi_{i,j}$ in (Equation 8) as an $N \times N$ *matrix*, and Φ_j as an N-dimensional *vector*, so that solving (Equation 8) is precisely equivalent to matrix-vector multiplication in the max-product semiring. For a particular value $x_i = q$, (Equation 8) becomes

$$m_{A \to B}(q) = \Phi_i(q|y) \times \max_{x_j} \underbrace{\Phi_j(x_j|y)}_{\mathbf{v}_a} \times \underbrace{\Phi_{i,j}(q, x_j)}_{\mathbf{v}_b}, \tag{9}$$

which is precisely the 'max-product inner-product' operation that we claimed was critical in Section 1.

As we have previously suggested, it will be possible to solve (Equation 9) efficiently if we know the order statistics of $\mathbf{v}_a$ and $\mathbf{v}_b$, that is, if we know the permutations that sort Φ_j and every row of $\Phi_{i,j}$ in (Equation 8). Sorting Φ_j takes $\Theta(N \log N)$, whereas sorting every row of $\Phi_{i,j}$ takes $\Theta(N^2 \log N)$ ($\Theta(N \log N)$ for each of N rows). The critical point to be made is that $\Phi_{i,j}(x_i, x_j)$ *does not depend on the observation*, meaning that its order statistics can be obtained *offline* in several applications.

The following elementary lemma is the key observation required in order to solve (Equation 1), and therefore (Equation 9) efficiently:

Lemma 1 *For any index q, the solution to $p = \mathrm{argmax}_{i \in \{1 \dots N\}} \{\mathbf{v}_a[i] \times \mathbf{v}_b[i]\}$ must have $\mathbf{v}_a[p] \geq \mathbf{v}_a[q]$ or $\mathbf{v}_b[p] \geq \mathbf{v}_b[q]$. Therefore, having computed $\mathbf{v}_a[q] \times \mathbf{v}_b[q]$, we can find 'p' by computing only those products $\mathbf{v}_a[i] \times \mathbf{v}_b[i]$ where either $\mathbf{v}_a[i] > \mathbf{v}_a[q]$ or $\mathbf{v}_b[i] > \mathbf{v}_b[q]$.*

Algorithm 2 Find i such that $\mathbf{v}_a[i] \times \mathbf{v}_b[i]$ is maximized

Input: two vectors $\mathbf{v}_a$ and $\mathbf{v}_b$, and permutation functions p_a and p_b that sort them in decreasing order (so that $\mathbf{v}_a[p_a[1]]$ is the largest element in $\mathbf{v}_a$)

1: **Initialize:** $start \coloneqq 1$, $end_a \coloneqq p_a^{-1}[p_b[1]]$, $end_b \coloneqq p_b^{-1}[p_a[1]]$ {if $end_b = k$, then the largest element in $\mathbf{v}_a$ has the same index as the k^{th} largest element in $\mathbf{v}_b$}
2: $best \coloneqq p_a[1]$, $max \coloneqq \mathbf{v}_a[best] \times \mathbf{v}_b[best]$
3: **if** $\mathbf{v}_a[p_b[1]] \times \mathbf{v}_b[p_b[1]] > max$ **then**
4: $best \coloneqq p_b[1]$, $max \coloneqq \mathbf{v}_a[best] \times \mathbf{v}_b[best]$
5: **end if**
6: **while** $start < end_a$ {in practice, we could also stop if $start < end_b$, but the version given here is the one used for analysis in Appendix A} **do**
7: $start \coloneqq start + 1$
8: **if** $\mathbf{v}_a[p_a[start]] \times \mathbf{v}_b[p_a[start]] > max$ **then**
9: $best \coloneqq p_a[start]$
10: $max \coloneqq \mathbf{v}_a[best] \times \mathbf{v}_b[best]$
11: **end if**
12: **if** $p_b^{-1}[p_a[start]] < end_b$ **then**
13: $end_b \coloneqq p_b^{-1}[p_a[start]]$
14: **end if**
15: {repeat lines 8–14, interchanging a and b}
16: **end while** {this loop takes *expected time* $O(\sqrt{N})$}
17: **Return:** *best*

This observation is used to construct Algorithm 2. Here we iterate through the indices starting from the largest values of $\mathbf{v}_a$ and $\mathbf{v}_b$, stopping once both indices are 'behind' the maximum value found so far (which we then know is the maximum). This algorithm is demonstrated pictorially in Figure 4. Note that Lemma 1 only depends upon the *relative* values of elements in $\mathbf{v}_a$ and $\mathbf{v}_b$, meaning that the number of computations that must be performed is purely a function of their *order statistics* (i.e., it does not depend on the actual values of $\mathbf{v}_a$ or $\mathbf{v}_b$).

If Algorithm 2 can solve (Equation 9) in $O(f(N))$, then we can solve (Equation 8) in $O(Nf(N))$. Determining precisely the running time of Algorithm 2 is not trivial, and will be explored in depth in Appendix A. At this stage we shall state an upper-bound on the true complexity in the following theorem:

Theorem 2 *The* expected *running time of Algorithm 2 is* $O(\sqrt{N})$*, yielding a speed-up of at least* $\Omega(\sqrt{N})$ *in cliques containing pairwise factors. This expectation is derived under the assumption that* $\mathbf{v}_a$ *and* $\mathbf{v}_b$ *have independent order statistics.*

Algorithm 3 uses Algorithm 2 to solve (Equation 8), where we assume that the order statistics of the rows of $\Phi_{i,j}$ have been obtained offline.

While the offline cost of sorting is not problematic in situations where the model is to be repeatedly reused on several observations, it can be avoided in two situations. Firstly, many models have a 'homogeneous' prior, that is, the same prior is shared amongst every edge (or clique) of the model. In such cases, only a single 'copy' of the prior needs to be sorted, meaning that in any model

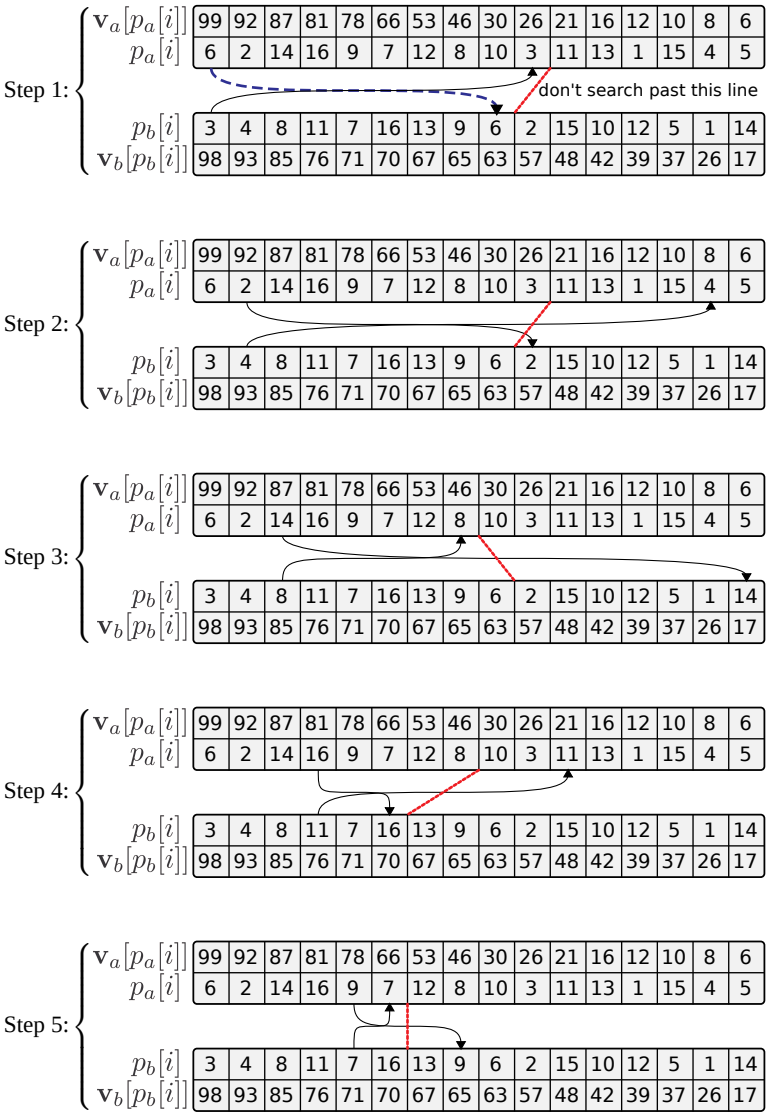


Figure 4: Algorithm 2, explained pictorially. The arrows begin at $p_a[start]$ and $p_b[start]$; the red dashed line connects end_a and end_b, behind which we need not search; a dashed arrow is used when a new maximum is found. Note that in the event that $\mathbf{v}_a$ and $\mathbf{v}_b$ contain repeated elements, they can be sorted arbitrarily.

containing $\Omega(\log N)$ edges, speed improvements can be gained over the naïve implementation. Secondly, where an iterative algorithm (such as loopy belief-propagation) is to be used, the sorting step need only take place prior to the *first* iteration; if $\Omega(\log N)$ iterations of belief-propagation are to be performed (or in a homogeneous model, if the number of edges multiplied by the number of

Algorithm 3 Solve (Equation 8) using Algorithm 2

Input: a potential $\Phi_{i,j}(a,b) \times \Phi_i(a|y_i) \times \Phi_j(b|y_j)$ whose max-marginal $m_i(x_i)$ we wish to compute, and a set of permutation functions $\mathbf{P}$ such that $\mathbf{P}[i]$ sorts the i^{th} row of $\Phi_{i,j}$ (in decreasing order).

1: compute the permutation function p_a by sorting Ψ_j {takes $\Theta(N \log N)$}
2: **for** $q \in \{1 \dots N\}$ **do**
3: $(\mathbf{v}_a, \mathbf{v}_b) \coloneqq (\Psi_j, \Phi_{i,j}(q, x_j | y_i, y_j))$
4: $best \coloneqq Algorithm2(\mathbf{v}_a, \mathbf{v}_b, p_a, \mathbf{P}[q])$ {$O(\sqrt{N})$}
5: $m_{A \to B}(q) \coloneqq \Phi_i(q) \times \Phi_j(best) \times \Phi_{i,j}(q, best | y_i, y_j)$
6: **end for** {this loop takes *expected time* $O(N\sqrt{N})$}
7: **Return:** $m_{A \to B}$

iterations is $\Omega(\log N)$), we shall again gain speed improvements even when the sorting step is done online.

In fact, the second of these conditions obviates the need for 'conditional factorizability' (or 'data-independence') altogether. In other words, in *any* pairwise model in which $\Omega(\log N)$ iterations of belief-propagation are to be performed, *the pairwise terms need to be sorted only during the first iteration*. Thus these improvements apply to those models in Figure 1, so long as the number of iterations of belief-propagation is $\Omega(\log N)$.

4. Latently Factorizable Models

Just as we considered the simplest *conditionally factorizable* model in Section 3, we now consider the simplest nontrivial *latently factorizable* model: a clique of size three containing pairwise factors. In such a case, our aim is to compute

$$m_{i,j}(x_i, x_j) = \max_{x_k} \Phi_{i,j,k}(x_i, x_j, x_k), \tag{10}$$

which we have assumed takes the form

$$m_{i,j}(x_i, x_j) = \max_{x_k} \Phi_{i,j}(x_i, x_j) \times \Phi_{i,k}(x_i, x_k) \times \Phi_{j,k}(x_j, x_k).$$

For a particular value of $(x_i, x_j) = (a, b)$, we must solve

$$m_{i,j}(a,b) = \Phi_{i,j}(a,b) \times \max_{x_k} \underbrace{\Phi_{i,k}(a, x_k)}_{\mathbf{v}_a} \times \underbrace{\Phi_{j,k}(b, x_k)}_{\mathbf{v}_b}, \tag{11}$$

which again is in precisely the form shown in (Equation 1).

Just as (Equation 8) resembled matrix-vector multiplication, there is a close resemblance between (Equation 11) and the problem of matrix-matrix multiplication in the max-product semiring (often referred to as 'tropical matrix multiplication', 'funny matrix multiplication', or simply 'max-product matrix multiplication'). While traditional matrix multiplication is well-known to have a subcubic worst-case solution (see Strassen, 1969), the version in (Equation 11) has no known subcubic solution (the fastest known solution is $O(N^3/\log N)$, but there is no known solution that runs in $O(N^{3-\varepsilon})$ (Chan, 2007); Kerr (1970) shows that no subcubic solution exists under certain models of computation). The worst-case complexity of solving (Equation 11) can also be shown to be

Algorithm 4 Use Algorithm 2 to compute the max-marginal of a 3-clique containing pairwise factors

Input: a potential $\Phi_{i,j,k}(a,b,c) = \Phi_{i,j}(a,b) \times \Phi_{i,k}(a,c) \times \Phi_{j,k}(b,c)$ whose max-marginal $m_{i,j}(x_i,x_j)$ we wish to compute

1: **for** $n \in \{1 \dots N\}$ **do**
2: compute $\mathbf{P}^i[n]$ by sorting $\Phi_{i,k}(n,x_k)$ {takes $\Theta(N \log N)$}
3: compute $\mathbf{P}^j[n]$ by sorting $\Phi_{j,k}(n,x_k)$ {$\mathbf{P}^i$ and $\mathbf{P}^j$ are $N \times N$ arrays, each row of which is a permutation; $\Phi_{i,k}(n,x_k)$ and $\Phi_{j,k}(n,x_k)$ are functions over x_k, since n is constant in this expression}
4: **end for** {this loop takes $\Theta(N^2 \log N)$}
5: **for** $(a,b) \in \{1 \dots N\}^2$ **do**
6: $(\mathbf{v}_a, \mathbf{v}_b) \coloneqq \left(\Phi_{i,k}(a,x_k), \Phi_{j,k}(b,x_k)\right)$
7: $(p_a, p_b) \coloneqq \left(\mathbf{P}^i[a], \mathbf{P}^j[b]\right)$
8: $best \coloneqq Algorithm2\,(\mathbf{v}_a, \mathbf{v}_b, p_a, p_b)$ {takes $O(\sqrt{N})$}
9: $m_{i,j}(a,b) \coloneqq \Phi_{i,j}(a,b) \times \Phi_{i,k}(a,best) \times \Phi_{j,k}(b,best)$
10: **end for** {this loop takes $O(N^2\sqrt{N})$}
{the total running time is $O(N^2 \log N + N^2\sqrt{N})$, which is dominated by $O(N^2\sqrt{N})$}
11: **Return:** $m_{i,j}$

equivalent to the all-pairs shortest-path problem, which is studied in Alon et al. (1997). Although we shall not improve the worst-case complexity, Algorithm 2 leads to far better *expected-case* performance than existing solutions.

In principle Strassen's algorithm could be used to perform *sum-product* inference in the setting we discuss here, and indeed there has been some work on performing sum-product inference in graphical models that factorize (Park and Darwiche, 2003). Interestingly, there is also a sub-quadratic solution to sum-product matrix-vector multiplication that requires preprocessing (Williams, 2007), that is, the sum-product version of the setting we discussed in Section 3.

A prescription of how Algorithm 2 can be used to solve (Equation 10) is given in Algorithm 4. As we mentioned in Section 3, the expected-case running time of Algorithm 2 is $O(\sqrt{N})$, meaning that the time taken to solve Algorithm 4 is $O(N^2\sqrt{N})$.

5. Extensions

So far we have only considered the case of *pairwise* graphical models, though as mentioned our results can in principle be applied to any conditionally or latently factorizable models, no matter the size of the factors. Essentially our results about matrices become results about tensors. We first treat latently factorizable models, after which the same ideas can be applied to conditionally factorizable models.

5.1 An Extension to Higher-Order Cliques with Three Factors

The simplest extension that we can make to Algorithms 2, 3, and 4 is to note that they can be applied even when there are several overlapping terms in the factors. For instance, Algorithm 4 can

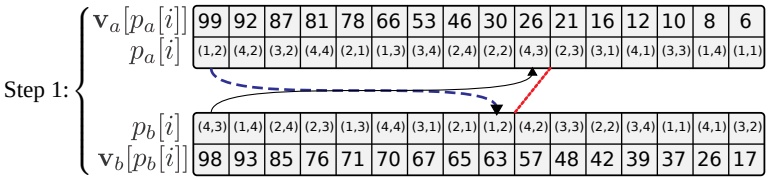


Figure 5: The reasoning applied in Algorithm 2 applies even when the elements of p_a and p_b are multidimensional indices.

be adapted to solve

$$m_{i,j}(x_i, x_j) = \max_{x_k, x_m} \Phi_{i,j}(x_i, x_j) \times \Phi_{i,k,m}(x_i, x_k, x_m) \times \Phi_{j,k,m}(x_j, x_k, x_m), \tag{12}$$

and similar variants containing three factors. Here both x_k and x_m are shared by $\Phi_{i,k,m}$ and $\Phi_{j,k,m}$. We can follow precisely the reasoning of the previous section, except that when we sort $\Phi_{i,k,m}$ (similarly $\Phi_{j,k,m}$) for a fixed value of x_i, we are now sorting an *array* rather than a *vector* (Algorithm 4, lines 2 and 3); in this case, the permutation functions p_a and p_b in Algorithm 2 simply return *pairs* of indices. This is illustrated in Figure 5. Effectively, in this example we are sorting the variable $x_{k,m}$ whose domain is $\mathrm{dom}(x_k) \times \mathrm{dom}(x_m)$, which has state space of size N^2.

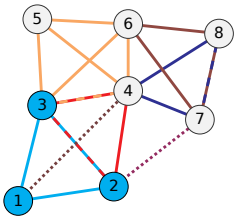
As the number of shared terms increases, so does the improvement to the running time. While (Equation 12) would take $\Theta(N^4)$ to solve using Algorithm 1, it takes only $O(N^3)$ to solve using Algorithm 4 (more precisely, if Algorithm 2 takes $O(f(N))$, then (Equation 12) takes $O(N^2 f(N^2))$, which we have mentioned is $O(N^2\sqrt{N^2}) = O(N^3)$). In general, if we have S shared terms, then the running time is $O(N^2\sqrt{N^S})$, yielding a speed-up of $\Omega(\sqrt{N^S})$ over the naïve solution of Algorithm 1.

5.2 An Extension to Higher-Order Cliques with Decompositions Into Three Groups

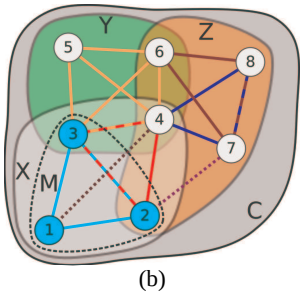
By similar reasoning, we can apply our algorithm to cases where there are more than three factors, in which the factors can be separated into three *groups*. For example, consider the clique in Figure 6(a), which we shall call G (the entire graph is a clique, but for clarity we only draw an edge when the corresponding nodes belong to a common factor). Each of the factors in this graph have been labeled using either differently colored edges (for factors of size larger than two) or dotted edges (for factors of size two), and the max-marginal we wish to compute has been labeled using colored nodes. We assume that it is possible to split this graph into three groups such that every factor is contained within a single group, along with the max-marginal we wish to compute (Figure 6, (b)). If such a decomposition is not possible, we will have to resort to further extensions to be described in Section 5.3.

Ideally, we would like these groups to have size $\simeq |G|/3$, though in the worst case they will have size no larger than $|G| - 1$. We call these groups X, Y, Z, where X is the group containing the max-marginal M that we wish to compute. In order to simplify the analysis of this algorithm, we shall express the running time in terms of the size of the largest group, $S = \max(|X|, |Y|, |Z|)$, and the largest difference, $S_\setminus = \max(|Y \setminus X|, |Z \setminus X|)$. The max-marginal can be computed using Algorithm 5.

The running times shown in Algorithm 5 are loose upper-bounds, given for the sake of expressing the running time in simple terms. More precise running times are given in Table 2; any of the

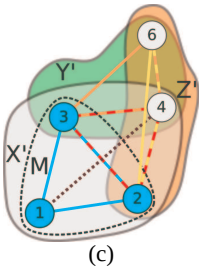


(a)

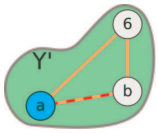


(b)

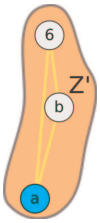
(a) We begin with a set of factors (indicated using colored lines), which are assumed to belong to some clique in our model; we wish to compute the max-marginal with respect to one of these factors (indicated using colored nodes); (b) The factors are split into three groups, such that every factor is entirely contained within one of them (Algorithm 5, line 1).



(c)

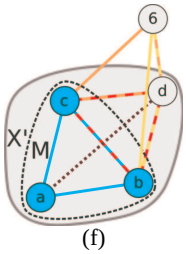


(d)

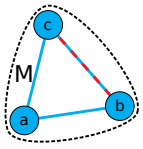


(e)

(c) Any nodes contained in only one of the groups are marginalized (Algorithm 5, lines 2, 3, and 4); the problem is now very similar to that described in Algorithm 4, except that *nodes* have been replaced by *groups*; note that this essentially introduces maximal factors in Y' and Z'; (d) For every value $(a,b) \in \text{dom}(x_3,x_4)$, $\Psi^Y(a,b,x_6)$ is sorted (Algorithm 5, lines 5–7); (e) For every value $(a,b) \in \text{dom}(x_2,x_4)$, $\Psi^Z(a,b,x_6)$ is sorted (Algorithm 5, lines 8–10).



(f)



(g)

(f) For every $\mathbf{n} \in \text{dom}(X')$, we choose the best value of x_6 by Algorithm 2 (Algorithm 5, lines 11–16); (g) The result is marginalized with respect to M (Algorithm 5, line 17).

Figure 6: Algorithm 5, explained pictorially. In this case, the most computationally intensive step is the marginalization of Z (in step (c)), which takes $\Theta(N^5)$. However, the algorithm can actually be applied *recursively* to the group Z, resulting in an overall running time of $O(N^4\sqrt{N})$, for a max-marginal that would have taken $\Theta(N^8)$ to compute using the naïve solution of Algorithm 1.

Algorithm 5 Compute the max-marginal of G with respect to M, where G is split into three groups

Input: potentials $\Phi_G(\mathbf{x}) = \Phi_X(\mathbf{x}_X) \times \Phi_Y(\mathbf{x}_Y) \times \Phi_Z(\mathbf{x}_Z)$; each of the factors should be contained in exactly one of these terms, and we assume that $M \subseteq X$ (see Figure 6)

1: **Define:** $X' \coloneqq ((Y \cup Z) \cap X) \cup M$; $Y' \coloneqq (X \cup Z) \cap Y$; $Z' \coloneqq (X \cup Y) \cap Z$ {X' contains the variables in X that are shared by at least one other group; alternately, the variables in $X \setminus X'$ appear only in X (sim. for Y' and Z')}

2: compute $\Psi^X(\mathbf{x}_{X'}) \coloneqq \max_{X \setminus X'} \Phi_X(\mathbf{x}_X)$ {we are marginalizing over those variables in X that do not appear in any of the other groups (or in M); this takes $\Theta(N^S)$ if done by brute-force (Algorithm 1), but may also be done by a recursive call to Algorithm 5}

3: compute $\Psi^Y(\mathbf{x}_{Y'}) \coloneqq \max_{Y \setminus Y'} \Phi_Y(\mathbf{x}_Y)$

4: compute $\Psi^Z(\mathbf{x}_{Z'}) \coloneqq \max_{Z \setminus Z'} \Phi_Z(\mathbf{x}_Z)$

5: **for** $\mathbf{n} \in \mathrm{dom}(X \cap Y)$ **do**

6: compute $\mathbf{P}^Y[\mathbf{n}]$ by sorting $\Psi^Y(\mathbf{n}; \mathbf{x}_{Y' \setminus X})$ {takes $\Theta(S_\setminus N^{S_\setminus} \log N)$; $\Psi^Y(\mathbf{n}; \mathbf{x}_{Y' \setminus X})$ is free over $\mathbf{x}_{Y' \setminus X}$, and is treated as an array by 'flattening' it; $\mathbf{P}^Y[\mathbf{n}]$ contains the $|Y' \setminus X| = |(Y \cap Z) \setminus X|$-dimensional indices that sort it}

7: **end for** {this loop takes $\Theta(S_\setminus N^S \log N)$}

8: **for** $\mathbf{n} \in \mathrm{dom}(X \cap Z)$ **do**

9: compute $\mathbf{P}^Z[\mathbf{n}]$ by sorting $\Psi^Z(\mathbf{n}; \mathbf{x}_{Z' \setminus X})$

10: **end for** {this loop takes $\Theta(S_\setminus N^S \log N)$}

11: **for** $\mathbf{n} \in \mathrm{dom}(X')$ **do**

12: $(\mathbf{v}_a, \mathbf{v}_b) \coloneqq \left(\Psi^Y(\mathbf{n}|_{Y'}; \mathbf{x}_{Y' \setminus X'}), \Psi^Z(\mathbf{n}|_{Z'}; \mathbf{x}_{Z' \setminus X'})\right)$ {$\mathbf{n}|_{Y'}$ is the 'restriction' of the vector $\mathbf{n}$ to those indices in Y' (meaning that $\mathbf{n}|_{Y'} \in \mathrm{dom}(X' \cap Y')$); hence $\Psi^Y(\mathbf{n}|_{Y'}; \mathbf{x}_{Y' \setminus X'})$ is free in $\mathbf{x}_{Y' \setminus X'}$, while $\mathbf{n}|_{Y'}$ is fixed}

13: $(p_a, p_b) \coloneqq \left(\mathbf{P}^Y[\mathbf{n}|_{Y'}], \mathbf{P}^Z[\mathbf{n}|_{Z'}]\right)$

14: *best* $\coloneqq$ *Algorithm2*$(\mathbf{v}_a, \mathbf{v}_b, p_a, p_b)$ {takes $O(\sqrt{S_\setminus})$}

15: $m_X(\mathbf{n}) \coloneqq \Psi^X(\mathbf{n}) \times \Psi^Y(\textit{best}; \mathbf{n}|_{Y'}) \times \Psi^Z(\textit{best}; \mathbf{n}|_{Z'})$

16: **end for**

17: $m_M(\mathbf{x}_M) \coloneqq$ *Algorithm1*(m_X, M) {i.e., we are using Algorithm 1 to marginalize $m_X(\mathbf{x}_X)$ with respect to M; this takes $\Theta(N^S)$}

terms shown in Table 2 may be dominant. Some example graphs, and their resulting running times are shown in Figure 7.

5.2.1 APPLYING ALGORITHM 5 RECURSIVELY

The marginalization steps of Algorithm 5 (lines 2, 3, and 4) may further decompose into smaller groups, in which case Algorithm 5 can be applied recursively. For instance, the graph in Figure 7(a) represents the marginalization step that is to be performed in Figure 6(c) (Algorithm 5, line 4). Since this marginalization step is the asymptotically dominant step in the algorithm, applying Algorithm 5 recursively lowers the asymptotic complexity.

Another straightforward example of applying recursion in Algorithm 5 is shown in Figure 8, in which a ring-structured model is marginalized with respect to two of its nodes. Doing so takes $O(MN^2\sqrt{N})$; in contrast, solving the same problem using the junction-tree algorithm (by triangulating the graph) would take $\Theta(MN^3)$. Loopy belief-propagation takes $\Theta(MN^2)$ per iteration, meaning

Description	lines	time
Marginalization of Φ_X, without recursion	2	$\Theta(N^{\lvert X\rvert})$
Marginalization of Φ_Y	3	$\Theta(N^{\lvert Y\rvert})$
Marginalization of Φ_Z	4	$\Theta(N^{\lvert Z\rvert})$
Sorting Φ_Y	5–7	$\Theta(\lvert Y'\backslash X\rvert N^{\lvert Y'\rvert}\log N)$
Sorting Φ_Z	8–10	$\Theta(\lvert Z'\backslash X\rvert N^{\lvert Z'\rvert}\log N)$
Running Algorithm 2 on the sorted values	11–16	$O(N^{\lvert X'\rvert}\sqrt{N^{\lvert(Y'\cap Z')\backslash X'\rvert}})$

Table 2: Detailed running time analysis of Algorithm 5; any of these terms may be asymptotically dominant

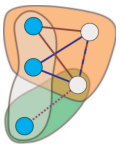
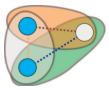
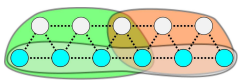
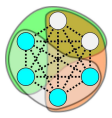
Graph:					{A complete graph K_M, with pairwise terms}
	(a)	(b)	(c)	(d)	(e)
Algorithm 1:	$\Theta(N^5)$	$\Theta(N^3)$	$\Theta(N^{11})$	$\Theta(N^6)$	$\Theta(N^M)$
Algorithm 5:	$O(N^3\sqrt{N})$	$O(N^2\sqrt{N})$	$O(N^6\sqrt{N})$	$O(N^5)$	$O(N^{5M/6})$
Speed-up:	$\Omega(N\sqrt{N})$	$\Omega(\sqrt{N})$	$\Omega(N^4\sqrt{N})$	$\Omega(N)$	$\Omega(N^{M/6})$

Figure 7: Some example graphs whose max-marginals are to be computed with respect to the colored nodes, using the three regions shown. Factors are indicated using differently colored edges, while dotted edges always indicate pairwise factors. (a) is the region Z from Figure 6 (recursion is applied *again* to achieve this result); (b) is the graph used to motivate Algorithm 4; (c) shows a query in a graph with regular structure; (d) shows a complete graph with six nodes; (e) generalizes this to a clique with M nodes.

that our algorithm will be faster if the number of iterations is $\Omega(\sqrt{N})$. Naturally, Algorithm 4 could be applied directly to the triangulated graph, which would again take $O(MN^2\sqrt{N})$.

5.3 A General Extension to Higher-Order Cliques

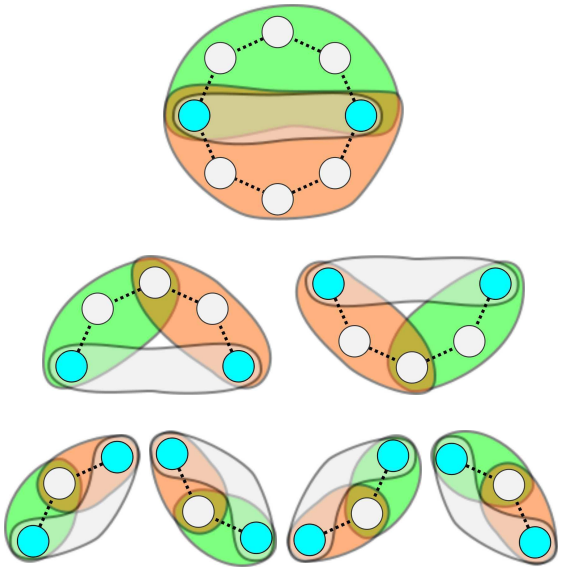
Naturally, there are cases for which a decomposition into three terms is not possible, such as

$$m_{i,j,k}(x_i,x_j,x_k) = \max_{x_m} \Phi_{i,j,k}(x_i,x_j,x_k) \times \Phi_{i,j,m}(x_i,x_j,x_m) \times \Phi_{i,k,m}(x_i,x_k,x_m) \times \Phi_{j,k,m}(x_j,x_k,x_m) \quad (13)$$

(i.e., a clique of size four with all possible third-order factors). However, if the model contains factors of size K, it must always be possible to split it into $K+1$ groups (e.g., four in the case of Equation 13).

Our optimizations can easily be applied in these cases simply by adapting Algorithm 2 to solve problems of the form

$$\max_{i\in\{1\ldots N\}} \{\mathbf{v}_1[i] \times \mathbf{v}_2[i] \times \cdots \times \mathbf{v}_K[i]\}. \quad (14)$$



$O(N^2)$

$+$

$O(2N^2\sqrt{N})$

$+$

$O(4N^2\sqrt{N})$ (by Algorithm 4)

Figure 8: In the above example, lines 2–4 of Algorithm 5 are applied recursively, achieving a total running time of $O(MN^2\sqrt{N})$ for a loop with M nodes (our algorithm achieves the same running time in the triangulated graph).

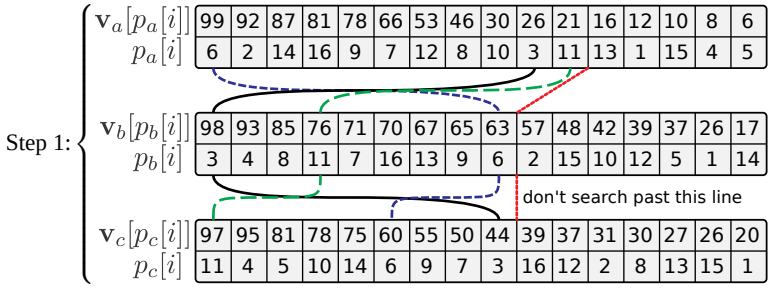


Figure 9: Algorithm 2 can easily be extended to cases including more than two sequences.

Pseudocode for this extension is presented in Algorithm 6. Note carefully the use of the variable *read*: we are storing which indices have been read to avoid re-reading them; this guarantees that our Algorithm is never asymptotically worse than the naïve solution. Figure 9 demonstrates how such an algorithm behaves in practice. Again, we shall discuss the running time of this extension in Appendix A. For the moment, we state the following theorem:

Theorem 3 *Algorithm 6 generalizes Algorithm 2 to K lists with an expected running time of* $O(KN^{\frac{K-1}{K}})$, *yielding a speed-up of at least* $\Omega(\frac{1}{K}N^{\frac{1}{K}})$ *in cliques containing K-ary factors. It is never worse than the naïve solution, meaning that it takes* $O(\min(N, KN^{\frac{K-1}{K}}))$.

Using Algorithm 6, we can similarly extend Algorithm 5 to allow for any number of *groups* (pseudocode is not shown; all statements about the groups Y and Z simply become statements

Algorithm 6 Find i such that $\prod_{k=1}^{K} \mathbf{v}_k[i]$ is maximized

Input: K vectors $\mathbf{v}_1 \dots \mathbf{v}_K$; permutation functions $p_1 \dots p_K$ that sort them in decreasing order; a vector *read* indicating which indices have been read, and a unique value $T \notin read$ {*read* is essentially a boolean array indicating which indices have been read; since *creating* this array is an $O(N)$ operation, we create it externally, and reuse it $O(N)$ times; setting $read[i] = T$ indicates that a particular index has been read; we use a different value of T for each call to this function so that *read* can be reused without having to be reinitialized}

Initialize: $start \coloneqq 1$,
$max \coloneqq \max_{p \in \{p_1 \dots p_K\}} \prod_{k=1}^{K} \mathbf{v}_k[p[1]]$,
$best \coloneqq \operatorname{argmax}_{p \in \{p_1 \dots p_K\}} \prod_{k=1}^{K} \mathbf{v}_k[p[1]]$
for $k \in \{1 \dots K\}$ **do**
  $end_k \coloneqq \max_{q \in \{p_1 \dots p_K\}} p_k^{-1}[q[1]]$
  $read[p_k[1]] = T$
end for
while $start < \max\{end_1 \dots end_K\}$ **do**
  $start \coloneqq start + 1$
  for $k \in \{1 \dots K\}$ **do**
    if $read[p_k[start]] \coloneqq T$ **then**
      continue
    end if
    $read[p_k[start]] \coloneqq T$
    $m \coloneqq \prod_{x=1}^{K} \mathbf{v}_x[p_k[start]]$
    if $m > max$ **then**
      $best \coloneqq p_k[start]$
      $max \coloneqq m$
    end if
    $e_k \coloneqq \max_{q \in \{p_1 \dots p_K\}} p_k^{-1}[q[start]]$
    $end_k \coloneqq \min(e_k, end_k)$
  end for
end while {see Appendix A for running times}
Return: *best*

about K groups $\{G_1 \dots G_K\}$, and calls to Algorithm 2 become calls to Algorithm 6). The one remaining case that has not been considered is when the sequences $\mathbf{v}_1 \cdots \mathbf{v}_K$ are functions of different (but overlapping) variables; naïvely, we can create a new variable whose domain is the product space of all of the overlapping terms, and still achieve the performance improvement guaranteed by Theorem 3; in some cases, better results can again be obtained by applying recursion, as in Figure 7.

As a final comment we note that we have not provided an algorithm for choosing *how* to split the variables of a model into $(K+1)$-groups. We note even if we split the groups in a naïve way, we are guaranteed to get *at least* the performance improvement guaranteed by Theorem 3, though more 'intelligent' splits may further improve the performance.

Furthermore, in all of the applications we have studied, K is sufficiently small that it is inexpensive to consider all possible splits by brute-force.

5.4 Extensions for Conditionally Factorizable Models

Just as in Section 5.2, we can extend Algorithm 3 to factors of any size, so long as the purely latent cliques contain more latent variables than those cliques that depend upon the observation. The analysis for this type of model is almost exactly the same as that presented in Section 5.2, except that any terms consisting of purely latent variables are processed offline.

As we mentioned in 5.2, if a model contains (non-maximal) factors of size K, we will gain a speed-up of $\Omega(\frac{1}{K}N^{\frac{1}{K}})$. If in addition there is a factor (either maximal or non-maximal) consisting of purely latent variables, we can still obtain a speed-up of $\Omega(\frac{1}{K+1}N^{\frac{1}{K+1}})$, since this factor merely contributes an additional term to (Equation 14). Thus when our 'data-dependent' terms contain only a single latent variable (i.e., $K=1$), we gain a speed-up of $\Omega(\sqrt{N})$, as in Algorithm 3.

6. Performance Improvements in Existing Applications

Our results are immediately compatible with several applications that rely on inference in graphical models. As we have mentioned, our results apply to *any model whose cliques decompose into lower-order terms*.

Often, potentials are defined only on *nodes* and *edges* of a model. A D^{th}-order Markov model has a tree-width of D, despite often containing only pairwise relationships. Similarly 'skip-chain CRFs' (Sutton and McCallum, 2006; Galley, 2006), and junction-trees used in SLAM applications (Paskin, 2003) often contain only pairwise terms, and may have low tree-width under reasonable conditions. These are examples of *latently factorizable* models. In each case, if the tree-width is D, Algorithm 5 takes $O(MN^D\sqrt{N})$ (for a model with M nodes and N states per node), yielding a speed-up of $\Omega(\sqrt{N})$.

Models for shape-matching and pose reconstruction often exhibit similar properties (Tresadern et al., 2009; Donner et al., 2007; Sigal and Black, 2006). In each case, third-order cliques factorize into second-order terms; hence we can apply Algorithm 4 to achieve a speed-up of $\Omega(\sqrt{N})$.

Another similar model for shape-matching is that of Felzenszwalb (2005); this model again contains third-order cliques, though it includes a 'geometric' term constraining all three variables. Here, the third-order term is *independent of the input data*, meaning that each of its rows can be sorted *offline*, as described in Section 3. This is an example of a *conditionally factorizable* model. In this case, those factors that depend upon the observation are pairwise, meaning that we achieve a speed-up of $\Omega(N^{\frac{1}{3}})$. Further applications of this type shall be explored in Section 7.4.

In Coughlan and Ferreira (2002), deformable shape-matching is solved approximately using loopy belief-propagation. Their model has only second-order cliques, meaning that inference takes $\Theta(MN^2)$ *per iteration*. Although we cannot improve upon this result, we note that we can typically do *exact* inference in a single iteration in $O(MN^2\sqrt{N})$; thus our model has the same running time as $O(\sqrt{N})$ iterations of the original version. This result applies to all second-order models containing a single loop (Weiss, 2000).

In McAuley et al. (2008), a model is presented for graph-matching using loopy belief-propagation; the maximal cliques for D-dimensional matching have size $(D+1)$, meaning that inference takes $\Theta(MN^{D+1})$ *per iteration* (it is shown to converge to the correct solution); we improve this to $O(MN^D\sqrt{N})$.

Interval graphs can be used to model resource allocation problems (Fulkerson and Gross, 1965); each node encodes a request, and overlapping requests form edges. Maximal cliques grow with the

Reference	description	running time	our method
McAuley et al. (2008)	D-d graph-matching	$\Theta(MN^{D+1})$ (iter.)	$O(MN^D\sqrt{N})$ (iter.)
Sutton and McCallum (2006)	Width-D skip-chain	$O(MN^D)$	$O(MN^{D-1}\sqrt{N})$
Galley (2006)	Width-3 skip-chain	$\Theta(MN^3)$	$O(MN^2\sqrt{N})$
Tresadern et al. (2009)	Deformable matching	$\Theta(MN^3)$	$O(MN^2\sqrt{N})$
Coughlan and Ferreira (2002)	Deformable matching	$\Theta(MN^2)$ (iter.)	$O(MN^2\sqrt{N})$
Sigal and Black (2006)	Pose reconstruction	$\Theta(MN^3)$	$O(MN^2\sqrt{N})$
Felzenszwalb (2005)	Deformable matching	$\Theta(MN^3)$	$\Theta(MN^{\frac{8}{3}})$ (online)
Fulkerson and Gross (1965)	Width-D interval graph	$O(MN^{D+1})$	$O(MN^D\sqrt{N})$

Table 3: Some existing work to which our results can be immediately applied (M is the number of nodes in the model, N is the number of states per node. 'iter.' denotes that the algorithm is iterative).

number of overlapping requests, though the constraints are only pairwise, meaning that we again achieve an $\Omega(\sqrt{N})$ improvement.

Finally, in Section 7.4 we shall explore a variety of applications in which we have pairwise models of the form shown in (Equation 7). In all of these cases, we see an (expected) reduction of a $\Theta(MN^2)$ message-passing algorithm to $O(MN\sqrt{N})$.

Table 3 summarizes these results. Reported running times reflect the *expected case*. Note that we are assuming that *max-product belief-propagation is being used in a discrete model*; some of the referenced articles may use different variants of the algorithm (e.g., Gaussian models, or approximate inference schemes). We believe that our improvements may revive the exact, discrete version as a tractable option in these cases.

7. Experiments

We present experimental results for two types of models: latently factorizable models, whose cliques factorize into smaller terms, as discussed in Section 4, and conditionally factorizable models, whose factors *that depend upon the observation* contain fewer latent variables than their maximal cliques, as discussed in Section 3.

We begin with an asymptotic analysis of the running time of our algorithm on the 'inner product' operations of (Equation 1) and (Equation 14), in order to assess Theorems 2 and 3 experimentally.

7.1 Comparison Between Asymptotic Performance and Upper-Bounds

For our first experiment, we compare the performance of Algorithms 2 and 6 to the naïve solution of Algorithm 1. These are core subroutines of each of the other algorithms, meaning that determining their performance shall give us an accurate indication of the improvements we expect to obtain in real graphical models.

For each experiment, we generate N i.i.d. samples from $[0,1)$ to obtain the lists $v_1 \dots v_K$. N is the domain size; this may refer to a single node, or a *group* of nodes as in Algorithm 6; thus large values of N may appear even for binary-valued models. K is the number of lists in (Equation 14); we can observe this number of lists only if we are working in cliques of size $K+1$, and then only if the factors are of size K (e.g., we will only see $K=5$ if we have cliques of size 6 with factors

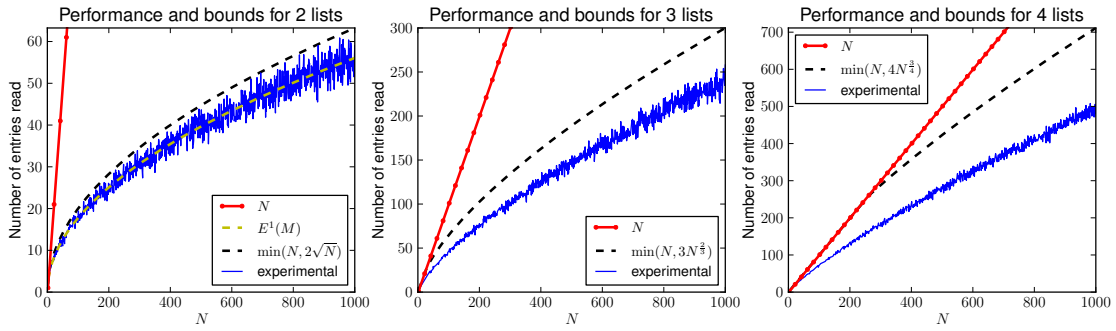


Figure 10: Performance of our algorithm and bounds. For $K = 2$, the exact expectation is shown, which appears to precisely match the average performance (over 100 trials). The dotted lines show the bound of (Equation 23). While the bound is close to the true performance for $K = 2$, it becomes increasingly loose for larger K.

of size 5); therefore smaller values of K are probably more realistic in practice (indeed, all of the applications in Section 6 have $K = 2$).

The performance of our algorithm is shown in Figure 10, for $K = 2$ to 4 (i.e., for 2 to 4 lists). When $K = 2$, we execute Algorithm 2, while Algorithm 6 is executed for $K \geq 3$. The performance reported is simply the number of elements read from the lists (which is at most $K \times \mathit{start}$). This is compared to N itself, which is the number of elements read by the naïve algorithm. The upper-bounds we obtained in (Equation 23) are also reported, while the true expected performance (i.e., Equation 19) is reported for $K = 2$. Note that the variable *read* was introduced into Algorithm 6 in order to guarantee that it can never be asymptotically slower than the naïve algorithm. If this variable is ignored, the performance of our algorithm deteriorates to the point that it closely approaches the upper-bounds shown in Figure 10. Unfortunately, this optimization proved overly complicated to include in our analysis, meaning that our upper-bounds remain highly conservative for large K.

7.2 Performance Improvement for Dependent Variables

The expected-case running time of our algorithm was derived under the assumption that each list has independent order statistics, as was the case for our previous experiment. We suggested that we will obtain worse performance in the case of negatively correlated variables, and better performance in the case of positively correlated variables; we shall assess these claims in this experiment.

Figure 11 shows how the order statistics of $\mathbf{v}_a$ and $\mathbf{v}_b$ can affect the performance of our algorithm. Essentially, the running time of Algorithm 2 is determined by the level of 'diagonalness' of the permutation matrices in Figure 11; highly diagonal matrices result in better performance than the expected case, while highly off-diagonal matrices result in worse performance. The expected case was simply obtained under the assumption that every permutation is equally likely.

We report the performance for two lists (i.e., for Algorithm 2), where each $(\mathbf{v}_a[i], \mathbf{v}_b[i])$ is an independent sample from a 2-dimensional Gaussian with covariance matrix

$$\Sigma = \begin{bmatrix} 1 & c \\ c & 1 \end{bmatrix},$$

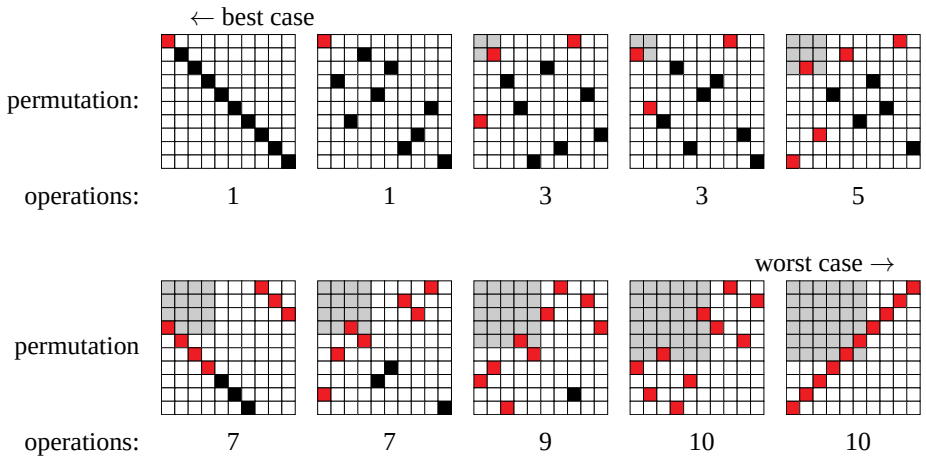


Figure 11: Different permutation matrices and their resulting cost (in terms of entries read/multiplications performed). Each permutation matrix transforms the *sorted* values of one list into the sorted values of the other, that is, it transforms $\mathbf{v}_a$ as sorted by p_a into $\mathbf{v}_b$ as sorted by p_b. The red (lighter) squares show the entries that must be read before the algorithm terminates (each corresponding to one multiplication). See Figure 23 for further explanation.

meaning that the two lists are correlated with correlation coefficient c (here we are working in the max-sum semiring). This dependence between the values of the two lists leads to a dependence in their order statistics, so that in the case of Gaussian random variables, the correlation coefficient precisely captures the 'diagonalness' of the matrices in Figure 11. Performance is shown in Figure 12 for different values of c ($c = 0$, is not shown, as this is the case observed in the previous experiment).

7.3 Message-Passing in Latently Factorizable Models

In this section we present experiments in models whose cliques factorize into smaller terms, as discussed in Section 4.

7.3.1 2-Dimensional Graph-Matching

Naturally, Algorithm 5 has additional overhead compared to the naïve solution, meaning that it will not be beneficial for small N. In this experiment, we aim to assess the extent to which our approach is faster in real applications. We reproduce the model from McAuley et al. (2008), which performs 2-dimensional graph-matching, using a loopy graph with cliques of size three, containing only second-order potentials (as described in Section 6); the $\Theta(NM^3)$ performance of McAuley et al. (2008) is reportedly state-of-the-art. We also show the performance on a graphical model with *random* potentials, in order to assess how the results of the previous experiments are reflected in terms of actual running time.

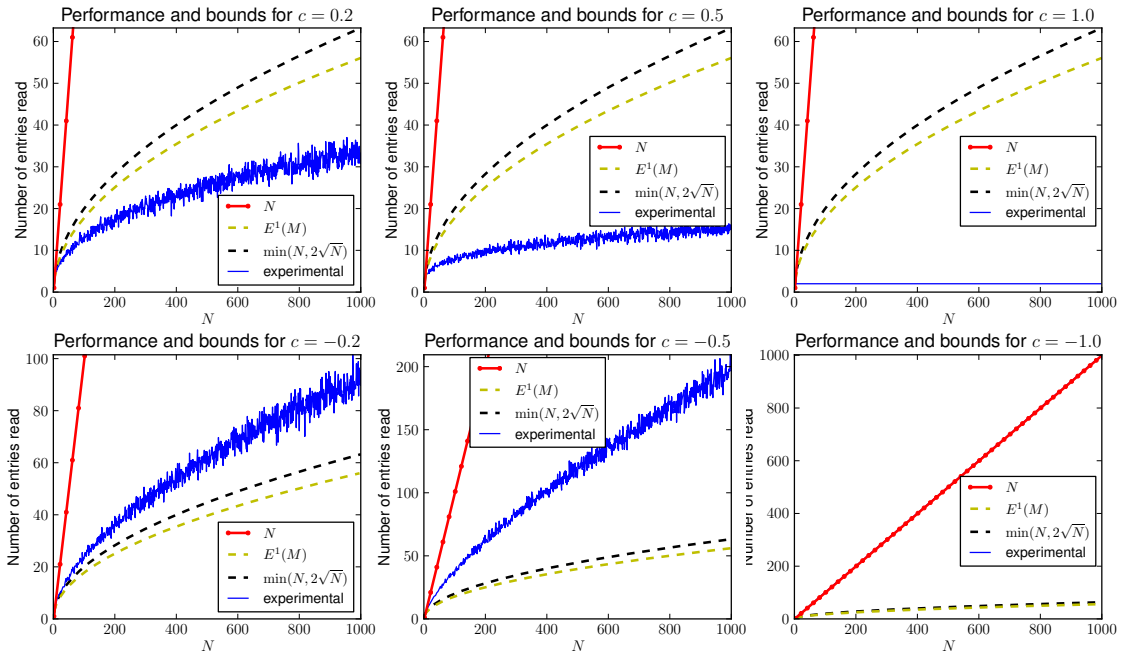


Figure 12: Performance of our algorithm for different correlation coefficients. The top three plots show positive correlation, the bottom three show negative correlation. Correlation coefficients of $c = 1.0$ and $c = -1.0$ capture precisely the best and worst-case performance of our algorithm, resulting in $O(1)$ and $\Theta(N)$ performance, respectively (when $c = -1.0$ the linear curve obscures the experimental curve).

We perform matching between a *template* graph with M nodes, and a *target* graph with N nodes, which requires a graphical model with M nodes and N states per node (see McAuley et al. 2008 for details). We fix $M = 10$ and vary N.

Figure 13 (left) shows the performance on random potentials, that is, the performance we hope to obtain if our model assumptions are satisfied. Figure 13 (right) shows the performance for graph-matching, which closely matches the expected-case behavior. Fitted curves are shown together with the actual running time of our algorithm, confirming its $O(MN^2\sqrt{N})$ performance. The coefficients of the fitted curves demonstrate that our algorithm is useful even for modest values of N.

We also report results for graph-matching using graphs from the MPEG-7 data set (Bai et al., 2009), which consists of 1,400 silhouette images (Figure 14). Again we fix $M = 10$ (i.e., 10 points are extracted in each template graph) and vary N (the number of points in the target graph). This experiment confirms that even when matching real-world graphs, the assumption of independent order statistics appears to be reasonable.

7.3.2 HIGHER-ORDER MARKOV MODELS

In this experiment, we construct a simple Markov model for text denoising. Random noise is applied to a text segment, which we try to correct using a prior extracted from a text corpus. For instance

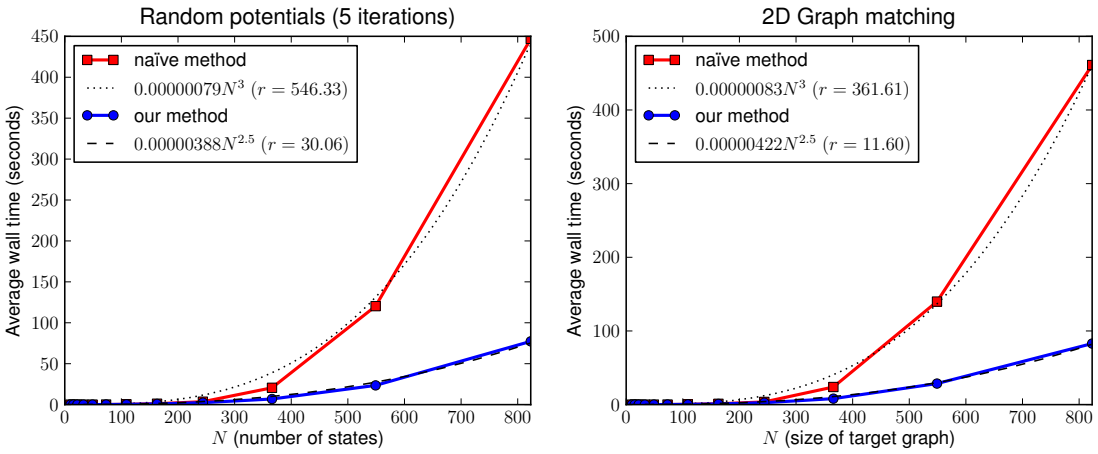


Figure 13: The running time of our method on randomly generated potentials, and on a graph-matching experiment (both graphs have the same topology). Fitted curves are also obtained by performing least-squares regression; the residual error r indicates the 'goodness' of the fitted curve.

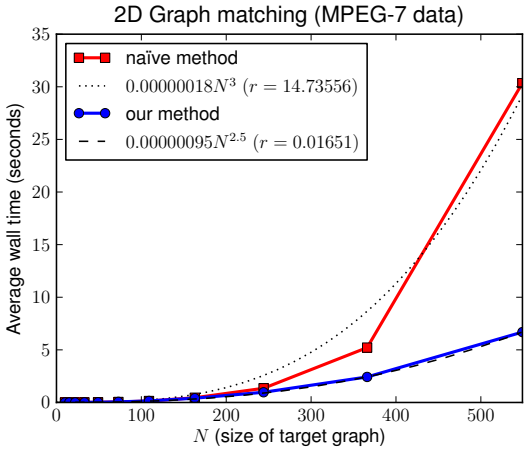


Figure 14: The running time of method our on graphs from the MPEG-7 data set.

`wondrous sight of th4 ivory Pequod` is corrected to `wondrous sight of the ivory Pequod`.

In such a model, we would like to exploit higher-order relationships between characters, though the amount of data required to construct an accurate prior grows exponentially with the size of the maximal cliques. Instead, our prior consists entirely of pairwise relationships between characters (or 'bigrams'); higher-order relationships are encoded by including bigrams of non-adjacent characters.

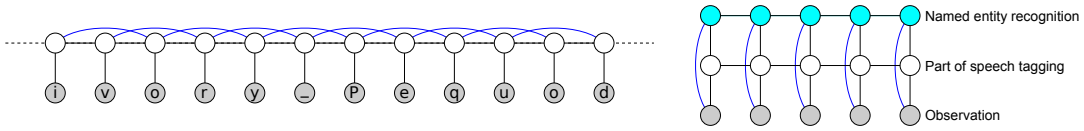


Figure 15: Left: Our model for denoising. Its computational complexity is similar to that of a skip-chain CRF, and models for named-entity recognition (right).

Specifically, our model takes the form

$$\Phi_X(\mathbf{x}_X) = \prod_{i=1}^{|X|-1} \Phi_{i,i+1}(x_i, x_{i+1}) \times \prod_{i=1}^{|X|-2} \Phi_{i,i+2}(x_i, x_{i+2})$$

where

$$\Phi_{i,j}(x_i, x_j) = \psi_{i,j}(x_i, x_j) p(x_i|o_i) p(x_j|o_j).$$

Here ψ is our *prior* (extracted from text statistics), and p is our 'noise model' (given the observation $\mathbf{o}$). The computational complexity of inference in this model is similar to that of the skip-chain CRF shown in Figure 3(b), as well as models for part-of-speech tagging and named-entity recognition, as in Figure 15. Text denoising is useful for the purpose of demonstrating our algorithm, as there are several different corpora available in different languages, allowing us to explore the effect that the domain size (i.e., the size of the language's alphabet) has on running time.

We extracted pairwise statistics based on 10,000 characters of text, and used this to correct a series of 25 character sequences, with 1% random noise introduced to the text. The domain was simply the set of characters observed in each corpus. The Japanese data set was not included, as the $\Theta(MN^2)$ memory requirements of the algorithm made it infeasible with $N \simeq 2000$; this is addressed in Section 7.4.1.

The running time of our method, compared to the naïve solution, is shown in Figure 16. One might expect that texts from different languages would exhibit different dependence structures in their order statistics, and therefore deviate from the expected case in some instances. However, the running times appear to follow the fitted curve closely, that is, we are achieving approximately the expected-case performance in all cases.

Since the prior $\psi_{i,i+1}(x_i, x_{i+1})$ is *data-independent*, we shall further discuss this type of model in reference to Algorithm 3 in Section 7.4.

7.4 Experiments with Conditionally Factorizable Models

In each of the following experiments we perform belief-propagation in models of the form given in (Equation 7). Thus each model is completely specified by defining the node potentials $\Phi_i(x_i|y_i)$, the edge potentials $\Phi_{i,j}(x_i, x_j)$, and the topology $(\mathcal{N}, \mathcal{E})$ of the graph.

Furthermore we assume that the edge potentials are *homogeneous*, that is, that the potential for each edge is the same, or rather that they have the same order statistics (for example, they may differ by a multiplicative constant). This means that sorting can be done *online* without affecting the asymptotic complexity. When subject to heterogeneous potentials we need merely sort them *offline*; the online cost shall be similar to what we report here.

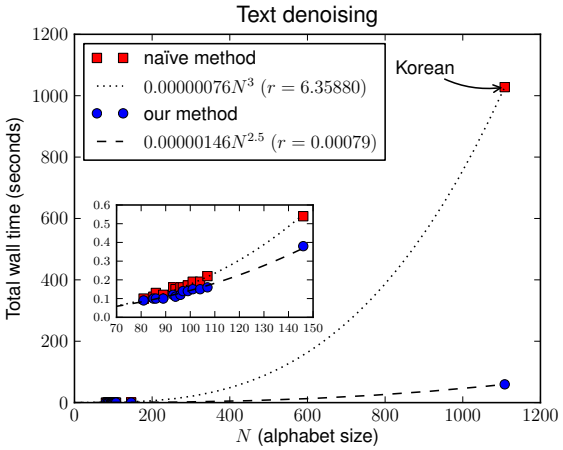


Figure 16: The running time of our method compared to the naïve solution. A fitted curve is also shown, whose coefficient estimates the computational overhead of our model.

7.4.1 Chain-Structured Models

In this section, we consider *chain-structured* graphs. Here we have nodes $\mathcal{N} = \{1 \dots Q\}$, and edges $\mathcal{E} = \{(1,2),(2,3) \dots (Q-1,Q)\}$. The max-product algorithm is known to compute the maximum-likelihood solution exactly for tree-structured models.

Figure 17 (left) shows the performance of our method on a model with *random* potentials, that is, $\Phi_i(x_i|y_i) = U[0,1)$, $\Phi_{i,i+1}(x_i,x_{i+1}) = U[0,1)$, where $U[0,1)$ is the uniform distribution. Fitted curves are superimposed onto the running time, confirming that the performance of the standard solution grows quadratically with the number of states, while ours grows at a rate of $N\sqrt{N}$. The residual error r shows how closely the fitted curve approximates the running time; in the case of random potentials, both curves have almost the same constant.

Figure 17 (right) shows the performance of our method on the text denoising experiment. This experiment is essentially identical to that shown in Section 7.3.2, except that the model is a chain (i.e., there is no $\Phi_{i,i+2}$), and we exploit the notion of data-independence (i.e., the fact that $\Phi_{i,i+1}$ does not depend on the observation). Since the same $\Phi_{i,i+1}$ is used for every adjacent pair of nodes, there is no need to perform the 'sorting' step offline—only a single copy of $\Phi_{i,i+1}$ needs to be sorted, and this is included in the total running time shown in Figure 17.

7.4.2 Grid-Structured Models

Similarly, we can apply our method to *grid-structured* models. Here we resort to loopy belief-propagation to approximate the MAP solution, though indeed the same analysis applies in the case of factor-graphs (Kschischang et al., 2001). We construct a 50×50 grid model and perform loopy belief-propagation using a random message-passing schedule for five iterations. In these experiments our nodes are $\mathcal{N} = \{1 \dots 50\}^2$, and our edges connect the 4-neighbors, that is, the node (i,j) is connected to both $(i+1,j)$ and $(i,j+1)$ (similar to the grid shown in Figure 2(a)).

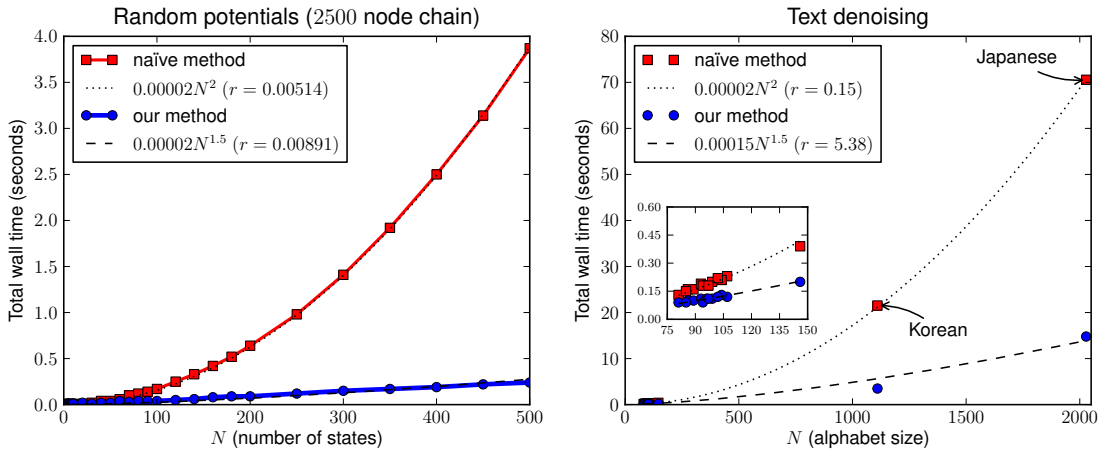


Figure 17: Running time of inference in chain-structured models: random potentials (left), and text denoising (right). Fitted curves confirm that the exponent of 1.5 given theoretically is maintained in practice (r denotes the sum of residuals, that is, the 'goodness' of the fitted curve).

Figure 18 (left) shows the performance of our method on a grid with random potentials (similar to the experiment in Section 7.4.1). Figure 18 (right) shows the performance of our method on an optical flow task (Lucas and Kanade, 1981). Here the states encode *flow vectors*: for a node with N states, the flow vector is assumed to take integer coordinates in the square $[-\sqrt{N}/2, \sqrt{N}/2)^2$ (so that there are N possible flow vectors). For the unary potential we have

$$\Phi_{(i,j)}(x|y) = \left\| Im_1[i,j] - Im_2[(i,j) + f(x)] \right\|,$$

where $Im_1[a,b]$ and $Im_2[a,b]$ return the gray-level of the pixel at (a,b) in the first and second images (respectively), and $f(x)$ returns the flow vector encoded by x. The pairwise potentials simply encode the Euclidean distance between two flow vectors. Note that a variety of low-level computer vision tasks (including optical flow) are studied in Felzenszwalb and Huttenlocher (2006), where the highly structured nature of the potentials in question often allows for efficient solutions.

Our fitted curves in Figure 18 show $O(N\sqrt{N})$ performance for both random data and for optical flow. Clearly the fitted curve for optical flow deviates somewhat from that obtained for random data; naturally the potentials are highly structured in this case, as exploited by Felzenszwalb and Huttenlocher (2006); it appears that some aspect of this structure is slightly harmful to our algorithm, though a more thorough analysis of this type of potential remains as future work. More 'harmful' structures are explored in the following section.

7.4.3 FAILURE CASES

In our previous experiments on graph-matching, text denoising, and optical flow we observed running times similar to those for random potentials, indicating that there is no prevalent dependence structure between the order statistics of the messages and the potentials.

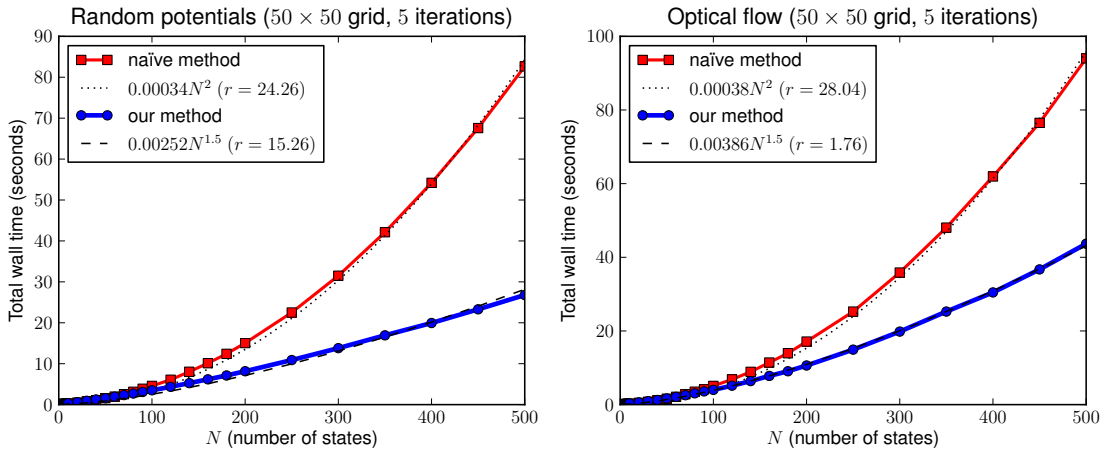


Figure 18: Running time of inference in grid-structured models: random potentials (left), and optical flow (right).

In certain applications the order statistics of these terms are highly dependent in a way that is detrimental to our algorithm. This behavior is observed for certain types of concave potentials (or convex potentials in a min-sum formulation). For instance, in a stereo disparity experiment, the unary potentials encode the fact that the output should be 'close to' a certain value; the pairwise potentials encode the fact that neighboring nodes should take similar values (Scharstein and Szeliski, 2001; Sun et al., 2003).

In these applications, the permutation matrices that transform the sorted values of $\mathbf{v}_a$ to the sorted values of $\mathbf{v}_b$ are block-off-diagonal (see the sixth permutation in Figure 11). In such cases, our algorithm only decreases the number of multiplication operations by a multiplicative constant, and may in fact be slower due to its computational overhead. This is precisely the behavior shown in Figure 19 (left), in the case of stereo disparity.

It should be noted that there exist algorithms specifically designed for this class of potential functions (Kolmogorov and Shioura, 2007; Felzenszwalb and Huttenlocher, 2006), which are preferable in such instances.

We similarly perform an experiment on image denoising, where the unary potentials are again convex functions of the input (see Geman and Geman, 1984; Lan et al., 2006). Instead of using a pairwise potential that merely encodes smoothness, we extract the pairwise statistics from image data (similar to our experiment on text denoising); thus the potentials are no longer concave. We see in Figure 19 (right) that even if a small number of entries exhibit some 'randomness' in their order statistics, we begin to gain a modest speed improvement over the naïve solution (though indeed, the improvements are negligible compared to those shown in previous experiments).

7.5 Other Applications of Tropical Matrix Multiplication

As we have mentioned, our improvements to message-passing in graphical models arise from a fast solution to matrix multiplication in the max-product semiring. In this section we discuss other

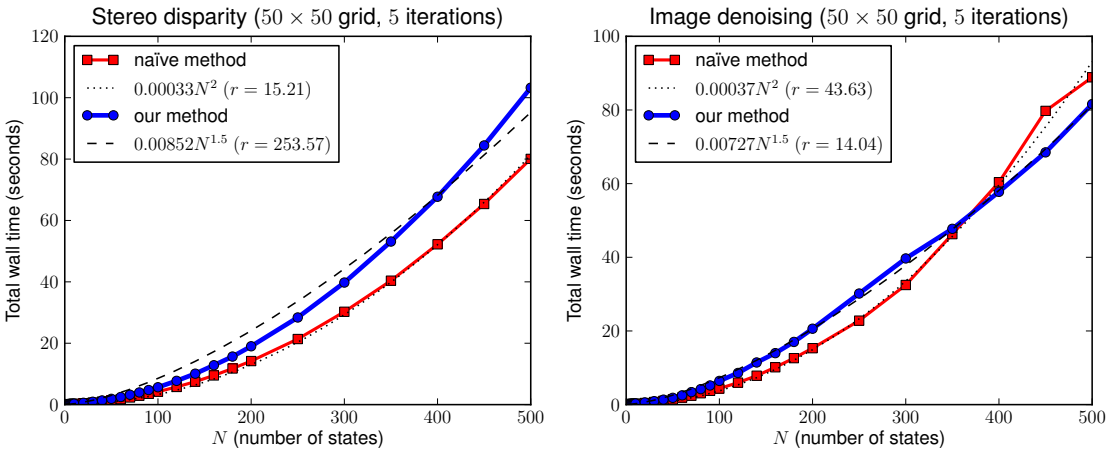


Figure 19: Two experiments whose potentials and messages have highly dependent order statistics: stereo disparity (left), and image denoising (right).

problems which include max-product (or 'tropical') matrix multiplication as a subroutine. Williams and Williams (2010) discusses the relationship between this type of matrix multiplication problem and various other problems.

7.5.1 MAX-PRODUCT LINEAR PROGRAMMING

In Sontag et al. (2008), a method is given for exact MAP-inference in graphical models using LP-relaxations. Where exact solutions cannot be obtained by considering only pairwise factors, 'clusters' of pairwise terms are introduced in order to refine the solution. Message-passing in these clusters turns out to take exactly the form that we consider, as third-order (or larger) clusters are formed from pairwise terms. Although a number of applications are presented in Sontag et al. (2008), we focus on protein design, as this is the application in which we typically observe the largest domain sizes. Other applications with larger domains may yield further benefits.

Without going into detail, we simply copy the two equations from Sontag et al. (2008) to which our algorithm applies. The first of these is concerned with passing messages between clusters, while the second is concerned with choosing new clusters to add. Below are the two equations, reproduced verbatim from Sontag et al. (2008):

$$\lambda_{c\to e}(x_e) \;\leftarrow\; -\frac{2}{3}\Big(\lambda_{e\to e}(x_e) + \sum_{c'\neq c, e\in c'} \lambda_{c'\to e}(x_e)\Big) + \frac{1}{3}\max_{x_{c\setminus e}}\Big[\sum_{e'\in c\setminus e}\Big(\lambda_{e'\to e'}(x_{e'}) + \sum_{c'\neq c, e'\in c'} \lambda_{c'\to e'}(x_{e'})\Big)\Big] \tag{15}$$

(see Sontag et al., 2008, Figure 1, bottom), which consists of marginalizing a cluster (c) that decomposes into edges (e), and

$$d(c) = \sum_{e\in c}\max_{x_e} b_e(x_e) - \max_{x_c}\left[\sum_{e\in c} b_e(x_e)\right], \tag{16}$$

(see Sontag et al., 2008, (Equation 4)), which consists of finding the MAP-state in a ring-structured model.

As the code from Sontag et al. (2008) was publicly available, we simply replaced the appropriate functions with our own (in order to provide a fair comparison, we also replaced their implementation of the naïve algorithm, as ours proved to be faster than the highly generic matrix library used in their code).

In order to improve the running time of our algorithm, we made the following two modifications to Algorithm 2:

- We used an *adaptive sorting algorithm* (i.e., a sorting algorithm that runs faster on nearly-sorted data). While Quicksort was used during the first iteration of message-passing, subsequent iterations used insertion sort, as the optimal ordering did not change significantly between iterations.

- We added an additional stopping criterion to the algorithm. Namely, we terminate the algorithm if $\mathbf{v}_a[p_a[start]] \times \mathbf{v}_b[p_b[start]] < max$. In other words, we check how large the maximum *could be* given the best possible permutation of the next elements (i.e., if they have the same index); if this value could not result in a new maximum, the algorithm terminates. This check costs us an additional multiplication, but it means that the algorithm will terminate faster in cases where a large maximum is found early on.

Results for these two problems are shown in Figure 20. Although our algorithm consistently improves upon the running time of Sontag et al. (2008), the domain size of the variables in question is not typically large enough to see a marked improvement. Interestingly, neither method follows the expected running time closely in this experiment. This is partly due to the fact that there is significant variation in the variable size (note that N only shows the *average* variable size), but it may also suggest that there is a complicated structure in the potentials which violates our assumption of independent order statistics.

7.5.2 ALL-PAIRS SHORTEST-PATH

The 'all-pairs shortest-path' problem consists of finding the shortest path between every pair of nodes in a graph. Although the most commonly used solution is probably the well-known Floyd-Warshall algorithm (Floyd, 1962), the state-of-the-art *expected-case* solution to this problem is that of Karger et al. (1993), whose expected-case running time is $O(N^2 \log N)$ when applied to graphs with distances sampled from the uniform distribution.

Unfortunately, the solution of Karger et al. (1993) requires a Fibonacci heap or similar data structure in order to achieve the reported running time (i.e., a heap with $O(1)$ insertion and decrease-key operations); such data structures are known to be inefficient in practice (Fredman and Tarjan, 1987). When their algorithm is implemented using a standard priority queue, it has running time $O(N^2 \log^2 N)$.

In Aho et al. (1983), a transformation is shown between the all-pairs shortest-path problem and min-sum matrix multiplication. Using our algorithm, this gives us an expected-case $O(N^2\sqrt{N})$ solution to the all-pairs shortest-path problem, assuming that the subproblems created by this transformation have i.i.d. order statistics; this assumption is notably different than the assumption of uniformity made in Karger et al. (1993).

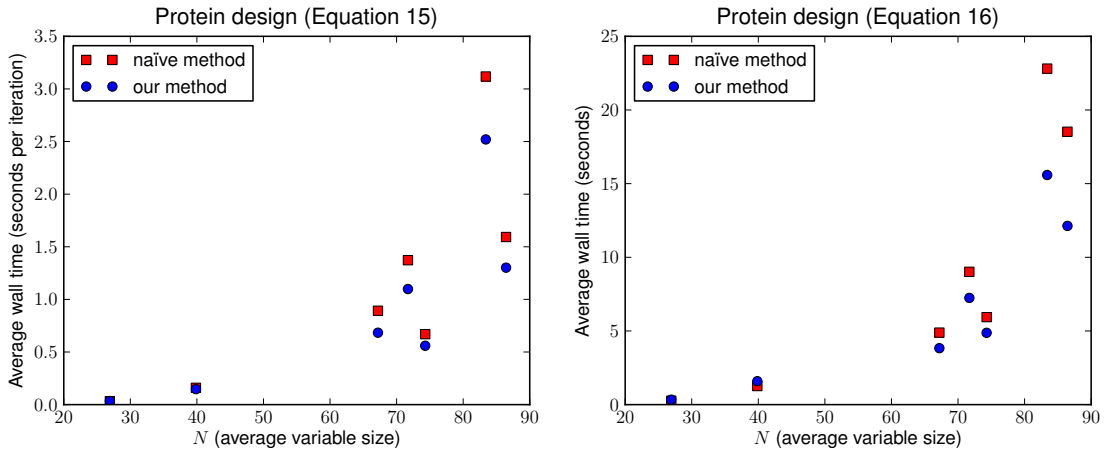


Figure 20: The running time of our method on protein design problems from Sontag et al. (2008). In this figure, N reflects the *average* domain size amongst all variables involved in the problem; fitted curves are not shown due to the highly variable nature of the domain sizes included in each problem instance.

In Figure 21, we show the performance of our method on i.i.d. uniform graphs, compared to the Floyd-Warshall algorithm, and that of Karger et al. (1993). On graph sizes of practical interest, our algorithm is found to give the fastest performance, in spite of its more expensive asymptotic cost. Our solution is comparable to that of Karger et al. (1993) for the largest graph size shown; larger graph sizes could not be shown due to memory constraints. Note that while these algorithms are fast in practice, each has $\Theta(N^3)$ *worst-case* performance; more 'exotic' solutions that improve upon the worst-case bound are discussed in Alon et al. (1997) and Chan (2007), among others, though none are truly subcubic (i.e., $O(N^{3-\varepsilon})$).

It should also be noted that the transformations given in Aho et al. (1983) apply in both directions, that is, solutions to the all-pairs shortest-path problem can be used to solve min-sum matrix multiplication. Thus any subcubic solution to the all-pairs shortest-path problem can be applied to the inference problems in graphical models presented in Section 4. However, the transformation of Aho et al. (1983) introduces a very high computational overhead (namely, solving min-sum matrix multiplication for an $N \times N$ matrix requires solving all-pairs shortest-path in a graph with $3N$ nodes), and moreover it violates the assumptions on the graph distribution required for fast inference given in Karger et al. (1993). In practice, we were unable to produce an implementation of min-sum matrix multiplication based on this transformation that was faster than the naïve solution.

Interestingly, a great deal of attention has been focused on expected-case solutions to all-pairs shortest-path, while to our knowledge ours is the first work to approach the expected-case analysis of min-sum matrix multiplication. Given the strong relationship between the two problems, it remains a promising open problem to assess whether the analysis from these solutions to all-pairs shortest-path can be applied to produce max-product matrix multiplication algorithms with similar asymptotic running times.

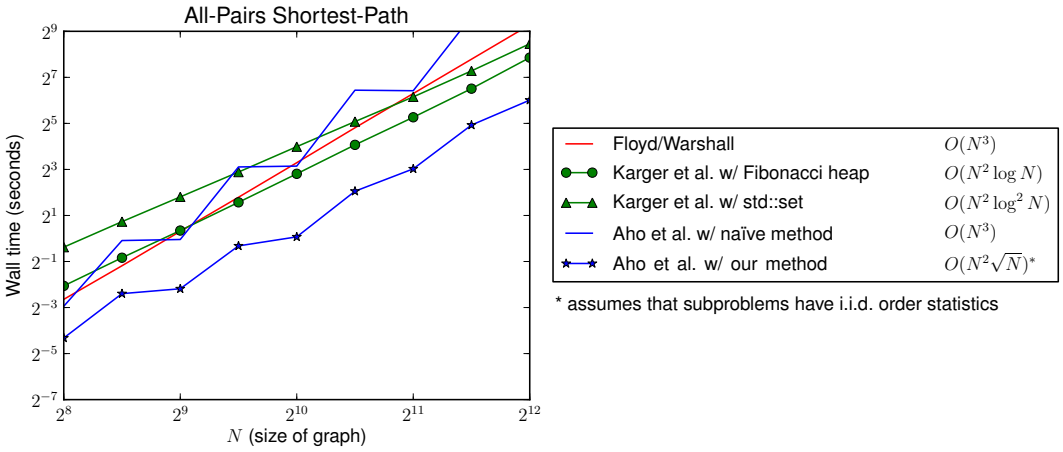


Figure 21: Our algorithm applied to the 'all-pairs shortest-path' problem. The expected-case running times of each algorithm are shown at right.

7.5.3 L^∞ Distances

The problem of computing an inner product in the max-sum semiring is closely related to computing the L^∞ distance between two vectors

$$||\mathbf{v}_a - \mathbf{v}_b||_\infty = \max_{i \in \{1...N\}} \left|\mathbf{v}_a[i] - \mathbf{v}_b[i]\right|. \tag{17}$$

Naïvely, we would like to solve (Equation 17) by applying Algorithm 2 to $\mathbf{v}_a$ and $-\mathbf{v}_b$ with the multiplication operator replaced by $a \times b = |a+b|$, however this violates the condition of (Equation 2), since the optimal solution may arise either when both $\mathbf{v}_a[i]$ and $-\mathbf{v}_b[i]$ are large, or when both $\mathbf{v}_a[i]$ and $-\mathbf{v}_b[i]$ are small (in fact, this operation violates the semiring axiom of associativity).

We address this issue by running Algorithm 2 *twice*, first considering the *largest* values of $\mathbf{v}_a$ and $-\mathbf{v}_b$, before re-running the algorithm starting from the *smallest* values. This ensures that the maximum solution is found in either case.

Pseudocode for this solution is given in Algorithm 7, which adapts Algorithm 4 to the problem of computing an L^∞ distance matrix. Similarly, we can adapt Algorithm 3 to solve L^∞ nearest-neighbor problems, where an array of M points in $\mathbb{R}^N$ is processed offline, allowing us compute the distance of a query point to all M other points $O(M\sqrt{N})$.

Figure 22 shows the running time of our algorithm for computing an L^∞ distance matrix (where $M = N$), and the online cost of performing a nearest-neighbor query. Again the expected speedup over the naïve solution is $\Omega(\sqrt{N})$ for both problems, though naturally our algorithm requires larger values of N than does Algorithm 4 in order to be beneficial, since Algorithm 2 must be executed *twice* in order to solve (Equation 17).

A similar trick can be applied to compute message in the max-product semiring even for potentials that contain negative values, though this may require up to four executions of Algorithm 2, so it is unlikely to be practical.

Algorithm 7 Use Algorithm 2 to compute an L^∞ distance matrix

Input: an $M \times N$ array $\mathbf{A}$ containing M points in $\mathbb{R}^N$

initialize an $M \times M$ distance matrix $\mathbf{D} := \mathbf{0}$
for $x \in \{1 \ldots M\}$ **do**
  compute $\overrightarrow{\mathbf{P}}[x]$ by sorting $\mathbf{A}[x]$ {takes $\Theta N \log N$}
  compute $\overleftarrow{\mathbf{P}}[x]$ by sorting $-\mathbf{A}[x]$ {i.e., $\overrightarrow{\mathbf{P}}[x]$ in reverse order}
end for {this loop takes $\Theta(MN \log N)$}
for $x \in \{1 \ldots M\}$ **do**
  for $y \in \{x+1 \ldots M\}$ **do**
    $best_1 := Algorithm2\left(\mathbf{A}[x], -\mathbf{A}[y], \overrightarrow{\mathbf{P}}[x], \overleftarrow{\mathbf{P}}[y]\right)$
    {takes $O(\sqrt{N})$; Algorithm 2 uses the operator $a \times b = |a+b|$}
    $best_2 := Algorithm2\left(\mathbf{A}[y], -\mathbf{A}[x], \overrightarrow{\mathbf{P}}[y], \overleftarrow{\mathbf{P}}[x]\right)$
    $\mathbf{D}[x,y] := \max\left(\left|\mathbf{A}[x,best_1] - \mathbf{A}[y,best_1]\right|, \left|\mathbf{A}[x,best_2] - \mathbf{A}[y,best_2]\right|\right)$
    $\mathbf{D}[y,x] := \mathbf{D}[x,y]$
  end for
end for {this loop takes expected time $O(M^2\sqrt{N})$}

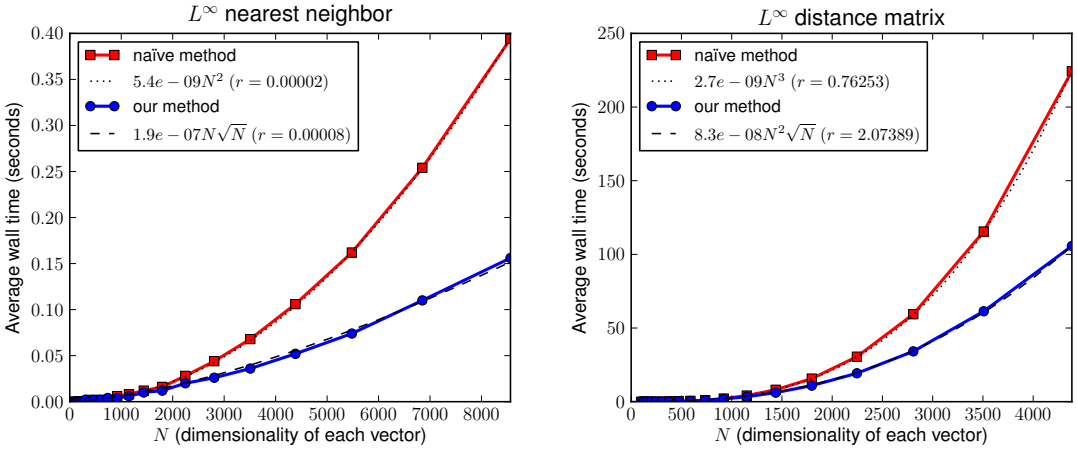


Figure 22: The running time of our method compared to the naïve solution. A fitted curve is also shown, whose coefficient estimates the computational overhead of our model.

8. Discussion and Future Work

We have briefly discussed the application of our algorithm to the all-pairs shortest-path problem, and also mentioned that a variety of other problems are related to max-product matrix multiplication via a series of subcubic transformations (Williams and Williams, 2010). To our knowledge, of all these problems only all-pairs shortest-paths has received significant attention in terms of expected-case analysis. The analysis in question centers around two types of model: the *uniform* model, where edge weights are sampled from a uniform distribution, and the *endpoint-independent model*, which

essentially makes an assumption on the independence of outgoing edge weights from each vertex (Moffat and Takaoka, 1987), which seems very similar to our assumption of independent order statistics. It remains to be seen whether this analysis can lead to better solutions to the problems discussed here, and indeed if the analysis applied to uniform models can be applied in our setting to uniform *matrices*.

It is interesting to consider the fact that our algorithm's running time is purely a function of the input data's *order statistics*, and in fact does not depend on the *data itself*. While it is pleasing that our assumption of independent order statistics appears to be a weak one, and is satisfied in a wide variety of applications, it ignores the fact that stronger assumptions may be reasonable in many cases. In factors with a high dynamic range, or when different factors have different scales, it may be possible to identify the maximum value very quickly, as we attempted to do in Section 7.5.1. Deriving faster algorithms that make stronger assumptions about the input data remains a promising avenue for future work.

Our algorithm may also lead to faster solutions for *approximately* passing a single message. While the stopping criterion of our algorithm *guarantees* that the maximum value is found, it is possible to terminate the algorithm earlier and state that the maximum has *probably* been found. A direction for future work would be to adapt our algorithm to determine the probability that the maximum has been found after a certain number of steps; we could then allow the user to specify an error probability, or a desired running time, and our algorithm could be adapted accordingly.

9. Conclusion

We have presented a series of approaches that allow us to improve the performance of exact and approximate max-product message-passing for models with factors smaller than their maximal cliques, and more generally, for models whose factors *that depend upon the observation* contain fewer latent variables than their maximal cliques. We are *always* able to improve the expected computational complexity in any model that exhibits this type of factorization, no matter the size or number of factors.

Acknowledgments

We would like to thank Pedro Felzenszwalb, Johnicholas Hines, and David Sontag for comments on initial versions of this paper, and James Petterson and Roslyn Lau for helpful discussions. NICTA is funded by the Australian Government's *Backing Australia's Ability* initiative, and the Australian Research Council's *ICT Centre of Excellence* program.

Appendix A. Asymptotic Performance of Algorithm 2 and Extensions

In this section we shall determine the expected-case running times of Algorithm 2 and Algorithm 6. Algorithm 2 traverses $\mathbf{v}_a$ and $\mathbf{v}_b$ until it reaches the smallest value of m for which there is some $j \leq m$ for which $m \geq p_b^{-1}[p_a[j]]$. If M is a random variable representing this smallest value of m, then we wish to find $E(M)$. While $E(M)$ is the number of 'steps' the algorithms take, each step takes $\Theta(K)$ when we have K lists. Thus the expected running time is $\Theta(KE(M))$.

To aid understanding our algorithm, we show the elements being read for specific examples of $\mathbf{v}_a$ and $\mathbf{v}_b$ in Figure 23. This figure reveals that the actual *values* in $\mathbf{v}_a$ and $\mathbf{v}_b$ are unimportant, and

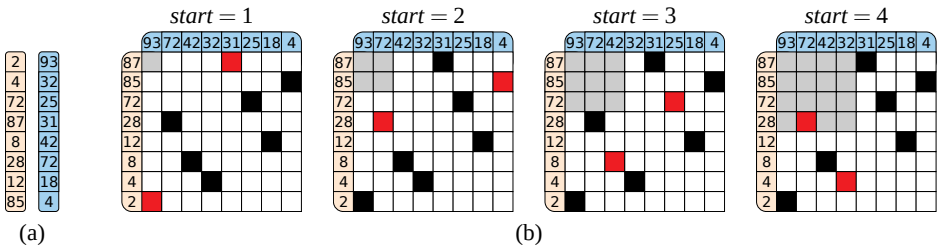


Figure 23: (a) The lists $\mathbf{v}_a$ and $\mathbf{v}_b$ before sorting; (b) Black squares show corresponding elements in the sorted lists ($\mathbf{v}_a[p_a[i]]$ and $\mathbf{v}_b[p_b[i]]$); red squares indicate the elements read during each step of the algorithm ($\mathbf{v}_a[p_a[start]]$ and $\mathbf{v}_b[p_b[start]]$). We can imagine expanding a gray box of size $start \times start$ until it contains an entry; note that the maximum is found during the first step.

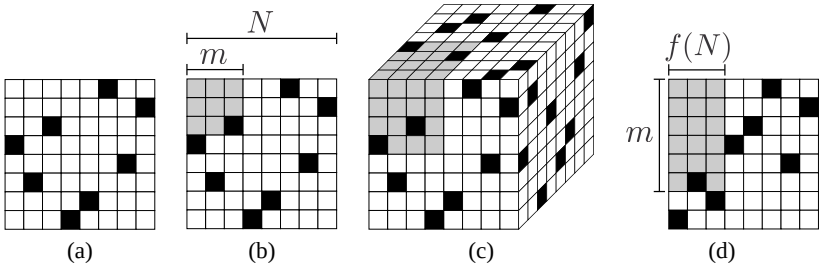


Figure 24: (a) As noted in Figure 23, a permutation can be represented as an array, where there is exactly one non-zero entry in each row and column; (b) We want to find the smallest value of m such that the gray box includes a non-zero entry; (c) A *pair* of permutations can be thought of as a cube, where every two-dimensional plane contains exactly one non-zero entry; we are now searching for the smallest gray cube that includes a non-zero entry; the faces show the projections of the points onto the exterior of the cube (the third face is determined by the first two); (d) For the sake of establishing an upper-bound, we consider a shaded region of width $f(N)$ and height m.

it is only the order statistics of the two lists that determine the performance of our algorithm. By representing a permutation of the digits 1 to N as shown in Figure 24 ((a), (b), and (d)), we observe that m is simply the width of the smallest square (expanding from the top left) that includes an element of the permutation (i.e., it includes i and $p[i]$).

Simple analysis reveals that the probability of choosing a permutation that does not contain a value inside a square of size m is

$$P(M > m) = \frac{(N-m)!(N-m)!}{(N-2m)!N!}. \tag{18}$$

This is precisely $1 - F(m)$, where $F(m)$ is the cumulative density function of M. It is immediately clear that $1 \leq M \leq \lfloor N/2 \rfloor$, which defines the best and worst-case performance of Algorithm 2.

Using the identity $E(X) = \sum_{x=1}^{\infty} P(X \geq x)$, we can write down a formula for the expected value of M:

$$E(M) = \sum_{m=0}^{\lfloor N/2 \rfloor} \frac{(N-m)!(N-m)!}{(N-2m)!N!}. \tag{19}$$

The case where we are sampling from multiple permutations simultaneously (i.e., Algorithm 6) is analogous. We consider $K-1$ permutations embedded in a K-dimensional hypercube, and we wish to find the width of the smallest shaded hypercube that includes exactly one element of the permutations (i.e., $i, p_1[i], \ldots, p_{K-1}[i]$). This is represented in Figure 24(c) for $K = 3$. Note carefully that K is the number of *lists* in (Equation 14); if we have K lists, we require $K-1$ permutations to define a correspondence between them.

Unfortunately, the probability that there is no non-zero entry in a cube of size m^K is not trivial to compute. It is possible to write down an expression that generalizes (Equation 18), such as

$$P^K(M > m) = \frac{1}{N!^{K-1}} \times \sum_{\sigma_1 \in S_N} \cdots \sum_{\sigma_{K-1} \in S_N} \bigwedge_{i=1}^{m} \left(\max_{k \in \{1 \ldots K-1\}} \sigma_k(i) > m \right) \tag{20}$$

(in which we simply enumerate over all possible permutations and 'count' which of them do not fall within a hypercube of size m^K), and therefore state that

$$E^K(M) = \sum_{m=0}^{\infty} P^K(M > m). \tag{21}$$

However, it is very hard to draw any conclusions from (Equation 20), and in fact it is intractable even to evaluate it for large values of N and K. Hence we shall instead focus our attention on finding an upper-bound on (Equation 21). Finding more computationally convenient expressions for (Equation 20) and (Equation 21) remains as future work.

A.1 An Upper-Bound on $E^K(M)$

Although (Equation 19) and (Equation 21) precisely define the running times of Algorithm 2 and Algorithm 6, it is not easy to ascertain the speed improvements they achieve, as the values to which the summations converge for large N are not obvious. Here, we shall try to obtain an upper-bound on their performance, which we assessed experimentally in Section 7. In doing so we shall prove Theorems 2 and 3.

Proof [Proof of Theorem 2] (see Algorithm 2) Consider the shaded region in Figure 24(d). This region has a width of $f(N)$, and its height m is chosen such that it contains precisely one non-zero entry. Let $\dot{M}$ be a random variable representing the height of the gray region needed in order to include a non-zero entry. We note that

$$E(\dot{M}) \in O(f(N)) \;\Rightarrow\; E(M) \in O(f(N));$$

our aim is to find the smallest $f(N)$ such that $E(\dot{M}) \in O(f(N))$. The probability that none of the first m samples appear in the shaded region is

$$P(\dot{M} > m) = \prod_{i=0}^{m} \left(1 - \frac{f(N)}{N-i}\right).$$

Next we observe that if the entries in our $N \times N$ grid do not define a permutation, but we instead choose a *random* entry in each row, then the probability (now for $\ddot{M}$) becomes

$$P(\ddot{M} > m) = \left(1 - \frac{f(N)}{N}\right)^m \tag{22}$$

(for simplicity we allow m to take arbitrarily large values). We certainly have that $P(\ddot{M} > m) \geq P(\dot{M} > m)$, meaning that $E(\ddot{M})$ is an upper-bound on $E(\dot{M})$, and therefore on $E(M)$. Thus we compute the expected value

$$E(\ddot{M}) = \sum_{m=0}^{\infty} \left(1 - \frac{f(N)}{N}\right)^m.$$

This is just a geometric progression, which sums to $N/f(N)$. Thus we need to find $f(N)$ such that

$$f(N) \in O\left(\frac{N}{f(N)}\right).$$

Clearly $f(N) \in O(\sqrt{N})$ will do. Thus we conclude that

$$E(M) \in O(\sqrt{N}).$$

■

Proof [Proof of Theorem 3] (see Algorithm 6) We would like to apply the same reasoning in the case of multiple permutations in order to compute a bound on $E^K(M)$. That is, we would like to consider $K-1$ *random* samples of the digits from 1 to N, rather than $K-1$ permutations, as random samples are easier to work with in practice.

To do so, we begin with some simple corollaries regarding our previous results. We have shown that in a permutation of length N, we expect to see a value less than or equal to f after N/f steps. There are now $f-1$ other values that are less than or equal to f amongst the remaining $N - N/f$ values; we note that

$$\frac{f-1}{N - \frac{N}{f}} = \frac{f}{N}.$$

Hence we expect to see the *next* value less than or equal to f in the next N/f steps also. A consequence of this fact is that we not only expect to see the *first* value less than or equal to f earlier in a permutation than in a random sample, but that when we sample m elements, we expect *more* of them to be less than or equal to f in a permutation than in a random sample.

Furthermore, when considering the *maximum* of $K-1$ permutations, we expect the first m elements to contain more values less than or equal to f than the maximum of $K-1$ random samples. (Equation 20) is concerned with precisely this problem. Therefore, when working in a K-dimensional hypercube, we can consider $K-1$ random samples rather than $K-1$ permutations in order to obtain an upper-bound on (Equation 21).

Thus we define $\ddot{M}$ as in (Equation 22), and conclude that

$$P(\ddot{M} > m) = \left(1 - \frac{f(N,K)^{K-1}}{N^{K-1}}\right)^m.$$

Thus the expected value of $\ddot{M}$ is again a geometric progression, which this time sums to $(N/f(N,K))^{K-1}$. Thus we need to find $f(N,K)$ such that

$$f(N,K) \in O\left(\left(\frac{N}{f(N,K)}\right)^{K-1}\right).$$

Clearly

$$f(N,K) \in O\left(N^{\frac{K-1}{K}}\right)$$

will do. As mentioned, each step takes $\Theta(K)$, so the final running time is $O(KN^{\frac{K-1}{K}})$. ■

To summarize, for problems decomposable into $K+1$ groups, we will need to find the index that chooses the maximal product amongst K lists; we have shown an upper-bound on the expected number of steps this takes, namely

$$E^K(M) \in O\left(N^{\frac{K-1}{K}}\right). \tag{23}$$

References

Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *Data Structures and Algorithms*. Addison-Wesley, 1983.

Srinivas M. Aji and Robert J. McEliece. The generalized distributive law. *IEEE Transactions on Information Theory*, 46(2):325–343, 2000.

Noga Alon, Zvi Galil, and Oded Margalit. On the exponent of the all pairs shortest path problem. *Journal of Computer and System Sciences*, 54(2):255–262, 1997.

Xiang Bai, Xingwei Yang, Longin Jan Latecki, Wenyu Liu, and Zhuowen Tu. Learning context-sensitive shape similarity by graph transduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5):861–874, 2009.

Timothy M. Chan. More algorithms for all-pairs shortest paths in weighted graphs. In *Annual ACM Symposium on Theory of Computing*, pages 590–598, 2007.

James M. Coughlan and Sabino J. Ferreira. Finding deformable shapes using loopy belief propagation. In *ECCV*, 2002.

René Donner, Georg Langs, and Horst Bischof. Sparse MRF appearance models for fast anatomical structure localisation. In *BMVC*, 2007.

Gal Elidan, Ian Mcgraw, and Daphne Koller. Residual belief propagation: informed scheduling for asynchronous message passing. In *UAI*, 2006.

Pedro F. Felzenszwalb. Representation and detection of deformable shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(2):208–220, 2005.

Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient belief propagation for early vision. *International Journal of Computer Vision*, 70(1):41–54, 2006.

Robert W. Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5(6):345, 1962.

Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34(3):596–615, 1987.

Delbert R. Fulkerson and O. A. Gross. Incidence matrices and interval graphs. *Pacific Journal of Mathematics*, (15):835–855, 1965.

Michel Galley. A skip-chain conditional random field for ranking meeting utterances by importance. In *EMNLP*, 2006.

Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distribution and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984.

David R. Karger, Daphne Koller, and Steven J. Phillips. Finding the hidden path: time bounds for all-pairs shortest paths. *SIAM Journal of Computing*, 22(6):1199–1217, 1993.

Leslie R. Kerr. The effect of algebraic structure on the computational complexity of matrix multiplication. *PhD Thesis*, 1970.

Kristian Kersting, Babak Ahmadi, and Sriraam Natarajan. Counting belief propagation. In *UAI*, 2009.

Uffe Kjærulff. Inference in bayesian networks using nested junction trees. In *Proceedings of the NATO Advanced Study Institute on Learning in Graphical Models*, 1998.

Vladimir Kolmogorov and Akiyoshi Shioura. New algorithms for the dual of the convex cost network flow problem with application to computer vision. Technical report, University College London, 2007.

Frank R. Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

M. Pawan Kumar and Philip Torr. Fast memory-efficient generalized belief propagation. In *ECCV*, 2006.

Xiang-Yang Lan, Stefan Roth, Daniel P. Huttenlocher, and Michael J. Black. Efficient belief propagation with learned higher-order markov random fields. In *ECCV*, 2006.

Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, 1981.

Julian J. McAuley and Tibério S. Caetano. Exact inference in graphical models: is there more to it? *CoRR*, abs/0910.3301, 2009.

Julian J. McAuley and Tibério S. Caetano. Exploiting within-clique factorizations in junction-tree algorithms. *AISTATS*, 2010a.

Julian J. McAuley and Tibério S. Caetano. Exploiting data-independence for fast belief-propagation. *ICML*, 2010b.

Julian J. McAuley, Tibério S. Caetano, and Marconi S. Barbosa. Graph rigidity, cyclic belief propagation and point pattern matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):2047–2054, 2008.

Alistair Moffat and Tadao Takaoka. An all pairs shortest path algorithm with expected time $O(n^2 \log n)$. *SIAM Journal of Computing*, 16(6):1023–1031, 1987.

James D. Park and Adnan Darwiche. A differential semantics for jointree algorithms. In *NIPS*, 2003.

Mark A. Paskin. Thin junction tree filters for simultaneous localization and mapping. In *IJCAI*, 2003.

Kersten Petersen, Janis Fehr, and Hans Burkhardt. Fast generalized belief propagation for MAP estimation on 2D and 3D grid-like markov random fields. In *DAGM*, 2008.

Daniel Scharstein and Richard S. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1–3):7–42, 2001.

Leonid Sigal and Michael J. Black. Predicting 3D people from 2D pictures. In *AMDO*, 2006.

David Sontag, Talya Meltzer, Amir Globerson, Tommi Jaakkola, and Yair Weiss. Tightening LP relaxations for MAP using message passing. In *UAI*, 2008.

Volker Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 14(3):354–356, 1969.

Jian Sun, Nan-Ning Zheng, and Heung-Yeung Shum. Stereo matching using belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(7):787–800, 2003.

Charles Sutton and Andrew McCallum. *Introduction to Conditional Random Fields for Relational Learning*. MIT Press, 2006.

Philip A. Tresadern, Harish Bhaskar, Steve A. Adeshina, Chris J. Taylor, and Tim F. Cootes. Combining local and global shape models for deformable object matching. In *BMVC*, 2009.

Yair Weiss. Correctness of local probability propagation in graphical models with loops. *Neural Computation*, 12:1–41, 2000.

Ryan Williams. Matrix-vector multiplication in sub-quadratic time (some preprocessing required). In *SODA*, pages 1–11, 2007.

Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix, and triangle problems. In *FOCS*, 2010.

Fast Matching of Large Point Sets Under Occlusions

Julian J. McAuley, Tibério S. Caetano

The authors are with the Automated Data Analysis program at NICTA, and the Research School of Information Sciences and Engineering, Australian National University.

Abstract

The problem of isometric point-pattern matching can be modeled as inference in small tree-width graphical models whose embeddings in the plane are said to be 'globally rigid'. Although such graphical models lead to efficient and exact solutions, they can not generally handle occlusions, as even a single missing point may 'break' the rigidity of the graph in question. In addition, such models can efficiently handle point sets of only moderate size. In this paper, we propose a new graphical model that is not only adapted to handle occlusions, but is much faster than previous approaches for solving the isometric point-pattern matching problem. We can match point-patterns with thousands of points in a few seconds.

Keywords: Point-Pattern Matching, Graphical Models, Global Rigidity

1. Introduction

Identifying a correspondence between two sets of point features is a fundamental problem in pattern recognition, with many applications in Computer Vision, including object detection [1], image categorization [2], scene reconstruction [3], fingerprint recognition [4], and shape matching [5]. There are also several applications in other fields, such as molecular biology [6, 7],

computational chemistry [8, 9], and astronomy [10].

A common setting of this problem is that of *isometric point-pattern matching*, where we assume that a 'copy' of a graph $\mathcal{G}$ (the 'template') appears isometrically transformed within $\mathcal{G}'$ (the 'target'). Although such a formulation allows for outliers in the target graph, it does not allow for outliers in the template graph, i.e., points that appear in $\mathcal{G}$ but not in $\mathcal{G}'$. In some scenarios, this is a realistic assumption, for instance if the user has manually specified a set of interest points in a template image. However, in many cases we are interested in identifying the *largest common isometric subgraph* between the two point sets, for instance where automatic interest point detectors have been used for a pair of images.

Amongst approaches concerned with isometric matching, there are further subtle differences in the formulations used. Fast algorithms exist for the case of *exact* isometric matching (i.e., no noise) [11, 12], but achieving an optimal solution for even a small, known amount of noise becomes a computationally difficult (albeit polynomial) problem [13]. Others seek solutions that are fast under certain conditions, but may be slow or inaccurate in other cases (e.g. graphs such as grids) [4, 14]. Here, we shall follow the line of work of [15], and later [16] and [17], which is concerned with algorithms that are *provably optimal and efficient* in the noise-free case, but which give empirically good performance in the case of noise.

Along these lines, we aim to make two contributions: firstly we specify a graphical model that allows us to solve the point-pattern matching problem more efficiently than the existing state-of-the-art. This is done by adapting previous results from [17] to use a different graph topology, which allows us

to use efficient search algorithms. In practice, doing so reduces the running time from $O(|\mathcal{G}||\mathcal{G}'|^3)$ in [17] to $O(|\mathcal{G}||\mathcal{G}'|^2 \log |\mathcal{G}'|)$. Furthermore, memory requirements are reduced from $O(|\mathcal{G}||\mathcal{G}'|^3)$ to $O(|\mathcal{G}'| \log |\mathcal{G}'|)$, which in practice allows the algorithm to be run on much larger point patterns (e.g. thousands rather than dozens of points).

Our second contribution is to demonstrate that this model can easily be adapted to handle occlusions, simply by running it multiple times under different configurations. The increase in running time depends on the number of occlusions N_{occ} that we wish to handle. For $N_{\text{occ}} < \lfloor |\mathcal{G}|/2 \rfloor$, we achieve a running time of $O(N_{\text{occ}}|\mathcal{G}||\mathcal{G}'|^2 \log |\mathcal{G}'|)$, with no increase in memory requirements. We can handle *any* number of occlusions in $O(|\mathcal{G}|^3|\mathcal{G}'|^2 \log |\mathcal{G}'|)$ time; in practice, this allows us to find the largest common isometric subgraph in graphs with several hundred nodes, requiring only a few seconds on a standard desktop machine.

2. Background

2.1. Point-Pattern Matching

The problem of *point-pattern matching* consists of finding an instance of a 'template' graph $\mathcal{G}$ within a 'target' scene $\mathcal{G}'$.[1] In this paper, we shall assume that this instance corresponds to an *isometric* transformation, meaning that

[1]Strictly, we consider *embeddings* of graphs in the plane, i.e., the nodes of $\mathcal{G}$ and $\mathcal{G}'$ encode 2-D coordinates. Similarly, by 'globally rigid graph' we mean a graph with a globally rigid embedding. Note that rather than using the common notation $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we consider $\mathcal{G}$ to be a set of points for simplicity, and say $(i, j) \in \mathcal{G}$ when we want to refer to edges.

$\mathcal{G}'$ can be formed by isometrically transforming $\mathcal{G}$, and possibly adding outlying points (and possibly applying some noise to the point coordinates).[2] More formally, we wish to find a total function $\hat{f} : \mathcal{G} \to \mathcal{G}'$ such that the distances between points in $\mathcal{G}$ are preserved by their image in $\mathcal{G}'$ under $\hat{f}$. That is, we wish to find

$$\hat{f} = \underset{f:\mathcal{G}\to\mathcal{G}'}{\operatorname{argmin}} \sum_{(i,j)\in\mathcal{G}} \Big| d(g_i, g_j) - d(f(g_i), f(g_j)) \Big|, \tag{1}$$

where $d(\cdot,\cdot)$ is simply a Euclidean distance function. While this would appear to be an instance of a *quadratic assignment problem*, which is NP-hard in general [18], it has been shown in [15, 16] that this problem can be solved in polynomial time by means of a *globally rigid* graphical model. In [16], a graphical model is presented which allows exact inference to be performed in $O(|\mathcal{G}||\mathcal{G}'|^3)$ memory and time; we will present this model and improve upon it in Section 3.

2.2. Largest Common Subgraph

The above problem can be adapted to allow for outliers in $\mathcal{G}$ *and* $\mathcal{G}'$, by allowing the function f to be a *partial function*. Of course, in the case of non-zero noise, the minimizing assignment of (eq. 1) then becomes the null-match (i.e., no points are mapped), so we must add some penalty κ for each point which is occluded in $\mathcal{G}$:

$$\hat{f} = \underset{f:\mathcal{G}\to\mathcal{G}'}{\operatorname{argmin}} \sum_{(i,j)\in\mathcal{G}} \Big| d(g_i, g_j) - d(f(g_i), f(g_j)) \Big| + \kappa \underbrace{\big(|\mathcal{G}| - |f(\mathcal{G})|\big)}_{\text{number of unmapped points}}. \tag{2}$$

[2] Additional points in $\mathcal{G}'$ are referred to as 'outliers', while additional points in $\mathcal{G}$ are 'occlusions'.

Assuming that κ is equal to the maximum amount of noise we are likely to observe (so that points perturbed by noise always incur a cost less than κ), and that we always favor mappings over non-mappings in the case of equal costs, this encodes the problem of identifying the largest common isometric subgraph between $\mathcal{G}$ and $\mathcal{G}'$. To our knowledge, this problem has not previously been approached using globally rigid graphical models, since the absence of a single node will generally 'break' the rigidity of the graph in question.

2.3. Matching with Globally Rigid Graphs

A graph $\mathcal{G}$ is said to be globally rigid if the pattern of distances for edges in $\mathcal{G}$ uniquely determine the distances in the graph complement $\bar{\mathcal{G}}$. In simpler terms, the only transformations that can be applied to the node coordinates while preserving the distances in $\mathcal{G}$ are isometries.

Several papers have proposed to solve the matching problem in (eq. 1) by using graphical models whose embedding in the plane are 'globally rigid' [15, 16, 17]. In these formulations, the nodes of the graphical model correspond to points in $\mathcal{G}$, while their assignments are points in $\mathcal{G}'$. If the globally rigid graph in question forms a *junction-tree* with small maximal clique size, then *min-sum belief propagation* [19] results in an exact and efficient solution to the matching problem.

Essentially, if we have a rigid graph $\mathcal{R}$ (which includes the same nodes as $\mathcal{G}$, but only a subset of its edges), it is not necessary to solve the problem described in (eq. 1), but only

$$\hat{f} = \underset{f:\mathcal{G}\to\mathcal{G}'}{\operatorname{argmin}} \sum_{(i,j)\in\mathcal{R}} \Big| d(g_i, g_j) - d(f(g_i), f(g_j)) \Big| \tag{3}$$

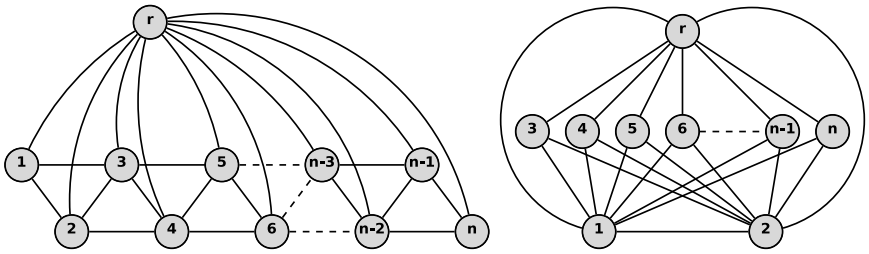


Figure 1: Models for rigid matching. The model from [17] is shown at left; the proposed model is shown at right (the nodes labeled **1** to **n** represent the points $g_1 \dots g_n$ in $\mathcal{G}$; **r** is a boolean variable encoding whether or not the pattern is reflected). Both models are globally rigid when embedded in the plane.

(similarly for (eq. 2) in the case of occlusions). Preserving the lengths of the edges in $\mathcal{R}$ implies that the lengths of *all* edges $(i, j) \in \mathcal{G}$ are preserved, due to the global rigidity result (see Theorem 1 from [15]).

Research in this direction has consisted of producing globally rigid graphs with smaller maximal clique sizes. [15] proposed using a '3-tree' model, which had maximal cliques of size 4, allowing inference to be run in $O(|\mathcal{G}||\mathcal{G}'|^4)$ time and space. [16] reduced this to $O(|\mathcal{G}||\mathcal{G}'|^3)$ using an iterative algorithm based on loopy belief propagation. It was shown in [17] that this could be achieved in a single iteration, resulting in a faster algorithm in practice. Examples of such models are shown in Figure 1.

3. Our Model

We start from the model described in [17]. There, the authors start with a graph that is rigid to translations and rotations (but not reflections). To handle isometric transformations, a third-order term is added to (eq. 1) which

either enforces or bans reflections:

$$\hat{f}, \hat{r} = \underset{f:\mathcal{G}\to\mathcal{G}', r\in\{1,-1\}}{\operatorname{argmin}} \sum_{(i,j)\in\mathcal{R}} \Big| d(g_i, g_j) - d(f(g_i), f(g_j)) \Big| + \underbrace{\sum_{i=1}^{|\mathcal{G}|-2} r \times \det \begin{bmatrix} (g_i - g_{i+1})^T \\ (g_i - g_{i+2})^T \end{bmatrix} \times \det \begin{bmatrix} (f(g_i) - f(g_{i+1}))^T \\ (f(g_i) - f(g_{i+2}))^T \end{bmatrix}}_{r \text{ 'selects' whether the determinants should agree or disagree in sign}}. \quad (4)$$

If the points (g_i, g_{i+1}, g_{i+2}) are reflected, it will change the sign of the determinant above; thus the second term in (eq. 4) ensures that the points have a consistent orientation. Either they should all be reflected in the target ($r = 1$), or none of them should be ($r = -1$).

We improve this model by making a simple modification to the topology found in [17]. Whereas the graph from [17] forms a chain, it allows for more efficient inference if this model is replaced by a 2-tree, such as the one from Figure 1 (right). We shall say that the variables g_1 and g_2 form the 'root' of the 2-tree. Although this graphical model still has maximal cliques of size 3, meaning that min-sum belief propagation would take $O(|\mathcal{G}||\mathcal{G}'|^3)$ time, we can achieve faster results by conditioning upon g_1, g_2, and r. By conditioning on these three variables, the expected locations of the remaining points are known; we can search for points close to these locations in logarithmic time using a KD-tree [20]. Thus we have $O(|\mathcal{G}'|^2)$ different points to condition on, $O(|\mathcal{G}|)$ points to search for, and an $O(\log|\mathcal{G}'|)$ search algorithm; this results in a final running time of $O(|\mathcal{G}||\mathcal{G}'|^2 \log|\mathcal{G}'|)$. The algorithm requires $O(|\mathcal{G}'| \log|\mathcal{G}'|)$ space, as required by the KD-tree [20].

The KD-tree also allows us to search for K-nearest-neighbors, multiplying the running time by K. While this is not necessary in the exact case, it may

prove beneficial under noise.

Pseudocode for our method is shown in Algorithm 1. Although this algorithm will not necessarily produce the same solution as running the $O(|\mathcal{G}||\mathcal{G}'|^3)$ min-sum algorithm for point-patterns subject to noise, it will produce the same solution *if an isometric instance exists*. Thus we replicate precisely the theoretical results reported in [15]; in the case of noise, we will show in Section 4 that the two algorithms perform comparably. Also, note that due to the `break` statement on Line 10, the algorithm will be much faster if a 'good' solution is found early on. Thus if *bestcost* is initialized using a reasonable search heuristic, we may observe running times closer to $O(|\mathcal{G}'|^2)$.

3.1. Handling Occlusions

At this stage, our model does not handle occlusions (i.e., outliers in the *template* pattern). However, the proposed 2-tree model does have the advantage that removing any nodes other than g_1 or g_2 (the 'root') will not 'break' the global rigidity of the model. Thus if we bound the distance in Algorithm 1, Line 8 by κ (the occlusion cost), this algorithm will solve the problem of matching with occlusions *assuming that we know two points that are not occluded*, which are used as the root of the tree.

Of course, we do not in general know any two points that are not occluded, so we would like to avoid such a requirement. Assuming that fewer than $\lfloor |\mathcal{G}|/2 \rfloor$ points are occluded, doing so is easy – we can just choose $\lfloor |\mathcal{G}|/2 \rfloor + 1$ independent edges (i.e., edges with no nodes in common), and use each of them as a root – at least one of these edges *must* contain a pair of non-occluded points. By rerunning Algorithm 1 with each of these edges as

Algorithm 1 Find the function f resulting in the best match.

Input: two graphs $\mathcal{G}$ and $\mathcal{G}'$ ($|\mathcal{G}| \leq |\mathcal{G}'|$)

Output: a function $f : \mathcal{G} \to \mathcal{G}'$

Initialize: $best = \varnothing$, $bestcost = \infty$
for each possible mapping $(f(g_1), f(g_2)) \in \mathcal{G}' \times \mathcal{G}'$ **do**
  for $r \in \{1, -1\} = \{\text{reflected}, \text{not reflected}\}$ **do**
    $cost = \big||g_1 - g_2| - |f(g_1) - f(g_2)|\big|$
    for each node $g_i \in \mathcal{G} \backslash \{g_1, g_2\}$ **do**
      Find the expected position p of $f(g_i)$, given $f(g_1)$, $f(g_2)$, and r
      Search for p's nearest-neighbor, n_p
      $cost = cost + \big||g_1 - g_i| - |f(g_1) - n_p|\big| + \big||g_2 - g_i| - |f(g_2) - n_p|\big|$
      if $cost > bestcost$ **then**
        break
      end if
    end for
    if $cost < bestcost$ **then**
      $best = f$
      $bestcost = cost$
    end if
  end for
end for
Return: $bestcost, best$

the root, the minimum cost solution amongst all reruns shall correspond to the desired solution. Thus for $N_{\text{occ}} < \lfloor |\mathcal{G}|/2 \rfloor$, we require $O(N_{\text{occ}})$ reruns, resulting in a running time of $O(N_{\text{occ}}|\mathcal{G}||\mathcal{G}'|^2 \log |\mathcal{G}'|)$.

It is not complicated to extend this result to $N_{\text{occ}} \geq \lfloor |\mathcal{G}|/2 \rfloor$. First, we note that if all edges within a maximal clique of size K are used as the root, then we can handle up to $K-2$ occlusions, since the remaining two points will be used as the root during some rerun. Now, suppose that we have C such cliques; we only require that *one* of these cliques contains a pair of non-occluded points, which is guaranteed if $N_{\text{occ}} < (K-1)C$. Thus using cliques of size K, we can handle $N_{\text{occ}} < (K-1)\lfloor |\mathcal{G}|/K \rfloor$ occlusions, which requires a total of $\lfloor |\mathcal{G}|/K \rfloor \binom{K}{2} \in O(|\mathcal{G}|K)$ reruns. This scales until we have $|\mathcal{G}| - 2$ occlusions, in which case we must consider *every* edge as the root, requiring precisely $|\mathcal{G}|^2$ reruns.

The behavior when $|\mathcal{G}|/K$ is not an integer is simple: to handle at most N_{occ} occlusions, we will need to use a combination of cliques of size $\lfloor |\mathcal{G}|/(|\mathcal{G}| - N_{\text{occ}} - 1) \rfloor$ and cliques of size $\lfloor |\mathcal{G}|/(|\mathcal{G}| - N_{\text{occ}} - 1) \rfloor + 1$. We will need $|\mathcal{G}| \bmod (|\mathcal{G}| - N_{\text{occ}} - 1)$ of the larger cliques, and $(|\mathcal{G}| - N_{\text{occ}} - 1) - (|\mathcal{G}| \bmod (|\mathcal{G}| - N_{\text{occ}} - 1))$ of the smaller cliques. Thus for $|\mathcal{G}| = 8$ and $N_{\text{occ}} = 4$ (for example), we would need 2 cliques of size 3, and 1 clique of size 2. A demonstration of the number of edges required for different values of N_{occ} is shown in Figure 2.

Pseudocode for our algorithm is shown in Algorithm 2. Remember that N_{occ} is the number of occlusions we wish to be able to *handle*; the actual number of occlusions must be no greater than this value. Furthermore, we adapt Algorithm 1, Line 1 so that it uses the value of *bestcost* from Algorithm

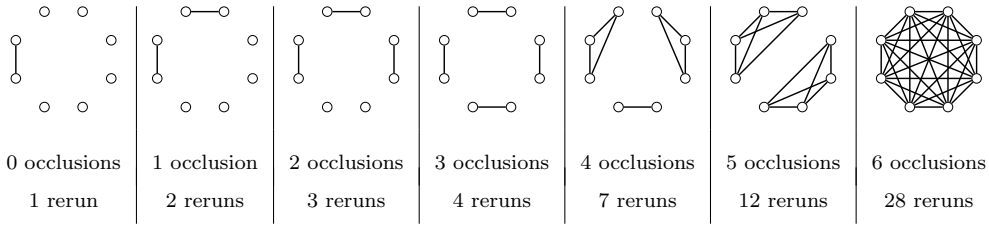


Figure 2: The number of reruns required for different values of N_{occ}, for a graph with $|\mathcal{G}| = 8$ nodes. Example choices of the edges to be used as the root are shown, though the nodes may be permuted in any order.

2; this way we do not waste time searching for solutions that have larger cost than the best solution already found.

A plot of the number of reruns required for different values of N_{occ} is shown in Figure 3, for $|\mathcal{G}| = 10000$. The function is bounded below by the linear function $y = N_{\text{occ}}$, and above by the quadratic function $y = \binom{N_{\text{occ}}}{2}$; it is not significantly worse than linear for reasonable values of N_{occ}.

Note that at this point the problem can be 'reversed', i.e., we can just as easily consider searching for $\mathcal{G}$ in $\mathcal{G}'$ as we can consider searching for $\mathcal{G}'$ in $\mathcal{G}$. Since our asymptotic running time has higher degree in $|\mathcal{G}|$ than in $|\mathcal{G}'|$ (at least for large values of N_{occ}), we shall consider $\mathcal{G}$ to be the smaller of the two graphs, without loss of generality.

The notion of constructing graphs that remain globally rigid when nodes or edges are deleted is not new, and is commonly referred to as 'redundant rigidity' [21]. Such a notion is useful in robotics applications (for example), whereby a group of robots may maintain a rigid formation even if some are unable to communicate. However, the goal in such applications is typically to minimize the number of *edges* in the graph (i.e., the total amount of

Algorithm 2 Find the maximal common isometric subgraph

Input: two graphs $\mathcal{G}$ and $\mathcal{G}'$, and $N_{\text{occ}} \leq |\mathcal{G}| - 2$

Output: a function $f : \mathcal{G} \to \mathcal{G}'$

Initialize: $best = \varnothing$, $bestcost = \infty$
$K_{\text{small}} = \lfloor |\mathcal{G}|/(|\mathcal{G}| - N_{\text{occ}} - 1) \rfloor$ (size of small cliques)
$K_{\text{large}} = K_{\text{small}} + 1$ (size of large cliques)
$N_{\text{large}} = |\mathcal{G}| \bmod (|\mathcal{G}| - N_{\text{occ}} - 1)$
$N_{\text{small}} = (|\mathcal{G}| - N_{\text{occ}} - 1) - N_{\text{large}}$
$ind = 1$
for size $\in \{\text{large}, \text{small}\}$ **do**
  for $c \in \{1 \dots N_{\text{size}}\}$ **do**
    for $(i, j) \in \binom{\{1 \dots K_{\text{size}}\}}{2}$ **do**
      run Algorithm 1 using the root (g_{i+ind}, g_{j+ind}),
      returning $cost, f$
      if $cost < bestcost$ **then**
        $best = f$
        $bestcost = cost$
      end if
      $ind = ind + K_{\text{size}}$
    end for
  end for
end for
Return: $bestcost, f$

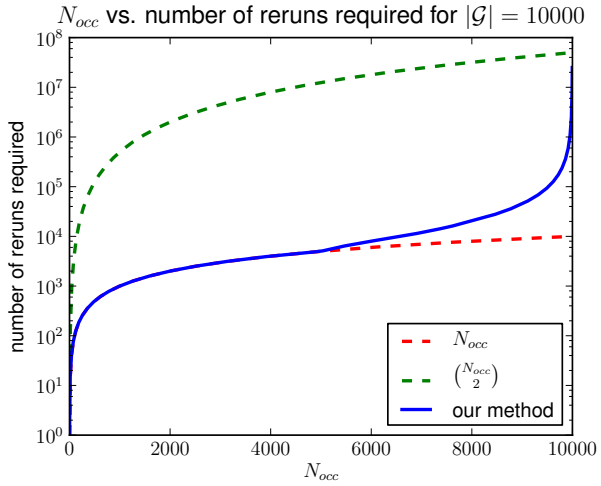


Figure 3: The number of reruns required for different values of N_{occ} when $|\mathcal{G}| = 10000$. The linear function $y = x$ and the quadratic function $y = \binom{N_{\text{occ}}}{2}$ bound the function below and above, and are shown in dotted lines (note the logarithmic scale).

communication), whereas we wish to minimize the maximal clique size (so that inference is efficient). Figure 4 (left) shows a graph that remains rigid after a single deletion, though it has maximal cliques of size 4. Instead, we are using a collection of graphs, each of which has maximal cliques of size 3. To continue the analogy with robotics, our approach corresponds to communication over several different channels – at least one of which will remain rigid even if some robots are unable to communicate.

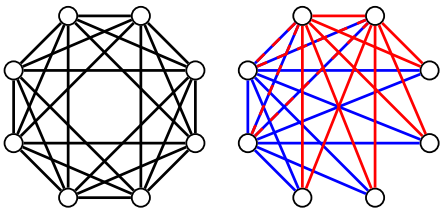


Figure 4: The 'redundantly rigid' graph (at left) has large maximal cliques, rendering inference intractable. Our approach (at right) uses a collection of graphs (in this case, two graphs for $N_{\text{occ}} \leq 1$). The graphs are shown using two different colors, where the dotted edges are shared.

4. Experiments

4.1. Graph Matching Without Occlusions

First, we compare the accuracy and running time of our method to those from [15], [16], [17], and [22].[3] For these experiments, we assume that all of the points in $\mathcal{G}$ have mappings in $\mathcal{G}'$, i.e., there are no occlusions.

4.1.1. House Dataset

Figure 5 shows the accuracy (top) and running time (bottom) of our method on the CMU 'house' dataset [23], which has been used in many computer vision papers [16, 24, 25, 26]. This dataset consists of a series of 111 frames of a toy house, subject to increasing rotation in 3-D. The same 30 points have been identified in each frame, so that the points in a pair of frames become $\mathcal{G}$ and $\mathcal{G}'$. The further apart these two frames appear in the video sequence, the more difficult the matching problem becomes. The error

[3]Note that we implemented the method of [22] in Octave, which is not directly comparable to the other methods, which were implemented in C++.

measurement is simply the 'Hamming error', i.e., the proportion of points incorrectly matched.

Figure 5 (top) shows how the accuracy of our method varies as the separation between frames increases. When searching only for nearest-neighbors (1NN), our method is worse than its competitors, though when performing a 10-nearest-neighbor search (10NN), it achieves similar performance to the other methods (it is worse for nearby frames, but better for distant frames). The running time is shown in Figure 5 (bottom); here we note that even the 10-nearest-neighbor version of our algorithm is *more than an order of magnitude* faster than its nearest competitor. Although our asymptotic bounds do not guarantee this level of improvement, our algorithm is able to achieve such results by terminating each search as soon as it has a higher cost than the best solution already found (see Algorithm 1, Line 10). By similar reasoning, the 10-nearest-neighbor version need not be significantly slower than the nearest-neighbor version, as it may find a low-cost solution earlier on. Consequently, the running time is not uniform across all baselines; it appears to become slightly slower when subject to high noise.

4.1.2. Exact Matching in Large Graphs

Here we take a small graph $\mathcal{G}$, with $|\mathcal{G}| = 10$. Points in $\mathcal{G}$ are generated by uniform sampling, and randomly transformed to form $\mathcal{G}'$. Outliers are added to $\mathcal{G}'$, whose size is increased until each method takes over 5 seconds to perform a single match. There is no jitter in the point coordinates, meaning that each method reported obtains the same (correct) solution; thus only the running time varies. Figure 6 shows the running time of each method as $|\mathcal{G}'|$ increases. Our method is able to run on graphs with ~ 10000 nodes in the

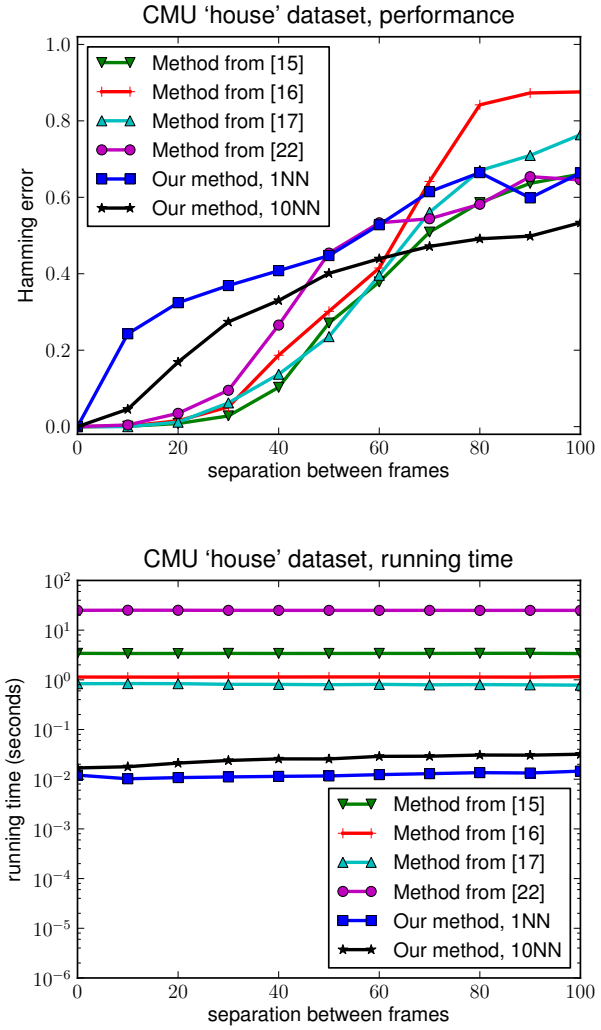


Figure 5: Performance (top) and running times (bottom) on the CMU 'house' dataset. Note the logarithmic scale of the timing plot. All methods obtain similar performance, though ours is more than an order of magnitude faster.

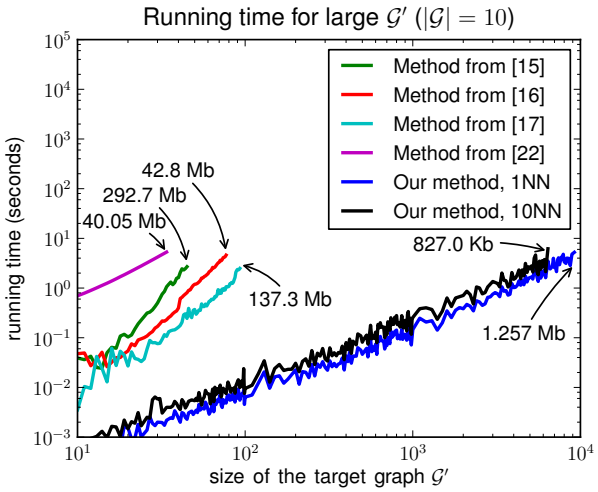


Figure 6: Running time on large graphs (average of 10 repetitions). Note that *both* axes are shown on a logarithmic scale. Annotations show the memory footprints for the largest graph sizes tested.

same time as others are able to run on only ~ 100. Our method exhibits a high variance in running time, which is again due to how quickly a 'good' solution is obtained.

Importantly, the other methods *could not* be run on larger graphs due to their memory footprints. The method from [15] has a memory footprint of $O(|\mathcal{G}||\mathcal{G}'|^4)$, meaning that it requires several hundred megabytes even for $|\mathcal{G}'| = 50$. Alternately, with memory requirements of only $O(|\mathcal{G}'| \log |\mathcal{G}'|)$, our method can in principle allow for graphs of several million nodes. Figure 6 has been annotated to show the memory requirements of each method, for the largest graph on which it was run; the memory footprint of the proposed method is significantly smaller than its competitors, even for much larger graphs.

4.2. Graph Matching With Occlusions

In this experiment, we create a graph $\mathcal{G}$ by uniform sampling in the square $[0, 1000)^2$, with $|\mathcal{G}| = 50$. The graph is randomly transformed to form $\mathcal{G}'$, though N_{occ} points are randomly replaced (again by uniform sampling). Random jitter from $[-5, 5)^2$ is applied to all of the point coordinates.

Figure 7 (top left) shows how each method performs on these graphs, for $0 \leq N_{\text{occ}} \leq |\mathcal{G}| - 3$ (beyond which the 'correct' match is not unique). Naturally, only our method and that of [22] are able to *detect* occlusions, whereas the other methods generate arbitrary (incorrect) assignments for the occluded points. To allow for a meaningful comparison, we only measure how many of the *unoccluded* points are correctly matched by each method. Although the other models produce reasonable results for up to about $N_{\text{occ}} \simeq 10$, this graph clearly demonstrates that the other models are invalid for large numbers of occlusions (in fact, they are no better than random). Alternately, the performance of our method does not degrade as N_{occ} increases, until about $N_{\text{occ}} \simeq 40$; however with so few unoccluded points, there may be spurious low-cost solutions, meaning that this result is to be expected.

The error on the remaining points (i.e., the proportion of occluded points incorrectly identified as *not* being occluded) is shown in Figure 7 (top right). Although the method of [22] is in principle able to handle occlusions using our energy function, it quickly becomes inaccurate as the number of occlusions increases; the problem appears to be that the method of [22] requires (eq. 2) to be expressed as an 'argmax' problem with positive potentials, meaning that when the potentials are rescaled to be positive, the 'important' part of the potential (i.e., the part encoding Euclidean distances) is concentrated within

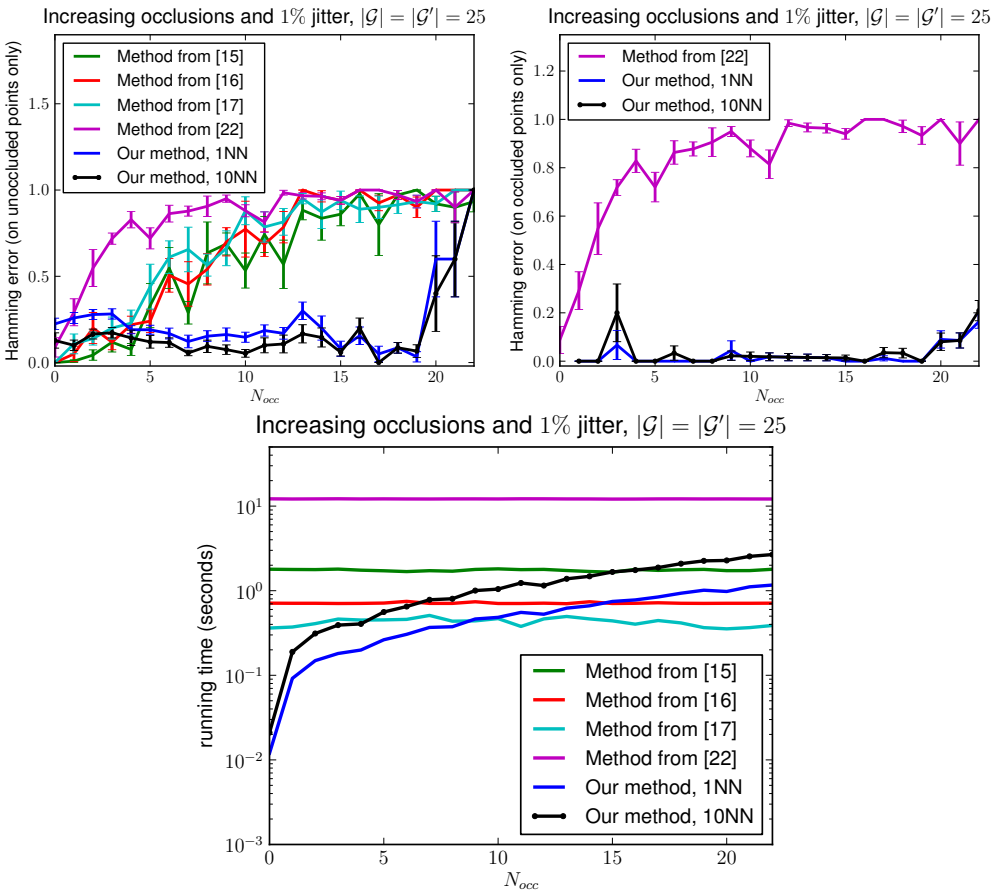


Figure 7: Performance (top) and running times (bottom) as N_{occ} varies. 10 repetitions were performed, with error bars indicating standard error. The top left plot shows the error amongst only the unoccluded points, while the top right plot shows the error amongst only the occluded points (i.e., whether they were correctly identified as occlusions); only our method and that of [22] are shown in the top right plot, as the others are unable to identify occlusions.

a small range of values, and is subsequently ignored by the approximation.

Figure 7 (bottom) shows the running time of each method. Naturally, for those models that do not encode occlusions, the running time is uniform as N_{occ} varies. Our method is faster up to about $N_{\text{occ}} \simeq |\mathcal{G}|/2$. However, at this point, the results obtained by the *other* methods are no better than random, as mentioned above. The running time of our method seems to closely follow the asymptotic prediction made in Figure 3, though it is slightly better in the case of large graphs due to the optimizations discussed in Section 4.1.1.

It is only after about $N_{\text{occ}} \simeq 40$ that our algorithm is slower by a significant margin. However, as our method is the *only* one that is able to perform matching under these conditions, this is a promising result.

5. Conclusion

We have presented an algorithm for isometric graph matching that is significantly faster than its nearest competitors, while maintaining similar results in the case of noise. The asymptotic improvements in running time and memory requirements allow our algorithm to be run on graphs which are orders of magnitude larger than those allowed by previous methods. Furthermore, a simple modification to our model allows it to handle isometric matching with *occlusions*, meaning that we are able to solve what could be called the *maximal common isometric subgraph* problem, which has thus far not been possible for similar models based on global rigidity.

References

[1] P. F. Felzenszwalb, Representation and detection of deformable shapes, IEEE Trans. on PAMI 27 (2) (2005) 208–220.

[2] S. Belongie, J. Malik, J. Puzicha, Shape matching and object recognition using shape contexts, IEEE Trans. on PAMI 24 (4) (2002) 509–522.

[3] R. I. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge University Press, 2004.

[4] P. van Wamelen, Z. Li, S. Iyengar, A fast expected time algorithm for the 2-D point pattern matching problem, Pattern Recognition 37 (8) (2004) 1699–1711.

[5] H. Chui, A. Rangarajan, A new algorithm for non-rigid point matching, in: CVPR, 2000, 44–51.

[6] T. Akutsu, K. Kanaya, A. Ohyama, A. Fujiyama, Point matching under non-uniform distortions, Discrete Applied Mathematics 127 (1) (2003) 5–21.

[7] R. Nussinov, H. J. Wolfson, Efficient detection of three-dimensional structural motifs in biological macromolecules by computer vision techniques, Proceedings of the National Academy of Sciences 88 (1991) 10495–10499.

[8] Y. Martin, M. Bures, E. Danaher, J. DeLazzer, I. Lico, A fast new approach to pharmacophore mapping and its application to dopaminer-

gic and bezodiazepine agonists, Journal of Computer-Aided Molecular Design 7 (1993) 83–102.

[9] P. W. Finn, L. E. Lavraki, J.-C. Latombe, R. Motwani, C. Shelton, S. Venkatasubramanian, A. Yao, RAPID: Randomized pharmacophore indentification for drug design, Symposium on Computational Geometry.

[10] F. Murtagh, A new approach to point-pattern matching, Astronomical Society of the Pacific 104 (674) (1992) 301–307.

[11] H. Alt, L. J. Guibas, Discrete geometric shapes: Matching, interpolation, and approximation: A survey, Tech. rep., Handbook of Computational Geometry (1996).

[12] P. J. de Rezende, D. T. Lee, Point set pattern matching in d-dimensions, Algorithmica 13 (1995) 387–404.

[13] H. Alt, K. Mehlhorn, H. Wagener, E. Welzl, Congruence, similarity and symmetries of geometric objects, Discrete Computational Geometry 3 (3) (1988) 237–256.

[14] M. T. Goodrich, J. S. B. Mitchell, Approximate geometric pattern matching under rigid motions, IEEE Trans. on PAMI 21 (4) (1999) 371–379.

[15] T. S. Caetano, T. Caelli, D. Schuurmans, D. A. C. Barone, Graphical models and point pattern matching, IEEE Trans. on PAMI 28 (10) (2006) 1646–1663.

[16] J. J. McAuley, T. S. Caetano, M. S. Barbosa, Graph rigidity, cyclic belief propagation and point pattern matching, IEEE Trans. on PAMI 30 (11) (2008) 2047–2054.

[17] T. S. Caetano, J. J. McAuley, Faster graphical models for point-pattern matching, Spatial Vision.

[18] K. Anstreicher, Recent advances in the solution of quadratic assignment problems, Mathematical Programming 97 (2003) 27–42.

[19] S. M. Aji, R. J. McEliece, The generalized distributive law, IEEE Trans. Information Theory 46 (2) (2000) 325–343.

[20] J. L. Bentley, Multidimensional binary search trees used for associative searching, Communications of the ACM 18 (9) (1975) 509–517.

[21] T. Jordán, Z. Szabadka, Operations preserving the global rigidity of graphs and frameworks in the plane, Computational Geometry 42 (6-7) (2009) 511–521.

[22] M. Leordeanu, M. Hebert, A spectral technique for correspondence problems using pairwise constraints, in: ICCV, Vol. 2, 2005, pp. 1482 – 1489.

[23] CMU 'house' dataset. [link].
URL `http://vasc.ri.cmu.edu/idb/html/motion/house/`

[24] J. Yarkony, C. Fowlkes, A. Ihler, Covering trees and lower-bounds on quadratic assignment, in: CVPR, 2010, pp. 887–894.

[25] O. Duchenne, F. Bach, I. Kweon, J. Ponce, A tensor-based algorithm for high-order graph matching, in: CVPR, 2009, pp. 1980–1987.

[26] M. Leordeanu, M. Hebert, Unsupervised learning for graph matching, in: CVPR, 2009, pp. 864–871.

Optimization of Robust Loss Functions for Weakly-Labeled Image Taxonomies: An ImageNet Case Study

Julian J. McAuley[1], Arnau Ramisa[2], and Tibério S. Caetano[1]

[1] Statistical Machine Learning Group, NICTA, and the Australian National University
{julian.mcauley, tiberio.caetano}@nicta.com.au
[2] Institut de Robòtica i Informàtica Industrial (CSIC-UPC). Spain, aramisa@iri.upc.edu

Abstract. The recently proposed *ImageNet* dataset consists of several million images, each annotated with a single object category. However, these annotations may be imperfect, in the sense that many images contain *multiple* objects belonging to the label vocabulary. In other words, we have a multi-label problem but the annotations include only a single label (and not necessarily the most prominent). Such a setting motivates the use of a *robust* evaluation measure, which allows for a limited number of labels to be predicted and, as long as one of the predicted labels is correct, the overall prediction should be considered correct. This is indeed the type of evaluation measure used to assess algorithm performance in a recent competition on ImageNet data. Optimizing such types of performance measures presents several hurdles even with existing structured output learning methods. Indeed, many of the current state-of-the-art methods optimize the prediction of only a single output label, ignoring this 'structure' altogether. In this paper, we show how to directly optimize continuous surrogates of such performance measures using structured output learning techniques with latent variables. We use the output of existing binary classifiers as input features in a new learning stage which optimizes the structured loss corresponding to the robust performance measure. We present empirical evidence that this allows us to 'boost' the performance of existing binary classifiers which are the state-of-the-art for the task of object classification in ImageNet.

1 Introduction

The recently proposed ImageNet project consists of building a growing dataset using an image taxonomy based on the WordNet hierarchy (Deng et al., 2009). Each node in this taxonomy includes a large set of images (in the hundreds or thousands). From an object recognition point of view, this dataset is interesting because it naturally suggests the possibility of leveraging the image taxonomy in order to improve recognition beyond what can be achieved independently for each image. Indeed this question has been the subject of much interest recently, culminating in a competition in this context using ImageNet data (Berg et al., 2010; Lin et al., 2011; Sánchez and Perronnin, 2011).

Although in ImageNet each image may have several objects from the label vocabulary, the annotation only includes a single label per image, and this label is not necessarily the most prominent. This imperfect annotation suggests that a meaningful performance measure in this dataset should somehow not penalize predictions that contain

legitimate objects that are missing in the annotation. One way to deal with this issue is to enforce a *robust* performance measure based on the following idea: an algorithm is allowed to predict more than one label per image (up to a maximum of K labels), and as long as one of those labels agrees with the ground-truth label, no penalty is incurred. This is precisely the type of performance measure used to evaluate algorithm performance in the aforementioned competition (Berg et al., 2010).

In this paper, we present an approach for directly optimizing a continuous surrogate of this robust performance measure. In other words, we try to optimize the very measure that is used to assess recognition quality in ImageNet. We show empirically that by using binary classifiers as a starting point, which are state-of-the-art for this task, we can boost their performance by means of optimizing the structured loss.

1.1 Literature Review

The success of visual object classification obtained in recent years is pushing computer vision research towards more difficult goals in terms of the number of object classes and the size of the training sets used. For example, Perronnin et al. (2010) used increasingly large training sets of Flickr images together with online learning algorithms to improve the performance of linear SVM classifiers trained to recognize the 20 Pascal Visual Object Challenge 2007 objects; or Torralba et al. (2008), who defined a gigantic dataset of 75,062 classes (using all the nouns in WordNet) populated with 80 million tiny images of only 32×32 pixels. The WordNet nouns were used in seven search engines, but without any manual or automatic validation of the downloaded images. Despite its low resolution, the images were shown to still be useful for classification.

Similarly, Deng et al. (2009) created ImageNet: a vast dataset with thousands of classes and millions of images, also constructed by taking nouns from the WordNet taxonomy. These were translated into different languages, and used as query terms in multiple image search engines to collect a large amount of pictures. However, as opposed to the case of the previously mentioned 80M Tiny Images dataset, in this case the images were kept at full resolution and manually verified using Amazon Mechanical Turk. Currently, the full ImageNet dataset consists of over 17,000 classes and 12 million images. Figure 1 shows a few example images from various classes.

Deng et al. (2010) performed classification experiments using a substantial subset of ImageNet, more than ten thousand classes and nine million images. Their experiments highlighted the importance of algorithm design when dealing with such quantities of data, and showed that methods believed to be better in small scale experiments turned out to under-perform when brought to larger scales. Also a cost function for classification taking into account the hierarchy was proposed. In contrast with Deng et al. (2010), most of the works using ImageNet for large scale classification made no use of its hierarchical structure.

As mentioned before, in order to encourage large scale image classification using ImageNet, a competition using a subset of 1,000 classes and 1.2 million images, called the ImageNet Large Scale Visual Recognition Challenge (ILSVRC; Berg et al., 2010), was conducted together with the Pascal Visual Object Challenge 2010 competition. Notoriously, the better classified participants of the competition used a traditional one-versus-all approach and completely disregarded the WordNet taxonomy.

Fig. 1. Example images from ImageNet. Classes range from very general to very specific, and since there is only one label per image, it is not rare to find images with unannotated instances of other classes from the dataset.

Lin et al. (2011) obtained the best score in the ILSVRC'10 competition using a conventional one-vs-all approach. However, in order to make their method efficient enough to deal with large amounts of training data, they used Local Coordinate Coding and Super-Vector Coding to reduce the size of the image descriptor vectors, and averaged stochastic gradient descent (ASGD) to efficiently train a thousand linear SVM classifiers.

Sánchez and Perronnin (2011) got the second best score in the ILSVRC'10 competition (and a posteriori reported better results than those of Lin et al. (2011)). In their approach, they used high-dimensional Fisher Kernels for image representation with lossy compression techniques: first, dimensionality reduction using Hash Kernels (Shi et al., 2009) was attempted and secondly, since the results degraded rapidly with smaller descriptor dimensionality, coding with Product Quantizers (Jégou et al., 2010) was used to retain the advantages of a high-dimensional representation without paying an expensive price in terms of memory and I/O usage. For learning the standard binary one-vs-all linear classifiers, they also used Stochastic Gradient Descent.

The difficulty of using the hierarchical information for improving classification may be explained by the findings of Russakovsky and Fei-Fei (2010). In their work ImageNet is used to show that the relationships endowed by the WordNet taxonomy do not necessarily translate in visual similarity, and that in fact new relations based only on visual appearance information can be established between classes, often far away in the hierarchy.

2 Problem Statement

We are given the dataset $\mathcal{S} = \left\{(x^1, y^1), \ldots, (x^N, y^N)\right\}$, where $x^n \in \mathcal{X}$ denotes a feature vector representing an image with label y^n. Our goal is to learn a classifier $\bar{Y}(x;\theta)$ that for an image x outputs a set of K distinct object categories. The vector θ 'parametrizes' the classifier $\bar{Y}$; we wish to learn θ so that the labels produced by $\bar{Y}(x^n;\theta)$ are 'similar to' the training labels y^n under some loss function $\Delta(\bar{Y}(x^n;\theta), y^n)$. Our specific choice of classifier and loss function shall be given in Section 2.1.

We assume an estimator based on the principle of regularized risk minimization, i.e. we aim to find θ^* such that

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \Bigg[\underbrace{\frac{1}{N}\sum_{n=1}^{N} \Delta(\bar{Y}(x^n;\theta), y^n)}_{\text{empirical risk}} + \underbrace{\frac{\lambda}{2}\|\theta\|^2}_{\text{regularizer}} \Bigg]. \tag{1}$$

Our notation is summarized in Table 1. Note specifically that each image is annotated with a *single* label, while the output space consists of a *set* of K labels (we use y to denote a single label, Y to denote a set of K labels, and $\mathcal{Y}$ to denote the space of sets of K labels). This setting presents several issues when trying to express (eq. 1) in the framework of structured prediction (Tsochantaridis et al., 2005). Apparently for this reason, many of the state-of-the-art methods in the ImageNet Large Scale Visual Recognition Challenge (Berg et al., 2010, or just 'the ImageNet Challenge' from now on) consisted of binary classifiers, such as multiclass SVMs, that merely optimized the score of a *single* prediction (Lin et al., 2011; Sánchez and Perronnin, 2011).

Motivated by the surprisingly good performance of these binary classifiers, in the following sections we shall propose a learning scheme that will 'boost' their performance by re-weighting them so as to take into account the structured nature of the loss function from the ImageNet Challenge.

2.1 The Loss Function

Images in the ImageNet dataset are annotated with a single label y^n. Each image may contain multiple objects that are not labeled, and the labeled object need not necessarily be the most salient, so the method should not be penalized for choosing 'incorrect' labels in the event that those objects actually appear in the scene. Note that this is not an issue in some similar datasets, such as the Caltech datasets (Griffin et al., 2007), where the images have been selected to avoid such ambiguity in the labeling, or all instances of objects covered in the dataset are annotated in every image, as in the Pascal Visual Object Challenge (Everingham et al., 2010).

To address this issue, a loss is given over a *set* of output labels Y, that only penalizes the method if *none* of those labels is similar to the annotated object. For a training image with label y^n, the loss incurred by choosing the set of labels Y is given by

$$\Delta(Y, y^n) = \min_{y \in Y} d(y, y^n). \tag{2}$$

Table 1. Notation

Notation	Description
x	the feature vector for an image (or just 'an image' for simplicity)
x^n	the feature vector for the n^{th} training image
$\mathcal{X}$	the features space, i.e., $x^n \in \mathcal{X}$
F	the feature dimensionality, i.e., $F = \lvert x^n \rvert$
N	the total number of training images
y	an image label, consisting of a single object class
y^n	the training label for the image x^n
$\mathcal{C}$	the set of classes, i.e., $y^n \in \mathcal{C}$
C	the total number of classes, i.e., $C = \lvert\mathcal{C}\rvert$
$\bar{Y}(x;\theta)$	the *set* of output labels produced by the classifier
$\hat{Y}(x;\theta)$	the output labels resulting in the most violated constraints during column-generation
$\bar{Y}^n$	shorthand for $\bar{Y}(x^n;\theta)$
$\hat{Y}^n$	shorthand for $\hat{Y}(x^n;\theta)$
K	the number of output labels produced by the classifier, i.e., $K = \lvert\bar{Y}^n\rvert = \lvert\hat{Y}^n\rvert$
$\mathcal{Y}$	the space of all possible sets of K labels
θ	a vector parameterizing our classifier
θ^y_{binary}	a binary classifier for the class y
λ	a constant that balances the importance of the empirical risk versus the regularizer
$\phi(x,y)$	the joint parametrization of the image x with the label y
$\Phi(x,Y)$	the joint parametrization of the image x with a *set of labels* Y
$\Delta(Y,y^n)$	the error induced by the set of labels Y when the correct label is y^n
$d(y,y^n)$	a distance measure between the two classes y and y^n in our image taxonomy
Z^n	latent annotation of the image x^n, consisting of $K-1$ object classes distinct from y^n
Y^n	the 'complete annotation' of the image x^n, i.e., $Z^n \cup \{y^n\}$

In principle, $d(y,y^n)$ could be any difference measure between the classes y and y^n. If $d(y,y^n) = 1 - \delta(y = y^n)$ (i.e., 0 if $y = y^n$, 1 otherwise), this recovers the ImageNet Challenge's 'flat' error measure. If $d(y,y^n)$ is the shortest-path distance from y to the nearest common ancestor of y and y^n in a certain taxonomic tree, this recovers the 'hierarchical' error measure (which we shall use in our experiments).

For images with multiple labels we could use the loss $\Delta(Y,Y^n) = \frac{1}{|Y^n|}\sum_{y^n \in Y^n}\Delta(Y,y^n)$, though when using the ImageNet data we always have a single label.

2.2 'Boosting' of Binary Classifiers

Many of the state-of-the-art methods for image classification consist of learning a series of binary 'one vs. all' classifiers that distinguish a single class from all others. That is, for each class $y \in \mathcal{C}$, one learns a separate parameter vector θ^y_{binary}, and then performs classification by choosing the class with the highest score, according to

$$\bar{y}_{\text{binary}}(x) = \operatorname*{argmax}_{y \in \mathcal{C}} \left\langle x, \theta^y_{\text{binary}} \right\rangle. \tag{3}$$

In order to output a set of K labels, such methods simply return the labels with the highest scores,

$$\bar{Y}_{\text{binary}}(x) = \underset{Y \in \mathcal{Y}}{\text{argmax}} \sum_{y \in Y} \left\langle x, \theta^y_{\text{binary}} \right\rangle, \tag{4}$$

where $\mathcal{Y}$ is the space of sets of K distinct labels. The above equations describe many of the competitive methods from the ImageNet Challenge, such as Lin et al. (2011) or Sánchez and Perronnin (2011).

One obvious improvement is simply to learn a new set of classifiers $\{\theta^y\}_{y \in \mathcal{C}}$ that optimize the structured error measure of (eq. 1). However, given the large number of classes in the ImageNet Challenge ($|\mathcal{C}| = 1000$), and the high-dimensionality of standard image features, this would mean simultaneously optimizing several million parameters, which is not practical using existing structured learning techniques.

Instead, we would like to leverage the already good classification performance of existing binary classifiers, simply by re-weighting them to account for the structured nature of (eq. 2). Hence we will learn a *single* parameter vector θ that re-weights the features of every class. Our proposed learning framework is designed to extend any classifier of the form given in (eq. 4). Given a set of binary classifiers $\{\theta^y_{\text{binary}}\}_{y \in \mathcal{C}}$, we propose a new classifier of the from

$$\bar{Y}(x; \theta) = \underset{Y \in \mathcal{Y}}{\text{argmax}} \sum_{y \in Y} \left\langle x \odot \theta^y_{\text{binary}}, \theta \right\rangle, \tag{5}$$

where $x \odot \theta^y_{\text{binary}}$ is simply the Hadamard product of x and θ^y_{binary}. Note that when $\theta = \mathbf{1}$ this recovers precisely the original model of (eq. 4).

To use the standard notation of structured prediction, we define the joint feature vector $\Phi(x, Y)$ as

$$\Phi(x, Y) = \sum_{y \in Y} \phi(x, y) = \sum_{y \in Y} x \odot \theta^y_{\text{binary}}, \tag{6}$$

so that (eq. 4) can be expressed as

$$\bar{Y}(x; \theta) = \underset{Y \in \mathcal{Y}}{\text{argmax}} \left\langle \Phi(x, Y), \theta \right\rangle. \tag{7}$$

We will use the shorthand $\bar{Y}^n := \bar{Y}(x^n; \theta)$ to avoid excessive notation. In the following sections we shall discuss how structured prediction methods can be used to optimize models of this form.

2.3 The Latent Setting

The joint parametrization of (eq. 6) is problematic, since the energy of the 'true' label y^n, $\langle \phi(x^n, y^n), \theta \rangle$, is not readily comparable with the energy of a *set* of predicted outputs Y, $\langle \Phi(x^n, Y), \theta \rangle$.

To address this, we propose the introduction of a latent variable, $Z = \{Z_1 \dots Z_N\}$, which for each image x^n encodes *the set of objects that appear in x^n that were not annotated*. The full set of labels for the image x^n is now $Y^n = Z_n \cup \{y^n\}$. If our method outputs K objects, then we fix $|Z_n| = K - 1$, so that $|Y^n| = K$. It is now possible to

meaningfully compute the difference between $\Phi(x^n, Y)$ and $\Phi(x^n, Y^n)$, where the latter is defined as

$$\Phi(x^n, Y^n) = \phi(x^n, y^n) + \sum_{y \in Z_n} \phi(x^n, y). \tag{8}$$

The importance of this step shall become clear in Section 3.1, (eq. 13). Note that we still define $\Delta(Y, y^n)$ in terms of the single training label y^n, as in (eq. 2).

Following the programme of Yu and Joachims (2009), learning proceeds by alternately optimizing the latent variables and the parameter vector. Optimizing the parameter vector θ^i given the latent variables Z^i is addressed in Section 3.1; optimizing the latent variables Z^i given the parameter vector θ^{i-1} is addressed in Section 3.2.

3 The Optimization Problem

The optimization problem of (eq. 1) is non-convex. More critically, the loss is a piecewise constant function of θ.[3] A similar problem occurs when one aims to optimize a 0/1 loss in binary classification; in that case, a typical workaround consists of minimizing a surrogate convex loss function that upper-bounds the 0/1 loss, such as the hinge loss, which gives rise to support vector machines. We will now see that we can construct a suitable convex relaxation for the problem defined in (eq. 1).

3.1 Convex Relaxation

Here we use an analogous approach to that of SVMs, notably popularized in Tsochantaridis et al. (2005), which optimizes a convex upper bound on the structured loss of (eq. 1). The resulting optimization problem is

$$[\theta^*, \xi^*] = \underset{\theta, \xi}{\operatorname{argmin}} \left[\frac{1}{N} \sum_{n=1}^{N} \xi_n + \lambda \|\theta\|^2 \right] \tag{9a}$$

$$\text{s.t. } \langle \Phi(x^n, Y^n), \theta \rangle - \langle \Phi(x^n, Y), \theta \rangle \geq \Delta(Y, Y^n) - \xi_n \tag{9b}$$

$$\forall n, Y \in \mathcal{Y}.$$

It is easy to see that ξ_n^* upper-bounds $\Delta(\bar{Y}^n, y^n)$ (and therefore the objective in (eq. 9) upper bounds that of (eq. 1) for the optimal solution). First note that since the constraints (eq. 9b) hold for all Y, they also hold for $\bar{Y}^n$. Second, the left hand side of the inequality for $Y = \bar{Y}^n$ must be non-positive since $\bar{Y}(x; \theta) = \operatorname{argmax}_Y \langle \Phi(x, Y), \theta \rangle$. It then follows that $\xi_n^* \geq \Delta(\bar{Y}^n, y^n)$. This implies that a solution of the relaxation is an upper bound on the solution of the original problem, and therefore the relaxation is well-motivated.

The constraints (eq. 9b) basically enforce a loss-sensitive margin: θ is learned so that mispredictions Y that incur some loss end up with a score $\langle \Phi(x^n, Y), \theta \rangle$ that is smaller than the score $\langle \Phi(x^n, Y^n), \theta \rangle$ of the correct prediction Y^n by a margin equal to that loss (minus the slack ξ_n). The formulation is a generalization of support vector machines for the multi-class case.

[3] There are countably many values for the loss but uncountably many values for the parameters, so there are large equivalence classes of parameters that correspond to precisely the same loss.

There are two options for solving the convex relaxation of (eq. 9). One is to explicitly include all $N \times |\mathcal{Y}|$ constraints and then solve the resulting quadratic program using one of several existing methods. This may not be feasible if $N \times |\mathcal{Y}|$ is too large. In this case, we can use a constraint generation strategy. This consists of iteratively solving the quadratic program by adding at each iteration the constraint corresponding to the most violated Y for the current model θ and training instance n. This is done by maximizing the violation gap ξ_n, i.e., solving at each iteration the problem

$$\hat{Y}(x^n; \theta) = \underset{Y \in \mathcal{Y}}{\operatorname{argmax}} \left\{ \Delta(Y, y^n) + \langle \Phi(x^n, Y), \theta \rangle \right\}, \tag{10}$$

(as before we define $\hat{Y}^n \coloneqq \hat{Y}(x^n; \theta)$ for brevity). The solution to this optimization problem (known as 'column generation') is somewhat involved, though it turns out to be tractable as we shall see in Section 3.3.

Several publicly available tools implement precisely this constraint generation strategy. A popular example is SvmStruct (Tsochantaridis et al., 2005), though we use BMRM ('Bundle Methods for Risk Minimization'; Teo et al., 2007) in light of its faster convergence properties. Algorithm 1 describes pseudocode for solving the optimization problem (eq. 9) with BMRM. In order to use BMRM, one needs to compute, at the optimal solution ξ_n^* for the most violated constraint $\hat{Y}^n$, both the value of the objective function (eq. 9) and its gradient. At the optimal solution for ξ_n^* with fixed θ we have

$$\langle \Phi(x^n, Y^n), \theta \rangle - \left\langle \Phi(x^n, \hat{Y}^n), \theta \right\rangle = \Delta(\hat{Y}^n, y^n) - \xi_n^*. \tag{11}$$

By expressing (eq. 11) as a function of ξ_n^* and substituting into the objective function we obtain the following lower bound on the objective of (eq. 9a):

$$o_i = \frac{1}{N} \sum_n \Delta(\hat{Y}^n, y^n) - \langle \Phi(x^n, Y^n), \theta \rangle + \left\langle \Phi(x^n, \hat{Y}^n), \theta \right\rangle + \lambda \|\theta\|^2, \tag{12}$$

whose gradient with respect to θ is

$$g_i = \lambda\theta + \frac{1}{N} \sum_n (\Phi(x^n, \hat{Y}^n) - \Phi(x^n, Y^n)). \tag{13}$$

3.2 Learning the Latent Variables

To learn the optimal value of θ, we alternate between optimizing the parameter vector θ^i given the latent variables Z^i, and optimizing the latent variables Z^i given the parameter vector θ^{i-1}. Given a fixed parameter vector θ, optimizing the latent variables Z^n can be done greedily, and is in fact equivalent to performing inference, with the restriction that the true label y^n cannot be part of the latent variable Z^n (see Algorithm 2, Line 5). See Yu and Joachims (2009) for further discussion of this type of approach.

Algorithm 1 Taxonomy Learning

1: **Input:** training set $\{(x^n, Y^n)\}_{n=1}^N$
2: **Output:** θ
3: $\theta := \mathbf{0}$ {in the setting of Algorithm 2, θ can be 'hot-started' with its previous value}
4: **repeat**
5: **for** $n \in \{1 \dots N\}$ **do**
6: $\hat{Y}^n := \operatorname{argmax}_{Y \in \mathcal{Y}} \{\Delta(Y, y^n) + \langle \phi(x^n, Y), \theta \rangle\}$
7: **end for**
8: Compute gradient g_i (equation (eq. 13))
9: Compute objective o_i (equation (eq. 12))
10: $\theta := \operatorname{argmin}_\theta \frac{\lambda}{2} \|\theta\|^2 + \max(0, \max_{j \le i} \langle g_j, \theta \rangle + o_j)$
11: **until** converged (see Teo et al. (2007))
12: **return** θ

Algorithm 2 Taxonomy Learning with Latent Variables

1: **Input:** training set $\{(x^n, y^n)\}_{n=1}^N$
2: **Output:** θ
3: $\theta^0 := \mathbf{1}$
4: **for** $i = 1 \dots I$ **do**
5: $Z_i^n := \left\{\operatorname{argmax}_{Y \in \mathcal{Y}} \left\langle \Phi(x^n, Y), \theta^{i-1} \right\rangle\right\} \setminus \{y^n\}$ {choose only $K - 1$ distinct labels}
6: $\theta^i := \text{Algorithm 1}\left(\left\{\left(x^n, Z_i^n \cup \{y^n\}\right)\right\}_{n=1}^N\right)$
7: **end for**
8: **return** θ^I

3.3 Column Generation

Given the loss function of (eq. 2), obtaining the most violated constraints (Algorithm 1, Line 6) takes the form

$$\hat{Y}^n = \operatorname*{argmax}_{Y \in \mathcal{Y}} \left\{ \min_{y \in Y} d(y, y^n) + \sum_{y \in Y} \langle \phi(x^n, y), \theta \rangle \right\}, \tag{14}$$

which appears to require enumerating through all $Y \in \mathcal{Y}$, which if there are $C = |\mathcal{C}|$ classes amounts to $\binom{C}{K}$ possibilities. However, if we know that $\operatorname{argmin}_{y \in \hat{Y}^n} d(y, y^n) = c$, then (eq. 14) becomes

$$\hat{Y}^n = \operatorname*{argmax}_{Y \in \mathcal{Y}'} \left\{ d(c, y^n) + \sum_{y \in Y} \langle \phi(x^n, y), \theta \rangle \right\}, \tag{15}$$

where $\mathcal{Y}'$ is just $\mathcal{Y}$ restricted to those y for which $d(y, y^n) \ge d(c, y^n)$. This can be solved greedily by sorting $\langle \phi(x^n, y), \theta \rangle$ for each class $y \in \mathcal{C}$ such that $d(y, y^n) \ge d(c, y^n)$ and simply choosing the top K classes. Since we don't know the optimal value of c in advance, we must consider all $c \in \mathcal{C}$, which means solving (eq. 15) a total of C times. Solving (eq. 15) greedily takes $O(C \log C)$ (sorting C values), so that solving (eq. 14) takes $O(C^2 \log C)$.

Although this method works for any loss of the form given in (eq. 2), for the specific distance function $d(y, y^n)$ used for the ImageNet Challenge, further improvements are possible. As mentioned, for the ImageNet Challenge's hierarchical error measure, $d(y, y^n)$ is the shortest-path distance from y to the nearest common ancestor of y and y^n in a taxonomic tree. One would expect the depth of such a tree to grow logarithmically in the number of classes, and indeed we find that we always have $d(y, y^n) \in \{0 \dots 18\}$. If the number of discrete possibilities for $\Delta(Y, y_n)$ is small, instead of enumerating each possible value of $c = \text{argmin}_{y \in \hat{Y}^n} d(y, y^n)$, we can directly enumerate each value of $\delta = \min_{y \in \hat{Y}^n} d(y, y^n)$. If there are $|L|$ distinct values of the loss, (eq. 14) can now be solved in $O(|L|C \log C)$. In ImageNet we have $|L| = 19$ whereas $C = 1000$, so this is clearly a significant improvement.

Several further improvements can be made (e.g. we do not need to sort all C values in order to compute the top K, and we do not need to re-sort all of them for each value of the loss, etc.). We omit these details for brevity, though our implementation shall be made available at the time of publication.[4]

4 Experiments

4.1 Binary Classifiers

As previously described, our approach needs, for each class, one binary classifier able to provide some reasonable score as a starting point for the proposed method. Since the objective of this paper is not beating the state-of-the-art, but rather demonstrating the advantage of our structured learning approach to improve the overall classification, we used a standard, simple image classification setup. As mentioned, should the one-vs-all classifiers of Lin et al. (2011) or Sánchez and Perronnin (2011) become available in the future, they should be immediately compatible with the proposed method.

First, images have to be transformed into descriptor vectors sensible for classification using machine learning techniques. For this we have chosen the very popular Bag of Features model (Csurka et al., 2004): dense SIFT features are extracted from each image x^n and quantized using a visual vocabulary of F visual words. Next, the visual words are pooled in a histogram that represents the image. This representation is widely used in state-of-the-art image classification methods, and despite its simplicity achieves very good results.

Regarding the basic classifiers, a rational first choice would be to use a Linear SVM for every class. However, since our objective is to predict the correct class of a new image, we would need to compare the raw scores attained by the classifier, which would not be theoretically satisfying. Although it is possible to obtain probabilities from SVM scores using a sigmoid trained with the Platt algorithm, we opted for training Logistic Regressors instead, which directly give probabilities as output and do not depend on a separate validation set.

In order to deal with the computational and memory requirements derived from the large number of training images, we used Stochastic Gradient Descent (SGD) from

[4] see http://users.cecs.anu.edu.au/~julianm/

Bottou and Bousquet (2008) to train the classifiers. SGD is a good choice for our problem, since it has been shown to achieve a performance similar to that of batch training methods in a fraction of the time (Perronnin et al., 2010). Furthermore, we validated its performance against that of LibLinear in a small-scale experiment using part of the ImageNet hierarchy with satisfactory results. One limitation of online learning methods is that the optimization process iterations are limited by the amount of training data available. In order to add more training data, we cycled over all the training data for 10 epochs.

With this approach, the θ^y_{binary} parameters for each class used in the structured learning method proposed in this work were generated.

4.2 Structured Classifiers

For every image x^n and every class y we must compute $\langle \phi(x^n, y), \theta \rangle$. Earlier we defined $\phi(x, y) = x \odot \theta^y_{\text{binary}}$. If we have C classes and F features, then this computation can be made efficient by first computing the $C \times F$ matrix A whose y^{th} row is given by $\theta^y_{\text{binary}} \odot \theta$. Similarly, if we have N images then the set of image features can be thought of as an $N \times F$ matrix X. Now the energy of a particular labeling y of x^n under θ is given by the matrix product

$$\langle \phi(x^n, y), \theta \rangle = \left(X \times A^T \right)_{n,y}. \tag{16}$$

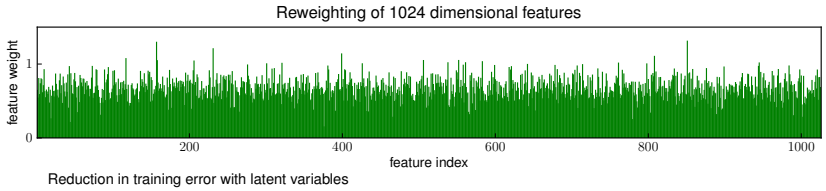
This observation is critical if we wish to handle a large number of images and high-dimensional feature vectors. In our experiments, we performed this computation using Nvidia's high-performance BLAS library CUBLAS. Although GPU performance is often limited by a memory bottleneck, this particular application is ideally suited as the matrix X is far larger than either the matrix A, or the resulting product, and X needs to be copied to the GPU only once, after which it is repeatedly reused. After this matrix product is computed, we must sort every row, which can be naïvely parallelized.

In light of these observations, our method is no longer prohibitively constrained by its running time (running ten iterations of Algorithm 2 takes around one day for a single regularization parameter λ). Instead we are constrained by the size of the GPU's onboard memory, meaning that we only used 25% of the training data (half for training, half for validation). In principle the method could be further parallelized across multiple machines, using a parallel implementation of the BMRM library.

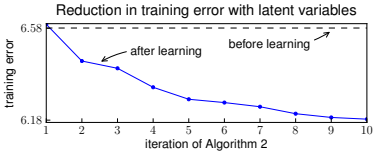
The results of our algorithm using features of dimension $F = 1024$ and $F = 4096$ are shown in Figures 2 and 3, respectively. Here we ran Algorithm 2 for ten iterations, 'hot-starting' θ^i using the optimal result from the previous iteration. The reduction in training error is also shown during subsequent iterations of Algorithm 2, showing that minimal benefits are gained after ten iterations. We used regularization parameters $\lambda \in \{10^{-1}, 10^{-2} \ldots 10^{-8}\}$, and as usual we report the test error for the value of λ that resulted in the best performance on the validation set. We show the test error for different numbers of nearest-neighbors K, though the method was trained to minimize the error for $K = 5$.

In both Figures 2 and 3, we find that the optimal θ is non-uniform, indicating that there are interesting relationships that can be learned between the features when a structured setting is used. As hoped, a reduction in test error is obtained over already good

(a)



(b)



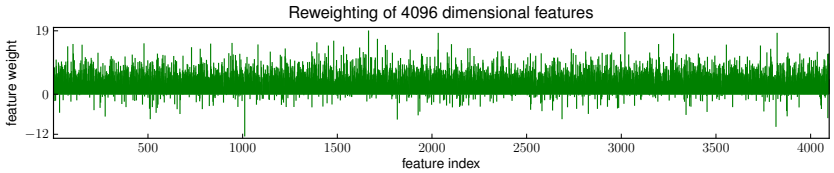
(c)

Test error:

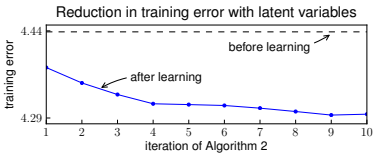
	1nn	2	3	4	5
Before learning	11.35	9.29	8.08	7.25	6.64
After learning	10.88	8.85	7.71	6.93	6.36

Fig. 2. Results for training with 1024 dimensional features. (a) feature weights; (b) reduction in training error during each iteration of Algorithm 2; (c) error for different numbers of nearest-neighbors K (the method was trained to optimize the error for $K = 5$). Results are reported for the best value of λ on the validation set (here $\lambda = 10^{-4}$).

(a)



(b)



(c)

Test error:

	1nn	2	3	4	5
Before learning	9.27	7.29	6.23	5.53	5.03
After learning	9.02	7.08	6.05	5.38	4.91

Fig. 3. Results for training with 4096 dimensional features. (a) feature weights; (b) reduction in training error during each iteration of Algorithm 2; (c) error for different numbers of nearest-neighbors K (the method was trained to optimize the error for $K = 5$). Results are reported for the best value of λ on the validation set (here $\lambda = 10^{-6}$).

classifiers, though the improvement is indeed less significant for the better-performing high-dimensional classifiers.

In the future we hope to apply our method to state-of-the-art features and classifiers like those of Lin et al. (2011) or Sánchez and Perronnin (2011). It remains to be seen whether the setting we have described could yield additional benefits over their already excellent classifiers.

5 Conclusion

Large scale, collaboratively labeled image datasets embedded in a taxonomy naturally invite the use of *both* structured *and* robust losses, to account for the inconsistencies in the labeling process and the hierarchical structure of the taxonomy. However, on datasets such as ImageNet, the state-of-the-art methods still use one-vs-all classifiers,

which do not account for the structured nature of such losses, nor for the imperfect nature of the annotation. We have outlined the computational challenges involved in using structured methods, which sheds some light on why they have not been used before in this task. However, by exploiting a number of computational tricks, and by using recent advances on structured learning with latent variables, we have been able to formulate learning in this task as the optimization of a loss that is both structured and robust to weak labeling. Better yet, our method leverages existing one-vs-all classifiers, essentially by re-weighting, or 'boosting' them to directly account for the structured loss. In practice this leads to improvements in the hierarchical loss of already good one-vs-all classifiers.

Acknowledgements

Part of this work was carried out when both AR and TC were at INRIA Grenoble, Rhône-Alpes. NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program. This work was partially funded by the QUAERO project supported by OSEO, French State agency for innovation and by MICINN under project MIPRCV Consolider Ingenio CSD2007-00018.

References

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009. 1, 2

Alex Berg, Jia Deng, and Fei-Fei Li. Imagenet large scale visual recognition challenge 2010. http://www.image-net.org/challenges/LSVRC/2010/index, 2010. 1, 2, 4

Yuanqing Lin, Fengjun Lv, Shenghuo Zhu, Ming Yang, Timothee Cour, and Kai Yu. Large-scale image classification: fast feature extraction and SVM training. In *IEEE Conference on Computer Vision and Pattern Recognition*, page (to appear), 2011. 1, 2, 3, 4, 6, 10, 12

Jorge Sánchez and Florent Perronnin. High-Dimensional Signature Compression for Large-Scale Image Classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, page (to appear), 2011. 1, 3, 4, 6, 10, 12

Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. *European Conference on Computer Vision*, pages 143–156, 2010. 2, 11

Antonio Torralba, Rob Fergus, and William T Freeman. 80 Million Tiny Images: a Large Data Set for Nonparametric Object and Scene Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1958–70, 2008. 2

Jia Deng, Alexander C. Berg, Kai Li, and Li Fei-Fei. What does classifying more than 10,000 image categories tell us? In *European Conference on Computer Vision*, 2010. 2

Qinfeng Shi, James Petterson, Gideon Dror, John Langford, Alex Smola, Alex Strehl, and Vishy Vishwanathan. Hash Kernels. In *Artificial Intelligence and Statistics*, 2009. 3

Herve Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2010. 3

Olga Russakovsky and L. Fei-Fei. Attribute learning in large-scale datasets. In *ECCV Workshop on Parts and Attributes*, 2010. 3

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005. 4, 7, 8

Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007. 4

Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The Pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010. 4

Chun-Nam John Yu and Thorsten Joachims. Learning structural svms with latent variables. In *International Conference on Machine Learning*, 2009. 7, 8

Choon Hui Teo, Alex Smola, S. V.N. Vishwanathan, and Quoc Viet Le. A scalable modular convex solver for regularized risk minimization. In *Knowledge Discovery and Data Mining*, 2007. 8, 9

Gabriela Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *ECCV Workshop on statistical learning in computer vision*, 2004. 10

Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In *Neural Information Processing Systems*, 2008. 11