Inaugural-Dissertation

zur
Erlangung der Doktorwürde
der
Naturwissenschaftlich–Mathematischen Gesamtfakultät
der
Ruprecht–Karls–Universität
Heidelberg

vorgelegt von
Dipl.–Inf. Jörg Hendrik Kappes
aus Heidelberg

Tag der mündlichen Prüfung: 18.04.2011

Inference on Highly-Connected Discrete Graphical Models with Applications to Visual Object Recognition

Gutachter: Prof. Dr. Christoph Schnörr
Prof. Dr. Gerhard Reinelt

Abstract

Object detection is one of the key components of modern computer vision systems. While the detection of a specific rigid object under changing viewpoints was considered hard just a few years ago, current research strives to detect and recognize classes of non-rigid, articulated objects. Hampered by the omnipresent problems due to clutter and occlusion, the focus has shifted from holistic approaches for object detection to representations of individual object parts linked by structural information, along with richer contextual descriptions of object configurations.

Along this line of research, we present a practicable and expandable probabilistic framework for parts-based object class representation, using probabilistic graphical models, enabling the detection of rigid and articulated object classes in arbitrary views. We investigate computational learning of this representation from labelled training images and infer globally optimal solutions to the contextual maximum a posteriori (MAP) detection problem for object recognition.

Both, learning the model parameters and inferring the optimal solution of such models, define combinatorial optimization problems that in general are NP-hard. Restrictions to feasible subclasses of models yield problems that can be solved efficiently. However, due to this limitation of modeling, not all relevant dependencies can be represented accurately. Thus, research has turned towards more general models that are associated with more complex optimization problems.

In this thesis, we propose novel methods for solving the MAP problem for models encountered in our object detection applications. Our first ansatz transforms the MAP problem into a shortest path problem, which is solved by an A^*-search with an admissible tree-based heuristic. As the A^*-method does not scale well to large problems, we investigate convex relaxations of the inference problems using the theory of exponential families. Since solving the corresponding convex problem is still computationally infeasible due to an enormous number of affine constraints, we present a novel dual decomposition approach to MAP problems for highly connected discrete graphical models. We show that decompositions into cyclic, k-fan structured subproblems significantly tighten the Lagrangian relaxation compared to the standard local polytope relaxation. Furthermore, efficient integer programming for solving the subproblems with our A^*–approach remains computationally feasible.

We evaluate the proposed algorithms on synthetic and real world data and compare their performance to standard methods from the field of computer vision as well as to commercial standard solvers for linear and mixed integer programs. With the exception of larger problem instances our A^*-approach outperforms all other methods in terms of run time and accuracy. Moreover, the dual decomposition methods show potential in terms of accuracy, but suffers from slower convergence.

Zusammenfassung

Das Erkennen und Finden von Objekten in Bildern ist eines der wichtigsten Teilprobleme in modernen Bildverarbeitungssystemen. Während die Detektion von starren Objekten aus beliebigen Blickwinkeln vor einigen Jahren noch als schwierig galt, verfolgt die momentane Forschung das Ziel, verformbare, artikulierte Objekte zu erkennen und zu detektieren. Bedingt durch die hohe Varianz innerhalb der Objektklasse, Verdeckungen und Hintergrund mit ähnlichem Aussehen, ist dies jedoch sehr schwer. Des Weiteren wird die Klassifikation der Objekte dadurch erschwert, dass die Beschreibung von ganzheitlichen Modellen häufig in dem dazugehörigen Merkmalsraum keine Cluster bildet. Daher hat sich in den letzten Jahren die Beschreibung von Objekten weg von einem ganzheitlichen hin zu auf Teilen basierenden Modellen verschoben. Dabei wird ein Objekt aus einer Menge von individuellen Teilen zusammen mit Informationen über deren Abhängigkeiten beschrieben.

In diesem Zusammenhang stellen wir ein vielseitig anwendbares und erweiterbares Modell zur auf Teilen basierenden Objekterkennung vor. Die Theorie über probabilistische graphische Modelle ermöglicht es, aus manuell notierten Trainingsdaten alle Modellparameter in einer mathematisch fundierten Weise zu lernen. Ein besonderer Augenmerk liegt des Weiteren auf der Berechnung der optimalen Pose eines Objektes in einem Bild. Im probabilistischem Sinne ist dies die Objektbeschreibung mit der maximalen *a posteriori* Wahrscheinlichkeit (MAP). Das Finden dieser wird auch als das MAP-Problem bezeichnet.

Sowohl das Lernen der Modellparameter als auch das Finden der optimalen Objektpose bedingen das Lösen von kombinatorischen Optimierungsproblemen, die in der Regel NP-schwer sind. Beschränkt man sich auf effizient berechenbare Modelle, können viele wichtige Abhängigkeiten zwischen den einzelnen Teilen nicht mehr beschrieben werden. Daher geht die Tendenz in der Modellierung zu generellen Modellen, welche weitaus komplexere Optimierungsprobleme mit sich bringen.

In dieser Arbeit schlagen wir zwei neue Methoden zur Lösung des MAP-Problems für generelle diskrete Modelle vor. Unser erster Ansatz transformiert das MAP-Problem in ein 'Kürzeste-Wege-Problem', welches mittels einer A^*-Suche unter Verwendung einer zulässigen Heuristik gelöst wird. Die zulässige Heuristik basiert auf einer azyklisch strukturierter Abschätzung des ursprünglichen Problems. Da diese Methode für Modelle mit sehr vielen Modellteilen nicht mehr anwendbar ist, betrachten wir alternative Möglichkeiten. Hierzu transformieren wir das kombinatorische Problem unter Zuhilfenahme von exponentiellen Familien in ein lineares Programm. Dies ist jedoch, bedingt durch die große Anzahl von affinen Nebenbedingungen, in dieser Form praktisch nicht lösbar. Daher schlagen wir eine neuartige Zerlegung des MAP Problems in Teilprobleme mit einer k-fan Struktur vor. Alle diese Teilprobleme sind trotz ihrer zyklischen Struktur mit unserer A^*-Methode effizient lösbar. Mittels der Lagrange-Methode und dieser Zerlegung erhalten wir bessere Relaxationen als mit der Standardrelaxation über dem lokalen Polytope.

In Experimenten auf künstlichen und realen Daten wurden diese Verfahren mit Standardverfahren aus dem Bereich der Bildverarbeitung und kommerzieller Software zum Lösen von lineare und ganzzahlige Optimierungsproblemen verglichen. Abgesehen von Modellen mit sehr vielen Teilen zeigte der A^*-Ansatz die besten Ergebnisse im Bezug auf Optimalität und Laufzeit. Auch die auf k-fan Zerlegungen basierenden Methode zeigte viel versprechende Ergebnisse bezüglich der Optimalität, konvergierte jedoch im Allgemeinen sehr langsam.

Danksagung

In meiner Zeit als Doktorand wurde ich von vielen Personen unterstützt, ohne die mein Promotionsvorhaben wohl kaum in dieser Form möglich gewesen wäre.

Schon während meines Studiums führte mich mein Doktorvater, Professor Christoph Schnörr, an das faszinierende Gebiet der digitalen Bildverarbeitung heran und gab mir schließlich die Möglichkeit, in seiner Forschungsgruppe zu promovieren. Neben seinen detaillierten Kenntnissen auf seinen Forschungsgebieten lernte ich auch seine menschliche Seite kennen und schätzen. Die Zusammenarbeit mit ihm war für mich eine wertvolle und positive Erfahrung. Ich danke ihm vielmals für diese Zeit und werde sie sicherlich in guter Erinnerung behalten.

Weiterhin geht mein Dank an alle meine ehemaligen und gegenwärtigen Kolleginnen und Kollegen der Universität Heidelberg. Speziell meine Mitstreiter in der IPA-Gruppe boten mir viele fruchtbare Diskussionen, aber auch Freundschaft und schöne gemeinsame Erlebnisse in- und außerhalb der Universität. Ganz besonders hervorheben möchte ich Florian Becker und Dirk Breitenreicher, die große Teile dieser Arbeit Korrektur gelesen haben und mir wertvolle Hinweise gaben. Des Weiteren möchte ich bei Martin Bergtholdt, Jan Lellmann, Stefan Schmidt, Björn Andres und Bogdan Savchynskyy für die gemeinsamen Arbeit, Diskussionen und Publikationen bedanken.

Bei den kleinen und größeren administrativen Problemen, standen mir Rita Schieker, Evelyn Verlinden und Barbara Werner mit Rat und Tat zur Seite, wofür ich ihnen an dieser Stelle noch einmal danken möchte.

Schließlich möchte ich mich bei meiner Frau Susanne Kappes-Jung bedanken, die mir während meiner Promotion immer zur Seite stand und mich immer wieder neu motivierte, wenn der Weg zu steinig zu sein schien. Mich während des Zusammenschreibens zu ertragen, mag an dem ein oder anderen Tag sogar schwerer gewesen sein als das Schreiben an sich – nicht nur dafür danke ich ihr von ganzem Herzen!

CONTENTS

CHAPTER 1

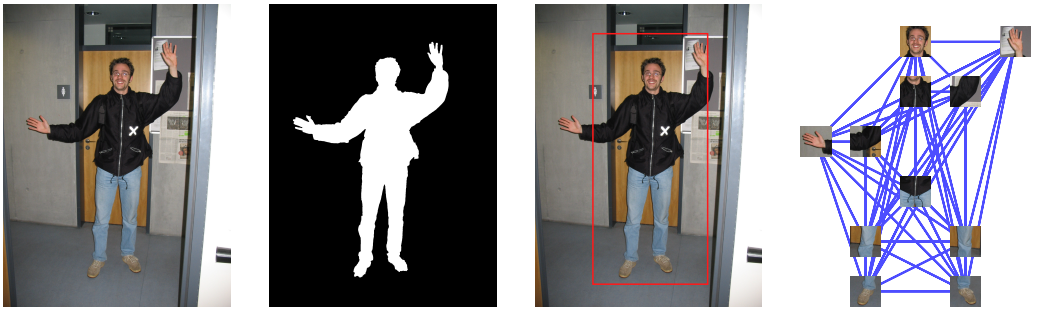
INTRODUCTION

1.1. Motivation

The recognition and detection of objects in images are one of the key components in modern computer vision systems and fundamental for image understanding. The recognition of the presence of instances of object classes are relevant for the classification of images, e.g. find all images which containing an object of a specific object class. Figure 1.1 shows some images in which prototypical object classes are visualized. The different objects in

Figure 1.1.: Examples for typical object classes for recognition and detection problems. Due to the different appearance and geometry of the objects – between but also within the classes – it is not obviously how objects can be represented in a efficient and preferably invariant manner.

Figure 1.1 have a dissimilar representation in the images, in terms of appearance and geometry. It is not obviously how to construct a probabilistic model which decide if an object of a specific class is present in the image. The problem becomes even more involved if we would like to detect the objects in the images, i.e. finding the position or the pose of an object in a scene.



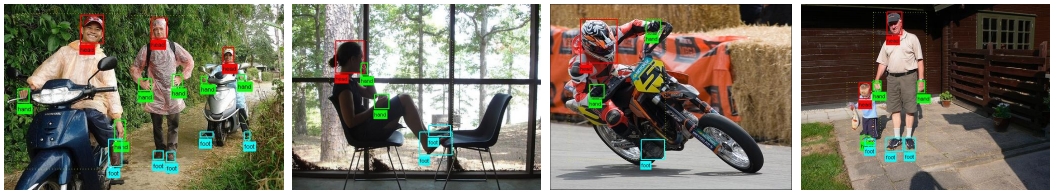
(a) (b) (c) (d)

Figure 1.2.: Possible representation of an object (here a person) in an image are a pixelwise segmentation of the image **(b)**, a bounding box around the object **(c)** or a part-based description by the positions of the object parts in the image **(d)**.

Figure 1.2 shows three possible ways to tackle this problem. In the first ansatz (Figure 1.2b), each pixel in the image is assigned to a certain class, in this simple case *person* or *background*. This segmentation is a very detailed description of the object. This high dimensional description implicate several complications in applications. Solving problems like detecting a person in an image which such models becomes often too complex if no further simplifications are assumed. A very simple description of an object is given by a single bounding box around the object, as shown in Figure 1.2c. This bounding box includes little information of the pose of the object and just indicates that the image area bound by the box contains a person. The image patch inside of this box is described by a feature vector. This feature vector has to cope with the variability of the single class as well as the change of background appearance. Hampered by the omnipresent confusion information due to clutter and occlusion as well as the hight intraclass variability, the focus of research has shifted from holistic models to models which represent an object by a set of individual object parts linked by structural information. Figure 1.2d illustrates the idea of such part-based models. Its underlying assumption is, that the detection of the single parts is much simpler than the detection of the complete object at once. This assumption is motivated by the fact that the intraclass variability of a single part is much smaller than the intraclass variability of the complete object. The structural information between the parts are usually geometric prior knowledge. The object itself is represented by the individual positions of the parts.

While for the bounding box ansatz usually a sliding window approaches with cascaded classifiers is used, it is common to use probabilistic graphical models as mathematical description for the other two representation. In these cases the state of each random variable is associated with the corresponding class of a pixel or the part position. Since the set of all object classes or part positions in the image is discrete, we are confronted with combinatorial problems, when we would like to learn the probabilistic model or infer the optimal representation of an object for a given image. Additional assumption on the conditional independences of the variables and the factorization of the distribution makes this problems computationally feasible. Therefore, many approaches use tree structured models exclusively, for which efficient inference methods exist. Simplifying the problem such that optimization becomes feasible causes usually weaker models. On the other hand more powerful models causes more complex optimization problems, which often can only be solved approximative. Even if the performance of approximative methods is adequate

(a)



(b)

Figure 1.3.: Detecting humans in images is a very challenging task. Different appearance, pose and scale of the person itself as well as highly cluttered and arbitrary background causes that this problem is very hard without any further restrictions on the problem. Images annotated by hand labeled ground truth information from the VOC person layout challenge [41] are shown in **(b)**.

for many applications, we face an essential problem: When the approach fails on some data, we do not know if this due to an insufficient model or approximative inference methods.

In this work we investigate a part-based approach for human recognition and pose detection in single images. Detecting the humans in the example images in Figure 1.3a is due to different appearance, pose and scale of the person itself as well as highly cluttered and arbitrary background very challenging. The corresponding hand labeled ground truth notation from the VOC person layout challenge is shown in Figure 1.3b. While we focus on the detection of the human pose, our model generalizes straight forward to many other object classes. Besides the human body model we consider a model of the human face, containing the parts eyes, nose and mouth corners, as simple toy example and a model of the human spine applied on 3D magnet resonance images, in which the parts are the intervertebral disks. In all cases we apply the same framework.

Figure 1.4 sketches the main idea of our approach. We decompose an object in a set of several parts, which are represented by their position in the image. Object detection in this context means to find for each part a candidate (position) such that the set of all part positions describes the image best. Local classifiers are used to reduce the number of possible candidates for each part to a few tens. However, for an object, which consists of several parts, the set of all possible configurations is too large.

We assume that the graphical model includes a fully connected graph and the underlying probability distribution factorizes into terms which depend on at most two random variables. Even if this is major simplification on the model, the model is still very powerful. Contrary to tree structured models, full connected models can for example model the occlusion of parts. For a set of images with hand annotated part positions we learn local probabilities which depend on one or two random variables. This includes local probabilities for the appearance of each part and the image region between the parts, which are associated with edges. Furthermore the model includes prior distributions, which model the geometric relations between the parts. The weight of each of this terms in the model can be selected manually or learned by optimizing a set of parameters which weight the

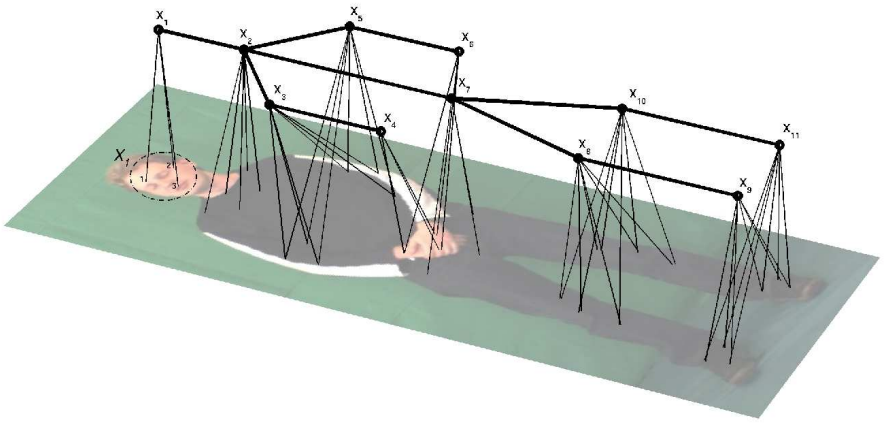


Figure 1.4.: The main idea of our approach is to represent an object, in this case the human body, by a set of parts, marked as nodes in the graph. The edges in this graph are just for illustration and mark the physical relations between the parts. In the image we detect several possible locations for each part. We would like to find for each part a candidate such the configuration of the human body in this image is the most probable one.

functions with respect to some criteria, e.g. maximizing the probability for the trainings data. However, the inference problems on these models are NP-hard. The inference problems of our interest are firstly the calculation of marginal distributions and secondly the calculation of the mode or most probable configuration and known as maximum a posteriori (MAP) problem or MAP-inference. We will introduce several methods which solves inference problems approximately and exactly. For the exact solution of the MAP-problem we discuss a fast Branch and Bound algorithm using tree based admissible heuristics. In another line of research, convex relaxations by Lagrangian decomposition are investigated. Surprisingly, standard method from integer programming seems to perform better than reported in the literature. However, problem specialized algorithms are significantly faster and approaches using convex relaxation can be applied to problem sizes for which standard mixed integer problem solvers are not applicable any more.

1.2. Related Work

1.2.1. Modeling

The related literature concerning models for object detection in 2D images is vast and we make no claim to give a complete overview here, we will only give a roughly overview and point out the most important works related to our approach. For the recent work on object specific pixel-wise segmentation and detection by bounding boxes in images we refer to the results of the corresponding PASCAL-VOC Challenges [42]. We will discuss the methods performing best [25, 45, 162] in Chapter 3.

People detection and human pose estimation have a large variety of applications such as automotive safety, surveillance, and image or video indexing. The research in pedestrian detection includes among others [165, 36, 109, 145, 116, 42] and focus on the detection of the human itself and not on its pose. Since we are interested in detecting the pose, we

will not follow this line of research. Moreover, we will not consider tracking approaches [4, 131, 51] which use of temporal information. Furthermore, our approach is view-based in order to keep its applicability to different object categories straightforward. We do not exploit category-specific 3D prior knowledge as e.g. in [21] for the case of humans, or as in [108, 136, 151, 6] in very detailed form.

Concerning the problem of human pose estimation, a major line of research dealing with pictorial structure models, which had been introduced by Fischler and Elschlager [52]. The idea of pictorial structure models is to represent an object by a set of parts arranged in a deformable configuration and model the appearance of each part separately. Given a configuration, which describes the pose of the object, an energy function that measure both, a match cost in each part and a deformation cost for all pairs of connected parts, can be evaluated. The configuration with the lowest energy is the best match for the object in the image. This idea has been used for pose estimate and object detection in several works [46, 5, 48, 148, 79, 66, 160, 11]. Most of these works use probabilistic models instead of energy based ones. Typically these probabilistic models include a data term, coding the object appearance by a product of independent local probabilities, and a prior on the geometry, also called shape of the object. The main advantage of probabilistic models is that sophisticated learning methods can be applied, see e.g. [103, 102].

To keep optimization computationally feasible many approaches use only tree structured prior for the geometric relations [46, 5, 48]. As shown in [133, 148, 79, 66, 174, 11, 160], this causes several limitations, e.g. assigning both legs to the same image region, in presence of noise and occlusion. Including additional terms, which yields to cyclic structures, may help to resolve this problem [133, 148, 79, 66, 174, 11, 160], but involve more complex problems for learning and detection. Alternatively, in [46] a tree model is used to sample several hypothesize and evaluate these on a global Chamfer distance [22]. However, sampling includes a random process and it is not clear which energy function is optimized. In other works which use tree structured models [5, 48], this problem is ignored or tried to bear down by stronger data terms. Pham et al. [126] introduced models with Gaussian distributed shape priors. While modeling occlusion and missing parts can be obtained very smart, the set shape prior is restricted to a unimodal distribution, which is not accurate for restricted set of human poses.

Modeling becomes much more complex when occluded or missing parts are present, as the assumption that each part is located in the image does not hold any more. Occlusion can be caused by self-occlusion or occlusion by another object. Furthermore, the part can be missing if it is locate outside the image. For tree structured models it is essential that the "kinematic chain" is not broken, otherwise the model is torn and the single connected components are treated without any geometric relation. Similar to our work, several approaches [174, 79] try to overcome this problem by including an additional candidate for each part. Since this candidate is not restricted to a fix image position, additional terms are required to ensure the "connectivity" of the model.

Contrary to the major part of these approaches for human pose estimation in 2D images, which use body limbs as parts, i.e. torso, lower arms, upper arms etc. , we use the joints of the human body, such as head, breast elbows, hands, etc. as parts. This has essential advantage as the representation of the single parts is much smaller (only position instead of position, orientation and aspect ratio). Since the appearance of the regions including these limbs is an important input feature for the model, we include pairwise appearance terms to our model which include this information of limb regions.

1.2.2. Optimization

Concerning the optimization task, for acyclic models dynamic programming is used in all applications. Even if the used methods named different, e.g. Viterbi algorithm, forward-backward algorithm and belief propagation, they always following the same dynamic programming strategy. For cyclic graphs, loopy belief propagation (LBP) has been widely used in computer vision. Due to its simple implementation and empirically good performance, it is very attractive, but for decades it was unclear what LBP optimize. Later investigations by Yedida et al. [186, 187] show that fix points of the LBP coincide with stationary points of the Bethe variational problem, known from statistical physics. For the MAP-problem Weiss and Freeman [176] has shown that fix points of LBP for the MAP-Problem it can be guaranteed that the resulting MAP configuration has higher probability than any other configuration in the "Single Loops and Trees" neighborhood.

If the objective function of the problem belong to some special class of functions[1], graph cut or iterative graph cut methods have become state of the art methods [91, 137, 89, 141, 95, 96, 92, 130, 85]. The main idea of these approaches is to transform the MAP-problem into a single or a sequence of st-min-cut-problems which can be solved in polynomial time. For submodular problems this leads to global optimal solutions of the MAP-problem.

Since in general problems belong not to these special classes we investigate a branch and bound algorithm in a A^*-search framework. Similar to the graph cut approaches, we transform the MAP-problem to a classical graph theoretical problem, i.e. the shortest path problem. However, the graph in which the shortest path between two nodes is searched, has exponential many paths between them. We overcome this problem by bounding the true costs for the paths such that not all paths have to be explored. We have applied this framework for MAP-inference on our pose estimation problem. While in general the complexity is exponential in the number of nodes, it empirically outperforms all state of the art algorithms including MIP-solvers, in run-time and optimality. Similar methods in the field of computer vision are used by Fergus et al. [50, 47] and Pham and Smeulders [126] in connection with Gaussian priors, for which lower bounds can be observed straight forward. In [12] we present a generalization of this method for arbitrary discrete models. In a later work [11], we presented tighter bounds for the heuristic. Currently, several other researches come up with similar ideas. Schlesinger [142] uses linear programs for the calculation of the bounds and Tian et al. [160] use a reserved branching strategy but calculate the bounds similar to our approach.

Furthermore, our A^*-approach is strongly related to branch and bound algorithm used in MIP-solvers. Similar to these approaches we subdivide the feasible set iteratively and bound the objective for these sets. Our method differ in two important points, firstly, we use dynamic programming instead of linear programming relaxations for bounding and secondly decompose the sets into several subsets in each step. For larger and more complex problems this can cause a decomposition in to many subsets. Contrary, state of the art MIP-solvers use heuristics which efficiently avoid this.

A broad class of methods are based on the principle of convex relaxations, in which the discrete optimization problems are relaxed to convex optimization problems over continuous domains. For a general overview of convex relaxations for inference problems on graphical models we refer to [100]. We will restrict ourself to linear programming (LP)

[1] This special functions are submodular, metric and semi-metric functions and will be discussed in Section 4.2.4

relaxations. The MAP-problem can be describe exactly by a LP. However the complexity of the problem causes in general an exponential number of constraints. Since the exact LP is in general intractable, outer relaxations on the polytope defined by the constraint set are used. A commonly used relaxation approximates the exact polytope by the so called local polytope. The local polytope is an outer relaxation which is only defined by local constraints. For general graphs, this first-order LP relaxation can be solved – at least in principle – by various standard algorithms from linear programming, including simplex and interior point methods [14, 18]. However, such general methods do not make use of the graphical structure of the problem, and hence do not scale favorably for large scale problems as reported by Yanover et al. [185].

In a series of works Wainwright and colleagues [167, 172, 169, 170, 87, 135, 173] has investigated in convex relaxations of the inference-problems and introduce message passing algorithms which solve these problem by block coordinate descent like scheme. Contrary to general purpose LP solvers, this tree reweighted message passing algorithm (TRW), also known tree reweighted belief propagation (TRBP), scales for larger problems. While TRW-algorithms working on the dual of the LP which is in general non smoothness, TRW methods can get stuck in points which satisfy some local criteria known as weak tree agreement [87]. Instead of sequentially sending messages as suggested by Kolmogorov [87], known as TRW-S, Wainwright [173] suggested to updated all messages in parallel. This avoids the overhead for scheduling the messages and enables to update messages straight forward in a parallel implementation. Empirically, Wainwright [173] observed that the parallel updating strategy with sufficient logarithmic damping of the messages convergences. The fix points for TRW are the same as for its sequential version. Both LBP and TRW can be generalized to higher order relaxations by including higher order cluster-based constraints into the relaxation. These Kikuchi or convexified Kikuchi-based methods are discussed in [173] and will not used in this thesis. Due to their higher complexity they are scarcely used in computer vision. A related line of research add incremental violated higher order constraints to the LP relaxation. For further details to this cutting plane methods see [180, 154].

In this context we follow an alternative line of research which addresses Lagrangian decomposition [65, 64] for graphical models, also known as dual decomposition [94]. Contrary to TRW-algorithms this method get not stuck in local fix points. However, solving the non-smooth dual problem by sub-gradient methods, includes the selection of a step sizes, which is not trivial and influence the speed of convergence. Typically, this methods have sub-linear convergence rates [14].

The calculation of the sub-gradients can be done by solving the MAP-problem for the subproblems, which are sufficiently simpler than the original problem. Komodakis [94] used this method together with tree-structure subproblems. This leads to the same relaxation as used by TRW-algorithms. In later work [93], he used simple cyclic-subproblems for which MAP-inference is feasible. Since more complex sub-tructures leads to tighter relaxations, these frameworks are from main interest for applications where simple relaxations are not sufficiently tight. Currently, Batra et al. [7] have used the dual decomposition technique together with outer-planar subproblems and Strandmark and Kahl [155], together with submodular subproblems. In this line of research we introduce a decomposition in k-fan-structured subproblems [82]. k-fans includes a clique of k nodes. All other nodes are only connected to the clique members. As long as k gets not to large this sub-problems can be solved by dynamic programming or branch and bound algorithms. For tight relaxation this method calculates the global optimal integer solution, otherwise it ends up with an integer solution and a bound given by the solution of the relaxed problem.

1.3. Contribution

We present a part-based model which is able to detect the pose of an object in an image in presence of clutter and collusion. To train the complete model we require only a set of images which include the position of the parts centers. The use of fully connected models allows strong models, including terms considering symmetry and occlusion. However, exact inference on such models was so far not possible in computer vision. We introduced a branch and bound algorithm based on a best first search strategy. This A^*-search algorithm uses an admissible tree-based heuristic for on-line search space reduction and outperforms state of the art algorithms in time and optimality. While this algorithm do not scale to larger problems we investigate Lagrangian decompositions and introduce a framework which use significant tighter relaxations than the standard relaxations which use local polytope relaxation. Although a common preconception in the field of computer vision is that, standard solvers from mixed integer programming (MIP) are not applicable to computer vision problems, we show the converse. On several computer vision problems MIP-solvers perform well. We show some examples where MIP-solvers, like CPLEX, can be apply directly on computer vision problems. In an empirical evaluation on real world and syntactic data, we compare the different algorithms and show their limitations.

1.4. Organization

This thesis is organized as follows. In Chapter 2, we start with the basic definitions from the field of graph theory, followed by an introduction to graphical models. We will discuss the correspondence of conditional independences of random variables to edges in a probabilistic graphical model. In the second part of this chapter we introduce factor-graphs which focus in contrast to the standard representation of probabilistic graphical models on the factorization properties of the underling distribution or energy function of the graphical model. As a fundamental working horse, we will introduce the theory of exponential families which builds the bridge to the area of convex analysis and variational inference. We especially discuss exponential families for discrete graphical models. Finally, we introduce conditional random fields and conclude Chapter 2 with some simple examples for illustration.

The importance of graphical models for visual object detection in the field of computer vision will be discussed in Chapter 3. Starting with fundamental modeling aspects, we introduce several approximating techniques for real world scenarios. These models are used to infer information required in computer vision applications. Starting with classical problems, which operate on grid-structured models, such as segmentation, stereo vision, or image denoising we proceed with detection problems and the problems tackled in the Visual Object Classes Challenge (VOC) [41]. Here, the goal is to detect objects of different classes in an image by segmentation, a single bounding box, or several bounding boxes. Our contribution in this context is a part-based model which detects the pose of an object, defined by the position of a set of parts. We will introduce this model in detail and show evaluations on 3 different object classes, i.e. faces, human body and human spine. The last model includes 3D-data and a 3D-model.

The optimization problems occurring in these frameworks are discussed in Chapter 4. We focus on calculating a maximum a posteriori probability (MAP) configuration, called MAP-inference. A MAP-configuration is a mode of the posterior distribution and for our

choice of models equivalent to the configuration which minimize the corresponding energy function. The second problem which we will discuss, is the calculation of the marginal distributions, which is required for maximum likelihood learning of the parameters of the probability distribution. We divide the algorithms into two main classes; while in the first class optimization makes use of special properties of the models and transforms the inference-problems into solvable problems well known in graph theory, the second class includes algorithms which are based on concepts from convex analysis. Finally, we will compare the inference algorithms for several graphical models, including syntactic models and models based on real world data. We end up this thesis by a final conclusion.

1.5. Notation

We assume that the reader is familiar with the standard notations from set theory, graph theory, probability theory, convex analysis, and vector calculus. Although we use standard notation, we will start with briefly introduce the basic notations, in order to avoid misunderstandings.

Sets will be denoted with capital letters, scalars and vectors with small ones. We will not distinguish between scalars and vectors since this should be clear from context. Let $\emptyset$ denote the empty set, $A \cup B$ is the union, $A \cap B$ the intersection, and $A \setminus B$ the difference of two sets. A set A contains an element $a \in A$ and includes a subset $B \subset A$. If B is a proper subset of A (i.e. $B \subset A$ and $B \neq A$) we write $B \subsetneq A$. We use the standard notation ($\mathbb{R}$) for real numbers and denote the natural numbers including the zero by $\mathbb{N}$. For two real values $a, b \in \mathbb{R}$ we denote the sets $[a, b] := \{c \in \mathbb{R} | a \leq c \leq b\}$ and $(a, b) := \{c \in \mathbb{R} | a < c < b\}$. Given two sets A and B we write $A \times B$ for the Cartesian product set $(\{(x, y) | x \in X, y \in Y\})$. For a Cartesian product of a sequence of sets, we use the notation $\bigotimes_i A^i$. The power set of a A, denoted by $\mathcal{P}(A)$, is the set of all subsets of A including per definition the empty set. The cardinality of a set A is noted by $|A|$ and the dimension of a vector x is noted by $|x|$. For a vector x, x_i is the i-th element of the vector and for a matrix A, A_{ij} is the element in the i-th row and j-th column of A. For a set S, we denote with x_S the sub-vector of x given by $(x_a)_{a \in S}$. The inner product of two vectors x and y is denoted by $\langle x, y \rangle = \sum_i x_i \cdot y_i$ and the standard L_2 norm of a vector x by $\|x\| = \sqrt{\sum_i (x_i)^2}$.

A graph, in its standard form, $G = (V, E)$ is a pair of a set of nodes V and a set of edges E which connect the nodes. The edges in a graph can be directed or undirected. A special kind of graph which is used in this thesis is the factor graph $G = (V, F, E)$. A factor graph is a bipartite graph represented by a triple (V, F, E), where V is a set of variable nodes, F is a set of factor nodes, and $E = V \times F$ is a set of edges.

We denote random variables and random vectors with capital letters X and the values taken by random variables with small letters x. All possible events of a random variable is denoted by calligraphic letters $\mathcal{X}$ and called sample space or domain of the random variable within this thesis. To each random variable X we associate a probability distribution (density or mass function) $p(x)$.

A graphical model is a pair (X, G) of a random vector X, including the probability distribution, and a graph $G = (V, E)$, coding the conditional independence and factorization of the distribution. To each node $a \in V$ corresponds a random variable X_a taking values in $\mathcal{X}_a$. For $A \subset V$ we denote the random sub-vector of X which includes the random

variables to the nodes in A by $X_A := (X_a)_{a \in A}$. One single value in $\mathcal{X}$ is called configuration, label or setting and we denote sub-configuration and sub-label for values of $\mathcal{X}_S$ accordingly. Further important definitions are given in Section 2.3.

Given the function $f(x) = f(x_1, \ldots, x_n)$, we define the gradient of f as the vector

$$\nabla f(x) = \left(\frac{\partial f(x)}{\partial x_i} \right)_{i=1,\ldots,n},$$

and the Hessian of f as the matrix of second derivatives

$$\nabla^2 f(x) = \left(\frac{\partial f(x)}{\partial x_i \partial x_j} \right)_{i,j=1,\ldots,n}.$$

For a vector map $\Lambda : \mathbb{R}^n \to \mathbb{R}^m$, we define the Jacobian as

$$\partial \Lambda(x) = \left(\frac{\partial \Lambda_j(x)}{\partial x_i} \right)_{j=1,\ldots,n;\ i=1,\ldots,m}.$$

A function f is called convex on U if $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y)$ for all $x \neq y$ in U and $0 < \lambda < 1$. It is called strictly convex if strict inequality holds. A set $A \subset \mathbb{R}^N$ is called convex if for all $a, b \in A$ and $\lambda \in [0,1]$ $(1-\lambda)a + \lambda b \in A$. For a set A we denote with $\operatorname{conv}(A)$ the convex hull, $\operatorname{aff}(A)$ the affine hull and $\operatorname{ri}(A)$ the relative interior of A.

An image I is associated with a function $I : \Omega \to \mathcal{D}$ which maps values from the image domain Ω into the co-domain $\mathcal{D}$. For discrete two-dimensional images with a height N and width M, the image domain is the set of all pixel positions $\Omega = \{1, \ldots, N\} \times \{1, \ldots, M\}$ and the co-domain $\mathcal{D}$ is the color space, for example $[0,1]$ or $\{1, \ldots, 256\}$ for gray valued images and $[0,1]^d$ or $\{1, \ldots, 256\}^d$ for color images. To avoid confusion, note that we will not use the term *image* as synonym for the co-domain, so *image* always denotes a mapping from the image-domain Ω into some data domain.

CHAPTER 2

GRAPHICAL MODELS : FUNDAMENTALS

2.1. Introduction

Graphical models [106, 26, 34, 86, 106, 124, 173] are commonly used in statistics and many other fields to denote modularity by dividing a complex model into a combination of simpler parts. Probability theory provides a theoretical basis for combining the parts into a consistent model of the complex system and gives a tool to learn a model from given data. The theory of probabilistic graphical models connects the probability theory with graph theory, where the later one provides a intuitively appealing interface into probability theory and data structures which simplify the design of efficient algorithms.

A probabilistic graphical model contains a set of random variables. The conditional independence in the joint distribution is given by a set of variable pairs, represented by a graph. This graph consists of a set of nodes and set of edges. Models which only include directed edges are also called Bayesian Networks or Belief Networks. The direction of the edge implies a causal relation. Thus, Bayesian Networks often model a naturally occurring random process. Contrary, models which only include undirected edges, known as Markov Random Fields (MRF), express symmetric relations. In such cases the joint distribution of the MRF is a Gibbs distribution [59]. We will see many examples for such models later.

Conditional independencies of random variables in a probabilistic graphical model are represented by its graph. Consequently, if one would state that some variables are independent given some others, it is sufficient to carry out some graph based algorithms. The conditional independence intuitively imply a factorization of the distribution. This factorization have not to be equivalent to the finest one of the distribution. It is common to use factor graphs [97, 112] to visualize this "real" factorization. Another main advantage of factor graphs is that they are not restricted to probabilistic graphical models and can be applied straight forward to non probabilistic graphical models [2].

A bridge between graphical models and convex analysis is built by the theory of exponential families [173, 181]. This enables us to project discrete and other graphical models into

the world of convex analysis and thus problems on graphical models can be formulated as convex optimization problems.

This chapter gives an overview of graphical models and provides a dictionary between the different descriptions of those models. We organized the chapter in the following way. In Section 2.2, the basic notations and definitions for graph theory are repeated. Probabilistic graphical models will be introduced in Section 2.3. We will discuss and compare the two most common model types, namely directed and undirected models. In Section 2.4 we introduce the factor graph representation for graphical models and link it with the probabilistic representations. The theory of exponential families and its relation to graphical models is presented in Section 2.5. We end up with the introduction of Conditional Random Fields (CRF) and some further examples visualizing the findings of this chapter.

2.2. Graph Theory

Graph theory is the study of mathematical structures which represent the relation between a set of object. In 1736, Leonhard Euler studied the problem of the "Seven Bridges of Königsberg", which is regarded as the first contribution to graph theory in history. Until now graph theory plays an fundamental roll in many research-fields. This causes a large specialized vocabulary, which is poorly ambiguous; some authors use the same word with different meanings, others use different words to denote the same thing. For clearness in the usage of vocabulary, we will define the required basics in graph theory.

A graph is an ordered pair of a set of nodes and a set which models a relation between them, see Definition 2.1.

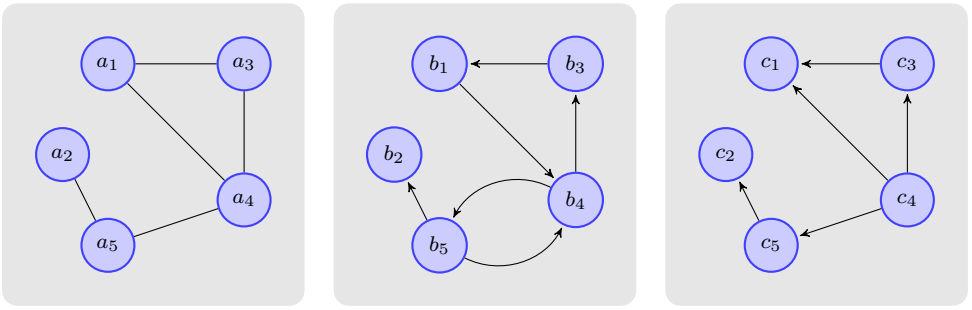
Definition 2.1. ***Graph:*** A graph is an ordered pair $G = (V, E)$ which gives an abstract representation of a set of objects represented as nodes (aka vertices) V together with binary relations between distinct nodes, represented by a set of edges $E \subset \{ab \in V \times V | a \neq b\}$.

We assume the usual case, where relations are represented by a set of pairs of nodes and call them edges. This set of edges is a subset of all pairs of distinct nodes. For hyper-graphs, the edge set is a subset of the power set of V.

For each subset $A \subset V$ we denote with $G_A = (A, E_A = E \cap (A \times A))$ the ***subgraph*** of G according to the node set A. The subgraph G_A includes the nodes A and all edges of G between nodes in A.

Undirected graphs (see Figure 2.1a), contain only undirected edges, which are represented by a set of unordered pairs $\{a, b\}$. The meaning of an undirected edge is that the two nodes associated with the edge have a symmetric relation. In contrast, the edges in a ***directed graph*** (see Figure 2.1b), also called arcs, have a parent-child-relation. A directed edge from a to b is represented by ordered pairs (a, b). We will use ab as a shortcut for the edges (a, b) or $\{a, b\}$. A graph with both directed and undirected edges is called ***mixed graph***. A subset $A \in V$ is called a ***chain component*** of G, if G_A includes only undirected edges.

To sum up; the meaning of an edge is that there is some relation between the nodes. If the edge is directed this relation is not symmetric and has a causal meaning. Some simple examples are shown in Figure 2.1.



(a) Undirected graph (b) Directed Graph (c) Directed Acyclic Graph

Figure 2.1.: Examples for graphs; **(a)** shows an undirected graph G^a. A directed graph G^b is shown in **(b)** and a directed acyclic graph G^c is shown in **(c)**. The formal definition of the two graphs is:
$G^a = (\{a_1, a_2, a_3, a_4, a_5\}, \{\{a_1, a_3\}, \{a_1, a_4\}, \{a_2, a_5\}, \{a_3, a_4\}, \{a_4, a_5\}\})$
$G^b = (\{b_1, b_2, b_3, b_4, b_5\}, \{(b_1, b_4), (b_4, b_3), (b_3, b_1), (b_4, b_5), (b_5, b_4), (b_5, b_2)\})$
$G^c = (\{c_1, c_2, c_3, c_4, c_5\}, \{(c_4, c_1), (c_4, c_3), (c_3, c_1), (c_4, c_5), (c_5, c_2)\})$

The number of edges connected to a node a is called the ***degree*** $deg(a)$. If all nodes have the same degree, the graph is called regular. Nodes with degree 1 are called leafs.

A node a is called ***child*** of a node b if there is a directed edge from b to a. An example is shown in Figure 2.1b: the node b_1 is a child of the node b_3. Alternatively, we can also say that the node b_3 is a ***parent*** of the node b_1. We use $\text{pa}(a)$ for the set of parents of a and $\text{ch}(a)$ for the set of children of a - in Figure 2.1b $\text{ch}(b_4) = \{b_3, b_5\}$ and $\text{pa}(b_4) = \{b_1, b_5\}$. The ***neighbors*** of a node a are given by the set of nodes which are linked with a by an edge, formally $\text{ne}(a) = \{b \in V | ab \in E\}$. For directed graphs the neighbors-set is the union of the parents and children. Furthermore, we call the set of nodes which have at least one child in common with the node a ***co-parents*** of a noted by $\text{cp}(a)$ and the set of all nodes which are a child, child of child, child of child of a child and so forth of a node a descendants of a denoted by $\text{de}(a)$.

A sequence of distinct nodes, in which the sequent nodes are adjacent in G is called a ***path***. In Figure 2.1a the sequence (a_1, a_4, a_5, a_2) is a path from a_1 to a_2 while (a_1, a_5, a_2) is not a path because there is no edge between a_1 and a_5. For a path $p = (a_1, \ldots, a_n)$ we denote with p_i the i-th node in the sequence p. The length of a path is the number of nodes in the sequence minus 1. Two nodes a and b are called ***connected*** if and only if there exists at least one path between a and b. If all edges on the path from node a to nodes b are directed to b we call this a directed path from a to b.

A node a is called a ***descendant*** of b if a directed path form b to a exists. A path from a node to itself which contains each node and edge only once is called a ***cycle***. Graphs containing cycles are called ***cyclic graphs***, graphs without cycles are called ***acyclic graphs***. A further important class are ***directed acyclic graph***, also called a ***DAG***. A DAG is a directed graph with no directed cycles. An example for a DAG is shown in Figure 2.1c. The cycle (c_4, c_3, c_1, c_4) in G^c is not a directed cycle, because the edge between c_1 and c_4 is not directed from c_1 to c_4.

An edge between two nodes of a cycle which itself is not part of the cycle is called a ***chord*** of the cycle. An undirected graph $G = (V, E)$ is called ***chordal*** or ***triangulated*** if every cycle with a length greater than three has a chord. If an undirected graph $G = (V, E)$ is

not chordal, than there exists a graph $G = (V, E')$ with $E \subset E'$ which is chordal. Building a chordal graph G' by adding edges to G is called triangulation. G' is called a minimal triangulation, if removing any edge from G' results in a non-chordal graph.

Many of the given definitions on single nodes can be generalized for a set of nodes $A \subset V$ as shown in Definition 2.2.

Definition 2.2. ***Relations in a Graph:*** For a graph $G = (V, E)$, where E can contain directed and undirected edges and $A \subset V$ and $a \in V$ we define

$$\mathrm{pa}(A) := \left\{b \in V \middle| \exists a \in A : (b, a) \in E, b \notin A\right\} \tag{2.1}$$
$$\mathrm{ch}(A) := \left\{b \in V \middle| \exists a \in A : (a, b) \in E, b \notin A\right\} \tag{2.2}$$
$$\mathrm{de}(A) := \left\{b \in V \middle| \exists (c_1, \ldots, c_n) : c_1 \in A,\ c_n = b\ \forall i = \{1, \ldots, n-1\}(c_i, c_{i+1}) \in E\right\} \tag{2.3}$$
$$\mathrm{ne}(A) := \left\{b \in V \middle| \exists a \in A : \{(b, a), (a, b), \{a, b\}\} \cap E \neq \emptyset, b \notin A\right\} \tag{2.4}$$
$$\mathrm{cp}(A) := \left\{b \in V \middle| \exists a \in A : b \in \mathrm{pa}(\mathrm{ch}(a)), b \notin A\right\} \tag{2.5}$$
$$deg(a) := |\mathrm{ne}(a)| \tag{2.6}$$

A graph in which for each node a path to all other node exists is called a ***connected*** graph. An acyclic undirected graph is called a ***tree*** if it is connected and is called a ***forest*** (union of trees) when it is unconnected. Trees have the property that between two nodes exactly one path exists. An undirected tree with a designated root-node is called a rooted tree, without a root-node it is called a free tree. we associate with each edge in a rooted tree a natural direction away from the root. The length of the path from the root to a node is called the ***depth*** of the node, the maximal depth of all nodes is the ***height*** of the tree.

A ***clique*** C in an undirected graph $G = (V, E)$ is a subset of nodes $C \subset V$ such that there is an edge between each of the nodes, that means $\forall a, b \in C, a \neq b : \{a, b\} \in E$. The clique is called ***maximum-clique*** if there is no further node which enlarges the clique. We denote the set of all maximum-cliques in G by $\mathcal{C}(G)$.

Any directed or mixed graph can be transformed into an undirected graph called ***moral graph***. The name stems the fact that parents should know each other, otherwise that would be unmoral. If in a graph the parents of a child are not connected by an edge we will call this an ***immorality***. To construct a moral graph from an directed graph, each co-parents have to be connected by an edge and then all edges have to be be transformed to undirected edges. A general definition of a moral graph to any mixed graph is given in Definition 2.3.

Definition 2.3. ***Moral Graph:*** A mixed graph $G = (V, E)$ with chain components $K = \{K_1, \ldots, K_n\}$ can be transformed into an undirected one, which is called moral graph $G^m = (V, E^m)$. The moral graph includes all edges from G and additionally edges between co-parents of chain components in G, that is

$$G^m = \left(V, \left\{\{a, b\} \in V \times V \;\middle|\; \begin{array}{l}(a, b) \in E \\ \text{or } (b, a) \in E \\ \text{or } \{a, b\} \in E \\ \text{or } \exists K_i \in K : a, b \in \mathrm{pa}(K_i), a \neq b\end{array}\right\}\right)$$

Another important property of undirected graphs is the separation property; a set of nodes $S \subset V$ separates two distinct nodes $a, b \in V \setminus S$ if at least one node in each path between

a and b is included in S. In other words, in the subgraph $G_{V\setminus S}$ exists no path between a and b. We say that S separates a from b. A more general definition of a separation is given in Definition 2.4. Using this we define a proper decomposition[1] of a graph in Definition 2.5.

Definition 2.4. ***Separation:*** Given an undirected graph $G = (V, E)$ we say that $S \subset V$ separates the disjunct sets $A, B \in V \setminus S$ if in the subgraph $G_{V\setminus S}$ there exists no path between any node in A and any node in B. If S separates A and B in G we denote this by $A \perp\!\!\!\perp_G B|S$.

Definition 2.5. ***Decomposition:*** A triple (A, B, S) of disjunct subsets of V is called a proper decomposition of an undirected graph $G = (V, E)$ if $A \cup B \cup S = V$, S is a clique of G and S separates A and B.

A graph is called decomposable if there exists a sequence of proper decompositions such that all As and Bs are cliques in G. This raises the question under which conditions an undirected graph is decomposable. The answer to this question is given in theorem 4.4 in [34], which shows that an undirected graph $G = (V, E)$ is decomposable if and only if G is triangulated.

Finally, we define for each undirected graph the so called ***Junction Tree***, also known as ***Clique Graph***. There might exist several, one or no junction tree according to an undirected graph G. In [34] theorem 4.6 it is shown that there exists at least one junction tree for graph $G = (V, E)$ if and only if G is decomposable.

Definition 2.6. ***Junction Tree:*** Given an undirected graph $G = (V, E)$ with the set of maximum-cliques $\mathcal{C}(G)$, the graph $T = (\mathcal{C}(G), E_T \subset \mathcal{C}(G) \times \mathcal{C}(G))$ is called Junction Tree if the intersection $C_a \cap C_b$ of each pair $C_a, C_b \in \mathcal{C}(G)$ is contained in every node of the unique path in T between C_a and C_b.

Let us stop this excursion at this point and come back to the fundamentals of graph theory. We will pick up this topic again in Section 4.2.2.

The ***tree-width*** of an undirected graph $\mathrm{tw}(G)$ is the size of the largest clique in G minus 1. For trees the tree-width is 1, for fullly connected graphs, i.e. graphs in which each node is linked to all other nodes, the tree-width is $|V| - 1$.

We can upgrade each graph by additionally assigning a weight to each edge. Such graphs are called ***weighted graphs***. Weighted graphs $G = (V, E, w)$ include a weight function $w : E \to \mathbb{R}$ defining a weight for each edge.

2.3. Probabilistic Graphical Models

A probabilistic graphical model is a pair $(X, G = (V, E))$ of a random vector X and a graph G. The random vector X is indexed by nodes of G, such that each node $a \in V$ is associated with a random variable X_a taking values $x_a \in \mathcal{X}_a$. The edge set E implies conditional independences on X and a factorization of the underlying probability distribution $p(x)$ with respect to some underlying measure.

[1] Contrary to the common literature which names this just 'decomposition', we use 'proper decomposition' to avoid confusion with future use of the term decomposition

Our notation of probability tries to make the split between formal mathematical definitions and a compact and clear notation. We will not differ between the probability distribution $P(X)$ and the density or mass function $p(x)$ and call both probability distribution. Indeed, for discrete variables the probability of an event x denoted by $P(X = x)$ is equivalent to $p(x)$.

If the random variable is discrete, then the sum of the probability of all events have to be 1, $\sum_{x\in\mathcal{X}} p(x) = 1$, and the expected value of a function $f(x)$ is given by $\mathbb{E}_p[f] := \sum_{x\in\mathcal{X}} p(x)f(x)$. For continuous random variables we have, $\int_{\mathcal{X}} p(x)dx = 1$ and the expected value of a function $f(x)$ is given by $\mathbb{E}_p[f] := \int_{\mathcal{X}} p(x)f(x)dx$. Given a joint distribution $p(x, y)$ the marginal distribution is given by $p(x) = \sum_{y\in\mathcal{Y}} p(x, y)$ and $p(x) = \int_{\mathcal{Y}} p(x, y)dy$, respectively.

Given the random vector X_B the conditional probability for a random vector X_A is denoted by $p(x_A|x_B)$. Its relation to the joint probability $p(x_A, x_B)$ is given by the ***Bayes Theorem*** [8]

$$p(x_A|x_B) = \frac{p(x_A, x_B)}{p(x_B)} = \frac{p(x_B|x_A)\, p(x_A)}{p(x_B)} \tag{2.7}$$

where $p(x_A)$ is the ***marginal distribution*** of $p(x_A, x_B)$ defined as

$$p(x_A) = \sum_{x_B\in\mathcal{X}_B} p(x_A, x_B) \tag{2.8}$$

in the discrete and

$$p(x_A) = \int_{\mathcal{X}_B} p(x_A, x_B)dx_B \tag{2.9}$$

in the continuous case.

Furthermore, we denote the expectation of a random variable X by $\mathbb{E}(X)$, the variance by $\mathrm{Var}(X)$, the covariance matrix by $\mathrm{Cov}(X)$ and the entropy by $H(X)$ (definitions given in Definition A.5, Definition A.6, Definition A.7 and Definition A.8).

Two random vectors X_A and X_B with a probability distribution $p(x_A, x_B)$ are called independent ($X_A \perp\!\!\!\perp X_B$) if and only if the joint distribution splits up into the product of their marginal distributions.

$$X_A \perp\!\!\!\perp X_B \Leftrightarrow \forall x_A \in \mathcal{X}_A, x_B \in \mathcal{X}_B : p(x_A, x_B) = p(x_A)\, p(x_B) \tag{2.10}$$

Accordingly, two random vectors X_A and X_B are called conditional independent given X_S if and only if they factorize with respect to X_S

$$X_A \perp\!\!\!\perp X_B|X_S \Leftrightarrow \forall x_A \in \mathcal{X}_A, x_B \in \mathcal{X}_B, x_S \in \mathcal{X}_S : p(x_A, x_B|x_S) = p(x_A|x_S)\, p(x_B|x_S) \tag{2.11}$$

As stated in [34], the ternary relation $X \perp\!\!\!\perp Y|Z$ has the following properties, where h denotes an arbitrary measurable function on $\mathcal{X}$:

(C1) if $X \perp\!\!\!\perp Y|Z$ then $Y \perp\!\!\!\perp X|Z$

(C2) if $X \perp\!\!\!\perp Y|Z$ and $U = h(X)$ then $U \perp\!\!\!\perp Y|Z$

(C3) if $X \perp\!\!\!\perp Y|Z$ and $U = h(X)$ then $X \perp\!\!\!\perp Y|(Z, U)$

(C4) if $X \perp\!\!\!\perp Y|Z$ and $X \perp\!\!\!\perp W|(Y,Z)$ then $X \perp\!\!\!\perp (Y,W)|Z$

A fifth property of conditional independence, does not hold in general, but under additional conditions, see [107] for a more detailed discussion.

(C5) if $X \perp\!\!\!\perp Y|Z$ and $X \perp\!\!\!\perp Z|Y$ then $X \perp\!\!\!\perp (Y,Z)$

If two random variables X and Y are not independent, we call them dependent and write $X \not\perp\!\!\!\perp Y$.

A graph can be understood as a description or map of the conditional independences of random variables. We wrote $A \perp\!\!\!\perp_G B|S$ if $S \subset V$ separates the disjunct subsets $A, B \in V \setminus S$ in the graph $G = (V, E)$. For undirected graphs this separation property has been defined in Definition 2.4, for directed graphs the d-separation property was introduced by Pearl [123] and is discussed in Section 2.3.1.

Definition 2.7. Let G be a graph and X a random vector with the joint distribution $p(x)$,

D-map then G is said to be a D-map (dependence map) of $p(x)$ if every conditional independence statement of X satisfied by $p(x)$ is represented in the G.

$$X_A \perp\!\!\!\perp X_B|X_S \Rightarrow A \perp\!\!\!\perp_G B|S$$

I-map if every conditional independence statement implied by G is satisfied by $p(x)$, then G is said to be an I-map (independence map) of $p(x)$.

$$A \perp\!\!\!\perp_G B|S \Rightarrow X_A \perp\!\!\!\perp X_B|X_S$$

P-map if all conditional independence properties of X are reflected in G, and vice versa, then G is said to be a P-map (perfect map) for $p(x)$. A perfect map is therefore both an I-map and a D-map.

$$A \perp\!\!\!\perp_G B|S \Leftrightarrow X_A \perp\!\!\!\perp X_B|X_S$$

A completely disconnected graph $G = (V, \emptyset)$ is a trivial D-map and a fully connected undirected graph $G = (V, V \times V)$ is a trivial I-map for any probability distribution $p(x)$ of a random vector X_V. The ***Markov blanket*** of a node $a \subset V$ is the minimal set of nodes $S \in V \setminus \{a\}$ which separates the node a from the rest of the nodes.

The general definition of a probabilistic graphical model, given in Definition 2.8, holds for directed, undirected and chain graphs models. The relation between a graph and the factorization and independence of a graphical model will be discussed for the different graph types in the following subsections.

Definition 2.8. ***Probabilistic Graphical Model*** A probabilistic graphical model is a pair (X, G) of a random vector X and a graph G. The graph G is an I-map of $p(x)$ and implies

i) a factorization of the probability distribution $p(x)$ into sub-sets $\mathcal{S} \subset \mathcal{P}(V)$

$$p(x) = \prod_{S \in \mathcal{S}} f_S(x_S), \tag{2.12}$$

and

ii) conditional independences between random variables, this is known as Markov Properties.

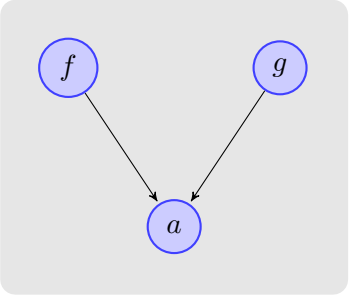


Figure 2.2.: Bellyache (a) can be caused by flatulence (f) or gastritis (g). This causal relation can be expressed by a directed graph.

Furthermore, we define ***proper*** and ***strict*** probabilistic graphical model.

Definition 2.9. ***Proper and Strict Models***

i) A graphical model $M = (X, G)$ is said to be proper if G is a P-map of $p(x)$.

ii) We call a graphical model ***strict*** if each term $f_S(x_S)$ of the factorization according to G, cannot be factorized any more. In other words

$$\forall f_S \ \nexists A, B \subsetneq S : f_S(x_S) = f_B(x_B) f_A(x_A)$$

2.3.1. Directed Graphical Models: Bayesian Networks

Many natural models include a natural factorization of the distribution, which is motivated by the Bayes Theorem. Each factor of the Bayes factorization represents a causal relation within the model. Directed graphical models express this causal relationship between random variables by a directed acyclic graph.

Definition 2.10. ***Directed Graphical Model*** A directed graphical model is a pair (X, G) of a random vector X and a directed acyclic graph $G = (V, E)$. The joint probability distribution $p(x)$ of X can be factorized into a product of its local conditioned distributions of the form

$$p(x) = \prod_{a \in V} p(x_a | x_{\mathrm{pa}(a)}).$$

Figure 2.2 shows a simple examples for such a causal relation. Bellyache is mainly caused by flatulence or gastritis. We can express this in a probabilistic form

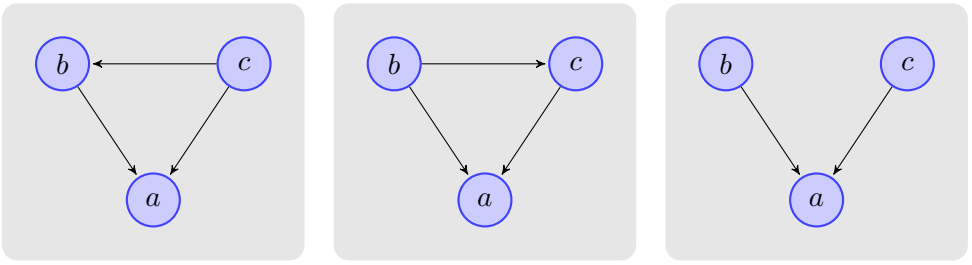
$$p(x_a, x_f, x_g) = p(x_a | x_f, x_g)\ p(x_f)\ p(x_g)$$

The random variable X_a models if a person has bellyache or not, the random variables X_f and X_g model if flatulence and gastritis is present respectively.

The factorization into a product of local functions $f_C(x_C)$ of the probability distribution of any directed model is given by

$$p(x) = \prod_{a \in V} p(x_a | x_{\mathrm{pa}(a)}) \tag{2.13}$$

$$= \prod_{a \in V} f_{\{a\} \cup \mathrm{pa}(a)}(x_{\{a\} \cup \mathrm{pa}(a)}). \tag{2.14}$$



(a) (b) (c)

Figure 2.3.: Examples for factorizations of $p(x_a, x_b, x_c)$. Since the graphs **(a)** and **(b)** includes no independent assumptions, graph **(c)** models the statistical independence of X_a and X_b if X_c is not given. The factorizations are **(a)** $p(x_a|x_b, x_c)\, p(x_b|x_c)\, p(x_c)$ **(b)** $p(x_a|x_b, x_c)\, p(x_c|x_b)\, p(x_b)$ **(c)** $p(x_a|x_b, x_c)\, p(x_b)\, p(x_c)$

Figure 2.3 shows some examples for directed graphs and the according factorization of the joint distribution.

Since the factorization of the distribution of a directed model follows directly from Definition 2.10, the separation property of undirected graphs which would imply conditional independence will be defined next. For a connected acyclic directed graph with three nodes there are three possibilities of edge-settings as shown in Figure 2.4. We will analyze the statistical relationship of the random variables X_b and X_c both when X_a is observed or not. If the random variable X_a is observed, i.e. we know the fix state of X_a, we condition the distribution on X_a and test for conditional independence of X_b and X_c with respect to $p(x_b, x_c|x_a)$ instead of $p(x_b, x_c, x_a)$ in the unobserved case. This six cases motivate and illustrate the key concepts of the directed separation (d-separation) property.

The first of the three examples is shown if Figure 2.4a and is called ***Head-To-Tail***. We say that node a is head-to-tail with respect to the directed path from b to c, because there is a directed path from b over a to c. The joint distribution corresponding to this graph factorize to

$$p(x_a, x_b, x_c) = p(x_c|x_a)p(x_a|x_b)p(x_b)$$

If none of the variables is observed, we can check whether X_b and X_c are independent by marginalizing over X_a. We get

$$p(x_b, x_c) = p(x_b) \sum_{x_a \in \mathcal{X}_a} p(x_c|x_a)p(x_a|x_b) = p(x_b)p(x_c|x_b)$$

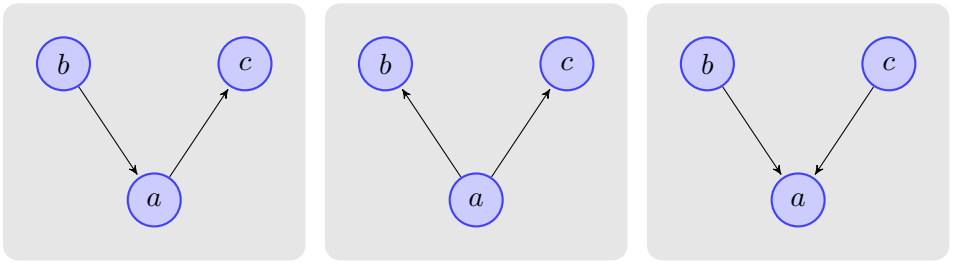
which in general does not factorize and so $X_b \not\perp\!\!\!\perp X_c|\emptyset$. Here the empty set $\emptyset$ denotes that no random variable was observed.

If we conditioned the distribution by fixing x_a, we can rewrite the conditional distribution

$$p(x_b, x_c|x_a) = \frac{p(x_c|x_a)p(x_a|x_b)p(x_b)}{p(p_a)} = p(x_c|x_a)p(x_b|x_a)$$

and obtain the conditional independence property $X_b \perp\!\!\!\perp X_c|X_a$.

In the presence of a head-to-tail path from b to c X_b and X_c are dependent. Knowing the value of the random variable to the head-to-tail node a blocks the path between b and c and implies conditional independence of X_b and X_c given X_a.



(a) (b) (c)

Figure 2.4.: The graph above show the tree types of dependencies which can be occur in an acyclic connected directed graph with three nodes. The three types are **(a)** called head-to-tail, **(b)** tail-to-tail and **(c)** head-to-head.

The second scenario is called ***Tail-To-Tail*** and shown in Figure 2.4b. The joint distribution associated with this graph is given by

$$p(x_a, x_b, x_c) = p(x_b|x_a)p(x_c|x_a)p(x_a).$$

Again let us supposed that none of the variables are observed and test for independence of X_b and X_c by marginalization over X_a.

$$p(x_b, x_c) = \sum_{x_a \in \mathcal{X}_a} p(x_c|x_a)p(x_b|x_a)p(x_a).$$

Since this in general does not factorize to $p(x_b)p(x_c)$ we obtain $X_b \not\perp\!\!\!\perp X_c|\emptyset$.

If we conditioned the distribution to a fixed x_a, it can be rewrite in the form

$$p(x_b, x_c|x_a) = p(x_c|x_a)p(x_b|x_a)$$

and again we obtain the conditional independence property $X_b \perp\!\!\!\perp X_c|X_a$.

As before, we can also give a graphical interpretation for this result. The node a is said to be tail-to-tail with respect to the path from b over a to c, because the path from b to c connects the two tails. The presence of such a path causes that X_b and X_c are not independent. However, when the state of X_a is observed, a blocks this path and X_b and X_c become conditional independent given X_a.

Finally, we consider the graph shown in Figure 2.4c, called ***Head-To-Head***. In contrast to the other two graphs this has a subtle definition of independence. The distribution to this graph can be factorized as

$$p(x_a, x_b, x_c) = p(x_a|x_b, x_c)p(x_b)p(x_c).$$

Again we first consider the unobserved case and marginalize over X_a and obtain

$$p(x_b, x_c) = p(x_b)p(x_c)$$

what causes X_b and X_c to be independent if no variable is observed, that is $X_b \perp\!\!\!\perp X_c|\emptyset$.

If we condition this model on X_a, the conditional distribution is

$$p(x_b, x_c|x_a) = \frac{p(x_a|x_b, x_c)p(x_b)p(x_c)}{p(x_a)}$$

which in general does not factorize into $p(x_b)p(x_c)$, and so $X_b \not\perp\!\!\!\perp X_c|X_a$.

Graphically, we say an node a is head-to-head with respect to the path from b to c, because it connects the heads of the arrows. If X_a is observed it 'unblocks the path between b and c. An unobserved X_a 'blocks' the path from b to c and makes X_b and X_c independent.

This third example has an opposite behavior compared to the first two and may be a little bit confusing since an unobserved path causes independence, while an observation destroy this independence. The bellyache example in Figure 2.2 is also an head-to-head scenario. One would not expect that there is a relation between flatulence and gastritis, but both cause bellyache. If we observe that someone has bellyache this has to be caused by something, most probably by flatulence or gastritis, but also other minor likely reasons are possible. That both appear at the same time is fortuity. This makes the random variables dependent if bellyache is observed, because if our patient has flatulence this would decrease the probability for gastritis.

However, for the definition of separation this is still a bit unusual and becomes much more complicated since even the observation of a descendant of a will 'unblock' the path from b to c and makes X_b and X_c independent. Assuming that $\mathrm{de}(a) = \{a_1, \ldots, a_n\}$ is the set of descendants of a such that the joint distributions decompose into

$$p(x) = p(x_{\mathrm{de}(a)}|x_a)p(x_a|x_b, x_c)p(x_b)p(x_c).$$

This distribution can be reformulated as

$$\begin{aligned} p(x) &= p(x_{\mathrm{de}(a)}|x_a)p(x_a|x_b, x_c)p(x_b)p(x_c) \\ &= p(x_{\mathrm{de}(a)}, x_a|x_b, x_c)p(x_b)p(x_c) \\ &= p(x_{\mathrm{de}(a)\setminus\{d\}}, x_a|x_d)p(x_d|x_b, x_c)p(x_b)p(x_c). \end{aligned}$$

If we fix an arbitrary descendant $d \in \mathrm{de}(a)$ of a, then the conditional distribution of x_b and x_c is

$$p(x_b, x_c|x_d) = \sum_{x_a, x_{\mathrm{de}(a)\setminus\{d\}}} \frac{p(x)}{p(x_d)} = \frac{p(x_d|x_b, x_c)p(x_b)p(x_c)}{p(x_d)}$$

which in general does not factorize into $p(x_b)p(x_c)$.

A more general statement about conditional independence in directed acyclic graphs is given by the ***d-separation property*** [123]. Let us suppose that $A, B, S \subset V$ are three disjunct sets in the graph G and we wish determine whether $X_A \perp\!\!\!\perp X_B|X_S$ holds.

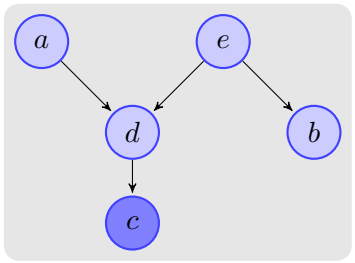
Theorem 2.1. *In a directed graph $G = (V, E)$ the set $S \subset V$ separates the disjunct subsets $A, B \in V \setminus S$, denoted by $A \perp\!\!\!\perp_G B|S$, if S blocks all paths between A and B. A path between a node of A and a node of B is said to be blocked if it includes a node s such that either*

i) $s \in S$ and s is head-to-tail or tail-to-tail node on this path, or

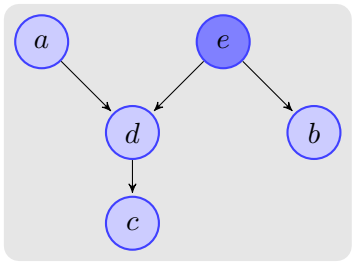
ii) $(\{s\} \cup de(s)) \cap S = \emptyset$ and s is head-to-head node on this path.

Proof. The proof builds on the 6 cases discussed for head-to-head, head-to-tail and tail-to-tail scenarios and can be found in [123]. □

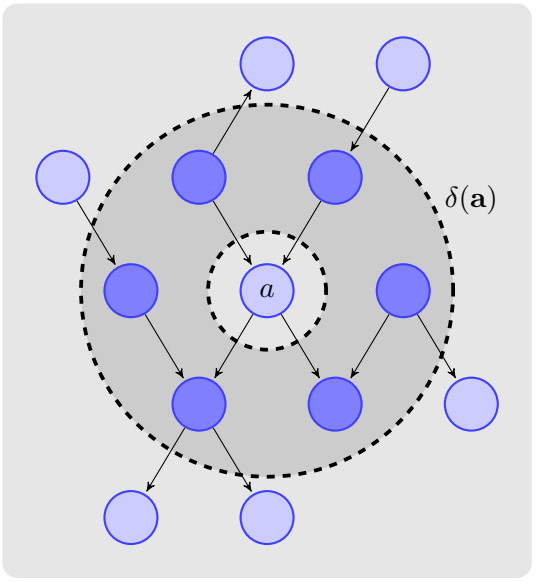
The Theorem 2.1 enables us to identify conditional independences of a distribution $p(x)$ just by inspecting the corresponding directed graph G, without any statistical calculations. For the bellyache example the graph in Figure 2.2 indicates that flatulence and gastritis



(a) $X_a \not\perp\!\!\!\perp X_b | X_c$



(b) $X_a \perp\!\!\!\perp X_b | X_e$



(c) $X_a \perp\!\!\!\perp X_{V \setminus (\delta(a) \cup \{a\})} | X_{\delta(a)}$

Figure 2.5.: The graphs (a) and (b) show the dependence of X_a and X_b given a observation marked with the dark colored nodes. For the independence of X_a and X_b we must observe X_e or neither observe X_c nor X_d. The graph (c) illustrate the Markov blank of X_a in a simple toy graph.

are dependent if we observe bellyache, because the bellyache node is a head-to-head node with respect to the path from the flatulence node to the gastritis node. A more complex example is shown in Figure 2.5a, where X_a and X_b are dependent given $X_c = x_c$ because the path from a to b is neither blocked by d nor by e. The observation of X_c, marked by darker color, unblocks the path in this scenario. Conditional independence is shown in Figure 2.5b. Here the path between a and b is blocked twice, once in d and once in e. An efficient algorithm for calculating the set conditional independent variables in a directed modes is given in [86] (Algorithm 3.1).

Theorem 2.1 can also be used to specify the Markov blanket for directed graphs. Figure 2.5c shows the Markov blanket of node a in a toy graph. In this example observing $x_{\delta(a)}$ enforce that X_a is independent from $X_{V \setminus (\delta(A) \cup \{a\})}$.

Theorem 2.2. *Given a directed model $(X, G = (V, E)$, then the Markov blanket of $a \in V$ is given by*

$$\delta(a) = pa(a) \cup ch(a) \cup cp(a).$$

Proof. This follows directly from Lemma 5.9 in [34]. □

2.3.2. Undirected Graphical Models: Markov Random Fields

The second major class of graphical models use an undirected graph-structure. Such models are called Markov random field (MRF), Markov networks or undirected graphical models. Just as for directed models we can specify rules for factorization and conditional independence. In contrast to directed models, directions of relations can not be modeled, since all edged are undirected. The pros and cons of this fact will become obvious in

the following. The separation property of undirected graphs is much simpler than the d-separation property and has been given in Definition 2.4. This results in the following definition for an undirected model.

Definition 2.11. An undirected graphical model is a pair $(X, G = (V, E))$ such that if two disjunct subsets $A, B \in V \setminus S$ are not connected in $G_{V \setminus S}$ then $X_A \perp\!\!\!\perp X_B | X_S$.

Testing for conditional independence in a undirected model is much simpler than in a directed model because undirected models have no directed edges and consequently no head-to-head nodes. The correspondence between the random variables and the edge set of the graph are often defined by the Markov properties of a random vector. For each undirected model $M = (X, G)$ the separation property of G implies conditional independence on X, such that the random vector X has the Markov property with respect to G by definition.

Definition 2.12. ***Markov Properties*** Given a undirected graph $G = (V, E)$ and a random vector X indexed by V, than we say that X has ...

- **(G)** the global Markov property with respect to G, if for each triple (A, B, S) of disjunct subsets of V such that S separates A and B in G the conditional independence $X_A \perp\!\!\!\perp X_B | X_S$ holds.
- **(L)** the local Markov property with respect to G, if for all nodes $a \in V$ the conditional independence $X_a \perp\!\!\!\perp X_{V \setminus (\mathrm{ne}(a) \cup \{a\})} | X_{\mathrm{ne}(a)}$ holds.
- **(P)** the pairwise Markov property with respect to G, if for all pairs of non-adjacent nodes a and b, the conditional independence $X_a \perp\!\!\!\perp X_b | X_{V \setminus \{a,b\}}$ holds.

While it is quite easy to show that (G) implies (L) and (P) (see Theorem 2.3) the reverse direction that shows that (P) implies (G) and consequently the three definitions of Markov property are equivalent, does not hold in general. A counter example is given in Figure 4.7 in [86].

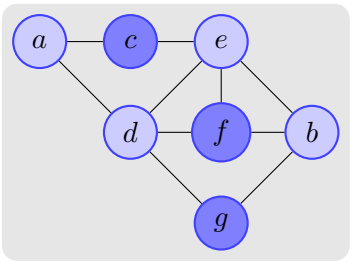
Theorem 2.3. *For an arbitrary random vector the global Markov property implies the local Markov property and the local Markov property implies the pairwise Markov property, i.e.* $(G) \Rightarrow (L) \Rightarrow (P)$

Proof.
$(G) \Rightarrow (L)$: Since $\mathrm{ne}(\{i\})$ separates $\{i\}$ and $V \setminus (\mathrm{ne}(i) \cup \{i\})$, (G) implies (L).
$(L) \Rightarrow (P)$: For non-adjacent nodes $i, j \in V$ we have $j \in V \setminus (\mathrm{ne}(i) \cup \{i\})$. □

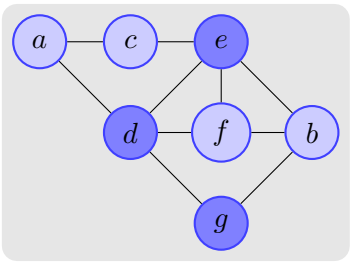
For the step from (P) to (G) we require the property (C5) for the random vector X. A sufficient assumption to make sure that (C5) holds is the strict positivity of the distribution $p(x)$, that means $\forall x \in \mathcal{X} : p(x) > 0$. For strict positive distributions (P) implies (G) and consequently all of this three definitions of a Markov Property are equivalent, i.e. $(G) \Leftrightarrow (L) \Leftrightarrow (P)$

Theorem 2.4. *For a random vector with a strict positive distribution $p(x)$, the pairwise Markov property implies the global Markov property, i.e.* $(P) \Rightarrow (G)$

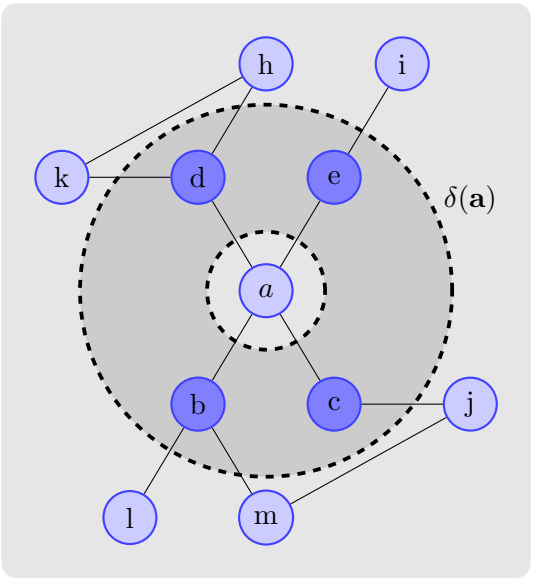
Proof. This theorem was originally formulated by Pearl and Paz [125], the proof is also given in theorem 5.5 in [34]. □



(a) $X_a \not\perp\!\!\!\perp X_b | X_{\{c,f,g\}}$



(b) $X_a \perp\!\!\!\perp X_b | X_{\{e,d,g\}}$



(c) $X_a \perp\!\!\!\perp X_{V \setminus (\delta(a) \cup \{a\})} | X_{\delta(a)}$

Figure 2.6.: The graphs **(a)** and **(b)** show the dependence of X_a and X_b given an observation marked by the dark colored node. While in **(a)** both random variables are dependent because the path (a, d, e, b) does not include an observed node in **(b)** all paths from a to b pass an observed node and so we know that they are conditional independent. The graph **(c)** illustrate the Markov blanket of X_a in a simple undirected toy graph.

From a practical point of view, the restriction that $p(x)$ is strictly positive is not a big problem. Even if an events is theoretical impossible or just impossible because it has not appear so far, we can fix this by assuming that the event has an infinitesimal small probability. On the other hand the strict positivity is a sufficient but not necessary criteria for $(P) \Rightarrow (G)$ and we will derive another criteria later on.

If $(P) \Rightarrow (G)$ holds the local, pairwise and global Markov properties can be used alternatively to construct a undirected model for some random vector or test for conditional independence. Examples for conditional dependence and conditional independence are shown in Figure 2.6a and Figure 2.6b.

The Markov blanket of a set $A \in V$ is simply the set of neighbors of A. This follows directly from the local Markov property. An example for the Markov blanket of an undirected graph is given in Figure 2.6c.

Like for directed models, there is a factorization rule which defines a factorization of the probability distribution $p(x)$ according to the undirected graph structure.

Definition 2.13. For an undirected graph $G = (V, E)$ a probability distribution $p(x)$ of a random vector X is said to factorize according to G if for each complete subset $C \subset \mathcal{C}(G)$ there exists a non-negative function ψ_C which depends on x through x_C only, such that X has the probability distribution $p(x)$ of the form:

$$p(x) = \prod_C \psi_C(x)$$

We say that a random vector X has ...

(F) the factorization property with respect to G if $p(x)$ factorizes according to G.

A further consequence of the definition of the factorization property is that it implies the global Markov property (see Theorem 2.5) such that we have in general the implication

$$(F) \Rightarrow (G) \Rightarrow (L) \Rightarrow (P).$$

Theorem 2.5. *If a random vector X has the factorization property with respect to an undirected graph G, then X has the global Markov property with respect to G, i.e. $(F) \Rightarrow (G)$*

Proof. Let A, B, S be three arbitrary disjoint subsets of V such that S separates A and B. Let $\tilde{A} \subset V \setminus S$ be the set of nodes which contains A and nodes which are in $G_{V \setminus S}$ connected to a node in A and let be $\tilde{B} = V \setminus (\tilde{A} \cup S)$. Since S separates A and B, any clique in G is either a subset of $\tilde{A} \cup S$ or $\tilde{B} \cup S$. Using the factorization property (F) we obtain

$$f(x) = \prod_{C \in \mathcal{C}} f_C(x_C) = \prod_{C \in \mathcal{C}_{\tilde{A} \cup S}} f_C(x_C) \prod_{C \in \mathcal{C}_{\tilde{B} \cup S}} f_C(x_C) = h(x_{\tilde{A} \cup S}) g(x_{\tilde{B} \cup S})$$

which implies $X_{\tilde{A}} \perp\!\!\!\perp X_{\tilde{B}} | X_S$. Since $A \subset \tilde{A}$ and $B \subset \tilde{B}$ it follows from (C1) and (C2) that $X_A \perp\!\!\!\perp X_B | X_S$ holds for any triple (A, B, S) of disjunct subsets of V for which S separates A and B in G, which is exactly the definition of the global Markov property. □

The reverse implication from the Markov properties to the factorization property does not hold in general. Again, this requires the property (C5) for the random vector X, which can be guaranteed by assuming a strict positive distribution. This result is usually attributed to Hammersley and Clifford [68] and known as the Hammersley-Clifford-Theorem.

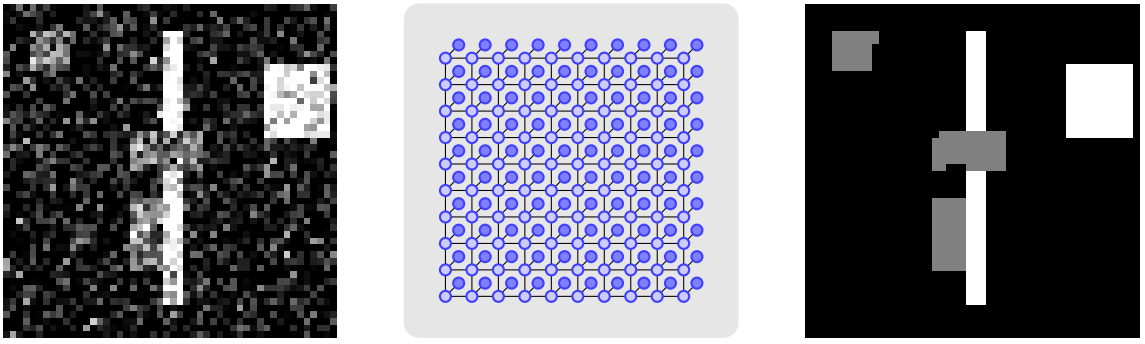
Theorem 2.6. Hammersley-Clifford-Theorem *A random vector with a strict positive distribution $p(x)$ satisfies the pairwise Markov property with respect to an undirected graph G if and only if it has the factorization property with respect to G, i.e. $(P) \Leftrightarrow (F)$.*

Proof. See theorem 3.9 in [107]. □

As already mentioned, the assumption that $p(x)$ is strict positive is sufficient but not necessary for the implication from (L) to (F) or from (L) to (G). An alternative sufficient assumption is that the model graph is chordal. The implication from (L) to (F) follows by the equivalence to a directed model, which we will discuss in the next section.

If neither $p(x)$ is strict positive nor G is chordal, we can add edges to G and get a chordal graph G'. If G is an I-map for X, G' is also an I-map for X because G' implies a subset of conditional independences that implied by G. Consequently we can apply the Hammersley-Clifford-Theorem with respect to G' and get a factorization of $p(x)$.

Example: Particle System The Hammersley-Clifford-Theorem brings us back to one of the earliest origins of graphical models. In statistic physics, Gibbs [59] models the behavior of large particle systems. Each particle can take different states and the total energy of the system $J(x)$ is composed by an external energy plus a energy influenced by the interaction of nearby particles or particle clouds. With each particle we associate a



(a) Noisy Image (b) Graph (c) Reconstruction

Figure 2.7.: To reconstruct the original image from a noisy image **(a)** it is common to use a graphical model **(b)** which includes further information of the statistics of the original image. We can use this model to infer the most probable original image **(c)**.

random variable and the state space of the variable is the set of states the particle can have. Two pixel respectively nodes are neighbors if their geometric distance is below some threshold. The relation of the particles is modeled by an undirected graph. If we assume that the distribution is strict positive or the energy of the system is bounded, then the distribution $p(x)$ and the total energy $J(x)$ of such a system determined its behavior by the so called Gibbs distribution or Gibbs energy.

$$p(x) = \frac{1}{Z} \exp\left(-\frac{1}{T} J(x)\right), \quad J(x) = \sum_{C \in \mathcal{C}(G)} f_C(x_C), \quad Z = \sum_{x \in \mathcal{X}} \exp\left(-\frac{1}{T} J(x)\right) \tag{2.15}$$

The parameter T is the temperature and f_C are the internal and external energy functions. The temperature controls the freedom of action of the particles. For higher temperatures the probability for settings with a higher energy increases while for low temperatures the mass gathers around the setting with the minimal total energy. This example also shows the strong relation between energy-based and statistical models/approaches.

Example: Image Denoiseing A typical example from the field of computer vision is the image denoiseing problem. For a given noisy image $I \in \Omega^{N \times M}$ we search the most probable image $\tilde{I} \in \tilde{\Omega}^{N \times M}$. In our simplified toy example (see Figure 2.7) for each pixel a in the image $\tilde{I}$ we define a random variable X_a taking values in $\{0, 0.5, 1\}$ and for each pixel a in I a random variable Y_a taking values in $[0, 1]$. With I_a and $\tilde{I}_a$ we denote the gray value of the pixel a. In the undirected graph $G = (V, E)$ each node corresponds to a pixel and edges exist between adjacent pixels (here we use a neighborhood of four pixels). Figure 2.7b sketches the graph structure of the undirected model $M = (X, G)$.

The joint distribution of the random vectors X and Y can be reformulated into

$$p(x, y) = p(y|x)\, p(x). \tag{2.16}$$

We assume that

$$p(y|x) \propto \exp\left(-\sum_{a \in V} |y_a - x_a|\right) \tag{2.17}$$

and

$$p(x) \propto \exp\left(-\lambda \sum_{ab \in E} |x_a - x_b|\right), \tag{2.18}$$

such that maximizing $p(x, y)$ and minimizing the energy $J(x, y)$ given by

$$p(x, y) = \frac{1}{Z} \exp(-J(x, y)) \tag{2.19}$$

$$J(x, y) = \sum_{a \in V} |x_a - y_a| + \lambda \sum_{ab \in E} |x_a - x_b| \tag{2.20}$$

are equivalent problems. Both force $\tilde{I}$, represented by x, to be pixel-wise similar to I, represented by y, and regularize $\tilde{I}$ to be smooth. The parameter λ controls whether similar or smooth solutions should be preferred.

Given a noisy image I as shown in Figure 2.7a, we can use the undirected graphical model to reconstruct the most probable image $\tilde{I}$. The result of this simple reconstruction of our toy example is shown in Figure 2.7c.

The definitions for conditional independence in undirected models are much simpler than those for directed ones, which rises several questions. Can we in general use undirected graphs instead of directed ones? Can directed models be transformed into undirected ones? And finally, does such a transformation cause a loss of information in terms of conditional independence?

2.3.3. Relations Between Undirected and Directed Models

In order to compare different graphical models we define some relations between them.

Definition 2.14. Given two graphs $G = (V, E)$ and $G' = (V, E')$ with the same set of nodes V. We say that G covers G' in terms of conditional independence, denoted by $G \geq_{\perp\!\!\!\perp} G'$, if all independence properties implied by G are also implied by G'. Formally this is

$$G \geq_{\perp\!\!\!\perp} G' \Leftrightarrow \big((A \perp\!\!\!\perp_G B|S) \Rightarrow (A \perp\!\!\!\perp_{G'} B|S)\big)$$

If $G \geq_{\perp\!\!\!\perp} G'$ holds, G is an I-map of every random vector for which G' is an I-map. As a result of this definition each fully connected graph, undirected or directed, covers all graphs with the same set of nodes. For undirected graphs we have again a simple graphical illustration, since an undirected graph $G = (V, E)$ covers another undirected graph $G' = (V, E')$ if and only if $E \subset E'$. Such a simple illustration does not exist for directed models since head-to-head nodes can introduce additional independence constraints.

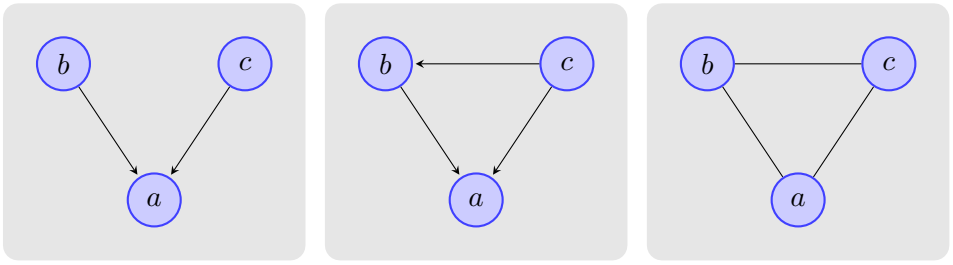
Following the definition of $\geq_{\perp\!\!\!\perp}$ we can define a equivalence-relation for graphs.

Definition 2.15. Given two graphs $G = (V, E)$ and $G' = (V, E')$ with the same set of nodes V. We say that G is equal to G' in terms of conditional independence, denoted by $G =_{\perp\!\!\!\perp} G'$, if and only if they cover each other. Formally this is

$$G =_{\perp\!\!\!\perp} G' \Leftrightarrow \big(G \geq_{\perp\!\!\!\perp} G' \wedge G' \geq_{\perp\!\!\!\perp} G\big)$$

If $G =_{\perp\!\!\!\perp} G'$ holds, then G is an I-map of X if and only if G' is an I-map of X. A simple example for two equivalent graphs in term of conditional independence is $G = (V = \{1, 2\}, E = \{\{1, 2\}\})$ and $G = (V = \{1, 2\}, E' = \{(1, 2)\})$.

Next, we will investigate if we can always transform directed to equivalent undirected models and vice versa.



(a) (b) (c)

Figure 2.8.: The directed graphs **(a)** and **(b)** have the same moral graph shown in **(c)**. With a undirected graph it is not possible to picture that the random variables X_b and X_c are dependent given X_c and otherwise independent, as shown in **(a)**.

Transforming Directed to Undirected Graphical Models

Undirected graphs have a less complex definition of the separation property than directed graphs which makes testing for conditional independence in undirected models easier than in directed ones. For a directed graphical model $M = (X, G = (V, E))$ the probability distribution factorizes into

$$p(x) = \sum_{a \in V} p(x_a | x_{\mathrm{pa}(a)}). \tag{2.21}$$

With $f_{a \cup \mathrm{pa}(a)}(x) = p(x_a | x_{\mathrm{pa}(a)})$ this can be written as

$$p(x) = \frac{1}{1} \sum_{a \in V} f_{a \cup \mathrm{pa}(a)}(x). \tag{2.22}$$

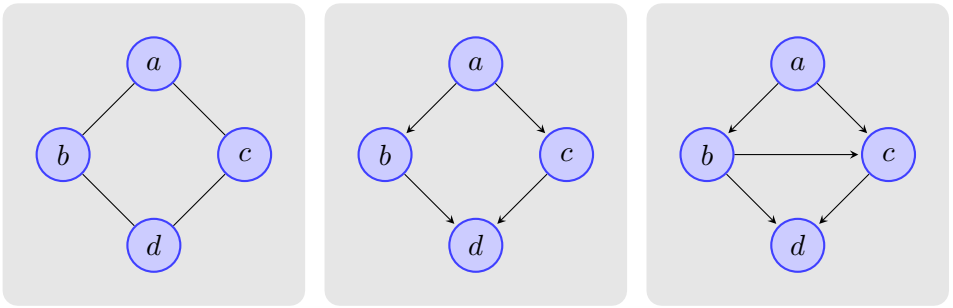
The factorization in (2.22) can be used to define a undirected model $M' = (X, G' = (V, E'))$, for the random variable X of the distribution in (2.21). The undirected graph G' has the same nodes as the directed graph and satisfies that for all $a \in V$ the set $a \cup \mathrm{pa}(a)$ is a clique in G'. The random vector X has by construction the factorization property and by Theorem 2.5 the Markov property with respect to the undirected graph G'.

To make sure that $\{a\} \cup \mathrm{pa}(a)$ is a clique, edges between the parents have to been added. The undirected graph G' is the moral graph of G denoted by G^m, see Definition 2.3.

By construction G^m is an I-map of $p(x)$ if G is an I-map of $p(x)$ as well, because G^m implies no conditional independences which is not implied by G. By the moralization of the graph the conditional independences, implied by head-to-head structures, are removed and not further implied by G^m. So the directed and undirected graph, achieved by moralization, are in general not equal in terms of conditional independence. More precisely, the directed graph G is covered by its moral graph G^m, i.e.

$$G^m \geq_{\perp} G$$

The directed model in Figure 2.8(a-b) are transformed into the same undirected model shown in (c). The conditional independence forced by the head-to-head node can not be expressed by an undirected model. The model in Figure 2.8c assumes more dependencies



(a) (b) (c)

Figure 2.9.: For the undirected graphical model shown in **(a)** there exists no undirected model, which can express the same terms of conditional dependencies for the same variables. The corresponding directed model always includes a immorality **(b)** which has to be fixed by an additional edge **(c)** to ensure that the directed graph is an I-map of the distribution.

than the one in Figure 2.8a, so we do nothing wrong by transforming the model, but loose some information in terms of conditional independence.

It can be shown that G^m is a minimal I-map of X if G is a minimal I-map. Furthermore G^m is a P-map of X if G is a P-map and has no head-to-head nodes. For the proofs see proposition 4.8 and 4.9 in [86].

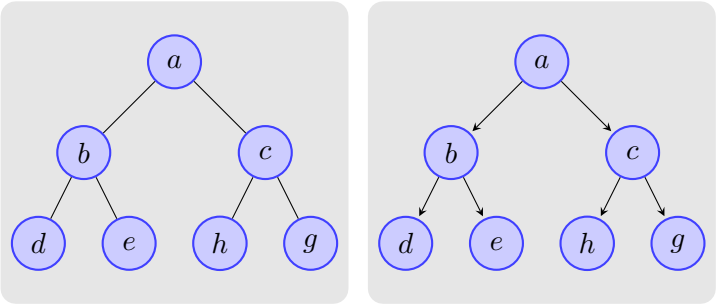
Transforming Undirected to Directed Graphical Models

In practice it might be unusual to transform an undirected model into a directed one, but from the theoretical point of view this step is of interest. Directed graphs can imply a set of conditional independences which can not be implied by undirected graphs, as shown in Figure 2.8. Surprisingly, some sets of conditional independences which are implied by undirected graphs can not be implied by directed graphs. That means for some random variables X exist an undirected graph but no directed graph which is P-map of X.

The independence properties implied by the graph $G = (V, E)$ shown in Figure 2.9a can not be expressed in terms of a directed acyclic graph $G' = (V, E')$ over the same variables. Since the directed graph has to be a DAG, supplementing each edge in G a direction causes an immorality as shown in Figure 2.9b. If the random variables X_a is observed, than the directed graph in Figure 2.9b implies a conditional independence of X_b and X_c, which is not implied by the undirected graph. To ensure that the directed graph covers the undirected one, an additional edge has to be added, as shown in Figure 2.9c. This directed graph covers G but is no P-map of the of the random vector, because X_b and X_c are then never independent.

This shows that undirected models can express conditional dependencies, which can not be expressed by directed models over the same variables.

However, for a subset of undirected models equivalent directed models exist. A trivial subclass of such models are acyclic undirected models, as shown in Figure 2.10. Each edge in the undirected graph can be given a direction without causing an immorality in the resulting directed graph.



(a) (b)

Figure 2.10.: A tree structured undirected model **(a)** can be transformed into an equivalent directed model **(b)** by selecting a root node in the undirected model and directing all edges with respect to this root node.

More generally it can be shown that all undirected models with a chordal graph structure can be transformed into an equivalent directed model. The proof makes use of the junction tree (see Definition 2.6) and is given in [86].

2.3.4. Chain Graph Models

Combining directed and undirected models result in so called chain graph models $M = (X, G = (V, E))$. The graph G is a mixed graph, which has no directed cycle between its chain components. We refer the reader to the standard literature [107, 34] for more details. The definition of the Markov and factorization properties are a similar to those for directed and undirected models.

Undirected and directed models are subclasses of chain graph models. The intersection of proper directed and undirected models is the set of models for which a proper chordal undirected model exists. Using chain graph models, proper models can be built not only in cases in which proper directed or undirected models exist, but can also be useed to build proper graphical models in cases where neither directed nor undirected models gives us a proper representation.

2.4. Factor Graph Models

2.4.1. Definition

While we consider only probabilistic graphical models so far, we will now introduce a more general form of graphical models. To this end, we introduce the concept of factor graphs [97, 2, 112]. Factor graph models are in two important points more general than undirected models. Firstly, they are defined on an arbitrary commutative monoids and thereby not restricted to probabilistic or energy based models. Secondly, they can describe a finer factorization of the objective functions than this can be done by undirected models. We will discuss these two aspects detailed in the following.

Definition 2.16. ***Factor Graph*** A factor graph is a bipartite graph $G = (V, F, E \subset (V \times F))$. With each variable node $a \in V$ we associate a variable $x_a \in \mathcal{X}_a$ and for each factor node $f \in F$ we associate with a small abuse of notation a function $f : \mathcal{X}_{\mathrm{ne}(f)} \to \Omega$.

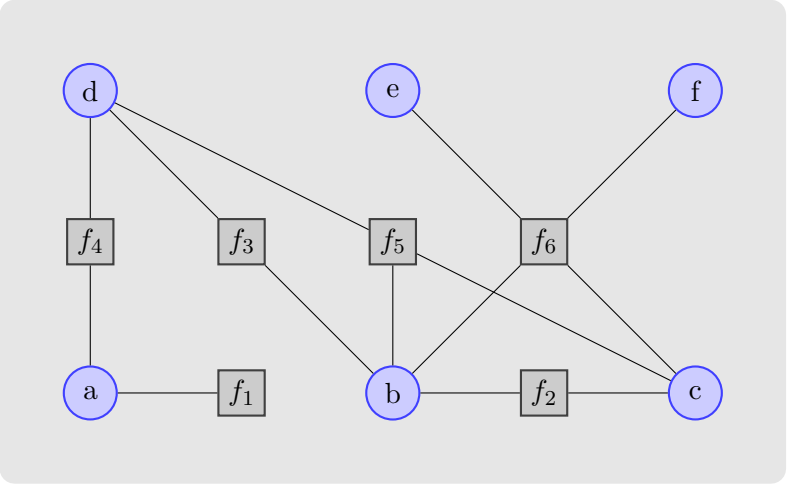


Figure 2.11.: The factor graph contains factor functions of the first (f_1), second (f_2, f_3, f_4), third (f_5) and fourth order (f_6), so the order of this factor graph is 4. The factor nodes are drawn as squares and the variable nodes as cycles. The graph can be simplified by merging f_2, f_3 and f_5, but the resulting factor graph still includes the cycle (b, f_5, c, f_6, b).

For each factor node, $\mathrm{ne}(f) := \{a \in V : (a, f) \in E\}$ denotes the set of all variable nodes that are connected to the factor. For a set of variable nodes $A \subseteq V$, $\mathcal{X}_A = \bigotimes_{a \in A} \mathcal{X}_a$ denotes the Cartesian product of the corresponding variable domains. Corresponding sequences of variables are written as $x_A = (x_a)_{a \in A}$.

In combination with a commutative monoid $(\Omega, \oplus)$, a formal definition of the commutative monoid is given in the appendix Definition A.1, the factor graph describes a factor graph model with an objective function over $\mathcal{X} := \mathcal{X}_V$ given by $\mathbf{f}_\oplus(x) = \bigoplus_{f \in F} f(x_{\mathrm{ne}(f)})$. We will use $\mathbf{f}(x)$ as a shorthand for $\mathbf{f}_\oplus(x)$ if the used operation is clear from the context. The operation $\oplus$ is associative and commutative. Typical candidates for the commutative monoid are $(\{0,1\}, \vee)$, $(\{0,1\}, \wedge)$, $(\mathbb{R}, \cdot)$ or $(\mathbb{R}, +)$.

Definition 2.17. ***Factor Graph models*** A factor graph and a commutative monoid $(\Omega, \oplus)$ define a factor graph model with the objective

$$\mathbf{f}_\oplus(x) = \bigoplus_{f \in F} f(x_{\mathrm{ne}(f)}).$$

Formal definitions on graphs such as neighborhood or the definition of a cycle can be assigned on a factor graph $G = (V, F, E)$ by applying the definitions on the graph $G' = (V \cup F, E)$. The maximum number of neighbors of a factor is called its order. The maximal order of all factor nodes is called the order of a factor graph and is defined as

$$\omega(G_F) = \max_{f \in F} |\mathrm{ne}(f)|.$$

In a visual representation of a factor graph the variable nodes are drawn as cycles while the factor nodes are squares. The factor graph in Figure 2.11 represents the factorization $\mathbf{f}_\oplus(x) = f_1(x_a) \oplus f_2(x_b, x_c) \oplus f_3(x_b, x_d) \oplus f_4(x_a, x_d) \oplus f_5(x_b, x_c, x_d) \oplus f_6(x_b, x_c, x_e, x_f)$.

The essential property of factor graphs is that they model the factorization of the objective function. In contrast, the previously discussed probabilistic models focus on the modeling of conditionally independence and provide a factorization as byproduct. With factor

graphs a much more detailed factorization of the objective function can be represented in terms of the (factor) graph.

2.4.2. Probabilistic Factor Graph Models

If a factor graph model should be used as a probabilistic model, the employed commutative monoid is the set of positive real numbers together with the multiplication $(\mathbb{R}^+, \cdot)$. The objective function $\mathbf{f}_\otimes(x)$ defines a factorization of the probability distribution $p(x)$. To model an energy function we would choose $(\mathbb{R}, +)$ as monoid.

To be consistence with our definition of a probabilistic graphical model (Definition 2.8) we introduce an alternative notation for a probabilistic factor graph model. It extends the model by a random vector which has to be consistent with the variable space $\mathcal{X}_a$ and the factor functions $f(\cdot)$.

Definition 2.18. ***Probabilistic Factor Graph Model*** A probabilistic factor graph model $(X, G = (V, F, E))$ is given by a pair of a random vector X and a factor graph model with the $(\mathbb{R}^+, \cdot)$ monoid.
The probability distribution $p(x)$ of the random vector X factorizes into

$$p(x) \propto \mathbf{f}(x).$$

The graph structure and the factorization of the distribution implies conditional independences on the random variables.

Theorem 2.7. *If for a probabilistic factor graph model $M = (X, G = (V, F, E))$, the set $S \subset V$ blocks all paths from $A \subset V$ to $B \subset V$ in G then*

$$X_A \perp\!\!\!\perp X_B | X_S.$$

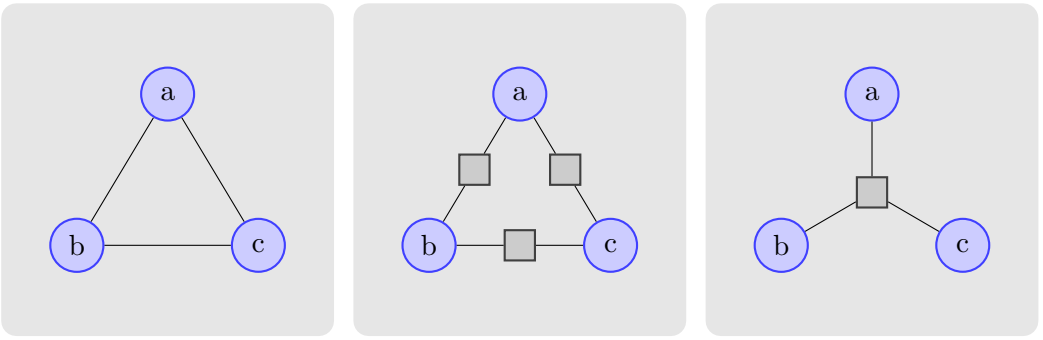
Proof. The random variable X has the factorization property with respect to the undirected graph $G' = (V, E' = \{ab \in V \times V | \exists f \in F : a \in \mathrm{ne}(f),\ b \in \mathrm{ne}(f)\})$ and by Theorem 2.5 also the global Markov property. □

If we assume that we have an undirected model $M = (X, G)$ with a strict positive distribution $p(x)$, it is easy to construct an equivalent probabilistic factor graph model $M' = (X, G')$, with

$$G' = (V, F = \{f_C | C \in \mathcal{C}(G)\}, E' = \{(a, f_C) \in V \times F | a \in C\}).$$

The construction makes use of the Hammersley-Clifford Theorem (Theorem 2.6). The global Markov property is equivalent to the factorization property for strict positive models. Thus, it is sufficient to construct the factor graph model according to the factorization property to fulfill equivalence.

While for strict positive distributions undirected and factor graph models are equivalent in term of conditional independence, factor graph models can express the factorization much more detailed, as illustrated in Figure 2.12. The factor graph in Figure 2.12c ensures a factorization $f_{123}(x_1, x_2, x_3)$ and the graph in Figure 2.12b a factorization $f_{1,2}(x_1, x_2) f_{1,3}(x_1, x_3) f_{2,3}(x_2, x_3)$ of $p(x)$. The factorization implied by the undirected



(a) (b) (c)

Figure 2.12.: Both factor graph models **(b)** and **(c)** are equivalent to the undirected model **(a)** in terms of conditional independence. Beside this Markov properties, factor graph models can represent the factorization of the distribution much more detailed.

model can not express the finer factorization, since the factorization property only guarantees that $p(x)$ factorizes into functions over the maximal clique. This is equivalent to the factor graph in Figure 2.12c.

Like undirected models factor graph modes can not visually express the conditional independence caused by head-to-head nodes in directed model. This information is hidden in the functions themselves. On the other hand, factor graph models, contrary to directed and undirected models, can represent the "real" factorization of a distribution.

2.5. Exponential Family

A broad class of probabilistic graphical models can be formulated as exponential families [173]. In particular all models with discrete variables, on which we will focus, can be represented by exponential families, see Section 2.5.2. Considering probabilistic graphical models from the view point of exponential families opens an entry point into the theory of convex analysis. Furthermore, it provides an alternative way to infer on graphical models as we will see in Section 4.3.

2.5.1. Definition

Given a random vector X which takes values $x \in \mathcal{X}$, let $\phi = (\phi_\alpha)_{\alpha \in \mathcal{I}}$ be the collection of functions $\phi_\alpha : \mathcal{X} \to \mathbb{R}$, called sufficient statistics. The index set $\mathcal{I}$ specifies the vector valued mapping $\phi : \mathcal{X} \to \mathbb{R}^{|\mathcal{I}|}$. For a sufficient statistic ϕ, let $\theta = (\theta_\alpha)_{\alpha \in \mathcal{I}}$ be the associated exponential parameter, which parametrize a distribution according to ϕ.

An exponential family associated with a sufficient statistic ϕ consists of the following parametrized collection of probability distributions

$$p_\theta(x) = \exp\left(\langle \theta, \phi(x) \rangle - A(\theta)\right), \tag{2.23}$$

taken with respect to the base measure ν. The log-partition function is defined by

$$A(\theta) = \log \int_{\mathcal{X}} \exp\left(\langle \theta, \phi(x) \rangle \nu(dx)\right). \tag{2.24}$$

If the integral is finite, this guarantees that p_θ is normalized, i.e. $\int_{\mathcal{X}} p_\theta(x)\nu(dx) = 1$. The exponential parameters θ of interest are those for which $A(\theta)$ is bounded, given by the set

$$\Omega := \{\theta \in \mathbb{R}^{|\mathcal{I}|} | A(\theta) < +\infty\}. \tag{2.25}$$

If Ω is an open set, then the exponential family is called regular. For a fixed sufficient statistic ϕ, each parameter vector $\theta \in \Omega$ represent one member of the exponential family.

An exponential family is called minimal if there exists no non-zero vector $a \in \mathbb{R}^{|\mathcal{I}|}$ such that $\langle a, \phi(x)\rangle$ is constant almost everywhere. For a minimal representation of an exponential family, each distribution is represented by exactly one exponential parameter θ. Non-minimal representations are also referred as overcomplete. For overcomplete representations each distribution can represented by an affine subset of exponential parameters $\{\theta | B\theta = 1, \overline{B}\theta = 0\} \subset \Omega$. Consequently, for arbitrary vectors $\alpha \in \mathbb{R}^N$ and $\beta \in \mathbb{R}^N$ the exponential parameters θ and $\theta + \alpha B + \beta\overline{B}$ represent the same distribution.

Definition 2.19. Suppose vectors θ and θ' define the same distribution, i.e. $p_\theta(x) = p_{\theta'}(x)$ for almost all configurations x. Then θ' is called a reparametrization of θ.

If the reparametrization of θ into θ' by

$$\theta' = \theta + \alpha B + \beta\overline{B} \tag{2.26}$$

uses only the homogeneous dependencies $\overline{B}\theta = 0$, that is $\alpha = 0$, we will call it a homogeneous reparametrization. Homogeneous reparametrizations additionally guarantee that for almost all $\phi(x)$ the value of $\langle a, \phi(x)\rangle$ does not change, while general reparametrizations only preserve the distribution. Reparametrizations which only use the inhomogeneous dependencies are called inhomogeneous reparametrizations.

We will show two essential properties of the exponential families. Firstly, the derivatives of $A(\theta)$ are the expectations of the sufficient statistic ϕ under $p_\theta(x)$. And secondly, $A(\theta)$ is a convex function of θ. In the context of exponential families we will use $\mathbb{E}_\theta(\cdot)$ as a shorthand for $\mathbb{E}_{p_\theta}(\cdot)$

Theorem 2.8. *The log-partition function*

$$A(\theta) = \log \int_{\mathcal{X}} \exp\langle \theta,\ \phi(x)\rangle \nu(dx)$$

associated with any regular exponential family has the properties:

a) The first two derivatives yields the moments of the random vectors $\phi(X)$ as follows:

$$\frac{\partial A}{\partial \theta_\alpha}(\theta) = \mathbb{E}_\theta[\phi_\alpha(X)] \tag{2.27}$$

$$\frac{\partial^2 A}{\partial \theta_\alpha \partial \theta_\beta}(\theta) = \mathbb{E}_\theta[\phi_\alpha(X)\phi_\beta(X)] - \mathbb{E}_\theta[\phi_\alpha(X)]\mathbb{E}_\theta[\phi_\beta(X)] \tag{2.28}$$

b) $A(\theta)$ is a convex function of θ on its domain Ω, and strictly convex if the exponential family is minimal.

Proof. See Section A.3 □

A fundamental result of Theorem 2.8 is, that $\nabla A(\theta)$ is a mapping from the parameter space Ω into the mean parameter space which is defined as

$$\mathcal{M} := \left\{ \mu \in \mathbb{R}^{|\mathcal{I}|} \,\middle|\, \exists\, p(x) \; s.t. \; p(x) \geq 0, \; \int_x p(x)\nu(dx) = 1, \; \mu = \mathbb{E}_p(\phi(x)) \right\}. \tag{2.29}$$

Let us emphasize that $p(x)$ is an arbitrary distribution, not necessary from the exponential family. However, any vector $\mu \in \mathcal{M}$ which does not lie on the boundary of $\mathcal{M}$ can be realized by a distribution from the exponential family. The boundary points of $\mathcal{M}$ are not covered because $\nabla A(\theta)$ approaches the boundary of the mean polytope when and only when some components of θ approach infinity [173].

The mean parameter space $\mathcal{M}$ is equivalent to the convex hull of the image of $\mathcal{X}$ under the mapping ϕ,

$$\mathcal{M} = conv(\{\phi(x) | x \in \mathcal{X}\}. \tag{2.30}$$

Consequently, $\mathcal{M}$ is convex. If in addition $\mathcal{X}$ is finite, $\mathcal{M}$ is a convex polytope. The vertices of this polytope correspond to values in $\mathcal{X}$. In such cases we will call $\mathcal{M}$ a marginal polytope. An alternative description using a finite set of linear inequality constraints is provided by the Minkiwski-Weyl theorem [134]

$$\mathcal{M} = \left\{ \mu \in \mathbb{R}^{|\mathcal{I}|} \,\middle|\, \langle a_j, \mu \rangle \geq b_j \forall j \in \mathcal{J} \right\}. \tag{2.31}$$

In this representation the non-redundant inequality constraints, indexed by $\mathcal{J}$, correspond to the facets of the marginal polytope.

The computation of the forward mapping $\nabla A(\theta)$ can become very difficult since the computation of the integral might be very costly. This is not surprising, since the forward mapping solves one of the fundamental inference problems on graphical models (see Chapter 4).

The opposite direction, the backward mapping, maps a mean parameter of the relative interior of the mean space $\mu \in \text{ri}(\mathcal{M})$ to an exponential parameter $\theta \in \Omega$. Using conjugate duality from convex analysis [134] yields the backwards mapping. The conjugate dual function of $A(\theta)$ is defined by

$$A^*(\mu) := \sup_{\theta \in \Omega} \langle \theta, \mu \rangle - A(\theta) \tag{2.32}$$

and since $A(\theta)$ is a proper, convex and continues function on Ω we have

$$A(\theta) = \sup_{\mu \in \mathcal{M}} \langle \theta, \mu \rangle - A^*(\mu). \tag{2.33}$$

The backward mapping is given by the subdifferential $\partial A^*(\mu)$ of the dual function. For minimal representations, $\nabla A(\theta)$ is a bijective function from Ω to $\text{ri}(\mathcal{M})$ and the reverse mapping from $\text{ri}(\mathcal{M})$ to Ω is given by the gradient $\nabla A^*(\mu)$. For overcomplete representations, $\nabla A(\theta)$ is not bijective, because all parametrization of θ are mapped to the same mean parameter and the backward mapping is a multivalued function. For any mean parameter on the boundary of the closure of $\mathcal{M}$, we have $A^*(\mu) = \lim_{n\to\infty} A^*(\mu^n)$ taken over a sequence $\{\mu^1, \mu^2, \ldots\} \subset \text{ri}(\mathcal{M})$ converging to μ (see Appendix B in [173] for more details and proofs). Figure 2.13 provides an idealized illustration of the bijective mappings for minimal representations, based on the gradient mappings $(\nabla A(\theta), \nabla A^*(\mu))$.

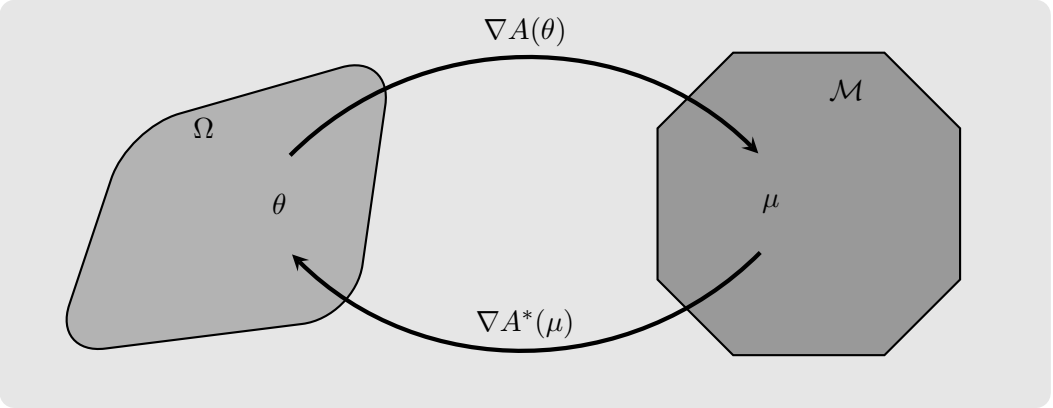


Figure 2.13.: Idealized illustration of the mapping between the parameter space Ω and the relative interior of the mean space $\mathrm{ri}(\mathcal{M})$. Each exponential parameter $\theta \in \Omega$ is mapped by $\nabla A(\theta)$ into the mean space. A mean parameter $\mu \in \mathrm{ri}(\mathcal{M})$ is mapped to a single parameter $\theta \in \Omega$ for minimal representations or to a set of parameters for overcomplete representations.

For a statistical interpretation of the backward mapping, we assume that an empirical mean parameter $\hat{\mu}$ is given for a set of samples $x^1, \ldots, x^n$ by $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x^i$. The standard approach for finding the optimal distribution $p_\theta(x) = p(x|\theta)$ in an exponential family is to maximize the log-likelihood of the data, which is equivalent to evaluating the dual of the log partition function as we will show in (2.34). Estimating the optimal parameter θ can be seen as a backward mapping of $\hat{\mu}$ by calculating $\partial A^*(\mu)$.

$$\sup_{\theta\in\Omega} l(\theta, \{x^i\}_{i=1,\ldots,n}) = \sup_{\theta\in\Omega} \sum_{i=1}^{n} \frac{1}{n} \log p_\theta(x^i) \tag{2.34}$$

$$= \sup_{\theta\in\Omega} \sum_{i=1}^{n} \frac{1}{n} \langle \theta, \phi(x^i)\rangle - A(\theta) \tag{2.35}$$

$$= \sup_{\theta\in\Omega} \langle \theta, \hat{\mu}\rangle - A(\theta) \tag{2.36}$$

$$= A^*(\hat{\mu}) \tag{2.37}$$

Let us further consider the entropy $H(X)$ of a random vector X is given by the integral over $\mathcal{X}$ with respect to some base measure ν,

$$H(X) = -\int_{x\in\mathcal{X}} p(x) \log p(x) \nu(dx). \tag{2.38}$$

For the discrete case this simplifies as follows:

$$H(X) = -\sum_{x\in\mathcal{X}} p(x) \log p(x). \tag{2.39}$$

For any mean parameter in the relative interior of the mean space $\mu \in \mathrm{ri}(\mathcal{M})$, $A^*(\mu)$ is the largest value of the negative entropy for a member $p_\theta(x)$ of the exponential family which has the mean value μ.

2.5.2. Exponential Family for Discrete Graphical Models

Since in this work we will focus on discrete graphical models we will explain how this is related to the exponential family representation. Let us assume that we have an undirected graphical model (X, G) with density $p(x)$ which factorize into $p(x) = \frac{1}{Z}\prod_{C\in\mathcal{C}(G)} f_C(x_C)$. The corresponding exponential family is defined by:

$$\mathcal{I}(G) := \{(C;i)|C \in \mathcal{C}(G),\ i \in \mathcal{X}_C\} \tag{2.40}$$

$$\phi_{(C;i)}(x) := \begin{cases} 1 & \text{if } x_C = i \\ 0 & \text{else} \end{cases} \tag{2.41}$$

$$\theta_{(C;i)} := \log(f_C(i)) \tag{2.42}$$

$$A(\theta) := \log(Z) = \log\left(\sum_{x\in\mathcal{X}} \langle\theta, \phi(x)\rangle\right) \tag{2.43}$$

The base measure is a counting measure such that we can replace the integral by a sum. To proof this we insert (2.40)-(2.43) into (2.44) and get

$$p_\theta(x) = \exp\left(\langle\theta,\ \phi(x)\rangle - A(\theta)\right) \tag{2.44}$$

$$= \exp\left(\sum_{(C;i)\in\mathcal{I}(G)} \log(f_C(i))\phi_{(C;i)}(x) - \log(Z)\right) \tag{2.45}$$

$$= \prod_{(C;i)\in\mathcal{I}(G)} f_C(i)\phi_{(C;i)}(x)\frac{1}{Z} \tag{2.46}$$

$$= \frac{1}{Z}\prod_{C\in\mathcal{C}(G)} f_C(x_C) \tag{2.47}$$

For a probabilistic factor graph model $(X, G = (V, F, E))$ there is a similar method to transform it into a member of an exponential family. This exponential family makes use of the detailed factorization of the factor graph:

$$\mathcal{I}(G) := \{(f;i)|f \in F,\ i \in \mathcal{X}_{\mathrm{ne}(f)}\} \tag{2.48}$$

$$\phi_{(f;i)}(x) := \begin{cases} 1 & \text{if } x_{\mathrm{ne}(f)} = i \\ 0 & \text{else} \end{cases} \tag{2.49}$$

$$\theta_{(f;i)} := \log(f(i)) \tag{2.50}$$

$$A(\theta) := \log(Z) = \log\left(\sum_{x\in\mathcal{X}} \langle\theta, \phi(x)\rangle\right) \tag{2.51}$$

the proof follows the same way as the previous

$$p_\theta(x) = \exp\left(\langle\theta,\ \phi(x)\rangle - A(\theta)\right) \tag{2.52}$$

$$= \exp\left(\sum_{(f;i)\in\mathcal{I}(G)} \log(f(i))\phi_{(f;i)}(x) - \log(Z)\right) \tag{2.53}$$

$$= \prod_{(f;i)\in\mathcal{I}(G)} f(i)\phi_{(f;i)}(x)\frac{1}{Z} \tag{2.54}$$

$$= \frac{1}{Z}\prod_{f\in F} f(x_{\mathrm{ne}(f)}) \tag{2.55}$$

2.6. Markov Random Fields vs. Conditional Random Fields

The random variables of a Markov Random Field (MRF) can be split into two random vectors X and Y. With X and Y we denoted the hidden and observed variables, respectively. They have the joint distribution $p(x, y)$.

MRFs are generative models, because given the state of the hidden variables we can generate observations by sampling from $p(y|x)$. Usually, the enumeration of all possible observations is intractable. Unless observed random variables are represented as isolated units, independent from each other, or other simplifications are assumed, it is not manageable to model $p(y)$ in a useful fashion. An example for such a conditional independence in the observed variables given all hidden variables was shown in Figure 2.7. In this model the observation at each pixel depends only on the hidden variable associated with this pixel. This is very simple and includes no statistic of the global texture of the image.

In most real world applications, the observed random variables are represented in terms of multiple interacting features of the data (e.g. image) with long-range dependencies. Thus we will not model the joint distribution $p(x, y)$ but the conditioned distribution $p(x|y)$ for a given observation y of the random vector X. Such a model is called Conditional Random Field (CRF) [104], because the distribution of X is conditioned on the observation y. The CRF-framework is a discriminative framework, which does not model the prior distribution of the observed variables. CRFs can only be used to discriminate between different settings of a hidden variable for a given observation.

Definition 2.20. A Conditional Random Field (CRF) is a triple (X, Y, G) of two random vectors X and Y and an undirected graph $G = (V, E)$ such that $X = (X_a)_{a \in V}$, and conditioned on Y, the random vector X has the Markov Property with respect to G.

If we assume that $p(x|y)$ is strictly positive, the distribution factorizes according to

$$p(x|y) = \frac{1}{Z} \exp\left(\sum_{C \in \mathcal{C}(G)} \theta_C(x_C) \right). \tag{2.56}$$

Furthermore, we will assume that each local function $\theta_C(x_c)$ is given by the weighted sum of functions $g_{C;k}(\cdot)$ with $k \in \mathcal{F}_C$, i.e.

$$\theta_C(x_c) = \sum_{k \in \mathcal{F}_C} \lambda_{C;k} \cdot g_{C;k}(x_C, y). \tag{2.57}$$

Training a conditional random field requires to find the set of parameters λ which fits best, e.g. maximize the likelihood. We will come back to this point in Section 3.3.2.

We will conclude this chapter with two examples. The first example is a generative MRF which models the noisy transmission of a signal. An overcomplete representation of the signal can be used for robust decoding. In the second example we introduce a part based model for object detection with a discriminative CRF. In Section 3.2 we will have a closer look on this part-based models. These two examples demonstrate the concepts of MRF and CRF, and relight the whole chapter from a more practical point of view.

Example: Signal Decoding A simple example for the use of generative graphical models is the decoding of noisy signals. Instead of the raw data usually a redundant representation of the data is transmitted. This redundancy allows that the original data can be

reconstructed even if some bits of the redundant representation are flipped. Such codes are called error correcting codes [77][2]. The causal process of encoding and noisy transmission can be described by the directed graphical model shown in Figure 2.14a. All random variables in this model are binary. Given the original data $x_d = (x_{d1}, x_{d2}, x_{d3}, x_{d4})$, the redundant codeword $x_s = (x_{s1}, x_{s2}, x_{s3}, x_{s4}, x_{s5})$ is uniquely defined by the function χ with $x_{si} = \chi_i(x_d)$. The bits x_s are sent and the codeword $x_r = (x_{r1}, x_{r2}, x_{r3}, x_{r4}, x_{r5})$ is observed by the receiver. The probability that the transmitted bit is equal to the received one depends on the noise and is modeled by the conditioned probability $p(x_{ri}|x_{si})$. Contrary to naive models, which calculated the data x_d minimizing the hamming distance between all valid codewords x_s and the received codeword x_r, we model the statistics for each bit independently. That will be helpful if each bit uses its own wire with a specific noise statistic. One can also think about correlations between adjacent wires or temporal relations. The distribution is defined by

$$p(x_{si}|x_d) = \begin{cases} 1 & \text{if } \chi_i(x_d) = x_{si} \\ 0 & \text{else} \end{cases} \tag{2.58}$$

$$p(x_{ri}|x_{si}) = \begin{cases} 1 - \beta_{i;0} & \text{if } x_{ri} = 0,\ x_{si} = 0 \\ \beta_{i;0} & \text{if } x_{ri} = 1,\ x_{si} = 0 \\ \beta_{i;1} & \text{if } x_{ri} = 0,\ x_{si} = 1 \\ 1 - \beta_{i;1} & \text{if } x_{ri} = 1,\ x_{si} = 1 \end{cases} \tag{2.59}$$

The directed graph for this model is shown in Figure 2.14a.

With this generative model we are able to simulate the transmission for a given noise model defined by β. We sample codewords x_d according to $p(x_d)$ calculate x_s and flip the bit x_{si} with a probability of $\beta_{i;x_{si}}$.

Since all random variables in this model are binary, the construction rules for a discrete model from Section 2.5.2 can be used to build the exponential family for this problem class represented by a probabilistic factor graph model

$$\mathcal{X}_a := \{0, 1\} \tag{2.60}$$

$$\mathcal{I}(G) := \{(f; i) | f \in F,\ i \in X_{\text{ne}(f)}\} \tag{2.61}$$

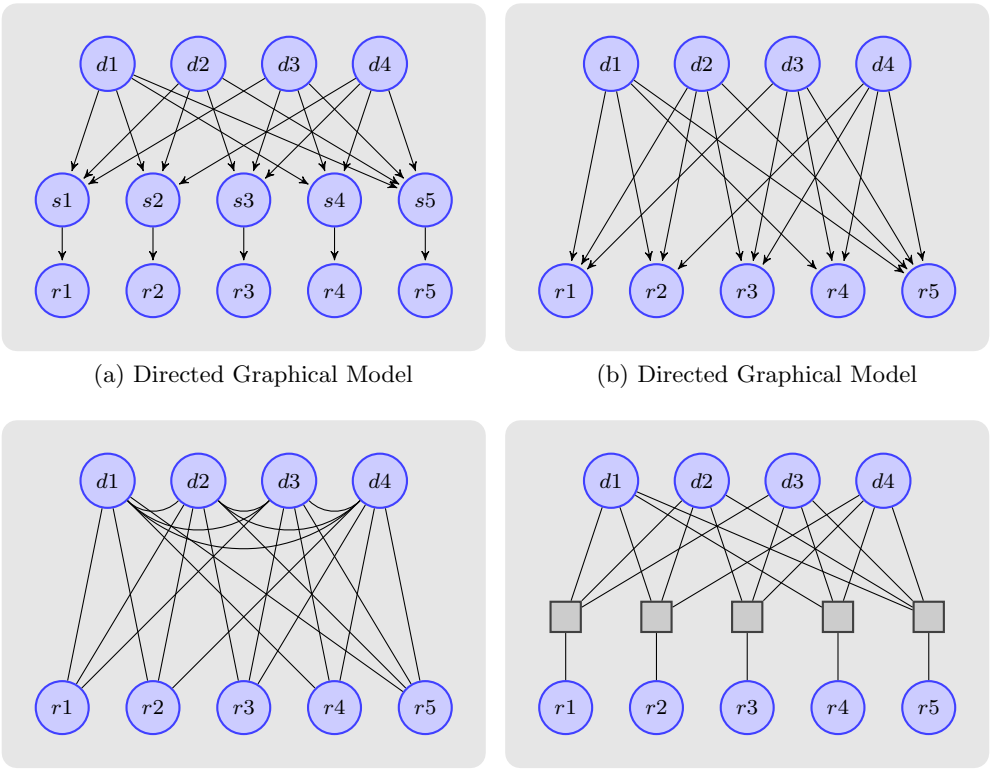
$$\phi_{f;i}(x) := \begin{cases} 1 & \text{if } x_{\text{ne}(f)} = i \\ 0 & \text{else} \end{cases} \tag{2.62}$$

$$\theta_{f;i} := \log(f(i)) \tag{2.63}$$

$$A(\theta) := \log(Z) = \log\left(\sum_{x \in \mathcal{X}} \langle \theta, \phi(x) \rangle\right) \tag{2.64}$$

There are two problems in this context which are from major interest. Firstly, we need to estimate the noise parameter β. For this we use a set of original data together with the received codeword and search the parameters which describes this behavior best. This problem is called the learning problem. From the viewpoint of exponential families we do a backward mapping from the empirical mean parameter into the exponential parameter space. We will light this problem in Section 3.3. The second problem is to find the most probable original data for a received code word. In Chapter 4 we will introduce several algorithms which can be used to solve this problem.

[2]The codes are called error correcting because with the assumption that only N bits are flipped, the original signal can be reconstructed. N depends on the redundancy of the representation.



(a) Directed Graphical Model

(b) Directed Graphical Model

(c) Undirected Graphical Model

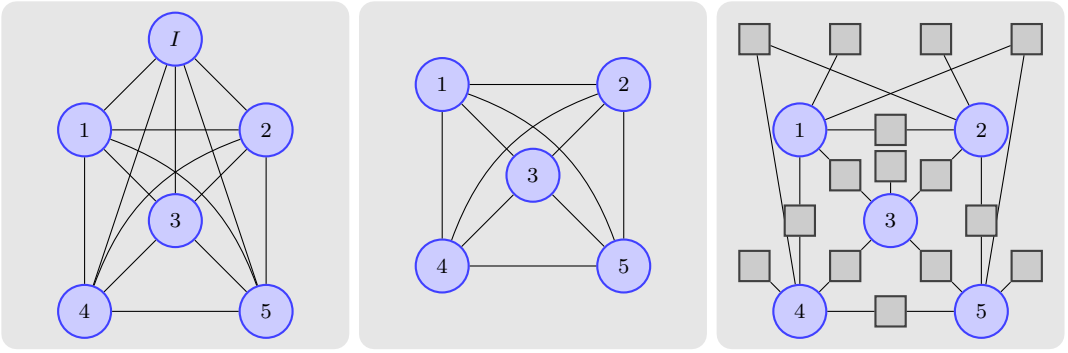
(d) Factor Graph Model

Figure 2.14.: The graphs above show the structure of graphical models for codeword-error-detection. A codeword x_d is transformed into a redundant representation x_s, sent over a noisy channel and observed as x_r. In **(a)** the directed graphical model according to this causal process is shown. Marginalization over x_s leads to the model shown in **(b)** which can be transformed into an undirected graph **(c)**. Figure **(d)** shows the corresponding representation as a factor graph. If we observe some codeword x_r we can use the graphical model to infer the most probable original data x_d.

Example: Part-Based Face Detection Assume the scenario that in a given image a face should be detected – more precisely the position of the eyes, the nose and the mouth corners. A part-based model assembles the face by a set of parts and relations between them.

We model the face by a CRF (X, Y, G), with a full connected graph $G = (V, E)$ with five nodes. Each node corresponds to one of the five face parts. The domain of the hidden random variables $\mathcal{X}$ is the image domain (set of all pixel positions). The observed random variable Y represents the image. The domain of Y is the set of all functions from the image domain into the color domain. If we assume a discrete representation of the image with $N \times M$ pixels and a color space $\{1, \ldots, 256\}^3$ the set of all possible images has the size $N \cdot M \cdot 256^3$. Modeling the distribution $p(y)$ or $p(y|x)$ is therefore hopeless without further simplifications.

Since we are not interested in the distribution over all images it is reasonable to avoid modeling $p(y)$ and use a CRF instead of a MRF. The graph to the MRF shown in Figure 2.15a includes 6 nodes. The nodes 1 to 5 represent the hidden variables x_i and encode



(a) Undirected Model (MRF) (b) Undirected Model (CRF) (c) Factor Graph Model (CRF)

Figure 2.15.: A simple part-based detection problem is the detection of a face in an image. The MRF **(a)** includes a random variable for the image. Conditioning the model on the image we get a CRF with the graph shown in **(b)**. If we further assume that the distribution $p(x|y)$ factorizes into first and second order terms we get the factor graph model shown in **(c)**.

the position of the face parts. The node I represent the observed variable y coding the image. If we condition the model to the image we get the CRF representing the conditioned distribution $p(x|y)$. The graph to the CRF is shown in Figure 2.15b. If we further assume that this distribution factorizes into terms of first and second order, i.e.

$$p(x|y) = \frac{1}{Z} \underbrace{\prod_{a \in V} \exp(\theta_a(x_a|y))}_{\propto \frac{p(x|y)}{p(x)}} \cdot \underbrace{\prod_{ab \in E} \exp(\theta_{ab}(x_a, x_b))}_{\propto p(x)}, \tag{2.65}$$

we get a manageable model which can be described by the factor graph show in Figure 2.15c.

The unary terms $\theta_a(x_a)$ measures how good the image region around x_a fits to a previously learned model for the appearance of the part a. Let the binary terms $\theta_{a,b}(x_a, x_b)$ be independent of y and encode a geometric or shape prior for the position of the face parts.

The unary terms are ambiguous and have high measurements also in other positions than the correct one (e.g. right and left eye are very similar). Thus using only unary terms will not be sufficient for good detections. We overcome this problem by adding a geometric prior represented by the binary terms $\theta_{a,b}(x_a, x_b)$. For a fixed scale of the image, the distance between the parts is distributed around the mean distance with a small variance. For fixed orientation of the face, the alignment of two parts will also be very similar for different faces. Such geometric or shape information can been encoded in the pairwise terms in (2.65).

Again, we can reformulated this problem into an exponential family representation, if we

assume a finite number of parts position $P_a \subset \mathbb{R}^2$.

$$\mathcal{X}_a := P_a \tag{2.66}$$

$$\mathcal{I}(G) := \{(C;i)|C \in V \cup E,\ i \in X_C\} \tag{2.67}$$

$$\phi_{C;i}(x) := \begin{cases} 1 & \text{if } x_C = i \\ 0 & \text{else} \end{cases} \tag{2.68}$$

$$\theta_{C;i} := \theta_C(i) \tag{2.69}$$

$$A(\theta) := \log(Z) = \log\left(\sum_{x \in \mathcal{X}} \langle \theta, \phi(x) \rangle\right) \tag{2.70}$$

Again, the two major problems are i) the learning problem, which estimates the functions $f(x_C|y)$ and the parameters λ which describe the distribution $p_\theta(x|y)$ best and ii) the problem of finding the most probable part based description x of the image.

CHAPTER 3

GRAPHICAL MODELS FOR VISUAL OBJECT DETECTION

3.1. Overview

Whenever someone would like to infer some information from an image, e.g. which objects are contained in the image or if and where a specific object is located in an image, it is important to understand the causality of the physical process which generates the image data.

These process of capturing an image can be described by a graphical model as shown in Figure 3.1. The scene s is described by a set of objects $\{o_i\}_i$ which are illuminated by a set of light source $l = \{l_i\}_i$. The image I is captured by a sensor, usually a camera, with internal and external parameters, denoted by c in the graph. In some settings we will have two or more cameras which capture several pictures of the scene, at the same time. The corresponding graph is shown in Figure 3.2.

If the settings of scene x_s, the light x_l and the camera parameters x_c are known, it is possible to render the corresponding image. Even if we can model this causal process and represent the conditional dependences by an undirected graph, it is in general practically impossible to give the probability distribution of this model without making any simplifications.

In a typical problem setting the image is given and sometimes we also observed the camera parameters (e.g. in stereo vision problems) or a description of the observe scene and lighting conditions (camera calibration). The unobserved variables are unknown and can be theoretical eliminated by marginalization if we are not interested in them. On the remaining variables we would like to infer, for example find a setting which maximize the probability which is conditioned on the observation.

If we want to infer on the scene, we have the problem of defining a representation of the scene. For this purpose, often a simplified representation of the scene is used, which includes the information of interest. An additional random variable X_r which represent this

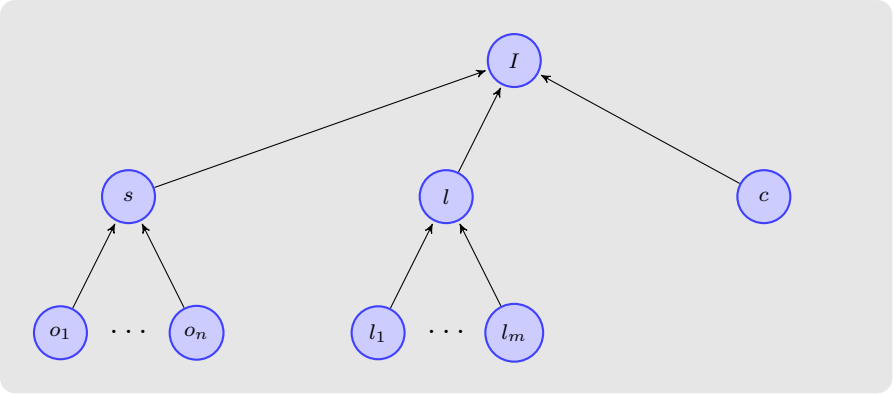


Figure 3.1.: The process of image acquisition can be described by a graphical model. A set of objects o_i describe the scene s which is illuminated by a set of lighting sources l_i. A sensor c captures an image I from this illuminated scene.

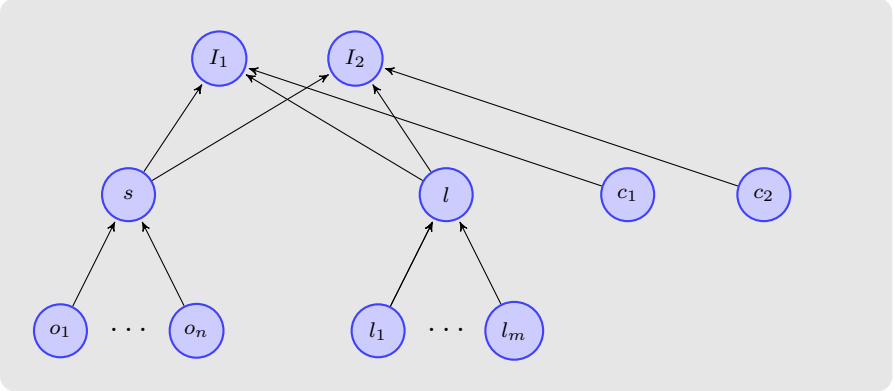


Figure 3.2.: The process sketched in Figure 3.1 can be easily extended to multiple sensors. While none pair of the sensors a co-parents in the directed graph, each single sensor c_i is a co-parent of the scene s and lighting sources l.

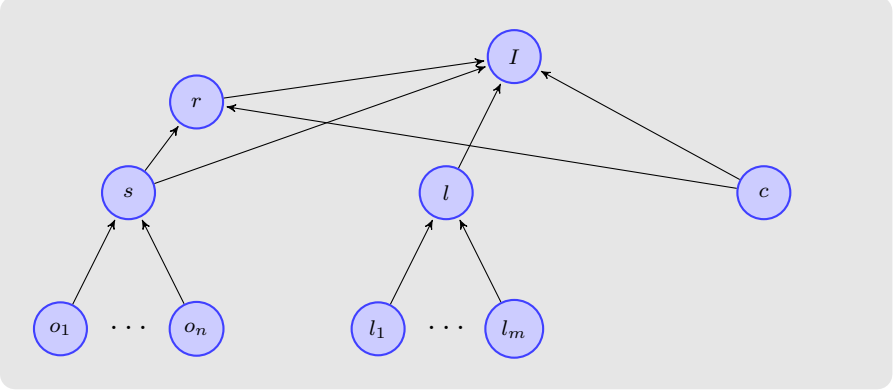


Figure 3.3.: If we are only interested in a subpart of the information included in the description of the scene we can add an extra random variable which represents only this information. To this end a node r is added into the directed graph which depends on the scene and – if the reduced representation lives in the image domain – also on the sensor.

reduced representation is included into the statistical model. The conditional independences of the random variables are represented by the graph in Figure 3.3. The reduced representation of the scene is often defined on the image domain and thus depends on the camera parameters, i.e. we have an edge from c to r in the graph.

If an image I is observed, we are interested in the conditional probability distribution $p(x_r|I)$. Theoretically, this can be calculated by marginalization over the other hidden variables

$$p(x_r|I) = \int_{\mathcal{X}_c} \int_{\mathcal{X}_s} p(x_r, x_c, x_s|I)\, dx_c\, dx_s.$$

Since neither the distribution $p(x_r, x_c, x_s|I)$ is usually known nor the integrals can be computed, the distribution $p(x_r|I)$ has to be estimated, e.g. by learning the parameters of a CFR. Furthermore, it is common to simplify the problem by making additional assumptions on the conditional independence or factorization of $p(x_r|I)$. For a more compact notation let us denote x_r by x and $p(x_r|I)$ by $p(x|I)$.

A common reduced representation includes a random variable for each pixel which takes values over the set of objects (Image Segmentation), colors (Image Denoising or Inpainting), disparities between corresponding pixels in two images (Stereo Matching) or scenes (Photo-montage). For a detailed description of such models we refer the reader to the comparative study of Szeliski et al. [159] and the references therein.

All models used in [159] assume that the conditional independence of the corresponding CRF can be represented by a grid graph as shown in Figure 3.4. This major simplification is required to render MAP-inference tracable, see Chapter 4. Relations between non-adjacent pixels or larger set of pixels can not be directly represented in such models. This limitation allow simple factorization

$$p(x|I) \propto \exp\big(-\big(\underbrace{\sum_{a\in V} f_a(x_a|I)}_{\text{data term}} + \lambda \underbrace{\sum_{ab\in E} f_{ab}(x_a, x_b|I)}_{\text{regularizer}} \big)\big) \tag{3.1}$$

The data term $\sum_{a\in V} f_a(x_a|I)$ sums up the pixel-wise data cost. The so called smoothness term or regularizer, usually does not depend on the image, and measures the local regularity of x. For segmentation for example it might be more probable that neighbored pixels are generated by the same object. A common and simple choice for the regularizer is

$$f_{ab}(x_a, x_b|I) = \begin{cases} 0 & \text{if } x_a = x_b \\ 1 & \text{else} \end{cases} \tag{3.2}$$

The parameter λ balance the relative importance of the data and smoothness term.

A similar method is used by the current winners of the Visual Object Classes (VOC) Challenge 2009 [42] in the Segmentation Competition, Carreira et al. [25]. In this challenge an image should be segmented into several known classes, see Figure 3.5. Carreira et al. use a grid graph model with a local statistic on different foreground classes.

All this models have an essential drawback: They include only local terms and can not model more global shape priors. If for example in the binary segmentation problem the foreground object is a ball, we can not include priors into the model that the segmentation of the foreground should form a circle. As long as the data term is strong enough this limitation can be compensated by the model. Veksler [163] shows that a point inside the

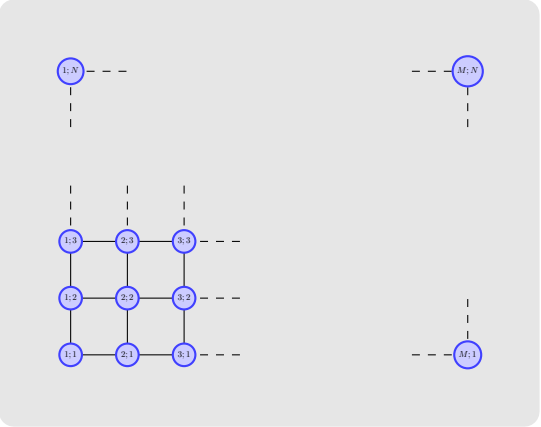
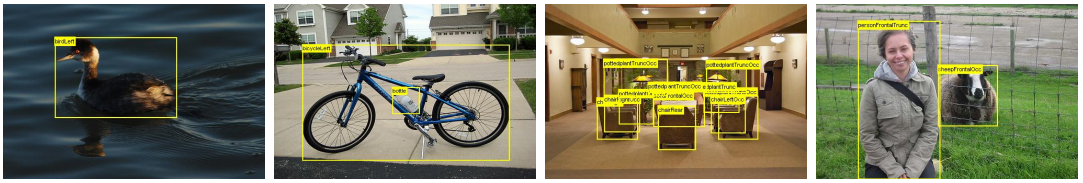


Figure 3.4.: Typically graphical models which associate random variables with pixels, assume that the conditional independences can be approximatively represented by a grid graph which connect only nodes corresponding to neighbored pixels. Dependences between distant variables are ignored. This approximation of the true conditional independence relation makes the problems tractable.

(a) Image (b) Ground Truth (c) Bonn [25]

(d) CVC [61] (e) UCI [184] (f) UoCTTI [45]

Figure 3.5.: The image **(a)** is an example from the VOC Segmentation Competition [42] with some ground truth notation **(b)**. Pixel belonging to a human are pink and pixels belonging to a table are yellow. The figures **(c-f)** show the result of different segmentation algorithms. Due to the highly clutter scene, the loss of global shape information can not be compensate by the data term, and none of the methods is able to give a good segmentation of the two persons. Images are taken from [42].



(a) (b) (c) (d)

Figure 3.6.: Hand-labeled data from the VOC Detection Competition 2009 [42]. The labeled objects are marked by bounding boxes. The challenge is to detect objects of several classes in unknown images. In this context, detecting means to find a bounding box around the object and assign it to the correct object class.

segment, which is obtained by user interaction, can be used to enforce a star shape prior on the segment by adding pairwise terms to the objective function.

Another line of research over-segments the image and associates the random variables with the segments, also known as super-pixels, instead of describing each pixel independently. In this context Borenstein et al. [16] presented a model which includes regional shape prior into the model by combining top-down and bottom-up segmentation.

In all this methods the random variables correspond to a fixed position in the image and their domain includes the possible explanations for this position. Roughly spoken, we fix *where* and ask *what* there is. Instead, we can also fix *what* we are searching and ask *where* it is in the image. The reduced representation of the scene is then given by a simple description of one or several parts in the image domain. Such a representation of the scene is often used for object detection. In the simplest form the model consists out of one part. Figure 3.6 shows some hand-labeled data from the VOC Detection Challenge [42]. An object is described by a bounding box around the object and its class (e.g. bird, person or sheep). The challenge is to find the bounding box and object class in images.

A simple and popular approach to solve this problem is the sliding window method. A classifier is applied on all scales, positions and sometimes also orientations in an image. However, testing for all this candidates for non-trivial classifiers can be computational expensive. A simple method addressing this problem involves applying a cascade of simple tests for each candidate [164, 84, 38, 17, 53, 57, 152]. This cascade eliminates the most of the candidates very quickly in an early stage. Before the classifier can be applied a descriptor of the image patch has to be computed. This descriptor is a representation of the image patch. The size of this descriptor is independent from the size of the image patch, and sometimes also invariant to scale, color-transformations or rotation. Common choices are Haar- [164], HoG- [36] or SIFT-descriptors [113].

Hampered by the omnipresent confusing information due to clutter and occlusion, as well as the high intraclass variability of several object classes, the focus in the field of object detection has shifted from holistic approaches to part-based ones. Bag of (visual) words approaches [37, 117, 150] describe an image patch by a histogram of visual words appearing in this patch. A face for example could be described by the visual words eyes, nodes, ears, mouth and background. Visual words have not to be parts in our usual understanding, visual words can also be some abstract classes calculated by a learning procedure, see [37]. For the classification of a patch, in the first step local features are computed for several positions in the patch. This positions can be obtained by a regular grid, an interest point detector or other methods (e.g. random sampling). The local features are usually

represented by a SIFT-descriptor. Then each of this features is mapped to one of the words such that the patch can be represented by the histogram of this words. This histogram is used as input for a classifier.

The main drawback of bag of word approaches is that they do not capture the relation between the visual words. It is not sufficient that detections for eyes, nose and a mouth are present in an image patch to imply that the patch contains a face. At least this parts should satisfy some geometric constraints. Contrary to bag of word approaches part-based approaches includes this relative information between parts. A part-based models consist of a predefined set of parts. The quantity and meaning of parts is fixed for a model. The domain of the random variables are positions or regions in the image. The distribution of this variables conditioned on the image is usually assumed to factorize into terms of first and second order as shown in (3.3). The appearance of the object is represented by unary and in some models also pairwise terms (feature functions). The geometric relation between the parts is always coded in pairwise terms (feature functions), which do not depend on the image.

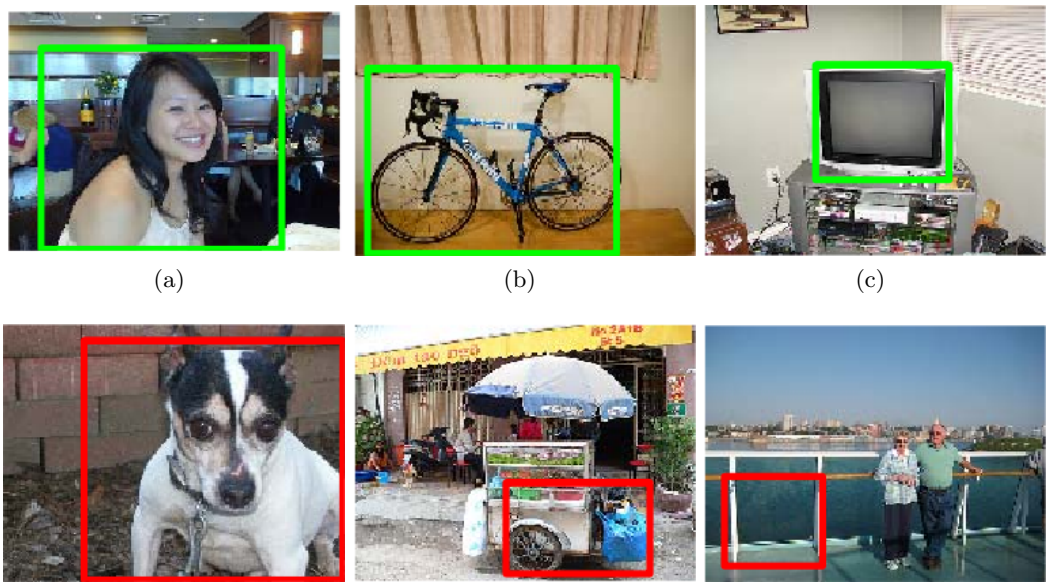
$$p(x|I) = \frac{1}{Z} \exp\left(\sum_{a \in V} \theta_a(x_a) + \sum_{ab \in E} \theta_{ab}(x_a, x_b)\right) \tag{3.3}$$

$$\theta_C(x_C) = \sum_k \lambda_{C;k} f_{C;k}(x_C) \tag{3.4}$$

The current two best performing approaches benchmarked on the VOC Detection Challenge are i) the part-based approach by Felzenszwalb et al. [45] which use a tree structured model and ii) the sliding window approach by Vedaldi et al. [162]. Both approaches merge modeling and optimization and use many other tricks for faster computation. The method of Felzenszwalb et al. includes a part covering the whole object and several smaller parts inside which are learned automatically and have no meaning in the real world. Appearance is only defined on single site terms, i.e. terms which depend on only one variable. The used descriptors are HoG-features, which are furthermore reduced by a principal component analysis (PCA) [81]. For fast performance the geometric relations are encoded such that the distance transform method [46] can be used. The most expensive part in this framework is the evaluation of the local appearance terms. Therefore, they use a cascade method. The sliding window method by Vedaldi et al. use a descriptor based on dense visual words, self-similarity descriptors, and edge based descriptors. For fast performance the descriptors are classified in a Viola-Jones-style [164]. Exemplary, we show some result of the approach by Felzenszwalb et al. in Figure 3.7.

While for the detection problem several good performing approaches exists another very related problem, seems to be much more complicated. The VOC Person Layout Challenge considering this problem and was started in 2007. This challenge contains only the object class "person", but a person is no longer represented by a single bounding box. A correct description includes the bounding boxes for the person, its head, hand and feet if they are present in the image, Figure 3.8 shows some hand labeled examples. The arbitrary pose, scale and appearance of the persons in a highly cluttered scene makes this problem very challenging. In combination with different lighting condition and arbitrary views from which the images are captured, the challenge is so hard that it is still unsolved. Until now we are the only group which has ever submitted results on this data.

The idea to decompose a visual object into its parts leads back to the early work on pictorial structures by Fischler and Elschlager [52] and was used by Felzenszwalb and



(a) (b) (c)

(d) (e) (f)

Figure 3.7.: Examples for correct and wrong detections of the part-based method by Felzenszwalb et al. [45]. The top row shows correct detections for the classes **(a)** person, **(b)** bike, and **(c)** monitor. Below we show examples in which the method states to detect a **(d)** person, **(e)** bike, and **(f)** monitor.

(a) (b) (c)

(d) (e) (f)

Figure 3.8.: Perhaps the most challenging task in the VOC Challenge [42] is the Person Layout Challenge. The challenge is to detect the person and the position of its head, hands and feet in arbitrary pose and appearance including different scales, partial occlusion and background clutter. The pictures above show hand labeled examples.

Huttenlocher [46] for object recognition. In the last years several methods based on part based models have been presented.

Regarding probabilistic models of spatial part configurations, Gaussian distributions have been proposed in [126]. However, this can only be accurate for restricted set of human poses. Likewise, computationally more convenient tree-structured models can not explicitly model relations between all object parts. As a consequence, they may tend to detect both arms and legs at the same position, for instance, and therefore have been mainly applied to views taken from a similar viewpoint [46, 48]. To overcome this shortcoming, in [46] configurations are sampled from the tree-structured distribution and evaluated by a global Chamfer distance [22]. In other works this problem is ignored or tried to bear down by stronger data terms, which may work in controlled environments, i.e. fixed background and object appearance. Recent work [148, 79, 66] has shown, however, that using additional relations between body parts, in terms of acyclic graphs, can enforce correct configurations, especially in less controlled settings. We report experiments comparing tree-structured and non-tree-structured models in Section 3.4.3.

However, tree-structured models benefit from a small number of parameters, and have recently shown to be extensible to weakly-supervised learning settings [49]. Furthermore, the paper [131] comprehensively elaborates on-line learning in order to adapt a general human model to specific detected object instances in the spatio-temporal context. We make no use of temporal information, since we consider single images as input and no image sequences.

A notable difference to our work concerns the meaning of nodes. Whereas most approaches, e.g. [51, 131, 148, 46], choose body parts as nodes in this work we use body joints. This results in a non-redundant parametrization of the articulated object and smaller domains for the random variables assigned to the nodes.

3.2. Part-Based Object Detection

In this section, we detail the components of our probabilistic representation of object views. After fixing some basic notation, we distinguish discriminative local models of object part appearance, and generative contextual models for the geometry of part configurations. Both components are combined in a probabilistic graphical model. Our model will be applied on 3 different object categories: faces, human body (single and multi view) and human spine in 3D medical image data. The decomposition of the objects into parts is done manually and for the learning of the parameters of the probability distribution we label the position of these parts for several images containing this object. Each part in the image is labeled with its position or as occluded if it is covered by another object or outside the image. This set is divided into a trainings dataset $\mathcal{D}_T$ and a validation dataset $\mathcal{D}_V$. Both sets include a set of images with their assigned ground truth notation.

We use always the same strategy:

i) manually decompose the model into a set of parts

ii) model the appearance terms using $\mathcal{D}_T$

iii) model the geometry of the part configurations using $\mathcal{D}_T$.

iv) combine local models in a probabilistic graphical model using $\mathcal{D}_V$.

v) for new data infer the maximum a-posteriori (MAP) configuration.

In the following we will focus on the steps i)-iii)[1]; the steps iv) and v) will be discussed in Section 3.3 and Chapter 4.

Our model represents the scene by a single object decomposed into a part-based description. Each part of the object is represented by its center position in the image domain. Faces are decomposed into the eyes, nose and mouth corners, humans into head, hands, elbows, etc. and the human spine into inter vertebra disks, see Figure 3.9.

To include this part-based representation in our scene model (Figure 3.1), we add a random variable, which represents the reduced description of the scene by the positions of the parts, see Figure 3.10. If we condition this model on the image I and marginalize over the random variables for the camera parameters c, lighting conditions $l, l_1, \dots l_m$ and the scene $s, o_1, \dots, o_n$, we get a conditional random field $M = (X, Y, G = (V, E))$ with graph structure shown in Figure 3.9. The full connectivity of the graphs is caused firstly by the conditioning on the image and secondly by the marginalization over the scene.

With each node $a \in V$ we associate a predefined part of the model and a random variable X_a. It takes values $x_a \in \mathcal{X}_a$, where $\mathcal{X}_a$ is a finite subset of the image domain. Additionally, $\mathcal{X}_a$ includes a candidate, denoted by $\emptyset$, which indicates the part a is not present in the image domain. The random variable Y represents an image. Since our model is conditioned on Y, the state of Y is fixed and will be further denoted by I.

The pose of an object is than given by the vector $x = (x_a)_{a \in V}$, where x_a encodes the position for the part a of the object. The probability of a pose x given the image I is modeled by the distribution

$$p(x|I) = \exp\left(\sum_{a \in V} \theta_a(x_a) + \sum_{ab \in E} \theta_{ab}(x_a, x_b) - A(\theta)\right), \tag{3.5}$$

with the normalizing log partition function $A(\theta)$ ensuring $\sum_{x \in \mathcal{X}} p(x|I) = 1$. To detect the most probable pose of the object we have to calculate the MAP configuration x^* by maximizing (3.5). We denote this optimization problem as MAP-inference defined by

$$x^* = \arg\max_{x \in \mathcal{X}} p(x|I). \tag{3.6}$$

In Chapter 4 we will discuss how to solve this efficiently. Let us assume at this point that we can calculate the MAP configuration efficiently and concentrate on the modeling aspect.

The formulation of (3.5) can also be transformed into an exponential family representation,

$$p(x|I) = \exp\left(\langle \theta, \phi(x) \rangle - A(\theta)\right). \tag{3.7}$$

With $\mathcal{C} = V \cup E$ the index set is defined as

$$\mathcal{I} = \{(a; i) |\ a \in V,\ i \in \mathcal{X}_a\} \cup \{(ab; ij) |\ ab \in E,\ i \in \mathcal{X}_a,\ j \in \mathcal{X}_b\}, \tag{3.8}$$

and the exponential parameters and sufficient statistics are

$$\theta_{C;i} = \theta_C(i), \qquad C \in \mathcal{C},\ i \in \mathcal{X}_C \tag{3.9}$$

$$\phi_{C;i}(x) = \begin{cases} 1 & \text{if } x_C = i \\ 0 & \text{else} \end{cases}, \qquad C \in \mathcal{C},\ i \in \mathcal{X}_C. \tag{3.10}$$

[1] the steps ii) and iii) reflects a publications together with Bergtholdt et al. [11] and Stefan Schmidt et al. [143]. The part of this approach considering the local feature functions based on the work of Martin Bergthold and Stefan Schmidt.

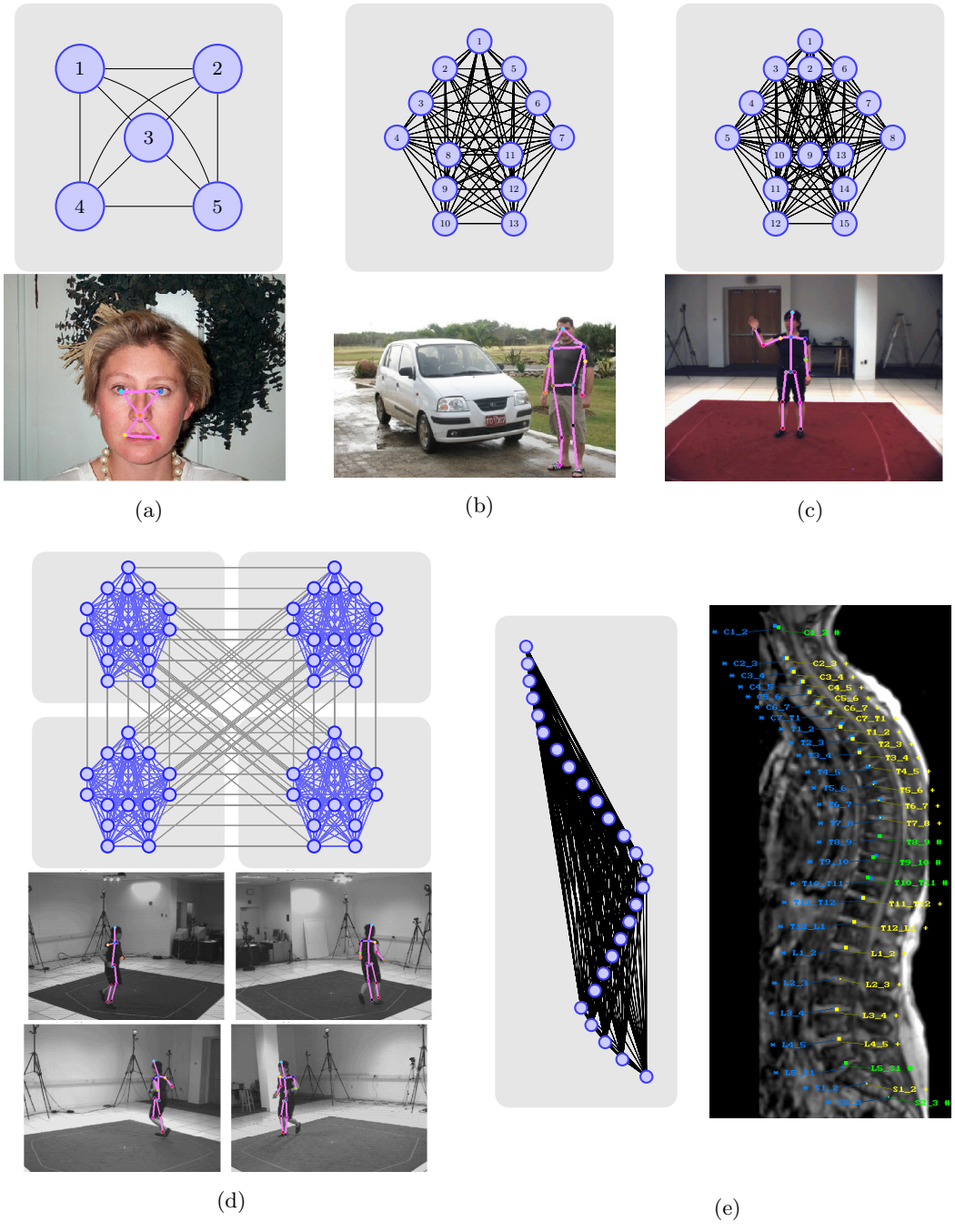


Figure 3.9.: The used datasets are **(a)** face, **(b)** human, **(c)** single view HumanEva, **(d)** multi view HumanEva (here with four cameras), and **(e)** 3D medical image data of the human spine. We show illustrations of the data and the corresponding graph of the CRF. The models in **(b)** and **(c)** differ in the set of parts. In **(c)** we add additional nodes for the middle of breast and hip. All these cases are handled by our approach in a uniform manner.

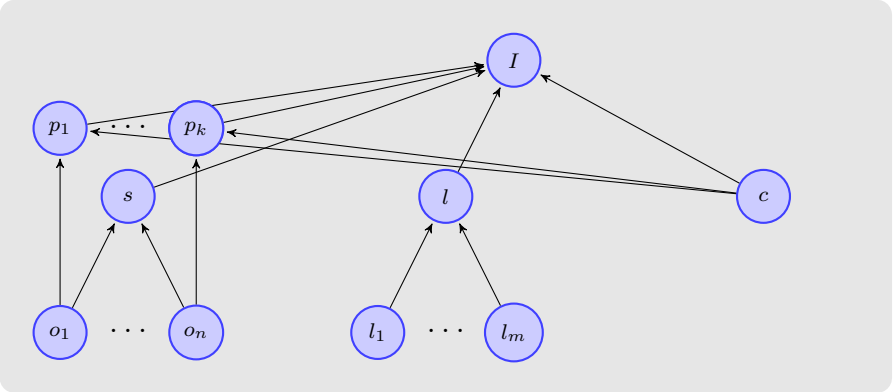


Figure 3.10.: By including variables p_i for the 2D-description of the objects into the model shown in Figure 3.1 we get the model above. Note that the reduced representation of the parts p_i represents only the position of this part in the image domain.

It is important to note that high probability of a configuration x given by $p(x|I)$ does not necessarily indicate that the image I contains an object of the modeled class with the configuration x. Since $p(x|I)$ sums up to 1 the probability of a configuration x depends also on the other possible configurations in the image. For example in an image with many persons the conditioned probability of a single person will be lower than in an image with a single person.

As an absolute, i.e. non-normalized, measure we will use the energy of a configuration x defined by

$$J(x) = -\langle \theta, \phi(x) \rangle. \tag{3.11}$$

Since the log partition function $A(\theta)$ depends not on x and the exponential function is strictly increasing, minimizing the energy $J(x)$ is equivalent to maximizing (3.5), i.e. MAP-inference can also be achieved by minimizing $J(x)$. The functional dependence of θ on observed image data I will be detailed below.

Exponential Parameter Function To cope with the large variability of the image data and the complex dependencies therein, we transform the image information into a set of *scalar-valued* feature functions. Each exponential parameter function $\theta_C(x_C)$ can be written as a weighted sum of the individual contributions:

$$\theta_C(x_C) = \sum_{k \in \mathcal{F}_C} \lambda_{C;k} f_{C;k}(x_C) \tag{3.12}$$

with model weights $\lambda_{C;k}$, feature functions $f_{C;k}(x_C)$, $C \in \mathcal{C}$, $k \in \mathcal{F}_C$, where the function types are

$$\mathcal{F}_C := \{\text{appearance, occlusion}\} \qquad \text{if } C \in V\ , \tag{3.13}$$

$$\mathcal{F}_C := \left\{ \begin{array}{l} \text{appearance, length, orientation,} \\ \text{epipolar, occlusion} \end{array} \right\} \qquad \text{if } C \in E\ . \tag{3.14}$$

We will use *app, len, ori, epi* and *occ* as shorthand for *appearance, length, orientation, epipolar* and *occlusion.*

Unary features or node features may depend on one variable x_a and the image I; Pairwise or edge features may depend on two variables (x_a, x_b) and the image I. Input features to the feature functions are: SIFT-features [113], color features, edge length, edge orientation, and epipolar residuals. All features have the property to depend at most on two variables, which allows us to compute exhaustively all unary terms and a sufficiently large set of edge terms. The features are described in detail in Sections 3.2.1, 3.2.2, and 3.2.3. Each feature function reduces a feature vector to a scalar.

3.2.1. Feature Functions for Object Appearance

Input features. The input feature vectors are computed from a window at each site. Figure 3.11 shows some example windows. For the 2D datasets (Human, HumanEva, Face, see Section 3.4) we compute

- SIFT features [113] with 8×8 spatial and 10 orientation bins at a fixed scale. Concatenation yields feature vectors of dimension $8 \times 8 \times 10 = 640$.
- Color features with 4×4 spatial bins in the L*a*b color space. Here each bin contains the average color of pixels falling inside it. Concatenation yields feature vectors of dimension $4 \times 4 \times 3 = 48$.

The same two features computed in windows aligned along the edge that connects two object parts have been used for pairwise appearance features, see Figure 3.11. For edges between physically connected body parts these pairwise appearance features are in fact "limb-like" as by construction they are invariant to translation, rotation and foreshortening.

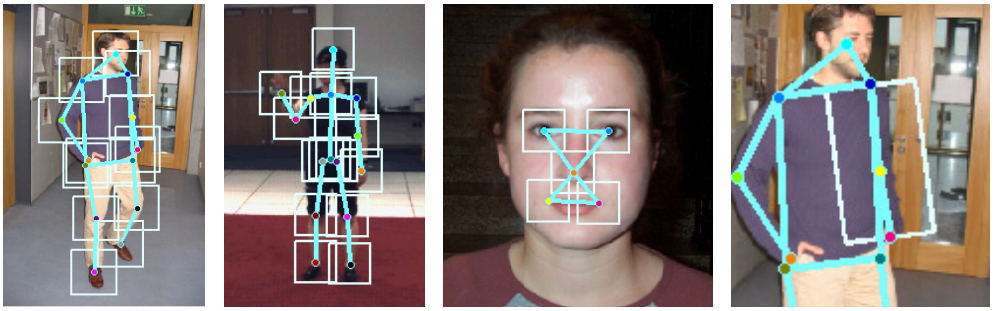
For the 3D dataset (Spine) we compute

- Intensity features with $15 \times 15 \times 15$ spatial bins which correspond one-to-one to the 3D window size of the input sub-volume. Concatenation yields feature vectors of dimension $15 \times 15 \times 15 = 3375$.

No pairwise appearance features have been used for this dataset.

Randomized classification trees. We use randomized classification trees [58] to compute scalar features given feature vectors. They allow for fast evaluation, are able to cope with large training data, and can be used to detect points of interest [110]. Randomized classification trees divide the feature space into arbitrary regions and build a statistic of the training features in each region. This process is repeated using many trees, e.g. about 100 in our case, and combining the individual statistics by averaging the class counts. In this way the effect of the hard region boundaries is reduced. Training such a classifier amounts to creating a set of decision trees and collecting the statistics of the training data under the trees classifications. The branching tests at tree-nodes that divide the feature space are chosen at random from a set of very simple tests, each involving only one or two feature space dimensions.

In particular, we have adopted three types of tests for branching



(a) (b) (c) (d)

Figure 3.11.: Ground truth configurations for the **(a)** Human, **(b)** HumanEva and **(c)** Face datasets with corresponding (uniform) window sizes for local appearance computations. **(d)** shows a window for computation of pairwise appearance between left-shoulder and left-hand. Note the cyan-edges are only for visualization, and features are generally computed between *all* pairs.

- T1: At each node of the tree, two dimensions dim_1 and dim_2 of an input feature vector v are chosen at random and their respective values compared. If $v(\text{dim}_1) < v(\text{dim}_2)$, we descend the left branch of the node, otherwise we descend the right branch.
- T2: At each node of the tree, one dimension dim of the input features and a threshold value val in the range of $v(\text{dim})$ are chosen randomly. If $v(\text{dim}) \leq val$, we descend the left branch of the node, otherwise the right branch.
- T3: This test is only used for pairs of input features. At each node, two dimensions dim_1 and dim_2 are chosen randomly. For two input features v_1 and v_2, if $v_1(\text{dim}_1) \leq v_2(\text{dim}_2)$, we descend the left branch, otherwise the right.

Among the three tree types we selected the best performing one as feature function generator for the input features. These are T1 for SIFT features and 3D Spine intensity features, T2 for color features, and T3 for pairs of color features for color similarity.

Building the statistics for the trees was stopped when the number of training samples falling into a leaf was smaller than a given threshold (a value of 10 was used throughout the experiments), or if it only contained samples of a single part. This method seemed favourable compared to others defining a maximum depth of the trees, as in our case the tree depth automatically adapts to the number of training samples.

The overall performance and robustness against noise results from aggregation of the statistics over a large number of such tests that are distributed over the ensemble of decision trees. For Human, HumanEva and Face, we used 100 trees for the unary features and 70 for the pairwise features; for the Spine 150 trees were used.

The class-specific scalar feature value is obtained by classifying a candidate feature vector, i.e. at each tree-node we descend into the corresponding sub-tree until we reach a leaf. The number of all training samples in the leafs, corresponding to the class and accumulated over all trees, divided by the number of all training samples accumulated over the respective leafs, yields the scalar feature. The final feature function value is obtained after a non-linear calibration method described in [11].

To reduce the number of functions for fitting and weight-parameters $\lambda_{C;k}$, we averaged the individual classifier scores of the SIFT and color classifiers separately for each vertex and each edge, by taking their geometric mean in order to obtain a combined classifier score. We calibrate these scores and refer to the resulting functions as "appearance probabilities" $p_{C;\text{app}}(C|x_C, I)$ for class $C \in \mathcal{C}$ at site location x_C for the image I. We have found that this yields also slightly better results with respect to classification than first calibrating to each feature type and taking the geometric mean afterwards.

Finally, in terms of the the exponential family (3.7) and (3.12), the logarithm of the appearance probability yields the feature function

$$f_{C;\text{app}}(x_C) := \log p_{C;\text{app}}(C|x_C, I), \qquad C \in \mathcal{C} \tag{3.15}$$

3.2.2. Feature Functions for Object Shape

The feature functions for object shape are derived from simple 1D histograms. As input features we used the Euclidean distance between pairs of sites constituting an edge in the graph, and the absolute edge orientation. Unary terms, e.g. absolute part locations, were not used. In other words, we only model object shape, not absolute location. Whereas these features are thus invariant to object translation, the edge-length feature is not invariant to changes in scale and the orientation feature is not invariant to in-plane rotations. For training of the geometric feature functions we normalized the scale by computing $r_{ab} = \frac{\mu_{ab}}{l_{ab}}$ for all available edges $ab \in E$ of the observed object, where μ_{ab} denotes a normalized edge length and l_{ab} is the observed length. We assume that variations in r_{ab} are due to global scale and foreshortening. The latter one causes r_{ab} to be overestimated as the observed l_{ab} is shorter than the true edge-length. To account for foreshortening we assume that at least one edge is not foreshortened so that l_{ab} is the true length under global scale of that edge. Thus taking the minimum of r_{ab} for all $ab \in E$ gives the global scale normalization factor r.

Clearly though, the effects of foreshortening will still hamper the length features. For inference on the test images we treat global object scale as a latent variable for the length features, i.e. the features are computed after normalization with the hidden/unknown scale parameter r, that has to be inferred.

In contrast we ignore the dependence of the orientation features to in-plane rotations for the following reasons: Our particular objects usually have one predominant orientation in images. Where this assumption does not hold, e.g. standing vs. lying humans, this will be reflected in the histograms since these do allow for multiple modes. Clearly configurations that do not correspond to major modes will be hard to detect with this approach.

We refer to the histogram outputs as length probability $p_{C;\text{len}}(\|x_{C_1} - x_{C_2}\||r)$ and orientation probability $p_{C;\text{ori}}(\angle(x_{C_1} - x_{C_2}))$, where $C \in E$, x_{C_1} and x_{C_2} denotes the two image sites corresponding to edge C and $\angle(u)$ the absolute orientation of the vector u. The feature functions for the energy formulation

$$f_{C;\text{len}}(x_C) := \log p_{C;\text{len}}(\|x_{C_1} - x_{C_2}\||r), \qquad C \in E \tag{3.16}$$

$$f_{C;\text{ori}}(x_C) := \log p_{C;\text{ori}}(\angle(x_{C_1} - x_{C_2})), \qquad C \in E \tag{3.17}$$

are again the logarithm of the histogram output.

3.2.3. Feature Functions for Epipolar Constraints

For the HumanEva dataset, up to 7 images from calibrated cameras were taken simultaneously from different directions. We made use of this additional information by combining the configurations of all available cameras into a single model. Each configuration in the images must satisfy additional pairwise constraints given by the epipolar geometry. For an image pair I_1, I_2 and two corresponding points u, v, i.e. u and v are projections of the same 3D world point, the epipolar constraint

$$v^\top F_{12}\, u = 0$$

must be satisfied, where F_{12} is the fundamental matrix [70] of the image pair.

We use for each view the original graph as depicted in Figure 3.9 (c), augmented by edges between all parts with the same meaning (e.g. head) in each combination of image pairs. Therefor, the corresponding model graph is not fully connected in this case as shown in Figure 3.9 (d). The input features for the additional edges are the algebraic residuals of the epipolar constraint $|x_{a,i}^\top F_{ij} x_{a,j}|$ for each part a and image pairs I_i, I_j. We compute 1D histograms of these features, analogously to the object shape features and refer to them as epipolar probability $p_{C;\mathrm{epi}}(|x_{a,i}^\top F_{ij} x_{a,j}|)$. With a slight abuse of notation:

$$f_{C;\mathrm{epi}}(x_C) := \log p_{C;\mathrm{epi}}(|x_{a,i}^\top F_{ij} x_{a,j}|), \qquad \begin{array}{r} C = ((a,i),(a,j)), \\ (a,i),(a,j) \in V,\ i \neq j. \end{array} \tag{3.18}$$

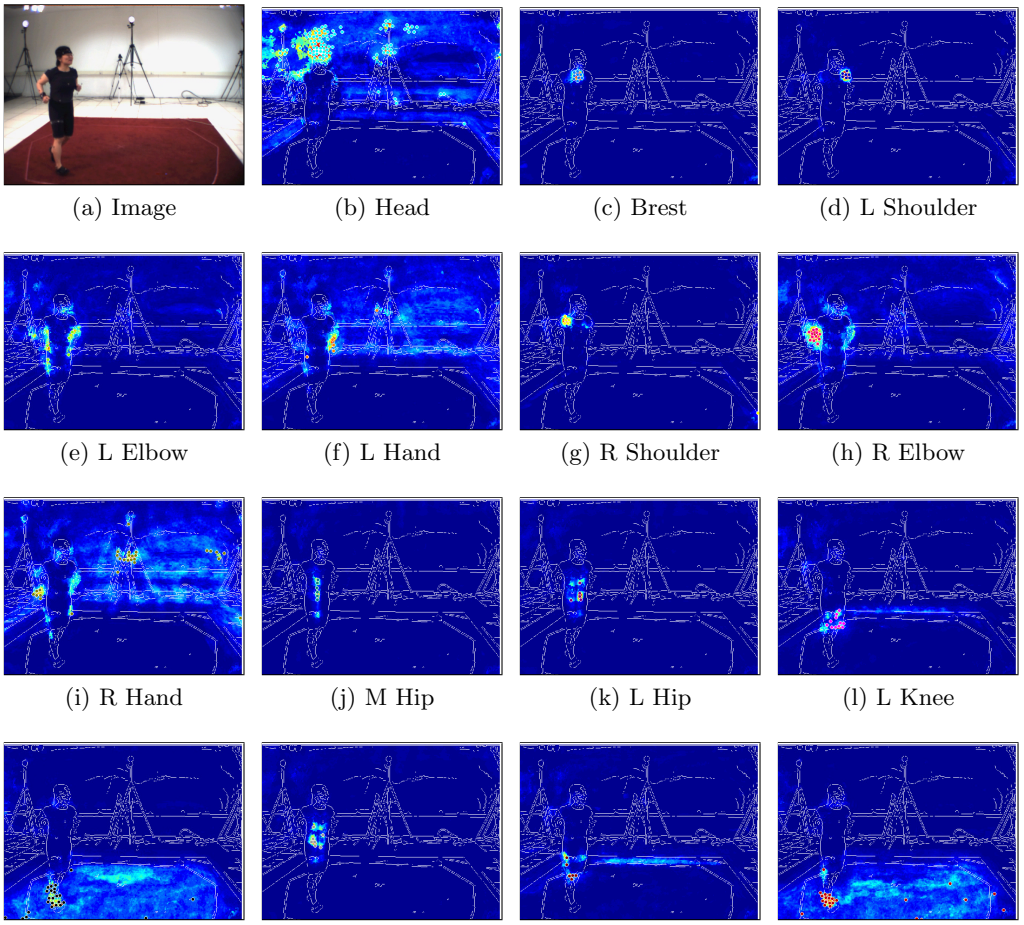
3.2.4. Problem Domain and Missing Parts

When we building a model instance corresponding to a given image, we proceed in a bottom-up manner. In each step we make use of previous computations to prune the problem to manageable size in terms of computational effort and memory. For a given test-image, we proceed as follows:

1. Compute the appearance probability $p_{a;\mathrm{app}}(a|x_a, I)$ for all parts $a \in V$ and all corresponding image sites x_a in the whole image domain.
2. For a fixed, per-part threshold T_a, sample a set of candidate part locations $\mathcal{X}_a$ by including all candidates x_a in the image domain with $p_{a;\mathrm{app}}(a|x_a, I) > T_a$. Additionally, we use non-maxima suppression to avoid closely located candidates.
3. For each edge $C \in E$ compute the lengths $p_{C;\mathrm{len}}(\|x_{C_1} - x_{C_2}\|r)$ and orientation probabilities $p_{C;\mathrm{ori}}(\angle(x_{C_1} - x_{C_2}))$ for all candidates. For the object scale r we have usually used 5 discrete settings. We employ hard thresholds for the edge-length, such that if it is smaller or larger than any observed length in the training set, the corresponding probability is set to zero.
4. For all edge candidates for which the length and orientation probabilities a non zero, compute the appearance probability $p_{C;\mathrm{app}}(C|x_C, I)$, $C \in E$.

Figure 3.12 shows examples of appearance probability maps and candidate samples.

By proceeding in a bottom-up manner, only a relevant subset of positions are considered for the object parts. Furthermore, the model has a locality property, as candidates with large distance get an edge probability of zero, which is used to speed up subsequent inference. The thresholds T_a, $a \in V$ can be set by the user. In order to get a good compromise



(a) Image (b) Head (c) Brest (d) L Shoulder

(e) L Elbow (f) L Hand (g) R Shoulder (h) R Elbow

(i) R Hand (j) M Hip (k) L Hip (l) L Knee

(m) L Foot (n) R Hip (o) R Knee (p) R Foot

Figure 3.12.: Appearance probability maps $p_{a;\mathrm{app}(a|x_a,I)}$ and candidate samples $\mathcal{X}_a$ for an image of the HumanEva dataset. Dark blue indicates low probabilities and green and yellow high probabilities. Selected candidates are marked by dots.

between missing detections and computational complexity, we have chosen the thresholds at the operating point where the individual classifiers maximize the F-measure [44] on all training features.

Missing parts. There are two natural reasons for missing parts. Firstly, the part can be occluded by another part or object. Secondly, the part may not be contained in the image. Moreover, by employing the thresholds T_a, some of the parts may be missed during sampling. Therefore, we include a special candidate $\emptyset$ for each part. "Special" means that $\emptyset$ has no location, hence feature functions cannot be computed in the usual way.

We define all feature functions to return 0 for this special candidate, only the feature function for occlusion is 1 if at least one site is missing and otherwise 0, i.e. $f_{C;k}(\emptyset) = 0$ if $k \neq \mathrm{occ}$ and

$$f_{C;\mathrm{occ}}(x_C) := \begin{cases} 1 & \text{if at least for one } a \in C: \ x_a = \emptyset \\ 0 & \text{else} \end{cases} \tag{3.19}$$

The value of the exponential parameter for a missing or occluded setting is only defined by

the weight parameter $\lambda_{C;\text{occ}}$. How to selected this parameter is discussed in Section 3.3.1, in cases of learning it is just treated as all other parameters.

In view of alternative approaches [156, 33, 31] that recreate a small number of candidates after few iterations of belief propagation, advantages of our model include independence of the inference method (any technique can be used), and feature functions for new candidates need not to be computed in each step. Natural occlusion, however, has still to be modeled by an extra candidate.

3.3. Learning

Learning the parameters of probabilistic models from labeled data, known as supervised learning, or unlabeled data, known as unsupervised learning, is a hot topic in current research. While for simple models with few parameters it might be possible to find good parameters by trial and error, this is not manageable for larger models. While unsupervised learning is not within the scope of our work, we consider a simple approach which optimizes the parameters with respect to some probabilistic measure, see Section 3.3.2 and compare this to hand tuned parameters, see Section 3.3.1.

When we learn parameters from data with expert ground truth, we have to deal with uncertainties in the expertise as well as limited amount of data. The latter one can cause that the model overfits the training data, i.e. the model describes also the specifics or noise in the training data and perform significant worse on other instances. Overfitting generally occurs when a model has too many degrees of freedom, in relation to the amount of data available.

We will not and can not give a complete overview of learning approaches in context with graphical models and introduce only one method used for our model. For an overview over learning in MRF and CRF we refer to [56, 71, 102]. The important point is, that learning usually uses inference methods as subroutines, as in our case.

3.3.1. Heuristics Parameter Estimation

In additional to the feature functions $f_{C;k}(x_C)$ we have to learn the model parameters $\lambda_{C;k}$ for the computation of $\theta_C(x_C)$, which has been defined in (3.12) as

$$\theta_C(x_C) = \sum_{k \in \mathcal{F}_C} \lambda_{C;k} f_{C;k}(x_C) \ .$$

For each node and edge, and for each corresponding feature, a model parameter has to be estimated. The reasoning for computing an initial guess is as follows:

Initially neglecting all structural information given by the edge terms, we can conceive a detector for recognizing an object by detecting its parts individually. Assuming that all part detectors are independent, the overall probability is the product of the individual probabilities

$$p_{V,\text{app}}(V|x, I) \propto \prod_{a \in V} p_{a;\text{app}}(a|x_a, I) \ .$$

We initialize all λ parameters corresponding to vertex appearance features with 1 in this case. Including the edge appearance probabilities gives us a complementary view of the

same probabilistic event. So we could set $p_{E,\text{app}}(E|x,I) \propto \prod_{C\in E} p_{C;\text{app}}(C|x_C;I)$. The number of edges is much higher than the number of nodes, however, and edge features correspond to image regions which may overlap by construction, i.e. the independence assumption of the individual classifiers does not hold. Therefore, if their individual contribution is comparable to the part probabilities, the overall final probability will be much lower than the one above, using parts alone. To account for this, we combine the probabilities in a "products of expert" model [75],

$$p_{E,\text{app}}(E|x,I) \propto \prod_{C\in E} p\left(C;\text{app}(C|x_c,I)\right)^{\lambda_{C;\text{app}}}.$$

Expecting edge appearance to be of similar quality as the one for nodes, we would set

$$\lambda_{C;\text{app}} := \frac{|V|}{|E|},\ \forall c \in E,$$

i.e. $\frac{2}{|V|-1}$ for a fully connected graph. Assuming further that length and orientation probabilities are equally informative, their respective λ parameters would be initialized in the same way.

So far we have consider each feature independently. To combine those, we assume that each type of feature (part appearance, edge appearance, length, orientation) gives rise to an *expert*, and the overall probability is again their combination using the "products of expert" model, where we weighted their contribution based on intuition as $\xi_{V,\text{app}} = 0.5$ for the node appearance, $\xi_{E,\text{app}} = 0.25$ for edge appearance, $\xi_{E,\text{len}} = 0.125$ for length, and $\xi_{E,\text{ori}} = 0.125$ for orientation. In conclusion we define the initial values of the λ parameters to be

$$\lambda_{c;k} := \begin{cases} \xi_{\text{V,k}}, & \text{if } c \in V \\ \frac{2}{(|V|-1)}\,\xi_{\text{E,k}}, & \text{if } c \in E \end{cases}. \tag{3.20}$$

If we use additional epipolar features we set the parameter $\lambda_{C;\text{epi}}$ such that the impact of epipolar edges is equivalent to the impact of edges inside images or the impact of nodes.

The remaining parameters $\lambda_{C;\text{occ}}$ defines the values of the exponential parameter for the candidate that model occlusion. Remember that $f_{C,\text{occ}}(x_C)$ is 1 if and only if x_C is occluded or missing and otherwise 0. Since calculating $\lambda_{C;\text{occ}}$ by marginalizing is intractable, we propose the following, efficient method:
First, a common approximation is to search for the maximally likely missing part. Note that the highest attainable appearance probability for the missing part is exactly the threshold T_a.

Next, we define the edge probabilities for the missing candidate. Assuming that the a miss is only caused by the local appearance probability lying below the threshold, but that pairwise edge probabilities would not be affected by this "failure" to recognize the part, we argue that the true part would lead to typical edge probabilities. Thus we define the edge probabilities by their typical values. We have chosen the mean of each of the appearance, length, and orientation probabilities for the three types of edge terms using a validation dataset $\mathcal{D}_V$ with x_c denoting the true locations of the parts in the respective image.

One might argue that the miss of the part may originate from other causes, in particular from occluding objects or self-occlusion, for which the part appearance probability at the true location will certainly be below T_a, as well as its edge-appearance counterparts. So

the estimates serve as an optimistic guess, and for experiments that rely on this heuristic, we have introduced the weight parameter $\gamma \leq 1$ by which we multiply the appearance probabilities for missing parts and edges.

So we set

$$\lambda_{a;\text{occ}} := \log\left(\gamma\ T_a\right),\ a \in V \tag{3.21}$$

and

$$\begin{aligned}\lambda_{C;\text{occ}} := \log(\gamma\ \lambda_{C;\text{app}}\ \frac{1}{|\mathcal{D}_V|}\sum_{d=1}^{|\mathcal{D}_V|} p_{C;\text{app}}(C|x_C^d, I^d)) \\ + \log(\lambda_{C;\text{len}}\ \frac{1}{|\mathcal{D}_V|}\sum_{d=1}^{|\mathcal{D}_V|} p_{C;\text{len}}(\|x_{C_1}^d - x_{C_2}^d\|)) \\ + \log(\lambda_{C;\text{ori}}\ \frac{1}{|\mathcal{D}_V|}\sum_{d=1}^{|\mathcal{D}_V|} p_{C;\text{ori}}(\angle x_{C_1}^d - x_{C_2}^d)) \\ + \log(\lambda_{C;\text{epi}}\ \frac{1}{|\mathcal{D}_V|}\sum_{d=1}^{|\mathcal{D}_V|} p_{C;\text{occ}}(|{x_{C_1}^d}^\top F_C x_{C_2}^d|)\ ) \quad \forall C \in E\end{aligned} \tag{3.22}$$

where we denote with x^d and I^d the ground truth and image of the d-th dataset in the validation data $\mathcal{D}_V$.

We have found that the above heuristic already gives quite reasonable results. Where ever model complexity or an insufficient number of training data did not allow for maximum likelihood learning as proposed in Section 3.3.1, we successfully used this method instead.

However, in practice, the assumptions made here do not strictly hold since image patches may overlap and features are in general not equally informative. Optimization of the λ parameters can be done in the conditional random field (CRF) framework by maximizing the log-likelihood of the ground truth for a set of training samples [104], as described next.

3.3.2. Maximum Likelihood Learning

In order to estimate the parameters λ, we need independent and identically distributed set of data including a set of images $\{I^1, \ldots, I^D\}$ with known hand labeled ground truth $\{\overline{x}^1, \ldots, \overline{x}^D\}$. We can not use the original trainings set $\mathcal{D}_T$ for learning λ, because the feature functions $f_{C;k}(x_C)$ are learned using $\mathcal{D}_T$ and the performance of the different feature functions would not reflect the performance on novel data. Instead, we will use the validation set $\mathcal{D}_V$ for this parameter estimation. One can now argue that $\mathcal{D}_V$ was already used for the calibration of the features and therefor learning may be biased. Due to the limited amount of labeled data, we decide to ignore this and use $\mathcal{D}_V$ for the training of λ. The alternative would be to split the set of all labeled images into a feature-training-, feature-validation- and parameter-learning-set. The size of the single sets would be smaller. This would boost the problem of overfitting for learning in all stages.

Given the validation set $\mathcal{D}_V = \{(I^d, \overline{x}^d) | d = 1, \ldots, D\}$, where I^d denotes the image and $\overline{x}^d$ the hand labeled ground truth, we follow the procedure in the previous sections and construct for each image I^d a CRF with a random variable X^d taking values $x^d \in \mathcal{X}^{\mathcal{D}}$. For each image I^d the conditional distribution $p(x^d|I^d)$ is given by the exponential parameter $\theta^d(\lambda)$ which depend from the unknown vector λ, which is identical for all images.

To estimate the parameter λ it is common to maximize the likelihood function

$$\begin{aligned} L(\lambda|\{I^d\},\{\overline{x}^d\}) &= \prod_{d=1}^{D} p(\overline{x}^d|\theta^d(\lambda)) \\ &= \prod_{d=1}^{D} \left(\exp\left(\langle\theta^d(\lambda),\phi(\overline{x}^d)\rangle - A(\theta^d(\lambda))\right)\right). \end{aligned} \tag{3.23}$$

While the optimization of equation (3.23), or equivalently its logarithm, is the commonly applied approach and a number of algorithms for exact and approximate solutions have been proposed, see e.g. [158, 129, 166, 111], there are two shortcomings of this formulation. First, in our case the number of parameters is large compared to the size of the data set used for learning[2]. This may cause the learning procedure to overfit to the validation data.

Secondly, the likelihood function does not variate smoothly in vicinity of the ground truth. In particular, when labeling ground truth configurations, a user is confronted with the difficult decision to label a single point on a joint or body part in arbitrary views, as well as deciding if a part is still visible or should be labeled as occluded. Without further proof, we feel that different users will most likely label different configurations for the same image. To tackle both problems at the same time, we relax (3.23) by also including configurations that are similar to the "ground truth" as positive examples. For this purpose we calculate weights for the part candidates $x_a^d \in \mathcal{X}_a^d$ decreasing with their distance to their ground truth location $\overline{x}_a^d$ until reaching a certain distance threshold D_a

$$\overline{w}_a^d(x_a^d) = \begin{cases} 1 - \frac{|x_a^d - \overline{x}_a^d|}{2D_a} & , \text{ if } |x_a^d - \overline{x}_a^d| < D_a \text{ ,} \\ \delta & , \text{ if } x_a = \emptyset \\ 0 & , \text{ otherwise,} \end{cases} \tag{3.24}$$

where $\delta \in [0, 0.5]$ reflects the possibility for hiding a part. See Figure 3.13 for a visualization of the distance thresholds used.

We normalize this weights such that they sum up to one and define a probability distribution, which cluster around the ground truth position. We define a local distribution for each node

$$w_a^d(x_a^d) = \frac{\overline{w}_a^d(x_a^d)}{\sum_{x_a \in \mathcal{X}_a} \overline{w}_a^d(x_a)},$$

such that

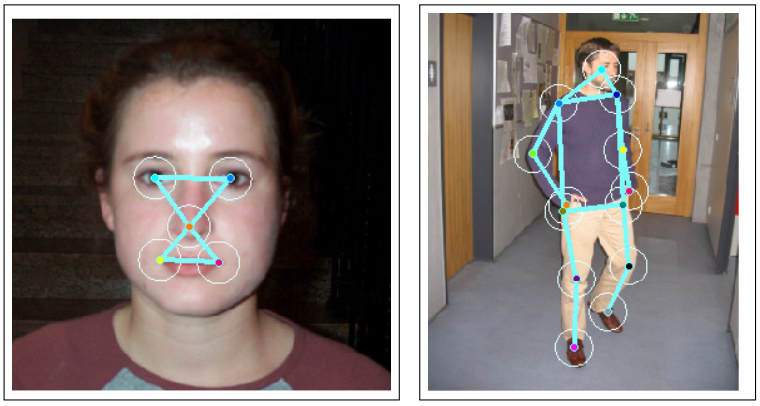
$$\forall a \in V : \sum_{x_a^d \in \mathcal{X}_a^d} w_a^d(x_a^d) = 1$$

and a global distribution for full configurations

$$W^d(x^d) := \prod_{a \in V} w_a^d(x_a^d), \quad \sum_{x^d \in \mathcal{X}^d} W^d(x^d) = 1.$$

The corresponding smoothed log-likelihood function behaves more gently in the neighborhood of the labeled ground truth, and is maximized using gradient ascent.

[2] For the human data we have 338 parameters and 429 parameters when using SIFT and color features as independent feature functions and only 717 trainings examples in the validation set.



(a) (b)

Figure 3.13.: Distance thresholds used for the **(a)** Face and **(b)** Human datasets both for learning and evaluation. Circles visualize the maximum distance to the manual ground truth location for a part to be considered as a positive hit.

We denote this smoothed log-likelihood function by $l(\lambda)$ as a shorthand for $l(\lambda|\{I^d\},\{\overline{x}^d\},\{\mathcal{X}^d\},W)$, defined by

$$\begin{aligned} l(\lambda) &= \sum_d \sum_{x^d \in \mathcal{X}^d} W^d(x^d) \cdot \ln p(x^d|\theta^d(\lambda)) \\ &= \sum_d \Big(\sum_{x^d \in \mathcal{X}^d} W^d(x^d) \cdot \big\langle \theta^d(\lambda), \phi(x^d) \big\rangle \Big) - A\big(\theta^d(\lambda)\big) \end{aligned} \tag{3.25}$$

In contrast to (3.23), where each ground truth configuration $\overline{x}^d$ is used only once, there is now a weighted summation over the configuration space $\mathcal{X}^d$. We have used the output after candidate sampling (c.f. Sec. 3.2.4) as configuration space $\mathcal{X}^d$, thus including the effects of candidate misses due to suboptimal classification.

However, the impact of the derived formulas is not clear at the first sight. Thus, we will consider the smoothed log-likelihood in terms of its statistic interpretation next. The smoothed log-likelihood function $l(\lambda)$ is identical to the Kullback-Leibler divergence $D_{KL}(W||p)$ [99] up to a constant and facor -1 between a desired distribution $W^d(x^d)$ and the estimates given by the model $p(x^d|\theta^d(\lambda))$.

$$\begin{aligned} -D_{KL}(W||p) &= -\sum_d \sum_{x^d \in \mathcal{X}^d} W^d(x^d) \cdot \ln \frac{W^d(x^d)}{p(x^d|\theta^d(\lambda))} \\ &= -\sum_d \sum_{x^d \in \mathcal{X}^d} W^d(x^d) \cdot \ln W^d(x^d) + \sum_d \sum_{x^d \in \mathcal{X}^d} W^d(x^d) \cdot \ln p(x^d|\theta^d(\lambda)) \\ &= -\text{ const} + \sum_d \sum_{x^d \in \mathcal{X}^d} W^d(x^d) \cdot \ln p(x^d|\theta^d(\lambda)) \\ &= -\text{ const} + l(\lambda)\,. \end{aligned} \tag{3.26}$$

The Kullback-Leibler divergence is a measure of the difference between the two probability distributions W and p. Thus maximizing of the smoothed log-likelihood results in parameters λ that force $p(x|I)$ to be "similar" to our artificial "ground truth distribution" $W(x)$.

Algorithm 3.1 Learn parameter λ

$\lambda^1 =$ Heuristic parameters
for $n = 1, \ldots, N$ **do**
  for $d = 1, \ldots, D$ **do**
    $\forall C \in \mathcal{C},\ x_C^d \in \mathcal{X}_C^d$ compute $b_C^d(x_C^d) \approx \mathbb{E}_{p(x^d|\theta^d(\lambda))}\left[\phi_{C;x_C^d}(x^d)\right]$
  end for
  for $C \in \mathcal{C}$ **do**
    for $k \in \mathcal{F}_C$ **do**

$$\lambda_{C;k}^{n+1} \leftarrow \lambda_{C;k}^{k} + \eta \sum_d \sum_{x_C^d \in \mathcal{X}_C^d} \left[\left(\left(\prod_{a \in C} w_a^d(x_a^d) \right) - b_C^d(x_C^d) \right) f_{C;k}^d(x_c^d) \right]$$

    end for
  end for
end for

Since the log partition function $A(\theta)$ is convex in θ and λ, $l(\lambda)$ is concave with respect to λ. The gradients are computed by the partial derivatives as:

$$\frac{\partial}{\partial \lambda_{C;k}} \left\langle \theta^d(\lambda), \phi(x^d) \right\rangle = f_{C;k}^d(x_C) \tag{3.27}$$

$$\frac{\partial}{\partial \lambda_{C;k}} A(\theta^d(\lambda)) = \sum_{x_C^d \in \mathcal{X}_C^d} \mathbb{E}_{p(x^d|\theta^d(\lambda))}\left[\phi_{C;x_C^d}(x^d)\right] f_{C;k}^d(x_C^d) \tag{3.28}$$

$$\begin{aligned} \frac{\partial}{\partial \lambda_{C;k}} l(\lambda) = \sum_d \Big(& \sum_{x_C^d \in \mathcal{X}_C^d} \left(\prod_{a \in C} w_a^d(x_a) \right) f_{C;k}^d(x_C) \\ & - \sum_{x_C^d \in \mathcal{X}_C^d} \mathbb{E}_{p(x^d|\theta^d(\lambda))}\left[\phi_{C;x_C^d}(x^d)\right] f_{C;k}^d(x_C^d) \Big). \end{aligned} \tag{3.29}$$

Note that, since the feature functions $f(x)$ also depend on the image I, i.e. $f(x)$ is a shorthand for $f(x|I)$, the parameter θ is implicitly dependent on I. Consequently, the distribution $p(x|\theta(\lambda))$ shadows the underlying distribution $p(x|I, \lambda)$. The mean parameter $\mathbb{E}\left[\phi_{C;x_C}(x)\right]$ are calculated with respect to the image I and the current value of λ.

The computation of $\mathbb{E}\left[\phi_{C;x_C}(x)\right]$ is known to be difficult for general graphs. However, we can obtain good approximations using e.g. loopy belief propagation (LBP) for fixed λ. We will discuss the approximate calculations in Chapter 4. Using this approximate gradient we perform a fixed number of gradient ascent steps to optimize the parameters. The value of the step size parameter was chosen sufficient small ($\eta = 0.01$).

The pseudo code of this algorithm is given in Algorithm 3.1. In the next section we will compare results to the heuristically estimated parameters when evaluating the models.

3.4. Evaluation of the Model

When we measure the performance of a model we will do this in different ways. First we measure the performance of the local classifier, because a good local detection of the parts is essential for a good overall performance. Second we evaluate the performance of the

model with a problem specific measure, e.g. evaluate the distance between ground truth and detected position of the parts.

Classifier performance. To estimate the performance of the part classifiers, we used several measures that are common in literature. Here we will consider only two of them:

- Precision recall curves (PR) and area under the curve (APR)
- Confusion matrix

We selected these because they are plain and meaningful. For exhaustive discussion we refer to [11, 143].

To measure the performance of the classifiers we extract the feature vectors belonging to the appearance terms of each part and the feature vectors belonging to background for the test dataset. Then we evaluate if this feature vectors are classified correctly.

Given the set of feature vector $u \in U^+$ belonging to class A, e.g. head, and the set feature vector $u \in U^-$ belonging not to class A, the precision of a classifier is defined as the number of vectors $u \in U^+$ which classified as class A divided by the total number of vectors $u \in U^+ \cup U^-$ classified as class A. The recall is defined as the number of vectors $u \in U^+$ which classified as class A divided by the total number of elements in U^+. In other words, the precision is the percentage of the elements classified as class A that are belonging to class A and the recall is the percentage of elements in U^+ which are detected by the classifier as class A.

The values for precision and recall for different classifier threshold are illustrated by the precision recall curves (PR). A single measure for the quality of a classifier is the area under the curve (APR), which would be 1 for a perfect classifier.

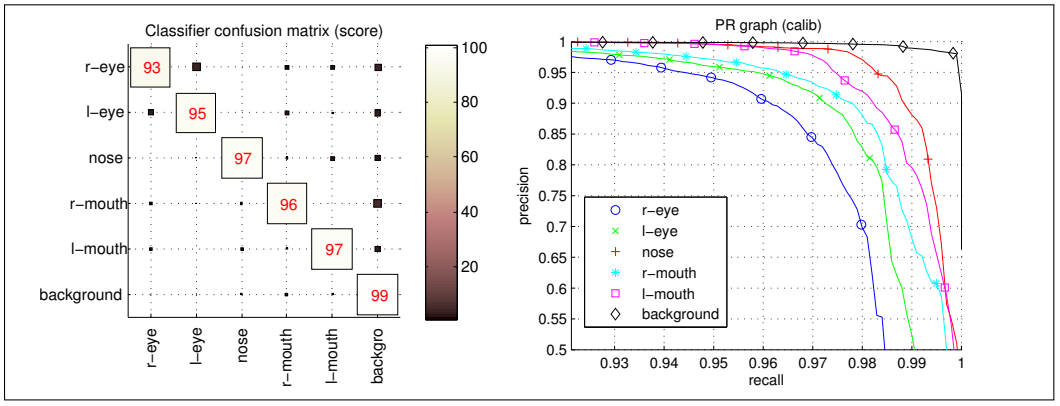
The confusion matrix reflects the outcome of classification. Each row gives the instances of an actual class and each column the instances of the prediction. We normalize the confusion matrix by its row-sums, i.e. each row shows how many percent of instances of that respective class have been predicted as any of the classes. The diagonal of the confusion matrix gives the accuracy for each class. Cohen's κ value [28] is a summary measure for a confusion matrix and can be interpreted as the level of agreement between truth and prediction where 1 is total agreement.

The confusion matrices and precision recall curves of the classifiers are shown in Figure 3.14, for discussion see the text to the model evaluation.

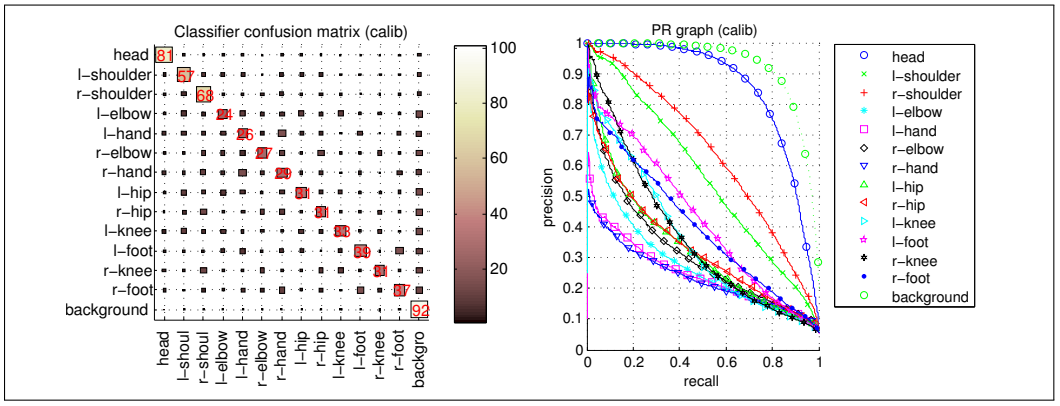
Localization performance. To measure the performance of the whole framework with respect to localizing an object by its parts, we use the following measures.

For the face and human dataset we consider

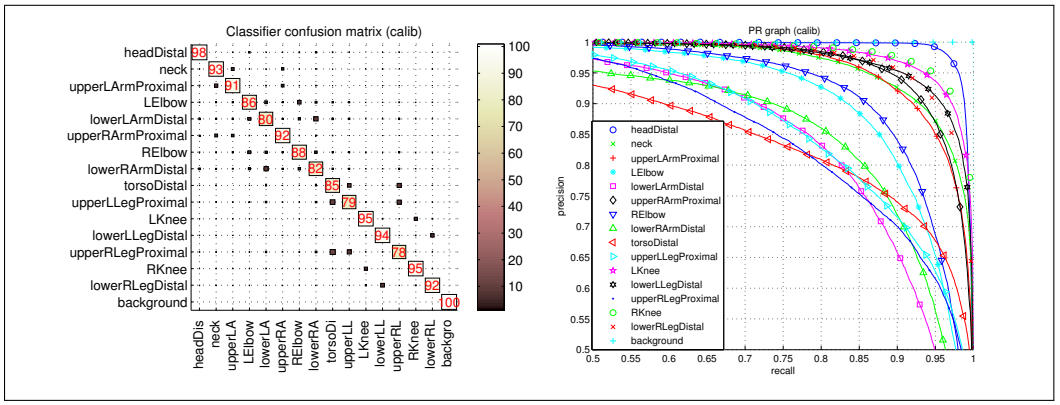
- the number of true positives (TP), i.e. ground truth is present and the estimated position is within the distance threshold (c.f. Figure 3.13).
- the number of outliers (OUT), i.e. ground truth is present and the estimated position is outside the distance threshold.
- the number of false negatives (FN), i.e. ground truth is present but the part has been labelled missing.



(a) Face dataset



(b) Human body dataset (PASCAL Challange)



(c) Human body dataset (HumanEva

Figure 3.14.: The classifier confusion matrix (left) and the precision recall graph (right) show the different performance of the part classifiers on the 3 datasets. Due to the simplicity of the face parts, the results for faces are much better than for human bodies. If we compare the results on the two human body datasets, we see that the classifiers performs significant better on the HumanEva dataset. This is not surprising since they contain less background clutter compared to the PASCAL data.

- the number of true negatives (TN), i.e. ground truth is occluded and part has been labelled missing.
- the number of false positives (FP), i.e. ground truth is occluded and part has been labelled present.
- the 2D relative localization error for TP: The distance of the part, after MAP inference, to its ground truth location normalized by an instance-specific distance. Normalization for the face dataset is with respect to the distance between the eyes. For the human dataset it is the mean of the distance between the left-hip and left-shoulder, and the distance between the right-hip and right-shoulder. We have chosen these normalization distances because they rarely suffer from foreshortening.

For the HumanEva dataset we use the performance measures joint with this dataset

- 2D localization error: The distance of the part, after MAP inference, to its ground truth location in pixel.
- 3D localization error: The distance in mm between the 3D ground truth location and the 3D location after triangulation using several synchronized 2D images.

The 3D localization error is also used for the Spine dataset.

3.4.1. Face

The model for the Face dataset consists of 5 parts: left and right eye, nose, left and right side of the mouth, see Figure 3.11. We used the Caltech face dataset [175] consisting of 450 frontal faces. Frames 1 to 89 (4 subjects, 3 male, 1 female, no beards, no glasses) constitute the training set; frames 90 to 165 (5 subjects, 4 male – two with beards, 1 female, no glasses) constitute the validation set; frames 166 to 450 (18 subjects, 10 male – one with beard, one with glasses – and 8 female) constitute the test set, where frames 328 to 336 have been rescaled by factor 3 so that the faces appear approximately at the same scale as the rest and frames $[400, 402, 403]$ have been omitted from evaluation (artificial paintings) making a total of 282 test frames. The shape features are more susceptible to changes in scale than the appearance features. Thus, we compute the MAP over 5 discrete scale settings ($[0.8, 0.9, 1.0, 1.1, 1.2]$) for the edge-length features.

Generic background was obtained from 45 images without people/faces, but featuring scenes where people normally occur.

Part Detection. Results from the part classifiers are visualized in Figure 3.14a. The classification performance is very good, the mean APR is 99.89% and Cohen's κ value with 96.5% also quite high. Only very few mis-classifications remain in the background class and between left and right eyes.

Learning. We used the CRF learning algorithm (Algorithm 3.1) to compute optimal λ parameters on the training and validation sets and compare them to the heuristic initialization (3.20). For the approximation of the expectations in the learning procedure we use loopy belief propagation (LBP), δ in (3.24) was set to 0.5. We also compare a simpler tree graph in form of a star with the nose as center to the completely connected graph. For this model LBP calculates the exact marginals.

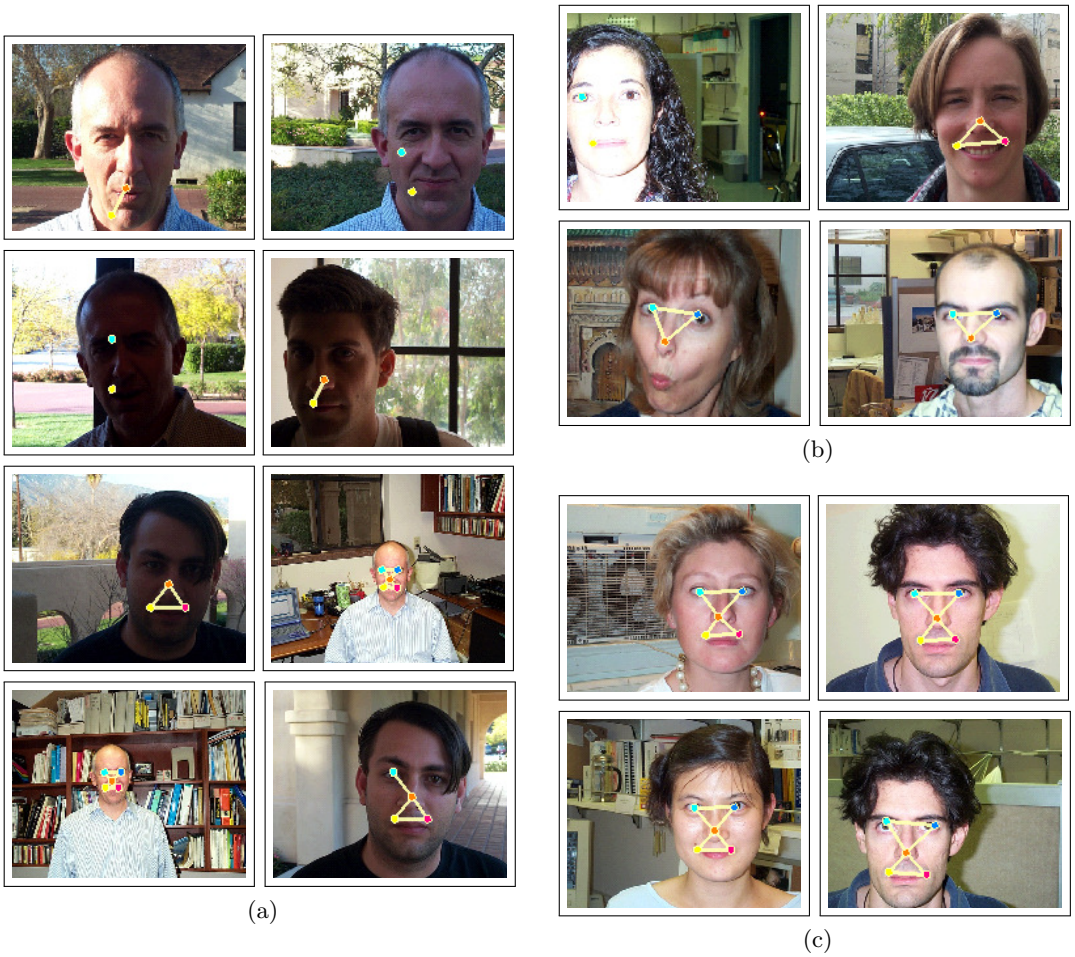


Figure 3.15.: Images with face configurations (slightly cropped). **(a)**: 8 worst configurations with respect to the mean distance to ground truth. **(b)** configurations with least parts detected not contained in **(a)**. **(c)**: 4 configurations with highest confidence, i.e. unnormalized probability. We throughout obtain good performance on this data as only the worst 3 images can be considered wrong configurations.

To compare λ values for different learning methods we rescaled them so that their sum equals one (MAP inference will not be affected by this). This means for the heuristic (3.20) that the sum of all λ values for nodes is $\sum_{a \in V} \lambda_{a;\text{app}} = \xi_{V,\text{app}} = 0.5$ and the sum of all λ values for edges is analogously $\xi_{E,\text{app}} + \xi_{E,\text{len}} + \xi_{E,\text{ori}} = 0.5$. Interestingly for the CRF learning these values shifted towards nodes 0.83 for CRF-LBP and 0.93 for the tree model. The edge terms are effectively zero for the edge appearance and length features (CRF-LBP: 0.01, 0; tree: 0, 0), compared to orientation (CRF-LBP: 0.16, tree: 0.06).

Localization. In Table 3.1 we show results for the localization performance on the test set for the different models. Baseline is given by ground truth, i.e. the user labeling and "baseline" which is the best localization possible given the reduced image information after candidate sampling (c.f. Section 3.2.4) by simply picking the nearest neighbor to the ground truth for each part in the set of candidates or the missing candidate if the ground truth is occluded. The models compared are: the complete graphs with two learning

ID	graph type	learning method	TP	OUT	FN	TN	FP	μ_d	σ_d
(1)	ground truth		1409	0	0	1	0	0	0
(2)	baseline		1406	3	0	1	0	0.04	0.04
(3)	complete	heuristic	1362	**5**	42	**1**	**0**	**0.08**	**0.05**
(4)	complete	CRF-LBP	**1375**	32	1	0	1	0.12	0.08
(5)	tree	heuristic	1358	25	26	0	1	0.10	0.16
(6)	tree	CRF-LBP	1374	30	5	0	1	0.12	0.08
(7)	decoupled	heuristic	1318	91	**0**	0	1	0.18	0.65

Table 3.1.: Localization performance for the face test set. Two models using complete graph as underlying structure have been learned using the heuristic (3) and CRF with LBP (4). Simpler graphs in form of a star with nose as center ("tree") (5+6) and a completely decoupled graph (7), where all nodes are independent, are also presented as comparison. Baseline is given in form of hand labeled ground truth (1) and "baseline" (2) which uses the nearest candidate to ground truth after sampling part-candidates. Positive results are in green, errors in red. In bold is the best number for each column, without (1) and (2). For discussion see Section 3.4.1 and Section 3.4.3.

methods (heuristic and CRF learning with LBP) (3)-(4); the tree graph with the heuristic initialization (taking into account the reduced number of edges in (3.20)) (5) and CRF learning using LBP (6). Finally, a decoupled graph without any structural edge information (7). Results are very close to optimality stemming from the good performance of the part-classifiers as indicated by the 93.5% of true positives obtained with the decoupled graph. Still more structure does improve the results: True positive rates increase for the tree models to 96.4% for the heuristic and 97.5% for the learned model, and to 96.7% (heuristic), 97.6% (CRF-LBP) for the complete graphs.

Overall there is no significant difference of the models given by the heuristic and the ones obtained by CRF learning on this dataset. Albeit the former tends to occlude more parts, but produces less outliers. This general trade-off is largely effected by the user parameters γ for the heuristic (here set to 0.8) and the penalty δ given to the missing candidate in (3.24). Also the tree models show comparable performance owing to the rigid structure of the faces in the images. We find that this is a comparatively simple dataset and our framework performs quite well. For comparable object-classes with simple structure the presented methods are well suited for detection.

Example configurations after MAP-inference are shown in Figure 3.15.

3.4.2. Human

As depicted in Figure 3.11, the model for the Human dataset consists of 13 parts or joints: 1 head, 2 shoulders, 2 elbows, 2 hands, 2 hip, 2 knees, and 2 feet. We have used a total of 2401 images consisting of images from private collections, images taken from the Internet and images of the PASCAL Visual Object Class (VOC) Challenge 2006 [43] and 2007 [41]. For the non-object class, we used the same background images as for the Face dataset. A total of 1243 images was used as training set to learn the feature functions for appearance and geometry, 717 images were used as validation set for the classifier calibration, and 441 images remained as test set. The test images were taken from the PASCAL VOC Challenge 2007 for the Person Layout task.

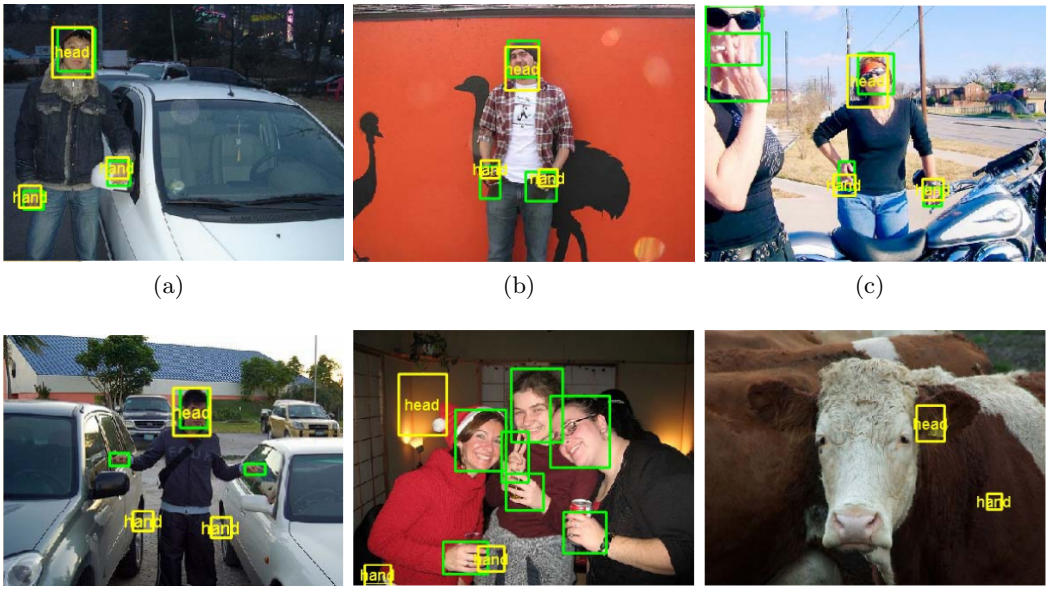


Figure 3.16.: Exemplary results evaluated on the Person Layout Change [41]. The yellow boxes are our estimates, the green boxes the hand labeled ground truth. A hit is shown in **(a)**. While our solutions was often nearly correct, we fail to have a significant overlap of the bounding boxes **(b)** or miss some parts of the second person **(c)**, that causes no hit on this image in the PASCAL evaluation. In **(d)** the data term in our model is not strong enough to escape the main geometric mode (hand hanging downwards). Figures **(e-f)** show some examples in which our approach fails.

Part Detection. Confusion matrix and precision recall graph for the calibrated classifiers are shown in Figure 3.14b. Cohen's κ values is 44.93% and the mean APR 45.18%. There is quite distinctive behavior for different body parts for this difficult dataset. Especially hands and elbows, whose appearance vary significantly due to articulations in the images, give poor results, followed by hip and knees. In contrast, the performance for head, shoulders and background is much superior. As the confusion matrix indicates, this is also true because of ambiguities between left and right body parts.

Compared to the results of the classifiers used for the face detection, our model has now to cope with worse local detectors.

Learning. We used the CRF learning algorithm (Algorithm 3.1) to compute optimal λ parameters on the calibration set, using LBP for learning on the complete graph (CRF-LBP) and on a simpler tree graph. δ in (3.24) was set to 0.01. The normalized λ values for the heuristic again sum up to 0.5 for nodes as well as for edges: $\sum_{a \in V} \lambda_{a;\text{app}} = \xi_{V,\text{app}} = 0.5$, $\xi_{E,\text{app}} + \xi_{E,\text{len}} + \xi_{E,\text{ori}} = 0.25 + 0.125 + 0.125 = 0.5$. The corresponding values for the complete model (CRF-LBP) are: 0.1818 for nodes and 0.8182 for edges (appearance: 0.0895, length: 0.2164, orientation: 0.5124). For the tree they are: 0.4247 for nodes and 0.5753 for edges (appearance: 0.0623, length: 0.2194, orientation: 0.2936). Contrary to the face dataset, the weights actually give more influence to the edge terms and less influence to the node terms. This is much more the fact for the complete graph than for the tree.

ID	graph type	learn. meth.	TP	OUT	FN	TN	FP	μ_d	σ_d
(1)	ground truth		2381	0	0	791	0	0	0
(2)	baseline		1494	887	0	791	0	0.27	0.36
(3)	complete	heuristic	628	940	813	526	265	0.55	0.58
(4)	complete	CRF-LBP	**734**	**868**	779	**541**	**250**	**0.41**	**0.45**
(5)	tree	heuristic	581	1196	604	298	493	0.77	0.74
(6)	tree	CRF-LBP	709	998	674	456	335	0.54	0.62
(7)	decoupled	heuristic	592	1762	**27**	90	701	0.94	0.79

Table 3.2.: Localization performance for the Human test set. Compared are the complete graph learned heuristically (3) and with CRF-LBP (4), the tree graph with heuristic (5) and learned (6), and the decoupled graph (7). Positive results are in green, errors in red. In bold is the best number for each column, without (1) and (2). For discussion see Sections 3.4.2 and 3.4.3.

It might indicate that shape is actually the more informative cue for this class as the part appearance is weaker for this difficult data.

Localization Figure 3.16 shows some of our results on the Pascal data. The evaluation of the PASCAL-Challange was done by measuring the overlap of the hand notated and inferred bounding boxes around the parts and the person. A detection was accepted as a hit if the overlap was above some threshold for all bounding boxes. Since our method infer only the position of the parts and no bounding boxes we have to estimate the height and width of the bounding boxes. Furthermore, we restrict ourself to find not more than one person per image. Together with ambiguity in occlusion (e.g. the right hand in Figure 3.8b may also be correctly labeled as occluded) we had an low average precision. While Figure 3.16(a) is a hit, (b) and (c) are considered as wrong since the overlap of the boxes is too small and the second person was not detected, respectively. In figure (b) we would claim that the yellow bounding boxes calculated by our approach are a better description than the ground truth notation. Of course our approach does sometimes fail and produce wrong detections, see Figure 3.16(d-f).

To avoid ambiguity in evaluation, we only consider images containing one person (244 frames). For input, images are rescaled using the method in Section 3.2.2. The shape term was optimized over 5 discrete scale settings ($[0.9, 0.95, 1.0, 1.05, 1.1]$). The results of the localization are summarized in Table 3.2. Figure 3.17 gives an overview over the number of correct parts per image, i.e. the number of true positives (TP) plus the number of true negatives (TN). Clearly, using CRF learning (Section 3.3.2) on the complete graph produces the strongest model for this dataset.

For the best performing model (CRF-LBP), Table 3.3 shows the localization results for each part individually. The rigid upper body parts head and shoulders are detected rather well, with moderate levels of outliers. Next come hip, followed by elbows. Most outliers occur with the hands that are the most difficult parts to detect, both with respect to appearance and shape due to articulation. Many of the images only contain partial configurations, especially images where only the upper body is visible and the rest is occluded. Our algorithm can handle these cases, indicated by the high levels of true negatives for the knees and feet.

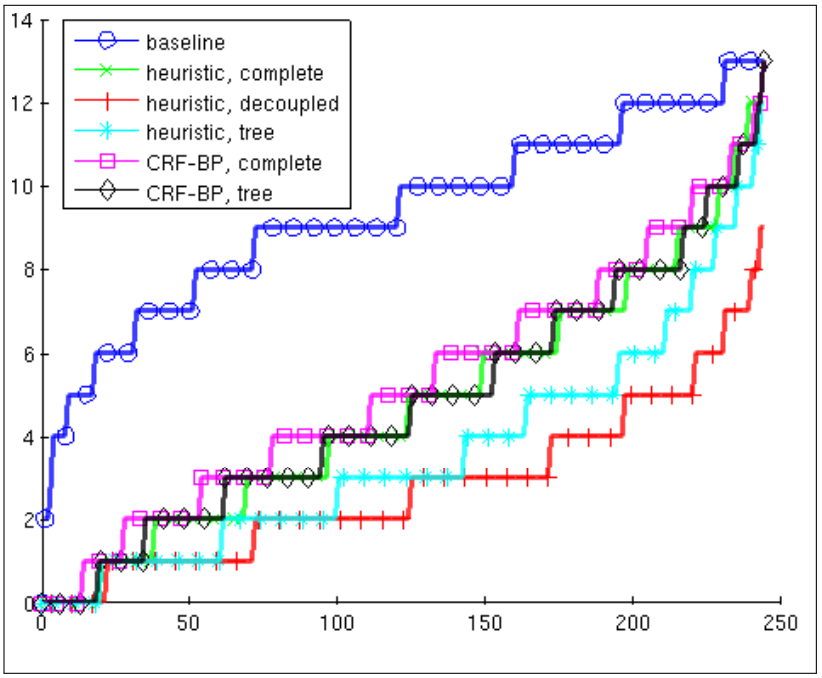


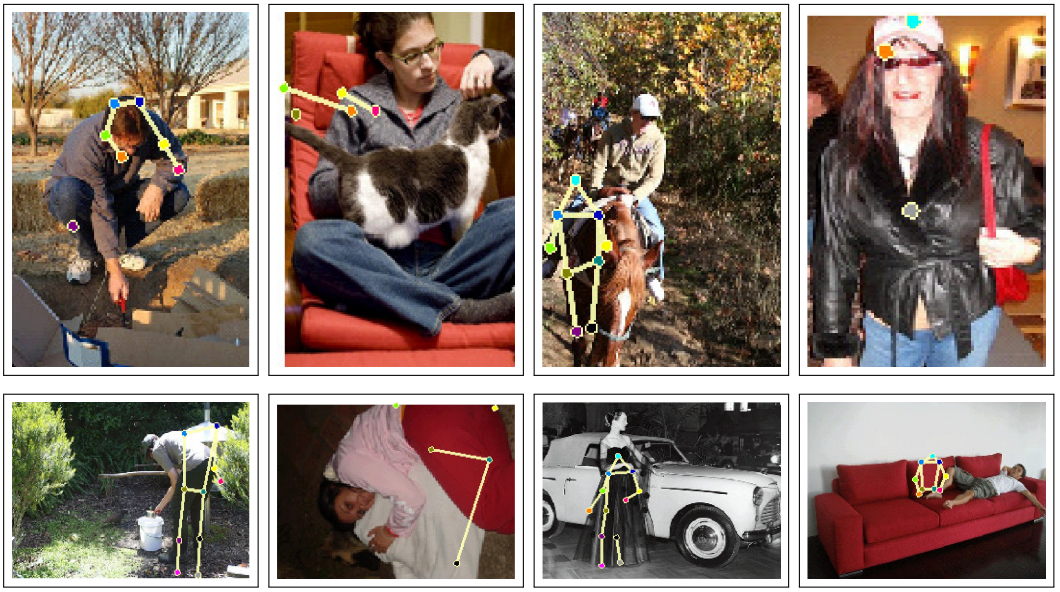
Figure 3.17.: Number of correct parts (TP+TN) per frame on 244 Human test images. Note that the integral under each curve corresponds to the total number of correct parts. On the x-coordinate we increase the number of images and give on the y-coordinate the correct number of parts for the n-th worse image. For further discussion see Sections 3.4.2 and 3.4.3.

Example configurations obtained after MAP-inference for the learned model are shown in Figure 3.18.

3.4.3. Comparison to Tree Graphs

As mentioned in previous sections, we compared the fully connected graphical models also to simpler tree structured models. Quantitative results are presented in Table 3.1 and Table 3.2 and also in Figure 3.17. Decline in localization performance is mainly due to our particular handling of occlusion/missing parts that is quite different to other methods: For a tree structured graph the inferred configuration is the result of two or more independent subgraphs if one of the parts is missing. For the face set with strong part classifiers this usually produces only marginally worse results, as most parts are found and there is hardly any occlusion. A few configurations where missing parts where inferred, however, illustrate this difficulty for tree-structured graphs, see Figure 3.19. For the human dataset where the framework has to deal with a high level of occlusion this can cause – for example – legs to appear inside the body, see Figure 3.20. This shows that for our framework dense graphs with additional structural information are a necessity.

The confusion between left and right body parts that are reported by others using tree models (e.g. [46]), and that usually require sampling from the tree-distribution and evaluation of another global cost-function, is immaterial in our framework, due to our particular shape features including absolute angles and appearance information on the edges.



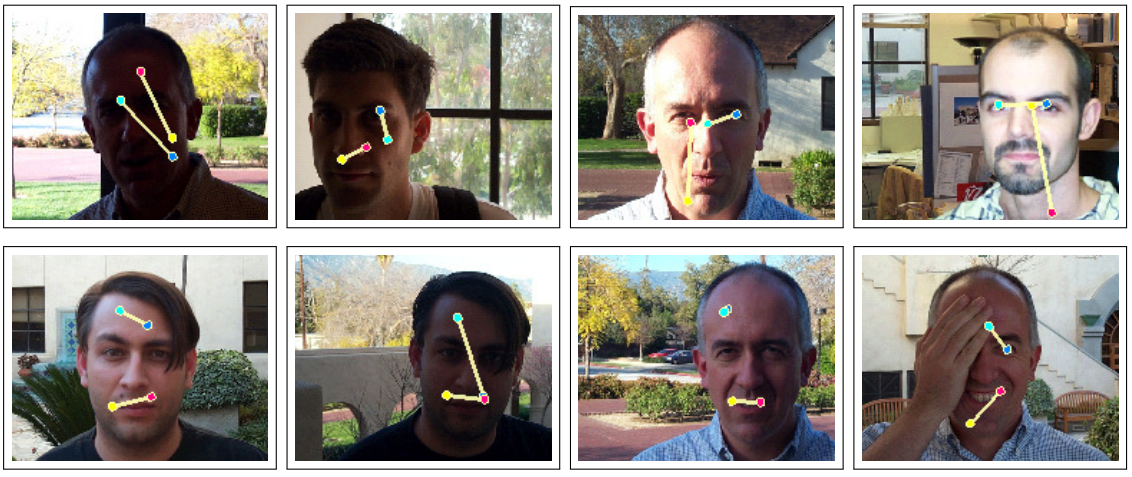
(a) Wrong Detections

(b) Good Detections

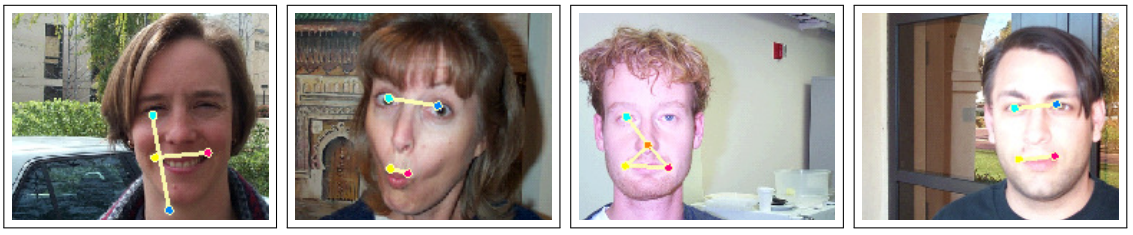
Figure 3.18.: Human configurations for the PASCAL VOC Challenge 2007 for the Person Layout task. **(a)**: configurations that were too difficult for our approach. **(b)**: 8 Configurations with high confidence. In general, standing humans without too much occlusion can be localized well as this is the predominant class in the training set.

	head	l-shoul	r-shoul	l-elbow	l-hand	r-elbow	r-hand	l-hip	r-hip	l-knee	l-foot	r-knee	r-foot	total
TP	160	94	125	41	31	47	31	58	56	26	22	24	19	734
OUT	55	51	73	76	94	92	100	79	74	42	26	58	48	868
FN	29	91	40	79	81	58	73	50	63	62	54	52	47	779
TN	0	6	2	25	16	18	16	37	28	87	116	87	103	541
FP	0	2	4	23	22	29	24	20	23	27	26	23	27	250

Table 3.3.: Localization performance for 244 Human test images, shown for each part individually for the model learned with CRF-LBP. Overall performance for head and shoulders is quite good, whereas the articulating parts, especially hands, are much harder to detect. This could be expected given the performance of the part classifiers and also the shape features are less discriminative for these parts.

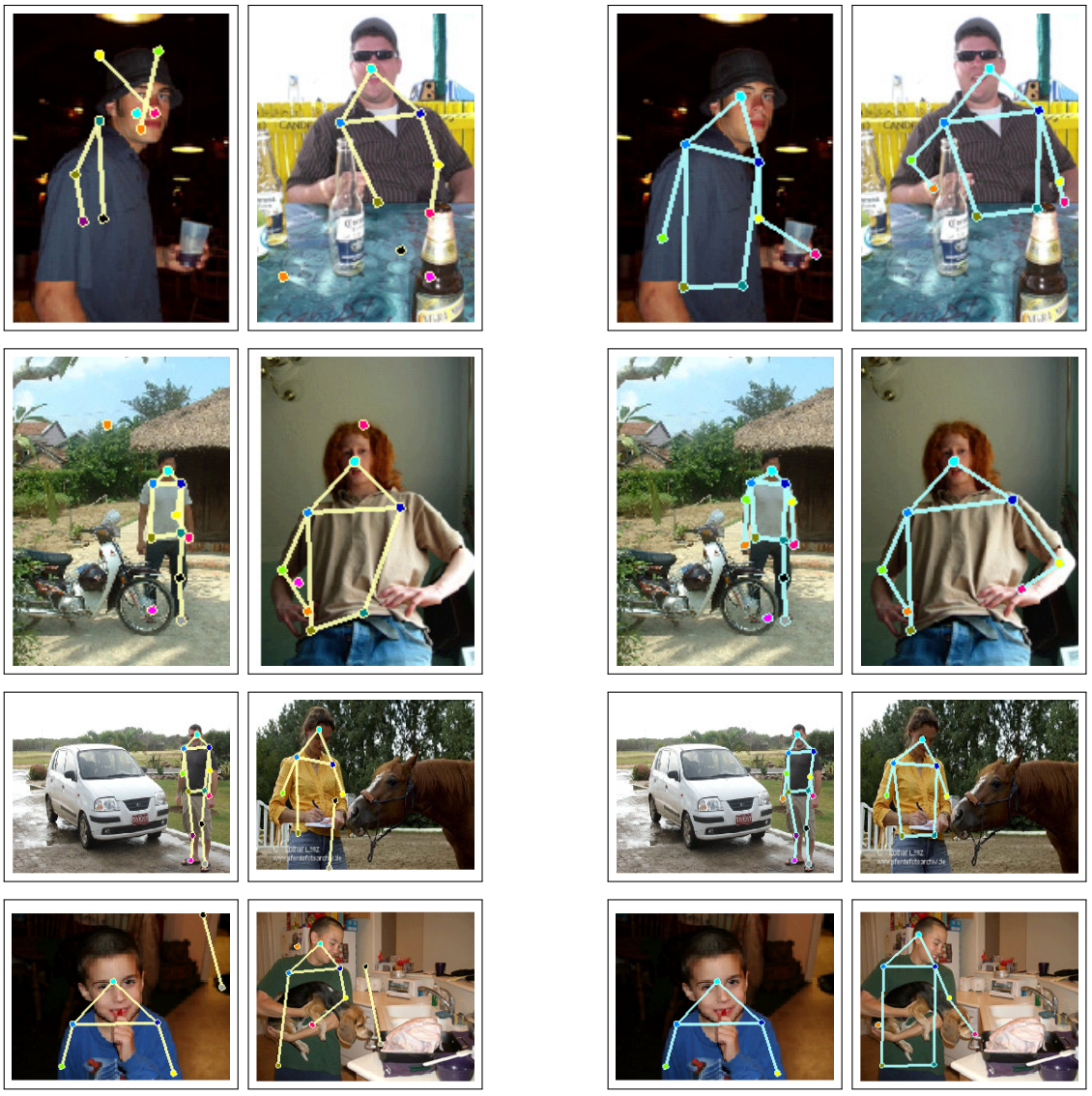


(a)



(b)

Figure 3.19.: Images with face configurations (slightly cropped) for the tree graph. The 8 worst configurations with respect to the mean distance to ground truth **(a)** and the first of the images with least parts detected **(b)**, can be considered wrong detections. Compare this to only 3 wrong configurations in Figure 3.15. Problems occur in cases when one or more parts are missing as then there is no structural information to other parts. This is especially severe in cases the nose is missing as it is the center of the star-shaped tree. Note that drawn edges do not correspond to the underlying tree-graph but are the same as in Figure 3.15.



(a) Tree-Structured Models

(b) Full-Connected Models

Figure 3.20.: Comparison of the tree graphs **(a)** to their respective fully connected models **(b)**. If parts are missing the tree-graphs get disconnected and independent subgraphs are matched, leading e.g. to spurious legs inside the body.

3.4.4. HumanEva

The HumanEva dataset [149] consists of several sequences of four subjects (S1, ..., S4) performing five different tasks (walking, jogging, gesturing, boxing and throwing/catching). The sequences for HumanEvaI were taken with 7 synchronized cameras (3 color, 4 grayscale), for HumanEvaII there are 4 synchronized color cameras. The sequences are separated into "Training", "Validation" and "Test" sets. The respective number of images in each set are given in [149]. The 2D labeling of parts (or joints) is similar to our labeling of the Human dataset, but here they do not correspond to visual features, but to the back-projection of their 3D counterparts, see Figure 3.11 for comparison. The 15 parts are shown in Figure 3.9.

The images in the trainings set have been used to train the local appearance classifiers, as well as the geometry terms. The validation set has been used for classifier calibration. In Figure 3.14c the confusion matrix and the precision recall curves are shown for the validation set, as ground truth for the test set is not available to us.

Classifier performance. Overall the classifier performance is much better than for the Human dataset as more training data is available and the test set is more similar to the training set. Mean APR is 95.43 and Cohen's κ is 90.35%, but note that the test set in this case is the same as used for calibration. The confusion matrix is here exemplary for the ambiguities between left and right body parts.

Due to the complex graphs, no CRF learning has been performed on this dataset. The presented results where obtained using only the initialization heuristic in Section 3.3.1.

Triangulation. The HumanEva dataset is the only dataset with 3rd-party ground truth data available, which allows objective measurements of the geometrical error. To obtain a 3D configuration for corresponding synchronized 2D images, we triangulate a 3D configuration as follows.

1. For all image pairs (I_i, I_j), $i, j \in \{1, \ldots, \#\text{cameras}\}$, $i < j$, of a synchronized frame, and for all parts $s \in V$ that are not occluded in either image, triangulate a 3D point $z_{s,ij}$ using standard stereo-triangulation.
2. Calculate the vector-median [177] to find the median 3D position, i.e. the 3D point z_s for part s that minimizes the sum of Euclidean distances to all candidate 3D points,

$$z_s = \min_{z_s} \sum_{ij} \|z_s - z_{s,ij}\| \,. \tag{3.30}$$

The vector-median has some beneficial properties: The median z_s lies in the convex hull spanned by $\{z_{s,ij}\}_{(ij)}$ and, like the scalar-median, is robust against outliers [177]. Additionally, it is rotationally invariant. For theoretical background and implementation details we refer to [9].

Localization performance. In Table 3.4 we summarize the results of the localization performance for the HumanEvaI and HumanEvaII dataset, by comparing two approaches: (1) inferring configurations for each image of a frame individually, and (2) using the additional epipolar constraints and inferring a configuration for all images of a frame at once

(indicated by the extension "e" in the table). Localization errors are reported for 3D and for 2D. For the shape terms, global scale r was inferred over 5 discrete scale settings ($[0.8, 0.9, 1.0, 1.1, 1.2]$). For 2D we also report the resulting error after back-projection of the inferred 3D locations which yields large improvements for the method without epipolar features, but almost no change in error when including epipolar features from the start, thus indicating that in deed the inclusion of epipolar features leads to more consistent 2D configurations. Sample configurations for different tasks after MAP inference are shown in Figure 3.21. Our results indicate that if the training and test data come from similar distributions, as it is the case for the HumanEvaI dataset, then our method works very well in almost all cases. Also when the test set changes, e.g. when the subjects have different clothing as is the case for HumanEvaII, but is still similar to the training, we can achieve competitive results with our method *without using background subtraction, temporal context or 3D kinematics*. Thus, our method can be used in contexts where the camera is not fixed and for (re-)initialization of tracking algorithms.

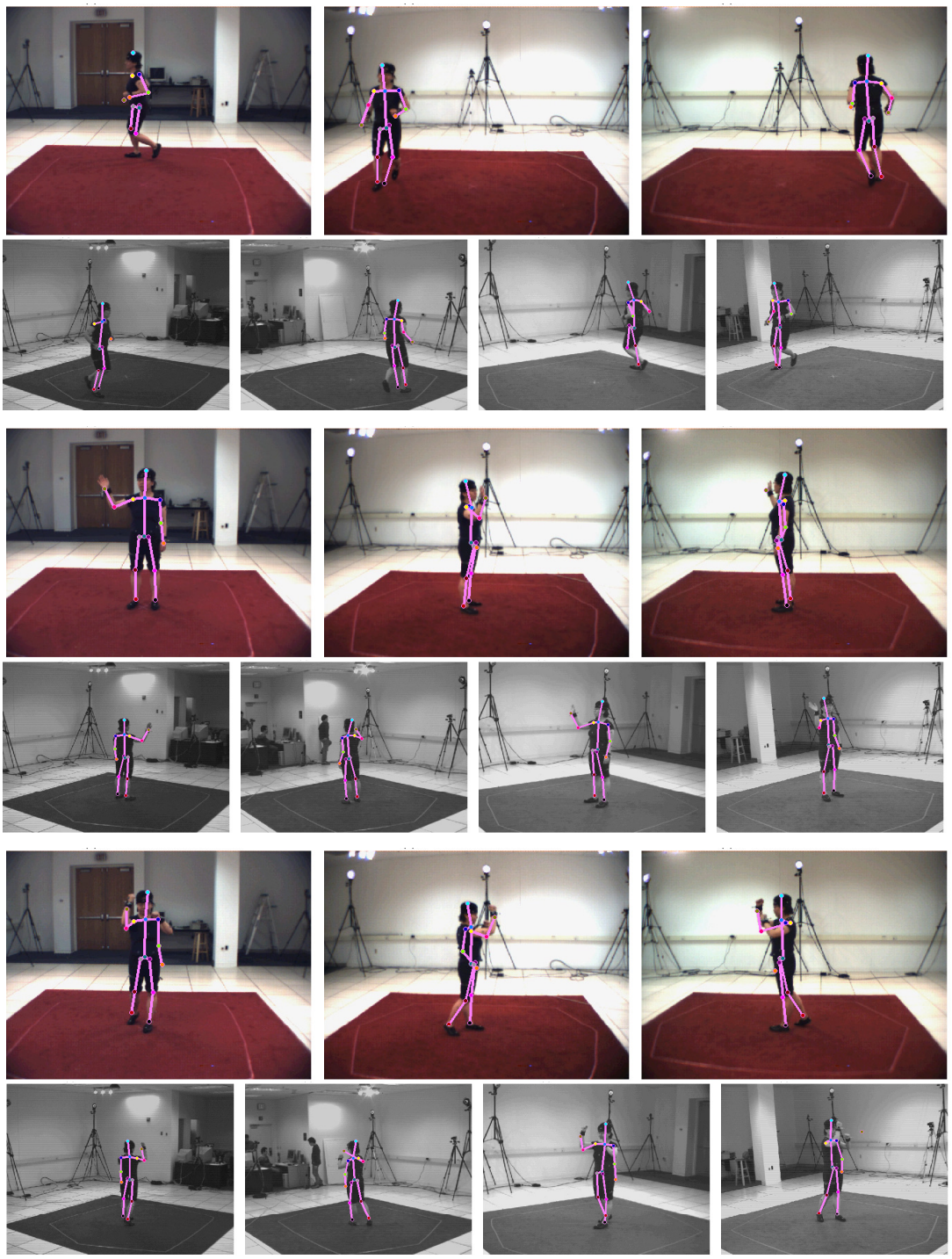


Figure 3.21.: Configurations for the HumanEva test set using MAP inference on the graphs including epipolar constraints. 3 complete frames are shown, i.e. all seven images of subject S1 performing the tasks jogging (rows 1-2), gestures (rows 3-4) and box (rows 5-6). We depict typical errors that can occur for some frames: 1st set: one foot is consistently matched to a wrong location (near the other foot). 3rd set: one arm is also matched consistently to a location near the hip, where it is often found during training and which leads to false detections in this case.

Data	3D				2D				3D→2D			
	μ	σ	0.5	0.9	μ	σ	0.5	0.9	μ	σ	0.5	0.9
I:S2	**39.49**	**3.28**	**39.41**	**43.28**	6.68	3.75	5.80	8.01	**4.82**	1.52	**4.64**	7.71
I:S3	95.33	48.30	84.21	183.36	15.15	9.63	12.97	29.37	11.74	**6.57**	10.00	**21.09**
I:S4	**197.96**	**61.77**	**192.32**	**292.84**	28.77	12.18	26.36	44.95	**24.58**	**9.74**	**22.60**	**36.61**
II:S2	207.48	90.85	**185.55**	338.71	29.75	15.51	26.16	50.66	**25.48**	13.18	**22.65**	**42.92**
II:S4	292.17	103.46	**283.49**	419.35	46.63	24.32	41.66	77.21	**38.82**	16.32	**36.30**	59.35
Data	3De				2De				3De→2D			
	μ	σ	0.5	0.9	μ	σ	0.5	0.9	μ	σ	0.5	0.9
I:S2	41.26	5.69	42.90	46.19	5.76	1.54	5.45	8.28	5.20	**1.42**	4.83	**7.67**
I:S3	**92.04**	**47.89**	**76.61**	**180.33**	12.17	6.99	9.93	21.91	**11.59**	6.73	**9.36**	21.19
I:S4	261.15	381.54	202.77	310.91	25.66	11.19	23.68	39.50	52.83	373.37	24.74	40.58
II:S2	211.40	**81.24**	200.51	**335.77**	27.76	12.65	25.42	46.47	27.13	**12.11**	25.18	45.32
II:S4	**290.98**	**78.56**	289.96	**397.70**	40.04	13.77	37.64	56.94	39.20	**13.31**	37.16	**55.24**

Table 3.4.: Localization results for the HumanEvaI (I:) and HumanEvaII (II:) test sets (Combo).
Settings: We took every 20th frame of the test set for each subject S2 to S4 for HumanEvaI and S2 and S4 for HumanEvaII. For subject S1 the gray-scale images were not available for action "Combo". We chose the "Cobmo" set as it includes all types of activities. We show the median and the mean error. 3D error is in millimeter, 2D errors are in pixel. If a part is labeled as occluded/missing it is not taken into account in the error-measure. The extension "e" indicates the model includes epipolar constraints. We report mean error (μ), standard deviation (σ), median (0.5) and 90 percent quantile (0.9). The bold values indicate the best 3D and 2D error for each subject.
Discussion: 2D error improves a lot with back projection when using the method without epipolar features. When epipolar features are included, then the back projection method does not seem to change much the 2D errors (except for I:S4 where it apparently produces one or more outliers – see mean and standard deviation for this case). This could indicate that the additional *epipolar features already enforce consistent results over the individual images* so that the back projection can not really improve the 2D error. 3D errors do not present significant differences neither do 2D errors after back projection or with epipolar features. Overall the resulting errors for HumanEvaII are a little higher than the ones reported by others [76, 27], *but without using background subtraction, temporal context or 3D kinematics.* Thus our method is especially attractive for (re-)initialization.

3.4.5. Spine Labeling in 3D Magnet Resonance Images

Our approach is not limited to face or person detection in 2D images. A related field that we investigate is the detection and labeling of anatomical structures in medical images. In this context, we experiment with magnetic resonance images of the human spine column, in which we automatically localize and identify the inter vertebral discs using the parts-based model described in this paper. Applications include labeled visualization, initialization of segmentation algorithm and statistical shape analysis for deformation-related pathologies (e.g. scoliosis). The 3D images are low-resolution ($224 \times 224 \times 180 \approx 9 \cdot 10^6$ voxels) T1-weighted fast field-echo images of the total spine. The fact that the sought discs have ambiguous local appearance, or, due to pathologies, might be degenerated or missing completely, is particularly challenging. Therefore, exploiting global context and permitting missing parts in the model are essential for successful labeling.

We used a simplified version of the described model for these data. Because of the limited training set of 30 3D-images, we modeled geometric features, i.e. pairwise part distances or pairwise displacements, as truncated Gaussians. The model contains 26 nodes corresponding to the center positions of the intervertebral structures, 23 of them being discs in the anatomical sense lying between mobile vertebrae. We trained an ensemble of 150 randomized trees without calibration on a set of volume patches around the ground truth locations as image features, augmented by resampled, deformed copies, and background.

Using a leave-one-out procedure for evaluation, the model was fit to the test image by generating the 10 candidates for each part having the strongest responses in the tree classifier, and infer the optimal configuration. The global scale parameter r was here estimated based on the first fit and used to refine the geometry prior, leading to a better subsequent fit by compensating for patient height. Finally, for parts marked as missing, we predict a position relative to the ones found in a postprocessing step. Figure 3.22 shows an example result of the procedure, illustrating localization and labeling of the model's parts in an unseen image.

We achieved an average part detection rate above 90% and an average part distance to ground truth of 5.1mm. Details are reported in [143].

The computational bottleneck of the detector in this application is not the inference stage, as one might expect, but rather the application of the tree classifier on each of the $\sim 9 \cdot 10^6$ voxels. Cascaded classifiers and parallelization might be viable approaches to resolve this issue.

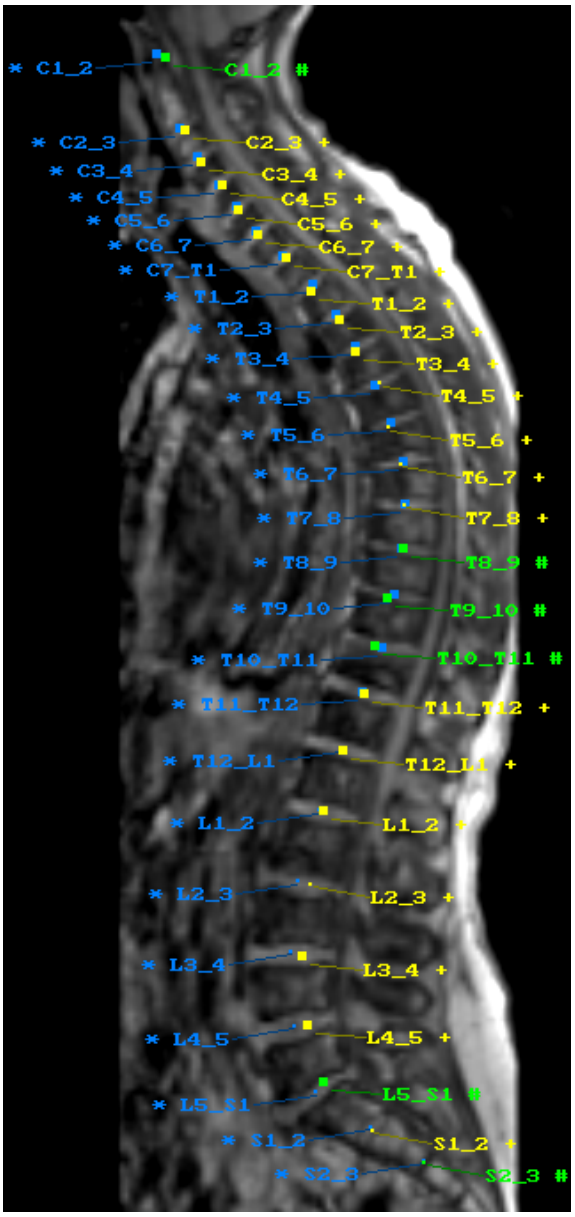


Figure 3.22.: MR spine labeling result.
Yellow labels show the MAP estimate(+), green labels represent parts inferred by postprocessing(#), and blue labels show the ground truth annotation(*), done independently. Larger dots indicate positions more closely located to the viewing plane. We achieved an average part detection rate above 90% and an average part distance to ground truth of 5.1mm. Details are reported in [143].

CHAPTER 4

INFERENCE ON DISCRETE MODELS

4.1. Introduction

In Chapter 2 we introduced graphical models to represent probability distributions and energy functions. In Chapter 3 we showed how graphical models can be used to model problems in computer vision. Let us now discuss how to solve the inference problems on the graphical models.

We assume that our model is given either as a probabilistic graphical model or as an energy function, both represented by a factor graph $G = (V, F, E)$ which forms together with an operation $\oplus$, multiplication or summation, a factor graph model. Accordingly the probability distribution and energy function are defined by

$$p(x|y) = \exp(\langle \theta, \phi(x) \rangle - A(\theta)) \tag{4.1}$$

$$\text{and} \qquad J(x|y) = -\langle \theta, \phi(x) \rangle, \tag{4.2}$$

respectively, where according to Section 2.5

$$\langle \theta, \phi(x) \rangle = \sum_{f \in F} \sum_{x_{\mathrm{ne}(f)} \in \mathcal{X}_{\mathrm{ne}(f)}} \theta_{f;x_{\mathrm{ne}(f)}} \phi_{f;x_{\mathrm{ne}(f)}}(x_{\mathrm{ne}(f)}). \tag{4.3}$$

The exponential parameters θ depend indirectly on the observation y since y is an argument of the feature functions. However, for reasons of readability we will omit this connection to y. Furthermore, it is common in the literature to use $J(x)$ as a shorthand for $J(x|y)$. In the posterior distribution $p(x|y)$, y is a fixed set of noisy observations in the image. For a fixed observation $p(x|y)$ and $J(x|y)$ are related by

$$p(x|y) \propto \exp\left(-J(x|y)\right). \tag{4.4}$$

Inference problems can be written in a general form as accumulation of an objective function $\mathbf{f}(x)$ over a subset of configurations $S \subset \mathcal{X}$. In our problems $\mathbf{f}(x)$ is either a

probability distribution $p(x|y)$ or energy function $J(x|y)$. If we denote the accumulation function by $\odot$, inference problems can be generally expressed by

$$\bigodot_{x \in S} \mathbf{f}(x) \tag{4.5}$$

or as searching the argument x^* for which

$$\mathbf{f}(x^*) = \bigodot_{x \in S} \mathbf{f}(x) \tag{4.6}$$

where

$$\mathbf{f}(x) = \bigoplus_{f \in F} f(x_{\mathrm{ne}(f)}). \tag{4.7}$$

The set S is usually so large that accumulation over all $x \in \mathcal{X}$ is intractable. An important property of $\odot$ and $\oplus$ is, that they define together with some set Ω a commutative semiring $(\Omega, \oplus, \odot)$ – see Definition A.2. Consequently, we have distributivity of $\oplus$ over $\odot$, e.g. $a \cdot \max\{b \cdot c\} = \max\{a \cdot b, a \cdot c\}$ with $\oplus = \cdot$ and $\odot = \max$. The distributivity is an important and indispensable property of the inference problems we will discuss.

Furthermore, any discrete graphical model (X, G) can be transformed into an equivalent model (X', G) with $\mathcal{X}'_a = \{1, \ldots, |\mathcal{X}'_a|\}$ by using a bijective function $\chi_a : \mathcal{X}_a \to \{1, \ldots, |\mathcal{X}_a|\}$. We will call $\mathcal{X}'$ the labeling space of the random variable X'_a. The inverse function $\chi_a^{-1} : \{1, \ldots, |\mathcal{X}_a|\} \to \mathcal{X}_a$ maps the labeling back to the corresponding value of the original space. With out loss of generality we will furthermore assume that for all $a \in V$ the space $\mathcal{X}_a = \{1, \ldots, L_a\}$.

Since all problems we will consider in this work are based of models, which decompose into functions depending on not more than two variables, we will often use an alternative representation. This representation is common in the literature and was already introduced in Chapter 3. Therefore, we define our problem as an undirected graphical model $(X, G = (V, E))$ with an objective function defined by

$$\mathbf{f}(x) = \bigoplus_{a \in V} f_a(x_a) \oplus \bigoplus_{ab \in E} f_{ab}(x_a, x_b). \tag{4.8}$$

This undirected model can be easily transformed into a factor graph model. However, to be consistent with the notation in the standard literature, we will sometimes use the from defined in (4.8) instead of (4.7).

4.1.1. Marginalization-Problem

The first class of inference-problems is to calculate a marginal distribution over a subset of variables of a model, also known as marginalization- or marginal-problem. The general definition of a marginal distribution with respect to a node set $A \in V$ and the $\odot$-$\oplus$-semiring is

$$b(x_A) = \bigodot_{x_{V \setminus A} \in \mathcal{X}_{V \setminus A}} \bigoplus_{f \in F} f(x_{\mathrm{ne}(f)}). \tag{4.9}$$

For the problems considered in this work three different type of marginal distributions are of interest. Each of them belongs to a different semirings.

Marginal: For $A \subset V$, the classical marginal is the marginal distribution $p(x_A|y)$ of the distribution $p(x|y)$ defined by

$$p(x_A|y) := \sum_{x_{V\setminus A} \in \mathcal{X}_{V\setminus A}} p(x|y) \tag{4.10}$$

and corresponds to the sum-product-semiring $([0,\infty), +, \cdot)$. We will call this marginal distribution just marginal. For a specific $x_A \in \mathcal{X}_A$, the marginal $p(x_A|y)$ is the probability, that the random variables belonging to A take the value x_A given y. We have encountered this problem in Section 3.3.2, since the marginals are related with the derivative of the log partition function. Learning the parameters λ of the CRF-model in the maximum likelihood framework involves to solve (4.10) several times.

Max-Marginal: A second marginal distribution we will consider, corresponds to the max-product-semiring $([0,\infty), \max, \cdot)$. The max-marginals are defined by

$$p^{\max}(x_A|y) := \max_{x_{V\setminus A} \in \mathcal{X}_{V\setminus A}} p(x|y). \tag{4.11}$$

The max-marginal of x_A has the same value as the probability of the most probable configuration for which the random variables of A takes the value x_A. The difference between (4.10) and (4.11) is, that the accumulating operation has changed. The max-marginal $p^{\max}(z_A|y)$ is an upper bound on the probability $p(x|y)$ for all configurations x for which $x_A = z_A$. Furthermore, max-marginals can be used for the calculation of the mode of $p(x|y)$, see Section 4.1.2.

Min-Marginal: Instead of the posteriori probability often energy functions are used. The main advantage is the lack of a normalizing term. Furthermore, the energy is given by a summation over local functions, instead of a product, which can be written as an inner product without an exponential function. Thus, we define the equivalent to the max-marginal in terms of energy by the min-marginal which correspond to the min-sum-semiring $((-\infty,\infty], \min, +)$ and is defined by

$$J^{\min}(x_A|y) := \min_{x_{V\setminus A} \in \mathcal{X}_{V\setminus A}} J(x|y). \tag{4.12}$$

Similar to the max-marginal, the min-marginal $J^{\min}(z_A|y)$ is a lower bound on the energy $J(x|y)$ for all configurations x for which $x_A = z_A$.

Just as the probability and energy the max- and min-marginals are related by

$$p^{\max}(x_A|y) \propto \exp\left(-J^{\min}(x_A|y)\right) \tag{4.13}$$

if $p(x|y)$ and $J(x|y)$ correspond to the same model.

4.1.2. MAP-Problem

The second major inference problem we consider, is the calculation of the mode of a model. The mode is the configuration which is most probable or has the lowest energy. Both, probabilistic and energy based models are related as shown in (4.4) and we can transform one in the other. The problem of finding the mode is also known as maximum

a posterior probability (MAP) problem, since we would like to calculate the maximizing elements of the posterior distribution $p(x|y)$.

The mode of $p(x|y)$ has not to be unique, there can exist several optimal solutions with identical values for $p(x|y)$. We denoted the set of all of this values by X^* and use x^* for a element of X^*.

$$x^* \in X^* := \arg\max_{x \in \mathcal{X}} p(x|y) \tag{4.14}$$

If a distribution $p(x|y)$ has exactly one unique mode we will call it unimodal and otherwise multimodal. Due to the strict monotony of the logarithmic function, the mode can also be calculated by

$$x^* \in X^* := \arg\min_{x \in \mathcal{X}} J(x|y). \tag{4.15}$$

It is well known, that the MAP-Problem is NP-hard [29, 146]. In general, we can not expect, that we will be able to solve the MAP-problem exactly. However, the NP-hardness does not imply, that the problem is in general not efficiently solvable. Even if the optimal solution can not be calculated in suitable time, one can find approximate solutions. In such cases we will always favor methods which give upper and lower bounds on the optimal posterior probabilities and energies, respectively.

For unimodal models, we can use the max- or min-marginals to calculate the mode of a model for each node $a \in V$ independently by

$$x_a^* = \arg\max_{x_a \in \mathcal{X}_a} p^{\max}(x_a|y) \tag{4.16}$$

$$x_a^* = \arg\min_{x_a \in \mathcal{X}_a} J^{\min}(x_a|y). \tag{4.17}$$

For multimodal models, this straight forward way will not work in general. If for example the set of modes is $X^* = \{(0,1,0),(1,0,1)\}$, then the above procedure can select the optimal labels for different variables from different modes and find the labeling $(0,0,0)$, which is not in $\mathcal{X}^*$. However, for acyclic models we can overcome this problem by a strategy, with the slightly higher complexity $O(|V| \cdot \max_{f \in F} |\mathcal{X}_{\mathrm{ne}(f)}|)$. The idea is to select the candidates with respect to the marginals defined on the factor nodes to ensure that we do not mix up different modes, see Algorithm 4.2. However, this shifts only the complexity from the calculation of the mode to the calculation of the max- or min-marginals.

Alternatively, we tackle the MAP-problem directly, instead of solving the max-marginal problem first. For several subclasses of models polynomial time algorithms for the MAP-problem exist. Unfortunately, our part-based object detections belong to neither of these classes. However, we apply branch and bound algorithms which show empirically good results and guarantee optimality.

As another line of research, we will consider to transform the MAP-inference problem into a convex optimization problem. Relaxing the problem may lead to fractal solutions. However, at least we obtain bounds on the optimal objective value. When fractal solution occur, rounding schemes have to be applied to obtain an integer solutions.

4.1.3. Related Work

In the following we will discuss how these inference problems can be solved and introduce several existing algorithms. The applicability and performance of these algorithms depend

on several characteristics of the graphical model. While for models with acyclic graph all these inference problems are manageable, for cyclic graphs this is in general not the case.

The algorithms we will consider can be divided in two main classes, namely algorithms based on combinatorial optimization and algorithms based on variational inference and convex optimization.

We will discuss several of those algorithm in the next sections. Nevertheless a short overview should be given. To simplify the notation for the asymptotic runtime [30] let us assume that $\forall a \in V : |\mathcal{X}_a| = L$ and let $o = \max_{f \in F} \mathrm{ne}(f)$ be the order of the model, given by the maximal number of neighbors of the factor nodes.

Let us first consider the problem of calculating the min-, max- and classical marginals. Since for discrete graphical models the domain of the random variable is finite. The most naive approach for calculation of the marginal distributions would be to accumulate for all relevant $x \in \mathcal{X}$ with fixed x_A. This Brute-Force-Algorithm will terminate and find the correct solution of the marginal inference problems in $O(L^{|V|})$ - which is impracticable even for small problems. This motivates approaches which make use of the structure of the problem given by G and solve the problem by graph based algorithms.

Dynamic Programming: For acyclic models the technique of dynamic programming can be used for fast inference. The complexity depends on the order of the model and is bounded by $O(|V| \cdot L^o)$. We will discuss this algorithms in Section 4.2.1. For models with a cyclic graph G we can construct an equivalent model over the junction tree $G' = (V', E')$ of G and apply dynamic programming on this acyclic model, see Section 4.2.2. The complexity for the computation rises to $O(|V'| \cdot L^{|tw(G^m)+1|})$. Since even a $n \times n$-grid graph has a tree-width of n and for our object detection problems the tree-width is $|V| - 1$, the junction tree algorithm has minor practical relevance. However, it solves the inference problems and is asymptotically faster than brute force search for sparse connected models.

Motivated by the dynamic programming approach, Pearl [123] proposed to use message passing algorithm called Loopy Belief Propagation (LBP) also for cyclic graphs. Originally LBP was suggested for the sum-product-semiring and later also applied on other semirings. For acyclic models it acts like the dynamic programming approach, if messages are send serial in an optimal order. Even for parallel updates it guarantees to converge to the optimal solution, but with some overhead of calculations. If cycles exist, the behavior of the algorithm is more or less undefined and convergence is not guaranteed. For decades LBP was applied on cyclic graphs for different semirings and showed convincing results. However, the question what LBP really optimizes was unclear for a long time. Investigations by Yedida et al. [186, 187] showed that fix points of the sum-product-LBP coincide with stationary points of the Bethe variational problem which dates back to the work of Bethe [15]. The main operation of LBP is sending messages between the nodes of the graph. This messages turn out to be directly related with the Lagrange multipliers of the Bethe variational problem and can be understood as a re-weighting of the objective function [182, 183]. However, in general the Bethe variational problem is not convex. Since the practical results with LBP are quite good, it has been used often. The asymptotic runtime of a single iteration of LBP is $O(|F| \cdot L^o)$.

Convex Relaxations for the Marginal-Problem: In a series of works [167, 172, 169, 170, 87, 135, 173] Wainwright and colleagues have introduced algorithms which are similar to LBP but based on a convex relaxation of the variational problem. Instead of replacing the

log partition function by a non-convex approximation, as done for the Bethe variational problem, they consider a convex upper bound. In an alternative view on their approaches, they give a description based on a decomposition of the original problem into several tree-structured subproblems. By reparametrization of the decomposition, which reweights the trees, an bound obtained by this decomposition is optimized. In the literature this family of algorithms is known as TRW or TRBP. It can be shown, that the dual problem to finding the optimal reweighting is a convex optimization problem including constraints over an outer relaxation of the marginal polytope, which consists of a polynomial number of facets.

Similar to the LBP, Wainwright [173] present message passing algorithms for the calculation of approximative marginal distributions for the sum-product- and max-product-semiring. While TRBP process fix-point updates, which are similar to a block coordinate descent, the algorithm can get stuck in local optima satisfied the weak tree agreement [87] if the function is non-smooth. Furthermore, TRBP has no guarantee for convergence if all messages are calculated parallel. Sequential updating of the messages (TRW-S), as suggested by Kolmogorov [87], guarantee to converge. However, for efficient implementation it requires that all trees a consistent with some ordering of the nodes. While for grid graphs this ordering is obviously, it restricts the choice of the decomposition defining trees for full connected graphs significantly. That is why we will follow a damping strategy by Wainwright [173], which leads for TRBP to better convergence behavior. We will discuss this approach detailed in Section 4.3.

A huge class of computer vision problems, including those discussed in Chapter 3, can be reduced to a MAP-problem. LBP or TRBP can be used to calculate the exact or approximated max-marginals, also known as pseudo max-marginals. Ignoring that the pseudo-max-marginals are not exact we can estimated the mode by $x_a^* = \arg\max_{x_a \in \mathcal{X}_a} p^{\max}(x_a|y)$.

Search-Based Algorithms: Of cause there are also other approaches which tackle the MAP-problem directly. An important class of problems in computer vision can be solved by algorithms based on the Max-Flow- or the Min-Cut problem. MAP-problems with binary random variables and a factor order less than or equal to three can be transposed into a Min-Cut problem [91]. If the energy function $J(x)$ of the model is submodular, all weights in the graph in which the cut is performed, can be chosen non-negative and the Min-Cut problem can be solved in polynomial time. For non-binary or non-submodular problems several modifications exist [91, 137, 89, 141, 95, 96, 92, 130, 85] which will be discussed in Section 4.2.4. For many important problems on grid-graphs, graph-cut algorithms have become state of the art, since they are very fast and show impressive results [159]. However, for these problems it is essential, that the pairwise energy terms in the models are at least semimetrics. In many application, e.g. our part-based object detection problem, the pairwise terms are highly non-submodular and far away from being a semimetric.

If $J(x)$ is highly non-submodular, the domain of the random variables is large, and the graph is highly connected, none of the previous mentioned algorithms is able to calculate the mode of such graphical models in an acceptable time. Since exactly such models appear in our framework for part-based object detection we furthermore investigate this class of problems. The remarkable property of this models is that even if $|\mathcal{X}|$ is very huge, a small subset of $\mathcal{X}$ has significant lower energies than the other configurations in $\mathcal{X}$.

We use this property and transform the MAP problem into a shortest path problem. This is solved by the A^*-algorithm with a heuristic based on a tree-approximation of the

graphical model. The A^*-algorithm belongs to the family of branch and bound algorithms and guarantees to converge to the optimal solution. In the worst case the runtime is still exponential in $|V|$, but empirically the A^*-approach is fast for models of moderate size. However, for larger models the A^*-approach will become very slow.

While a common preconception in the field of computer vision is, that standard solvers for mixed integer programs (MIP) are not applicable for computer vision problems, MIP-solvers was to our knowledge not applied on the MAP-problem so far in the field of computer vision. Motivated by the results of our A^*-approach, which is very similar to methods used in MIP, we test the commercial software CPLEX [1] on our problems. Surprisingly, the performance of the CPLEX-solver was high, but was still outperformed by our A^* approach for moderate size problems. However, the memory requirements for MIP-solvers are enormously for huge problems as used in [159]. Therefore, general purpose MIP-solvers are not applicable for a large class of computer vision problems. Nevertheless, our experiments show that MIP-solvers can solve many computer vision problems exactly. Many medium sized problems, which so far are only solved approximatively, can be solved optimally by standard MIP-solvers.

Convex Relaxations for the MAP-Problem: Another major line of research tries to relax the original MIP using the theory for discrete exponential families. For the calculation of the mode one "just" has to solve a convex problem with a linear objective and affine inequality constraints:

$$\max_{\mu \in \mathcal{M}(G)} \langle \theta, \mu \rangle \tag{4.18}$$

However, this problem is also intractable. The combinatorial complexity of the integer problem is shifted into the constraint-set of the LP. The number of affine inequality constraints defining $\mathcal{M}(G)$ is the number of facets of this marginal polytope. Even approximations of this problem by replacing the marginal polytope by a simpler outer polytope, known as the local polytope [173], becomes quickly too large for standard LP solvers.

Several methods, including TRBP [173] and TRW-S [87], try to tackle this problem over the local polytope, by solving the dual problem. The main attention is memory efficiency, e.g. by avoiding storing the affine constraints, and making use of the special structure on the graphical model.

A common technique in convex optimization is to decompose the problem in a set of simpler interdependent subproblems and force their consistency by additional constraints. This is known as Lagrangian or dual decomposition [65, 64, 94]. The dual problem is typically solved by sub-gradient methods. Contrary to TRW-algorithms, this method does not get stuck in local fix points. However, solving the non-smooth dual problem by sub-gradient methods includes the selection of step sizes, which is not trivial and influences the speed of convergence. Typically, these methods have sub-linear convergence rates [14].

The calculation of the sub-gradients can be done by solving the MAP-problem for subproblems, which are sufficiently simpler than the original problem. Komodakis [94] used this method together with tree-structure subproblems. This leads to the same relaxation as used by TRW-algorithms. In a later work [93], he used simple cyclic-subproblems for which MAP-inference is computationally feasible. Since more complex sub-problems may lead to tighter relaxations, these frameworks are mainly from interest for applications where simple relaxations are not sufficiently tight. Recently, Betra et al. [7] and Strandmark and

Kahl [155] have used the technique of dual decomposition with outer planar graphs and submodular subproblems, respectively.

We will revise this class of algorithm from the view point of exponential families, and introduce a decomposition in k-fan subproblems, which ends up in relaxations which are tighter than the local polytope relaxation.

In addition to linear programming relaxations of the MAP-problem several other convex relaxations have been suggested in the literature. Among those are methods based on quadratic programming [132, 83], second order cone programming [101] or semidefinite programming [140, 171]. For a review of convex relaxations for the MAP-problem we refer to [100].

4.1.4. Organization

The rest of this chapter is organized in the following way.

In Section 4.2, we introduce combinatorial methods for solving the inference problems. Starting with classical dynamic programming which solves the problems exactly for acyclic models, we sketch the idea of the junction tree algorithm to transform problems with cyclic structures into acyclic ones. Next we will introduce a heuristic extension of the dynamic programming method to cyclic structures by loopy belief propagation. In the second part of this section we investigate cyclic problems for which the MAP-problem can still solved exactly, including min-cut methods, A^*-search and mixed integer programming.

In Section 4.3, we revisit the inference-problems from the view point of variational inference and exponential families. This gives us a alternative view on the LBP-algorithm. Furthermore we discuss tree re-weighted message passing algorithms and convex relaxations of the MAP-problem. Finally, we introduce a Lagrangian decomposition approach, which decompose the graphical model into several simpler models and solve the dual problem by subgradient descent methods.

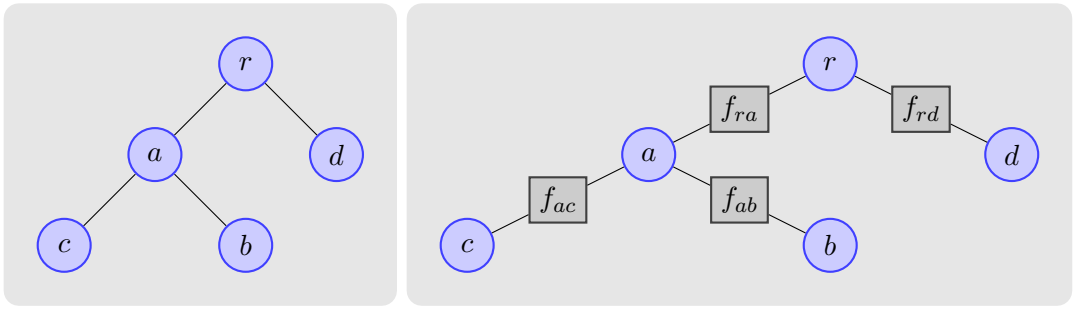
In Section 4.4 we sum up with an empirical evaluation of the introduced inference algorithms for the MAP-problem on real world and synthetic data.

4.2. Combinatorial Optimization

4.2.1. Dynamic Programming

Dynamic programming [10] is a method of solving problems, by breaking them down into sequence of simple calculations. Each calculation depends on the result on some other calculations. If an ordering exists such that each calculation only depends on calculations with a lower order, executing the calculations in this order leads to the solution of the problem. We can distinguish between two types of dynamic programming. Bottom-Up dynamic programming involves formulating the calculation as a recursive series of calculations, while Top-Down dynamic programming includes storing the results of certain calculations and reuse it later. It is also possible to take account of both types.

To illustrate the use of dynamic programming for inference problems on graphical models, let us consider the problem of computing the marginal $p(x_r|y)$ for the random vector X in a tree structured model with the graph G shown in Figure 4.1a.



(a) Undirected Graph (b) Factor Graph

Figure 4.1.: Example graphs – **(a)** Undirected graph and **(b)** factor graph for the same second order acyclic model. Without loss of generality we ignore unary factors in this factor graph model.

According to the graph G the probability distribution factorize into

$$p(x|y) = \frac{1}{Z} f_{ra}(x_r, x_a) f_{ab}(x_a, x_b) f_{ac}(x_a, x_c) f_{rd}(x_r, x_d). \tag{4.19}$$

Alternatively, this can be modeled by the factor graph in Figure 4.1b. For an efficient calculation of the marginal $p(x_r|y)$ we exploit that $([0, \infty), +, \cdot)$ defines a commutative semiring and so the multiplication distributes over the addition. Consequently, we can reformulate the calculation of $p(x_r|y)$ into a sequence of local operations.

$$p(x_r|y) = \sum_{x_{V\setminus\{r\}} \in \mathcal{X}_{V\setminus\{r\}}} p(x|y) \tag{4.20}$$

$$= \sum_{x_a \in \mathcal{X}_a} \sum_{x_b \in \mathcal{X}_b} \sum_{x_c \in \mathcal{X}_c} \sum_{x_d \in \mathcal{X}_d} \frac{1}{Z} f_{ra}(x_r, x_a) f_{ab}(x_a, x_b) f_{ac}(x_a, x_c) f_{rd}(x_r, x_d) \tag{4.21}$$

$$= \frac{1}{Z} \left(\sum_{x_a \in \mathcal{X}_a} \sum_{x_b \in \mathcal{X}_b} \sum_{x_c \in \mathcal{X}_c} f_{ra}(x_r, x_a) f_{ab}(x_a, x_b) f_{ac}(x_a, x_c) \right) \cdot \left(\sum_{x_d \in \mathcal{X}_d} f_{rd}(x_r, x_d) \right) \tag{4.22}$$

$$= \frac{1}{Z} \left(\sum_{x_a \in \mathcal{X}_a} f_{ra}(x_r, x_a) \Big(\sum_{x_b \in \mathcal{X}_b} f_{ab}(x_a, x_b) \Big) \Big(\sum_{x_c \in \mathcal{X}_c} f_{ac}(x_a, x_c) \Big) \right) \cdot \left(\sum_{x_d \in \mathcal{X}_d} f_{rd}(x_r, x_d) \right) \tag{4.23}$$

According to (4.23), we first sum up $f_{ab}(x_a, x_b)$ over x_b and $f_{ac}(x_a, x_c)$ over x_c and multiply these two results with $f_{ra}(x_r, x_a)$. This product only depends on x_r and x_a. We sum this function over x_a and multiply it with $\sum_{x_d \in \mathcal{X}_d} f_{rd}(x_r, x_d)$. If this product is divided by Z we get the marginal-distribution $p(x_r|y)$. Overall, this requires $4 \cdot L$ additions and $3 \cdot L + L^2$ multiplications. This is significant smaller number of operations than the L^4 additions and $4 \cdot L^4$ multiplications needed for evaluating (4.21) without using distributivity.

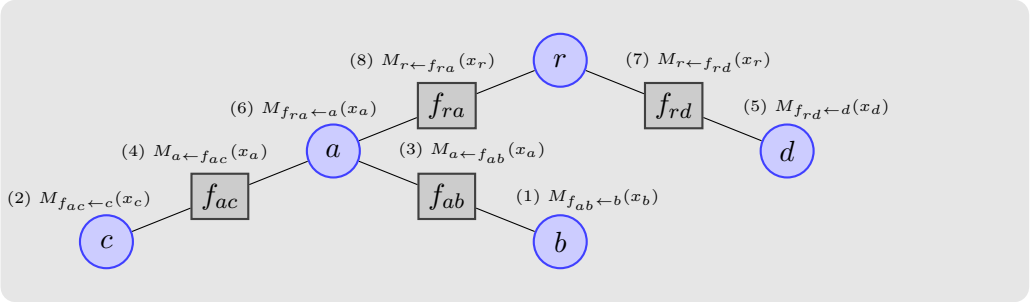


Figure 4.2.: Sequence of calculations for the example model in Figure 4.1b using dynamic programming. The messages $M_{*\leftarrow *}$ correspond to partial calculations of (4.23). The numbers in brackets give the order in which the messages have to be calculated.

Figure 4.2 sketch this sequence of calculations. We introduce an additional notation for the intermediate results

$$M_{a\leftarrow f_{ab}}(x_a) := \sum_{x_b\in\mathcal{X}_b} \left(f_{ab}(x_a, x_b) \cdot M_{f_{ab}\leftarrow b}(x_b) \right) \tag{4.24}$$

$$M_{f_{ab}\leftarrow a}(x_a) := \prod_{g\in \mathrm{ne}(a)\setminus\{f_{ab}\}} g(x_a), \tag{4.25}$$

which we call messages. The notation $M_{a\leftarrow f}(x_a)$ denotes that the message is sent from the factor node f to the variable node a and $M_{f\leftarrow a}(x_a)$ denotes that the message is sent from the variable node a to the factor node f.

This method is a Bottom-Up-dynamic programming approach. It divides the problem in a sequence of simpler subproblems. In addition to the computation of the marginal of one random variable, we can use a Top-Down-approach to compute the marginals for all single random variable by only twice the costs as for the calculations for one variable.

In the same manner, the marginals $p(x_{\mathrm{ne}(f)}|y)$ can be computed for all $f \in F$. In terms of messages this can be written as

$$p(x_{\mathrm{ne}(f)}|y) = f(x_{\mathrm{ne}(f)}) \cdot \prod_{a\in\mathrm{ne}(f)} M_{f\leftarrow a}(x_a) \quad \forall f \in F \tag{4.26}$$

This algorithm is known as the Sum-Product-Algorithm or Belief Propagation (BP) [123] – which should not be mixed up with Loopy Belief Propagation (LBP) which we will discussed in Section 4.2.3. Contrary to LBP, BP calculates each message only once. A message can be calculated, when all messages which are required for its calculation have been computed. If the model is cyclic, each cycle defines a 'waiting-cycle'[1] and consequently leads to a deadlock.

We can generalize this kind of dynamic algorithms for different commutative semirings and calculate different marginals.

- $([0,\infty), \max, \cdot) \rightarrow$ max-marginals
- $((-\infty,\infty], \min, +) \rightarrow$ min-marginals

[1] Each message in this cycle waits on another message in the cycle for its own calculation.

Algorithm 4.1 $\odot$-$\oplus$-Belief Propagation calculates the marginal distributions corresponding to the $\odot$-$\oplus$-semiring for any acyclic factor graph model.
In the first part it sequentially calculates all messages between neighbored nodes. These messages are then used to calculate the marginal distributions.

Require: Acyclic Factor Graph Model $(\oplus, G = (V, F, E))$
Ensure: $\forall A \in V \cup \{\mathrm{ne}(f) | f \in F\} : \quad b_A(x_A) = \bigodot_{x_{V \setminus A} \in \mathcal{X}_{V \setminus A}} \bigoplus_{f \in F} f(x_{\mathrm{ne}(f)})$

1: $M = \{(a, f) |\ f \in F, a \in \mathrm{ne}(f)\} \cup \{(f, a) |\ f \in F, a \in \mathrm{ne}(f)\}$
2: **while** $M \neq \emptyset$ **do**
3: Select $(z_1, z_2) \in M$ with $\forall z_3 \in \mathrm{ne}(z_2) \setminus \{z_1\} : (z_2, z_3) \notin M$
4: **if** $a = z_1 \in V$ and $f = z_2 \in F$ **then**
5: $M_{f \leftarrow a}(x_a) = \bigoplus_{g \in \mathrm{ne}(a) \setminus \{f\}} M_{a \leftarrow g}(x_a)$
6: **end if**
7: **if** $f = z_1 \in F$ and $a = z_2 \in V$ **then**
8: $M_{a \leftarrow f}(x_a) = \bigodot_{x_{\mathrm{ne}(f) \setminus \{a\}} \in \mathcal{X}_{\mathrm{ne}(f) \setminus \{a\}}} \left(f(x_{\mathrm{ne}(f)}) \oplus \bigoplus_{c \in \mathrm{ne}(f) \setminus \{a\}} M_{f \leftarrow c}(x_c) \right)$
9: **end if**
10: $M = M \setminus \{(z_1, z_2)\}$
11: **end while**
12: **for** $a \in V$ **do**
13: $b_a(x_a) = \bigoplus_{f \in \mathrm{ne}(a)} M_{a \leftarrow f}(x_a)$
14: **end for**
15: **for** $f \in F$ **do**
16: $b_{\mathrm{ne}(f)}(x_{\mathrm{ne}(f)}) = f(x_{\mathrm{ne}(f)}) \oplus \bigoplus_{a \in \mathrm{ne}(f)} M_{f \leftarrow a}(x_a)$
17: **end for**

- $([0, \infty), +, \cdot) \rightarrow$ marginals

The problem of finding the mode for an acyclic model can be reduce to the marginal problem. Since x^* is the optimum, x_a^* would also be an optimum for the marginal solution, i.e.

$$x_a^* \in \arg\max_{x_a \in \mathcal{X}_a} p^{\max}(x_a | y) \tag{4.27}$$

$$= \arg\min_{x_a \in \mathcal{X}_a} J^{\min}(x_a | y) \tag{4.28}$$

If we assume that the model is unimodal, the set of optima includes exactly one element. In general this is not the case and includes back tracking or a local search – see Algorithm 4.2.

Let us consider the general case for an arbitrary commutative semiring. Given an acyclic factor graph model $(\oplus, G = (V, F, E))$ together with an accumulative operation $\odot$ such that $(\Omega, \odot, \oplus)$ defines a commutative semiring. The marginal distribution according to this semiring can be calculated by Algorithm 4.1 in a dynamic programming style. This algorithm was introduced in a similar form in [2, 97]. Its idea is that the set M includes all messages which have to be sent. Iteratively, a messages which waits for no result of another message, is selected and sent. The selection of a message which is ready for sending can be implemented efficiently by sorting the messages into buckets, which indicate the number of messages for which they are waiting for.

To prove the correctness of Algorithm 4.1 we show that the algorithm terminates if and only if all messages are computed. Furthermore, we show that the computation of the marginals in the lines 13 and 16 are correct.

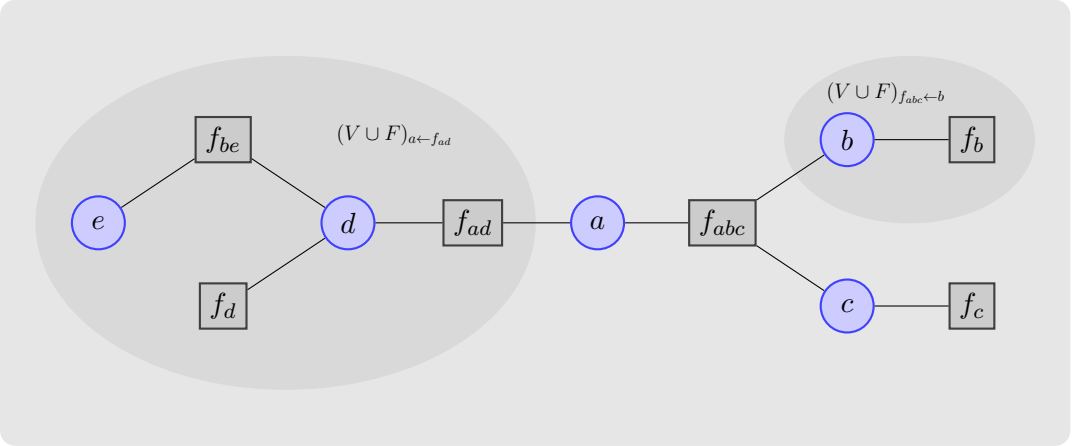


Figure 4.3.: Examples for sets $(V \cup F)_{z_1 \leftarrow z_2}$. The set of all nodes $n \in V \cup F$ for which the path from n to a includes f_{ad} are given by the set $(V \cup F)_{a \leftarrow f_{ad}} = \{e, d, f_{be}, f_d, f_{ad}\}$. As another example $(V \cup F)_{f_{abc} \leftarrow b} = \{b, f_b\}$.

Theorem 4.1. *For each acyclic factor graph $G = (V, F, E)$ Algorithm 4.1 converges if and only if all messages between neighbored nodes are calculated.*

Proof. Let us assume that Algorithm 4.1 does not converge. That means that in the set of non-calculated messages all messages still wait for the calculation of another message in M. This implies that the graph G has at least one cycle causing this blocking. If G is be acyclic, there has to be a node $a \in V \cup F$ with $ab \in M$, such that $M_{b \leftarrow a}$ can be calculated. Since we assume that G is acyclic the algorithm calculates all messages. □

Next we show that the calculation of the marginals in the lines 13 and 16 is correct. For this end, we have to introduce some further definitions. For all $a, b \in V \cup F$, let $F_{a \leftarrow b}$ be the set of all factor nodes $g \in F$ for which the unique path from g to a includes b and $V_{a \leftarrow b}$ the set of all variable nodes $c \in V$ for which the unique path from c to a includes b. The union of these two sets is denoted by $(V \cup F)_{a \leftarrow b}$. See Figure 4.3 for an illustration of these sets.

Now we can transform the recursive definition of messages

$$M_{f \leftarrow a}(x_a) := \bigoplus_{g \in \mathrm{ne}(a) \setminus \{f\}} M_{a \leftarrow g}(x_a) \tag{4.29}$$

$$M_{a \leftarrow f}(x_a) := \bigodot_{x_{\mathrm{ne}(f) \setminus \{a\}} \in \mathcal{X}_{\mathrm{ne}(f) \setminus \{a\}}} \left(f(x_{\mathrm{ne}(f)}) \oplus \bigoplus_{b \in \mathrm{ne}(f) \setminus \{a\}} M_{f \leftarrow b}(x_b) \right) \tag{4.30}$$

into an explicit form

$$M_{f \leftarrow a}(x_a) = \bigodot_{x_{V_{f \leftarrow a} \setminus \{a\}} \in X_{V_{f \leftarrow a} \setminus \{a\}}} \bigoplus_{g \in F_{f \leftarrow a}} g(x_{\mathrm{ne}(g)}) \tag{4.31}$$

$$M_{a \leftarrow f}(x_a) = \bigodot_{x_{V_{a \leftarrow f}} \in X_{V_{a \leftarrow f}}} \bigoplus_{g \in F_{a \leftarrow f}} g(x_{\mathrm{ne}(h)}). \tag{4.32}$$

The proof of the equivalence of the recursive and explicit form is given in the appendix (Theorem A.2) and uses a complete induction over the size of the subtree $|(V \cup F)_{a \leftarrow b}|$ from which the message is sent.

Finally, for the proof of correctness of Algorithm 4.1 we prove the correctness of the lines 13 and 16. Let $b_a(x_a)$ and $b_{\mathrm{ne}(f)}(x_{\mathrm{ne}(f)})$ be the marginals according to a $(\Omega, \odot, \oplus)$-semiring for a acyclic factor graph model $(\oplus, (V, F, E))$, then

$$b_a(x_a) := \bigodot_{x_{V\setminus\{a\}} \in \mathcal{X}_{V\setminus\{a\}}} \bigoplus_{f\in F} f(x_{\mathrm{ne}(f)}) \tag{4.33}$$

$$= \bigodot_{x_{V\setminus\{a\}} \in \mathcal{X}_{V\setminus\{a\}}} \bigoplus_{f\in \mathrm{ne}(a)} \left(\bigoplus_{g\in F_{a\leftarrow f}} g(x_{\mathrm{ne}(g)}) \right) \tag{4.34}$$

$$= \bigoplus_{f\in \mathrm{ne}(a)} \left(\bigodot_{x_{V_{a\leftarrow f}} \in X_{V_{a\leftarrow f}}} \bigoplus_{g\in F_{a\leftarrow f}} g(x_{\mathrm{ne}(g)}) \right) \tag{4.35}$$

$$= \bigoplus_{f\in \mathrm{ne}(a)} M_{a\leftarrow f}(x_a) \tag{4.36}$$

$$b_{\mathrm{ne}(f)}(x_{\mathrm{ne}(f)}) := \bigodot_{x_{V\setminus \mathrm{ne}(f)} \in \mathcal{X}_{V\setminus \mathrm{ne}(f)}} \bigoplus_{g\in F} g(x_{\mathrm{ne}(g)}) \tag{4.37}$$

$$= f(x_{\mathrm{ne}(f)}) \oplus \bigodot_{x_{V\setminus \mathrm{ne}(f)} \in \mathcal{X}_{V\setminus \mathrm{ne}(f)}} \left(\bigoplus_{a\in \mathrm{ne}(f)} \bigoplus_{g\in F_{f\leftarrow a}} g(x_{\mathrm{ne}(g)}) \right) \tag{4.38}$$

$$= f(x_{\mathrm{ne}(f)}) \oplus \bigoplus_{a\in \mathrm{ne}(f)} \left(\bigodot_{x_{V_{f\leftarrow a}\setminus\{a\}} \in \mathcal{X}_{V_{f\leftarrow a}\setminus\{a\}}} \bigoplus_{g\in F_{f\leftarrow a}} g(x_{\mathrm{ne}(g)}) \right) \tag{4.39}$$

$$= f(x_{\mathrm{ne}(f)}) \oplus \bigoplus_{a\in \mathrm{ne}(f)} M_{f\leftarrow a}(x_a). \tag{4.40}$$

We have shown, that the Algorithm 4.1 leaves the while-loop if and only if all massages have been computed and the calculation of the marginal distributions by local messages is correct. Overall this proves the correctness of Algorithm 4.1.

For the calculation of the optimal configuration x^* of $p(x|y)$, we first compute the max-marginal $p^{\max}(x_A|y)$ for all $A \in V \cup \{\mathrm{ne}(f) | f \in F\}$. If the optimal configuration is unique, we obtain x^* by $x^*_A = \arg\max_{x_A} p^{\max}(x_A|y)$. If the mode is not unique, i.e. when there are several configurations x for which $p(x|y) = p(x^*|y)$, then we start with an arbitrary node a and set x^*_a to an arbitrary element in $\arg\max_{x_a} p^{\max}_a(x_a|y)$. If we want to continue with the other nodes, we have to do this with respect to the previous decisions, in order to not switch between different modes. Thus we select a factor node which is adjacent to a processed node, i.e. for which at least one mode state is fixed. We condition the marginal distribution to the known mode states and select the mode of this conditioned marginal distribution. We repeat this procedure until all nodes are processed . The same procedure can be applied for the min-sum semiring. The algorithm for the max-product semiring is shown in Algorithm 4.2.

Since the most complex and dominant part of Algorithm 4.2 is calling Algorithm 4.1, both algorithms have the same asymptotic complexity. Overall we send over each edge two messages and the complexity of the calculation of such a message is in $O(|\mathcal{X}_{\mathrm{ne}(f)}|)$ and $O(|\mathrm{ne}(a)| \cdot L)$, respectively. Since the G is acyclic and bipartite we can bound the number of edges by $|E| \leq |F| + |V| \leq 3 \cdot |V|$ and give an asymptotic complexity bound by $O(|V| \cdot L^o + |V|^2 \cdot L)$.

Algorithm 4.2 Computing a mode x^* of $p(x|y) = \prod_{f \in F} f(x_{\mathrm{ne}(f)})$ for acyclic factor graph model $(\cdot, G = (V, F, E))$. The mode is calculated sequentially with respect to the previous decisions.

Require: Connected acyclic factor graph model $(\cdot, G = (V, F, E))$
Ensure: $x^* \in \arg\max_{x \in \mathcal{X}} p(x|y)$
  Run Algorithm 4.1 with $([0, \infty), \max, \cdot)$ on $(\cdot, G)$
  Select $a \in V$
  Select $x_a^* \in \arg\max_{x_a \in \mathcal{X}_a} p^{\max}(x_a|y)$
  for all $f \in \mathrm{ne}(a)$ **do**
    Append (f, a) at the end of the list L
  end for
  while L is not empty **do**
    Pop (f, a) from the front of L
    Select $x^*_{\mathrm{ne}(f)} \in \arg\max_{x_{\mathrm{ne}(f)} \in \mathcal{X}_{\mathrm{ne}(f)},\, x_a = x_a^*} p^{\max}(x_{\mathrm{ne}(f)}|y)$
    for all $b \in \mathrm{ne}(f) \setminus \{a\}$ **do**
      for all $g \in \mathrm{ne}(b) \setminus \{f\}$ **do**
        Append (g, b) at the end of L
      end for
    end for
  end while

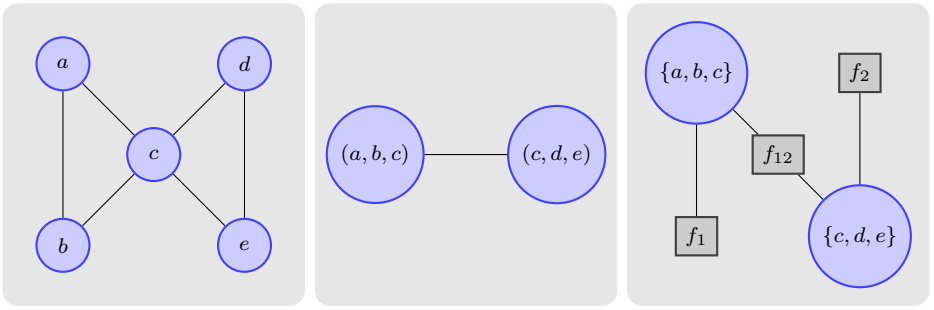
For further speed up one can avoid sending messages to factor nodes which depend only on one variable or parallelize the calculation of the messages. For parallel calculation a dependency graph has to be take into account. Furthermore, for models of the order two, messages between two variable nodes can be sent directly, without storing the message from variable node to factor node.

4.2.2. Junction Tree Algorithm

We have seen that dynamic programming is a very powerful tool if the graphical model is acyclic. However, when the model is cyclic the BP algorithms cannot be applied directly. In this section we will show that dynamic programming can also be applied on cyclic graphs as part of the Junction Tree Algorithm (JTA). The main idea of the JTA is to eliminate cycles by clustering the nodes of these cycles into super-nodes. This super-nodes correspond to nodes in the junction tree. We can apply dynamic programming algorithms to this junction tree model instead of inferring on the cyclic graph. Since the complexity of the junction tree algorithm is too large for our problems – same as brute force for fully connected models – we give only a brief outline of the junction tree algorithm and refer to [106] for more details.

Assume that $(X, G = (V, E))$ is a cyclic undirected graphical model. To construct the junction tree to G, G has to be chordal. If G is not chordal we add edges to G to make it chordal. From a statistical point of view this means, that we drop conditional independences of our model.

Without loss of generality we can assume that G is chordal and construct the corresponding junction tree $T = (V_T, E_T)$. We call the nodes in a junction tree super-nodes and denote them by capital letter, since each super-nodes is a maximal clique in the triangulated



(a) (b) (c)

Figure 4.4.: The figures above illustrates the construction of a junction tree factor graph model from a undirected model. The original cyclic model graph G, shown in **(a)**, implies a factorization $\mathbf{f}(x) = f_{abc}(x_{abc})f_{cde}(x_{cde})$. We construct the junction tree to this graph **(b)**. The factor graph **(c)** include 3 factors. While $f_{abc} = f_1$ and $f_{cde} = f_2$, f_{12} ensures the consistency in x_c.

graph G, see Definition 2.6. The random variable X_A, that corresponds to the super-node $A \in V_T$ takes values in $\mathcal{X}_A$. This is an equivalent representation of the random vector X_A for $A \subset V$ in the original model (X, G). By definition the node sets corresponding to two super-nodes $A, B \in V_T$ has not to be disjunct. If $A \cap B \neq \emptyset$, then there exists a unique path between A and B in T, such that each node C within this path includes $A \cap B$ as subset. This observation is very important and ensures consistency between the random variables in the junction tree model.

We define the junction factor graph model by $M' = (X', G' = (V', F', E'))$ with

$$V' := \{A | A \in \mathcal{C}(G)\} \tag{4.41}$$
$$F' := \{f_A | A \in V'\} \cup \{f_{AB} | AB \in E_T\} \tag{4.42}$$
$$E' := \{\{A, f_A\} | A \in V_T\} \cup \{\{A, f_{AB}\}, \{B, f_{AB}\} | AB \in E_T\} \tag{4.43}$$
$$X' := (X_A)_{A \in V'} \tag{4.44}$$
$$\mathcal{X}_A := \bigotimes_{a \in A \subset V} \mathcal{X}_a \tag{4.45}$$

Since the triangulated model (X, G) factorizes into

$$p(x|y) = \frac{1}{Z} \bigotimes_{C \in \mathcal{C}(G)} f_C(x_C) \tag{4.46}$$

each factor can be relocated into the unary factor f_C of the factor graph model. The consistency constraint of the duplicated random variables can be coded it the pairwise factor functions.

$$f_{AB}(x_A, y_B) = f_{AB}(x_{A \cap B}, y_{A \cap B}) = \begin{cases} \mathbf{1}_\oplus & \text{if } x_{A \cap B} = y_{A \cap B} \\ \mathbf{1}_\odot & \text{else} \end{cases} \tag{4.47}$$

Here $\mathbf{1}_\odot$ denotes the neutral element of the $\odot$-operation and $\mathbf{1}_\oplus$ the neutral element of the $\oplus$-operation.

A simple example is shown in Figure 4.4. However, it is not obviously that this additional functions ensure that all consistency constraints are fulfilled. In Figure 4.5 we show such a

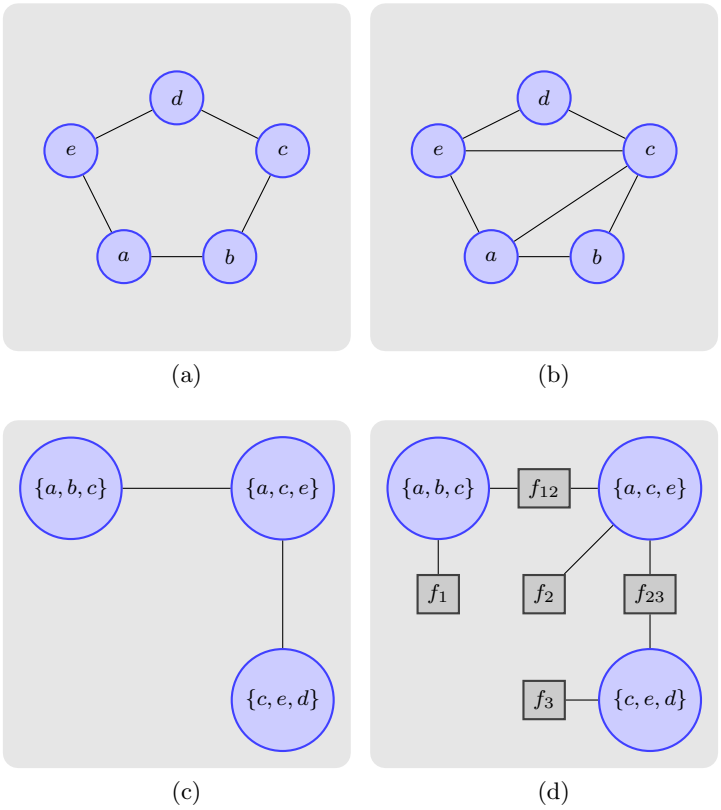


Figure 4.5.: For non-chordal graphs **(a)** in the first step the graph has to be triangulated. For the triangulated graph **(b)** we calculate the junction tree **(c)**. The consistency between the nodes is ensured by the pairwise functions f_{12} and f_{23} in the factor graph **(d)**, the objective function is included in the unary factors of the factor graph.

non-obvious example. The nodes $\{a, b, c\}$ and $\{c, d, e\}$ share c but there is no edge between the nodes in the junction tree in Figure 4.5c which ensures the consistency constraint. Since by definition, the intersection of two nodes in a junction tree is a subset of any node on the path between this nodes, the transitivity of the equivalence relation ensures that the model consistency. The pseudo code of the JTA is given in Algorithm 4.3.

The main trick for fast calculation is that the pairwise functions only depend on a small subset of variables and describe an equality constraint. Let us consider the message $M_{A \leftarrow f}(x_A)$; if the factor f connects the nodes A and B and their intersection is $C = A \cap B$, then the calculation of the message can be simplified by

$$M_{A \leftarrow f}(x_A) = M_{A \leftarrow f}(x_C) = \bigoplus_{x_{B \setminus C}} M_{f \leftarrow B}(x_B) \tag{4.48}$$

The alternative calculation of messages in (4.48) reduces the complexity of the JTA to $O(|V'|^2 \cdot L^{\mathrm{tw}(G)})$. For model graphs with large tree width the JTA is not feasible. However, the JTA can be applied for any graphical model and solves theoretically the inference problems faster than naive brute force approaches by considering the graph structure.

Algorithm 4.3 Computing a mode x^* of $(X, G = (V, E))$

Require: $(X, G = (V, E))$
Ensure: $y \in \arg\max_{x \in \mathcal{X}} p(x)$
  Calculate a triangulation of G
  Calculate the junction tree $T = (V_T, E_T)$ according to this triangulated graph
  Build the factor graph model $(X', (V', F', E'))$
  Run Algorithm 4.2 on $(X', (V', F', E'))$

4.2.3. Loopy Belief Propagation

If we would use the dynamic programming as suggested in Algorithm 4.1 on a cyclic model, the algorithm gets stuck in the while-loop, because each message in a cycle waits for another one in this cycle. If we would like to avoid this deadlock, we can send each message in each step in a parallel manner based on the messages from the last step. For acyclic models this approach will also lead to the correct solution after several iterations – of course with a substantial computational overhead. The main advantage of this parallel schedule strategy is, that for cyclic models the message passing does not get stuck in an endless loop. From the algorithmic point of view this parallel message passing on a cyclic graph is rather heuristic, because the change of the outgoing message change the incoming messages on which the calculation is based. However, as suggested by Pearl [125], many researches have used this heuristic method. Surprisingly, this method, known as loopy belief propagation (LBP), gives empirical good results. The the LBP-pseudo code for a factor graph model and an arbitrary commutative semiring is given in Algorithm 4.4.

In practice it is important to normalize the messages for numerical stability, otherwise overflow or underflow are likely to occur after a number of message updates. We normalize each message by a scalar κ, such that $\bigodot_{x_a \in \mathcal{X}_a} M_{a \leftarrow f}(x_a) = \mathbf{1}_\oplus$ and $\bigodot_{x_a \in \mathcal{X}_a} M_{f \leftarrow a}(x_a) = \mathbf{1}_\oplus$, where $\mathbf{1}_\oplus$ is the neutral element of $\oplus$. Consequently, we can only calculate relative marginals with respect to some normalization. For the sum-product semiring this normalization ensures that each marginal distribution sum up to one. For the min-sum semiring and the max-prod semiring it ensures that the minimal and maximal value are 0 and 1, respectively.

The update of the messages is repeated until the maximum number of iterations is achieved or the change of the messages from the last to the current step is below some threshold – we use $\epsilon = 10^{-7}$. If the messages do not change any more, LBP is converged and further iterations will not have any effect on the solution. However, in general LBP does not converge and if it converges this does not imply that the solution is optimal. Since LBP use an outer approximation the resulting marginals are called pseudo marginals. They are only an approximation of the true marginals.

A main problem of LBP is that it usually will not converge and the messages will start to oscillate. Murphy et al. [119] suggest to damp the massages by replacing the message sent at step t by a weighted average of the messages in step t and $t-1$. This causes that each message store some history and do not run greedily in some oscillating sequence. Let $\circledast$ be the hyper operation of $\oplus$, e.g. multiplication for addition and power for multiplication, then the weighted average of a and b is calculated by

$$a \circledast (1 - \alpha) \oplus b \circledast \alpha$$

with $\alpha \in [0, 1]$. Even if this method leads to empirical better results it does not guarantee convergence.

Algorithm 4.4 $\odot$-$\oplus$-Loopy Belief Propagation is a heuristic expansion of the BP for cyclic models. In each step all messages are computed in parallel based on the last iteration.

Require: Factor Graph Model $(\oplus, G = (V, F, E))$
Ensure: $\forall A \in V \cup \{\mathrm{ne}(f) | f \in F\}: \quad b_A(x_A) \approx \bigodot_{x_{V \setminus A} \in \mathcal{X}_{V \setminus A}} \bigoplus_{f \in F} f(x_{\mathrm{ne}(f)})$

$\forall af \in E: \quad M^0_{f \leftarrow a}(x_a) = \mathbf{1}_\oplus \quad M^0_{a \leftarrow f}(x_a) = \mathbf{1}_\oplus$
$t = 0$
repeat
  $t = t + 1$
  for all $af \in E$ **do**
    $M^t_{f \leftarrow a}(x_a) = \kappa_{fa} \oplus \bigoplus_{g \in \mathrm{ne}(a) \setminus \{f\}} M^{(t-1)}_{a \leftarrow g}(x_a)$
  end for
  for all $af \in E$ **do**
    $M^t_{a \leftarrow f}(x_a) = \kappa_{af} \oplus \bigodot_{x_{\mathrm{ne}(f) \setminus \{a\}} \in \mathcal{X}_{\mathrm{ne}(f) \setminus \{a\}}} \left(f(x_{\mathrm{ne}(f)}) \oplus \bigoplus_{c \in \mathrm{ne}(f) \setminus \{a\}} M^{(t-1)}_{f \leftarrow c}(x_c) \right)$
  end for
until $t > t_{max}$ or $\|M^{(t-1)} - M^t\|_\infty \leq \epsilon$
for all $a \in V$ **do**
  $b_a(x_a) = \kappa_a \oplus \bigoplus_{f \in \mathrm{ne}(a)} M^t_{a \leftarrow f}(x_a)$
end for
for all $f \in F$ **do**
  $b_{\mathrm{ne}(f)}(x_{\mathrm{ne}(f)}) = \kappa_f \oplus f(x_{\mathrm{ne}(f)}) \oplus \bigoplus_{a \in \mathrm{ne}(f)} M^t_{f \leftarrow a}(x_a)$
end for

For graphical models of order less than two, we can further simplify LBP and send messages only from node to node instead of sending messages over factor nodes. To simplify the notation let us represent the factorization by an undirected graph $G = (V, E)$ as $\mathbf{f}(x) = \bigoplus_{a \in V} f_a(x_a) \oplus \bigoplus_{ab \in E} f_{ab}(x_a, x_b)$. This representation of second order graphical models is common in the literature and also is used in Chapter 3. Using this representation we can rewrite Algorithm 4.5 for second order models into Algorithm 4.6 on the corresponding undirected model with the graph $G = (V, E)$.

For decades it was even unclear which objective LBP optimizes. Yedida et al. [186, 187] show, that fix points of the sum-product-LBP coincide with stationary points of the Bethe variational problem, which dates back to the work of Bethe [15]. Furthermore, they show that LBP can be generalize to so called Kikuchi-approximations, see [173] for a detailed discussion. The main operation of LBP is sending messages between the nodes of the graph. This messages turn out to be directly related with the Lagrange multipliers of the Bethe variational problem and can be understood as a re-weighting of the objective function [182, 183]. However, in general the Bethe variational problem is not convex. We will consider LBP again from the viewpoint of variational inference in Section 4.3.2.

4.2.4. Graph-Cuts

In the last decade, graph cut algorithms have become a state of the art method for optimizing important classes of energy functions in computer vision. The main idea of this algorithms is to reduce the energy minimization problem to a single or a sequence of min-cut problems. Graph cut methods for inference are not restricted to acyclic models, but require a special type of objective function. As we will see in this section the restrictions are so hard, that we can not apply graph cut-based algorithms on our inference problems.

Algorithm 4.5 Damped version of $\odot$-$\oplus$-Loopy Belief Propagation. The damping of the message updates decreases the oscillating of the messages, such that LBP converges more often.

Require: Factor Graph Model $(\oplus, G_F = (V, F, E))$
Ensure: $\forall A \in V \cup \{\mathrm{ne}(f) | f \in F\} : \quad b_A(x_A) \approx \bigodot_{x_{V\setminus A} \in \mathcal{X}_{V\setminus A}} \bigoplus_{f\in F} f(x_{\mathrm{ne}(f)})$

$\forall af \in E : \quad M^0_{f\leftarrow a}(x_a) = \mathbf{1}_\oplus \quad M^0_{a\leftarrow f}(x_a) = \mathbf{1}_\oplus$
$t = 0$
repeat
  $t = t + 1$
  for all $af \in E$ **do**
    $M^t_{f\leftarrow a}(x_a) = \kappa_{fa} \oplus \bigoplus_{g\in \mathrm{ne}(a)\setminus\{f\}} M^{(t-1)}_{a\leftarrow g}(x_a)$
    $M^t_{f\leftarrow a}(x_a) = M^t_{f\leftarrow a}(x_a) \circledast (1-\alpha) \oplus M^{(t-1)}_{f\leftarrow a}(x_a) \circledast \alpha$
  end for
  for all $af \in E$ **do**
    $M^t_{a\leftarrow f}(x_a) = \kappa_{af} \oplus \bigodot_{x_{\mathrm{ne}(f)\setminus\{a\}} \in \mathcal{X}_{\mathrm{ne}(f)\setminus\{a\}}} \left(f(x_{\mathrm{ne}(f)}) \oplus \bigoplus_{c\in \mathrm{ne}(f)\setminus\{a\}} M^{(t-1)}_{f\leftarrow c}(x_c) \right)$
    $M^t_{a\leftarrow f}(x_a) = M^t_{a\leftarrow f}(x_a) \circledast (1-\alpha) \oplus M^{(t-1)}_{a\leftarrow f}(x_a) \circledast \alpha$
  end for
until $t > t_{max}$ or $\|M^{(t-1)} - M^t\|_\infty \leq \epsilon$
for all $a \in V$ **do**
  $b_a(x_a) = \kappa_a \oplus \bigoplus f \in \mathrm{ne}(a) M^t_{a\leftarrow f}(x_a)$
end for
for all $f \in F$ **do**
  $b_{\mathrm{ne}(f)}(x_{\mathrm{ne}(f)}) = \kappa_f \oplus f(x_{\mathrm{ne}(f)}) \oplus \bigoplus_{a\in \mathrm{ne}(f)} M^t_{f\leftarrow a}(x_a)$
end for

Algorithm 4.6 Damped $\odot$-$\oplus$-Loopy Belief Propagation for second order undirected models. Contrary to LBP for factor graph models this algorithm sends messages directly between nodes.

Require: Undirected graph $G = (V, E)$ such that $\mathbf{f}(x) = \bigoplus_{a\in V} f_a(x_a) \oplus \bigoplus_{ab\in E} f_{ab}(x_a, x_b)$
Ensure: $\forall A \in V \cup E : \quad b_A(x_A) \approx \bigodot_{x_{V\setminus A} \in \mathcal{X}_{V\setminus A}} \mathbf{f}(x)$

$\forall ab \in E : \quad M^0_{b\leftarrow a}(x_b) = \mathbf{1}_\oplus \quad M^0_{a\leftarrow b}(x_a) = \mathbf{1}_\oplus$
$t = 0$
repeat
  $t = t + 1$
  for all $ab \in E$ **do**
    $M^t_{a\leftarrow b}(x_a) = \kappa_{ab} \oplus \bigodot_{x_b\in\mathcal{X}_b} f_b(x_b) \oplus f_{ab}(x_a, x_b) \oplus \bigoplus_{c\in \mathrm{ne}(b)\setminus\{a\}} M^{(t-1)}_{b\leftarrow c}(x_b)$
    $M^t_{a\leftarrow b}(x_a) = M^t_{a\leftarrow b}(x_a) \circledast (1-\alpha) \oplus M^{(t-1)}_{a\leftarrow b}(x_a) \circledast \alpha$
  end for
until $t > t_{max}$ or $\|M^{(t-1)} - M^t\|_\infty \leq \epsilon$
for all $a \in V$ **do**
  $b_a(x_a) = \kappa_a \oplus f_a(x_a) \oplus \bigoplus_{c\in \mathrm{ne}(a)} M^t_{a\leftarrow c}(x_a)$
end for
for all $ad \in E$ **do**
  $b_{ad}(x_{\mathrm{ne}(f)}) = \kappa_f \oplus f_a(x_a) \oplus f_d(x_d) \oplus f_{ad}(x_a, x_d) \oplus$
    $\bigoplus_{c\in \mathrm{ne}(d)\setminus\{a\}} M^t_{d\leftarrow c}(x_d) \oplus \bigoplus_{c\in \mathrm{ne}(a)\setminus\{d\}} M^t_{a\leftarrow c}(x_a)$
end for

However, due to its general importance in the field of computer vision and the fact that we can solve submodular problems of low order exactly, we will give a rough overview of the variants of graph cut algorithms in computer vision.

The st-min-cut problem is a classical graph theoretical problem. Given a directed weighted graph $G = (V, E, w)$ with a source $s \in V$ and a sink $t \in V$. A st-cut in this graph is a subset of edges $E_C \subset E$ such that each directed path from s to t contains at least one edge in E_C. In other words, if we remove the edges E_C from G it exists no directed path from s to t. The minimal cut E_C^* is the cut with the minimal weight. The weight of a cut is the sum of the weights of its edges. If all edge weights are non-negative the minimal cut can be calculated in polynomial time.

The dual of the min-cut problem is the max-flow problem. The maximal flow is the maximal capacity which can be transported from the source to the sink. The weights of the edges can be understood as the maximal capacity which can be transported trough this edge. The maximal flow is limited by the edges included in the minimal cut. Contrary, if the maximal flow uses the full capacity of an edge, than this edge is included in the minimal cut. Consequently we can solve the max-flow problem to determine the minimal cut. This result is also known as the Max-Flow Min-Cut Theorem [39, 54].

Several algorithms has been suggested to solve these problems, which cluster in two main classes. The Ford-Kruskal algorithm [55] is a representative of algorithms based on augmenting paths, while the Goldberg-Tarjan algorithms [60] is a member of the so call Push relabel algorithms. For a comparison of min-cut/max-flow algorithms for energy minimization in computer vision we refer the reader to [19].

The first use of graph cuts in computer vision goes back to the work of Greig et al. [62] in 1989. They transform the problem

$$\min_{x \in \{0,1\}^{|V|}} \sum_{i \in V} f_i(x_i) + \sum_{ij \in E} \beta_{ij} |x_i - x_j| \tag{4.49}$$

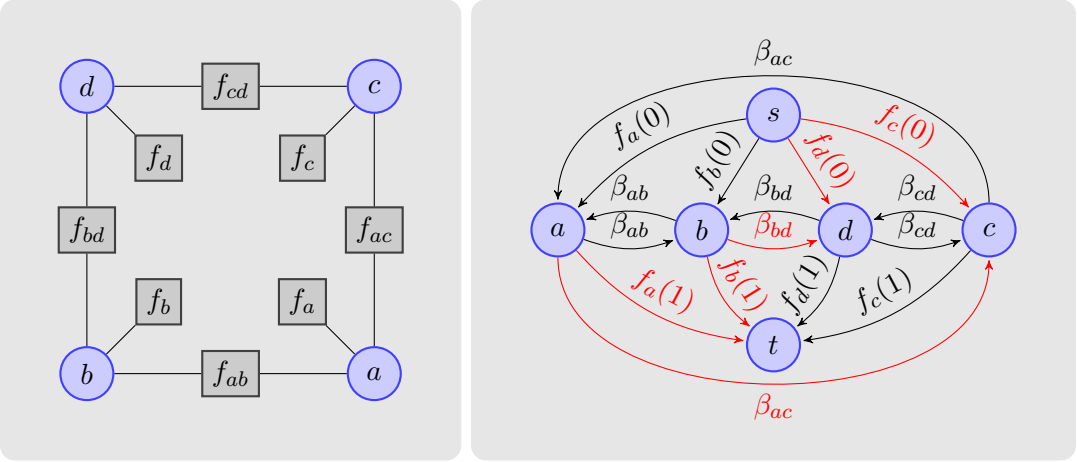
into a st-min-cut-problem. This fundamental work is the basis of several advanced algorithms that appeared in the last decade. Figure 4.6 illustrates the transformation of this binary labeling problem into a st-min-cut problem. The directed edges from s and to t encode the unary part of the energy functions, the remaining edges the pairwise terms. Each minimal cut which separates s from t corresponds exactly to one labeling. The variable x_a is labeled 0 if the cut includes the edge (s, a) and 1 if it includes (a, t). Obviously either (s, a) or (a, t) have to be included in the path. Otherwise there exists a path from s to t. If both edges (s, a) and (a, t) are contained in the cut one can be removed from the cut, since either all passes to t or from s are blocked by the cut.

A more general view on second order binary labeling problems is given by Kolmogorov et al. [91]. They show that it is possible to transform the minimization problem into a st-min-cut problem with non-negative weights if and only if for all edges $ab \in E$ the inequality

$$f_{ab}(0,0) + f_{ab}(1,1) \leq f_{ab}(0,1) + f_{ab}(1,0) \tag{4.50}$$

holds.

To sketch the proof we define a matrix $A \in \mathbb{R}^{2\times 2}$ with $A_{ij} = f_{ab}(i, j)$ and show that we can decompose this matrix into a sum of matrices which directly corresponds to the directed



(a) Factor Graph G_F (b) Min-Cut Graph G_C

Figure 4.6.: Figure **(a)** show the a toy factor graph. If we assume that the variables are binary and $f_{uv}(x_u, x_v) = \beta_{uv} \cdot |x_u - x_v|$, then the problem of finding the labeling that minimizes $J(x) = \sum_{f \in F} f(x)$ can be solved by finding the min cut on the graph shown in figure **(b)**. The variable x_a is labeled 0 if the min cut includes the edge sa. The cut marked in red corresponds to the labeling $x = (1, 1, 0, 0)$.

edges.

$$A = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} = \begin{pmatrix} 0 & \beta - \alpha \\ \gamma - \alpha & \delta - \alpha \end{pmatrix} + \begin{pmatrix} \alpha & \alpha \\ \alpha & \alpha \end{pmatrix} \tag{4.51}$$

$$= \begin{pmatrix} 0 & 0 \\ \gamma - \alpha & \delta - \beta \end{pmatrix} + \begin{pmatrix} 0 & \beta - \alpha \\ 0 & \beta - \alpha \end{pmatrix} + \begin{pmatrix} \alpha & \alpha \\ \alpha & \alpha \end{pmatrix} \tag{4.52}$$

$$= \underbrace{\begin{pmatrix} 0 & 0 \\ \gamma + \beta - \alpha - \delta & 0 \end{pmatrix}}_{\to w_{ab}} + \underbrace{\begin{pmatrix} 0 & 0 \\ \delta - \beta & \delta - \beta \end{pmatrix}}_{\to w_{sa}} + \underbrace{\begin{pmatrix} 0 & \beta - \alpha \\ 0 & \beta - \alpha \end{pmatrix}}_{\to w_{bt}} + \underbrace{\begin{pmatrix} \alpha & \alpha \\ \alpha & \alpha \end{pmatrix}}_{\to const.} \tag{4.53}$$

If either $\delta - \beta$ or $\beta - \alpha$ is negative one can use a substitution of the form

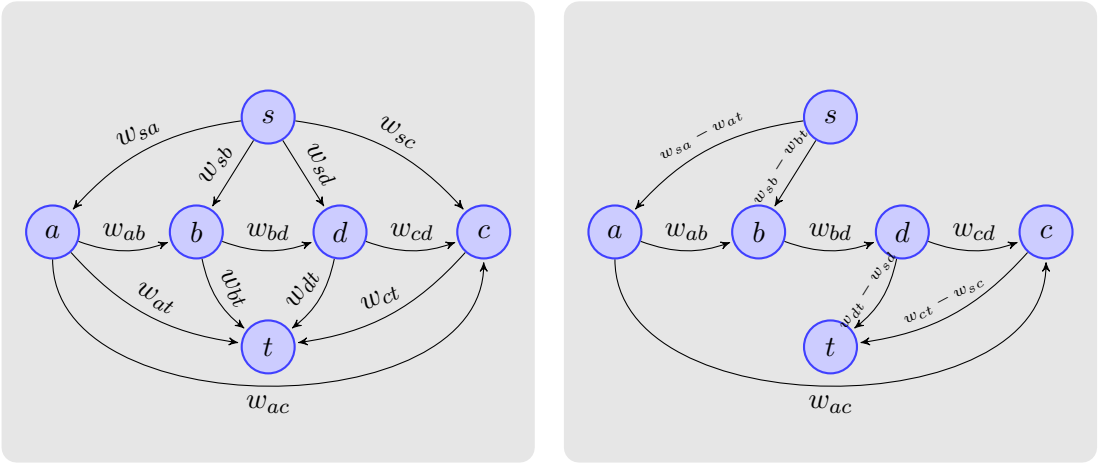
$$\underbrace{\begin{pmatrix} 0 & 0 \\ \delta - \beta & \delta - \beta \end{pmatrix}}_{\to w_{sa}} = \underbrace{\begin{pmatrix} \delta - \beta & \delta - \beta \\ \delta - \beta & \delta - \beta \end{pmatrix}}_{\to const} + \underbrace{\begin{pmatrix} \beta - \delta & \beta - \delta \\ 0 & 0 \end{pmatrix}}_{\to w_{at}} \tag{4.54}$$

to ensure positive weights. If also $\gamma + \beta - \alpha - \delta$ is non-negative we can use (4.53) and (4.54) to construct the corresponding st-cut-graph. The proof that there exists no non-negative st-cut-graph if $\gamma + \beta - \alpha - \delta < 0$ can be found in [91].

Applying the above rules ends up in a graph shown in Figure 4.7a. We can further simplify this st-min-cut problem by pushing as much flow for each $a \in V$ from s over a to t, such that one of the two edges is saturated. The resulting graph is shown in Figure 4.7b.

In [91] the functions having the property (4.50) are called regular functions and it is pointed out that there is a strong correspondences to the definition of submodularity.

Definition 4.1. A function $f : \mathcal{I} \to \mathbb{R}$ with $\mathcal{I} = \mathcal{I}_1 \times \ldots \times \mathcal{I}_d$ is submodular if for every $x, y \in \mathcal{I}$, $f(x \wedge y) + f(x \vee y) \leq f(x) + f(y)$. The join and meet operations are defined for



(a) Min-Cut Graph G_F (b) Min-Cut Graph G_C

Figure 4.7.: By using (4.54) one can transform a graph such that only one directed edge between all nodes $a, b \in V \setminus \{s, t\}$ exists. In a second step we push for each $a \in V \setminus \{s, t\}$ as much flow from s over a to t such that one of the to edges on the path is saturated and the other remains with a positive weight. The resulting graph is shown in **(b)**. It includes less edges than the graph in **(a)**.

every $x, y \in \mathcal{I}$ by

$$(x_1, \ldots, x_d) \wedge (y_1, \ldots, y_d) := (\max\{x_1, y_1\}, \ldots, \max\{x_d, y_d\}) \tag{4.55}$$

$$(x_1, \ldots, x_d) \vee (y_1, \ldots, y_d) := (\min\{x_1, y_1\}, \ldots, \min\{x_d, y_d\}) \tag{4.56}$$

where the max- and min-operation are used with respect to some orderings on $\mathcal{I}_d$.

Roughly spoken, submodularity can be understood as the discrete analogues to convexity. It is well known, that submodular functions can be minimized in polynomial time [120]. To our knowledge the fastest algorithm [120] that minimizes submodular functions has the complexity $O(n^5T + n^6)$ [2]. Since for computer vision problems n is too large, this algorithm for general submodular functions is computationally infeasible.

An alternative definition of submodularity for second order functions, which is much more intuitive in our context, makes use of Monge matrix.

Definition 4.2. A matrix $A \in \mathbb{R}^{N \times M}$ is called a Monge matrix if for every $1 \leq i < i' \leq N$ and $1 \leq j < j' \leq M$ we have $A_{i,j} + A_{i',j'} \leq A_{i,j'} + A_{i',j}$.

Each discrete second order function $g : L_1 \times L_2 \to \mathbb{R}$ can be represented by some matrix A. Theorem 4.2 shows that the function g is submodular if and only if A is a Monge matrix.

Theorem 4.2. *Let $L_1 = \{1, \ldots, N\}$ and $L_2 = \{1, \ldots, M\}$ be finite sets and $g : L_1 \times L_2 \to \mathbb{R}$ be a real-valued function defined by a matrix A by $f(i, j) = A_{ij}$. Then g is submodular if and only if A is a Monge matrix.*

[2] For definition of T and n see [120]. Due to a different definition of submodularity it is not straight forward to explain the meaning of n with our definition and will be skipped at this point to avoid further definitions

Proof. See [121] □

The submodularity of a function obviously depends on the choice of the ordering function. In this context Schlesinger [141] introduce the term of permuted submodular functions. A function is called permuted submodular if there exists an ordering such that the permuted function is submodular. An example is shown in (4.57). The matrix A is no Monge matrix. If we change the order of the rows we get the matrix B which is a Monge matrix.

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{4.57}$$

Even if each single potential-function in an energy-function is permuted submodular, this does not imply, that the energy-function itself is permuted submodular. While the sum of two submodular functions is still submodular [91] the sum of two permuted submodular functions is only permuted submodular if the functions share are permutation that make them submodular. Even for non-submodular functions approximations exist which are based on the roof duality [67]. We refer the reader to [137, 89] for more details in this context.

Also problems with non-binary variables can be solved by a min-cut. For discrete submodular functions Schlesinger [141] has shown that the minimization problem can be transformed into a submodular binary problem. The additional nodes in the constructed graph grows with the number of states of the discrete variables.

The restriction to submodular functions is too hard in most computer vision problems. Boykov et al. [20] introduce two algorithms which solve the MAP-problem by a sequence of min-cut problems to local optimality with less restrictions on the objective. Each of this sub-sequential problems calculates an optimal update on the current integer solution, which guarantee to decrease the energy. This update is also known as move. For the α-expansion the set of valid moves are moves, which change the setting of variable into α, while by the $\alpha\beta$-swap a valid move can change the setting of variables, which are set to α into β and vice versa. The algorithms stop if for all possible moves no improvement can be obtained any more. Although these two methods do not solve the problem exactly, they require only that the pairwise functions in the energy are a semi-metric ($\alpha\beta$-swaps) or a metric (α-expansion), which is fulfilled by several important computer vision problems. Komodakis et al. [95, 96, 92] developed the FastPD algorithm for semi-metric problems. This primal-dual method includes the α-expansion algorithm as a special case and gives a lower bound on the optimal energy. A mentionable note on this work is given by Kolmogorov [88]. His notation is more consistent with ours. Furthermore, functions with terms of higher order are also considered in the current graph cut literature, see e.g. [130, 85].

4.2.5. A^*- Search

Let us consider the problem of finding the mode of an energy function

$$J(x) = \sum_{f \in F} f(x_{\mathrm{ne}(f)}). \tag{4.58}$$

Searching over all possible configurations in $\mathcal{X}$ is impossible, because $\mathcal{X}$ is too large. Even for small problems with 5 nodes and 20 labels, $\mathcal{X}$ contains 64 million configurations.

However, it is not necessary to evaluate all $x \in \mathcal{X}$ if we can excluded configurations by a lower bound on their energy.

This idea is called branch and bound strategy, also used in mixed integer programming, see Section 4.2.6. Starting with the full set $\mathcal{X}$, in each iteration the subset with the smallest lower bound is decomposed into two ore more subsets. Then for each of these new subsets $S \subset \mathcal{X}$ a lower bound B is calculated, such that $B \leq \min_{x \in S} J(x)$ and $B = J(x)$ if $|S| = 1$. This procedure is repeated until the best subset S contains only one configuration. By definition the bound of the set S is exact if $|S| = 1$. Furthermore, all other configurations are included in other subsets, which are lower bounded by a value which is greater than or equal to the bound of S. Consequently, S includes a mode x^* of the energy function.

However, while this procedure looks very simple, its performance depends on tight bounds on the energies of the subsets and a good decomposition strategy. If for example, the sets are decomposed into n subsets of equal size, the bounds are tight, i.e. $B = \min_{x \in S} J(x)$, and the model unimodular, than the algorithm converges after $\lceil \log_n(|\mathcal{X}|) \rceil$ steps. Since we will not be able to calculate tight bounds, some gap between the calculated and optimal bound will remain. This gap causes that more than the $\lceil \log_n(|\mathcal{X}|) \rceil$ expansion steps may be required. However, when only some configurations have a similar energy to the mode and the most configurations have a significant higher energy, less tight bounds will cause less additional expansions steps.

Next, we introduce an algorithm for solving the MAP-problem which makes use of this branch and bound strategy. The algorithm is based on a best-first-search also known as A^*-search introduced by Hart [69] in 1968.

Related Work

A^*-algorithms for graphical models are strongly related to branch and bound algorithms used in mixed integer programming. Instead of evaluating bounds by solving linear programs, which relax the integer constraints, our A^*-approach calculates bounds by dynamic programming. For applications in computer vision we refer to e.g. [50, 32, 126, 47]. These approaches use Gaussian models and heuristics based on special properties of these models. More recently, Schlesinger [142] and Tian et al. [160] makes use of the branch and bound technique. While Schlesinger uses linear problems for bound calculations, Tian et al. use dynamic programming, but a different branching strategy compared to our A^*-approach.

The performance of the A^*-algorithm depends on devising a heuristic which estimates the minimal costs of an unexplored paths from the current node to a terminal node. Asymptotic bounds on the run-time of the A^*-algorithm can be given in some cases. For example if the estimated error does not grow faster than the logarithm of the true minimal costs, then the time complexity is polynomial [122, 138]. However, it is not clear how such statements can be obtained in practice, where such technical conditions are not fulfilled.

Problem Transformation

We will now transform the MAP-problem into the problem of calculating the shortest path between the nodes s and t in a directed weighted search-graph $G_S = (V_S, E_S, w)$. The cost of a path from a node v to a node u is the sum of the weights of the edges along the directed path from v to u. We assume, that the model is given by a factor graph $G = (V, F, E)$, and the set of nodes $V = \{1, \ldots, N\} \subset \mathbb{N}$ has a fixed order. Without

loss of generality, we can assume, that the order of nodes is induced by the order of the natural numbers, given by the relation $a < b$ for all $a, b \in V$. For a subset $A \in V$ we denote the node with the highest order by $\max(A)$. To avoid confusion, we point out the difference between the model-graph $G = (V, F, E)$, which describes the model structure and the search-graph $G_S = (V_S, E_S, w)$, which is used to reformulated the MAP-problem as as shortest path problem.

The formal definition of G_S is given in Definition 4.3 and illustrated in Figure 4.8. The node set V_S includes a distinct start-node s and distinct end-node t. To each node $u \in V_S$ we assign a level, denoted by $\langle u \rangle$. The level of u is equal to the number of edges of path from s to u. The level of s is zero and the level of t is $|V| + 1$. Each node on a path from s to t corresponds to a sub-configuration. With a slight abuse of notation, we will use the notation of a node $u \in V_S \setminus \{s, t\}$ also for the corresponding sub-configuration, i.e.

$$\mathcal{X}_{\{1,\dots,\langle u \rangle\}} \ni x_{\{1,\dots,\langle u \rangle\}} \stackrel{!}{=} u \in V_S \setminus \{s, t\}. \tag{4.59}$$

Consequently, a node $u \in V_S$ with the level n corresponds to a sub-configuration of the first n random variables, with respect to the order of V. Each path from s to t in G_S corresponds to a configuration $x \in \mathcal{X}$. The nodes along this path are the sub-configurations corresponding to x and the last node before t corresponds to the full configuration x.

The set of edges in the search-graph, denoted by E_S, includes directed edges from s to all nodes in level 1 and from all nodes in level $|V|$ to t. Furthermore, two nodes $u, v \in V_S \setminus \{s, t\}$ are connected by a directed edge (v, u) if v is a sub-configuration of u and $\langle v \rangle + 1 = \langle u \rangle$. The weight $w((v, u))$ which we assign to each edge (v, u) is equivalent to the energy caused by u minus the energy caused by v in the graphical model.

Definition 4.3. The search graph $G_S = (V_S, E_S, w)$ associated with the minimization problem $\min_{x \in \mathcal{X}} \sum_{f \in F} f(x_{\mathrm{ne}(f)})$ is defined by

$$V_S = \{s, t\} \cup \bigcup_{a \in V} \bigotimes_{b = V^a} \mathcal{X}_b \tag{4.60}$$

$$\langle u \rangle = \begin{cases} 0 & \text{if } u = s \\ |V| + 1 & \text{if } u = t \\ |u| & \text{else} \end{cases} \tag{4.61}$$

$$E_S = \left\{ (v, u) \in V_S \times V_S \;\middle|\; \begin{array}{l} v, u \in V_S \setminus \{s, t\},\ \langle v \rangle + 1 = \langle u \rangle,\ v = u_{V^{\langle v \rangle}} \\ \text{or } v = s,\ u \in \mathcal{X}_1 \\ \text{or } v \in \mathcal{X},\ u = t \end{array} \right\} \tag{4.62}$$

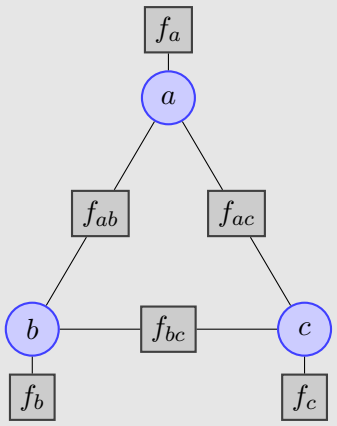
$$w((v, u)) = \sum_{f \in F^{\langle u \rangle} \setminus F^{\langle v \rangle}} f(u_{\mathrm{ne}(f)}) \tag{4.63}$$

with

$$V^n = \{a \in V | a \leq n\} \tag{4.64}$$

$$F^n = \{f \in F | \forall a \in \mathrm{ne}(f) : a \leq n\} \tag{4.65}$$

The set V^n is a subset of V and contains all nodes with an order lower than or equal to n. Accordingly, the set F^n contains all factors which depend only on variables in V^n. The number of nodes in the search-graph is $2 \cdot |\mathcal{X}| + 1$. Since we will never actually build the whole search-graph, this huge number does not form a problem – we use only the



$$\mathcal{X} = \{0,1\}^3$$

$$f_a(x_a) = \begin{cases} 5 & \text{if } x_a = 0 \\ 4 & \text{else} \end{cases}$$

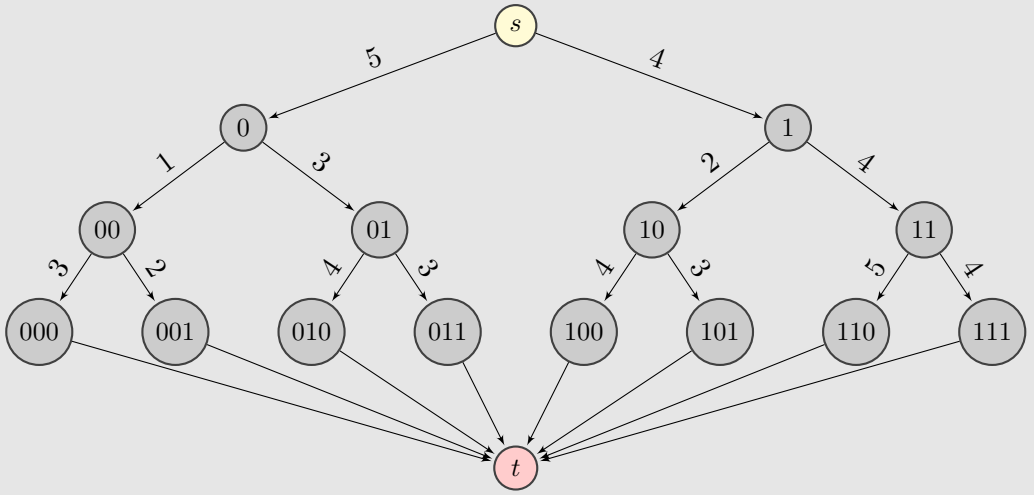
$$f_b(x_b) = \begin{cases} 1 & \text{if } x_b = 0 \\ 2 & \text{else} \end{cases}$$

$$f_c(x_c) = \begin{cases} 3 & \text{if } x_c = 0 \\ 0 & \text{else} \end{cases}$$

$$f_{nm}(x_n, x_m) = x_n + x_m, \qquad nm \in \{ab, bc, ac\}$$

$$J(x) = \textstyle\sum_{f \in F} f(x_{\text{ne}(f)})$$

(a)



(b)

Figure 4.8.: The problem of minimizing the energy function $J(x)$ defined by the factor graph model **(a)** can be transformed into a shortest path problem in the weighted search-graph G_S **(b)**. The weights in G_S are defined by the factor functions $f(\cdot)$ such that the shortest path in G_S from the node s to t corresponds to the configurations x minimizing $J(x)$. Using Definition 4.3 we obtain the weights
$w(s,0) = f_a(0) = 5$
$w(0,01) = f_{ab}(0,1) + f_b(1) = 3$
$w(01,010) = f_{ac}(0,0) + f_{bc}(1,0) + f_c(0) = 4$
and can calculate $J((0,1,0)) = w(s,0) + w(0,01) + w(01,010) = 12$.

search-graph to illustrated the underlying idea. Of course in the worst case we have to explore the whole search graph, but relevant real world problems are far away from this worst case scenario. The most important property of the search-graph is that the cost of each directed path from s to $v \in \mathcal{X}$, denoted by $d(s, v)$, is equivalent to the energy of the configuration v, and the cost of the shortest path from s to t is equal to the energy of the mode. For the proof see Theorem 4.3 and Theorem 4.4.

Theorem 4.3. *For all $v \in V_S \setminus \{s, t\}$, the cost of the path from s to v in G_S is equal to $\sum_{f \in F^{\langle v \rangle}} f(v_{ne(f)})$.*

Proof. For the unique path with nodes $q = (s, \ldots, v)$ in G_S, we obtain with (4.63)

$$d(s, v) = \sum_{n=1}^{\langle v \rangle} w(q_n, q_{n+1}) = \sum_{n=1}^{\langle v \rangle} \Big(\sum_{f \in F^n \setminus F^{n-1}} f(v_{\mathrm{ne}(f)}) \Big) = \sum_{f \in F^{\langle v \rangle}} f(v_{\mathrm{ne}(f)})$$

□

Theorem 4.4. *The energy of the mode $\min_{x \in \mathcal{X}} J(x)$ is equal to the weight of the shortest path from s to t.*

Proof. Since all edges containing t have the weight 0 and $\mathrm{ne}(t) = \mathcal{X}$,

$$\min\, d(s, t) := \min_{x \in \mathcal{X}} d(s, x) = \min_{x \in \mathcal{X}} J(x)$$

□

Efficient Search by Tree-Based Heuristics

As a consequence on Theorem 4.4 we can solve the MAP-problem by searching a shortest path between s and t in G_S. Due to the size of G_S it is not obvious, that this has any advantage. Indeed, we will not be able to calculate the mode efficiently in general. However, we will introduce a smart search strategy, which empirically performs well.

For the calculation of the shortest path between s and t in G_S we will use a best-first search algorithm, also known as A^* [69]. This algorithm starts in s and expands to all children of s. For each child v an approximation of the costs from s to t over v is calculated, which includes the correct costs of the unique path from s to v, denoted by $g(v)$, add an approximation $h(v)$ of the costs of the cheapest path from v to t. The minimal cost from v to t is denoted by $h^*(v)$. The expand step is repeated iteratively on the node with the lowest estimated costs. If $h(v)$ is a lower bound, i.e. $h(v) \leq h^*(v)$ for all $v \in V_S$, then $h(v)$ is called an admissible heuristic. For admissible heuristics the A^*-search is guaranteed to find the shortest path.

The connection to the more general description of branch and bound strategy is quite obvious. With each expansion step, we divide a set of possible configurations into several sets by fixing the next state of the configuration. Furthermore, we bound the minimal energy for each of these sets by underestimating the costs of all paths corresponding to each set.

However, it is still unclear, how this lower bound can be computed efficiently. Since the calculation of this lower bound will be executed very often, we need a fast method,

which produces preferably tight bounds. We use a tree-based admissible heuristic for this subroutine. The main idea is to transform the problem back to our factor graph model. All paths from s to t over a certain node v in G_S correspond to a subset of configurations in $\mathcal{X}$ defined by

$$\mathcal{X}(v) := \{x \in \mathcal{X} | \, x_{V^{\langle v \rangle}} = v\} \qquad\qquad u \in V_S \setminus \{t\}. \tag{4.66}$$

The weight of the shortest path from s to t over v can be calculated by

$$\min_{x \in \mathcal{X}(v)} J(x). \tag{4.67}$$

Alternatively, we can include the constraint $x_{V^{\langle v \rangle}} = v$ into the factor graph model by setting the variables $x_{V^{\langle v \rangle}}$ to v. We denote this conditioned energy function by

$$J(x|v) := \sum_{f \in F} f(x|v). \tag{4.68}$$

The notation $f(x|v)$ means, that we fix in the function $f(x)$ the arguments $x_{V^{\langle v \rangle}}$ to v, e.g. $f(x_2, x_3|(5, 9)) = f(9, x_3)$.

This procedure of fixing variables is illustrated in Figure 4.9. Nodes whose corresponding variables are fixed are drawn in black. Factors which contain this fixed variables can be simplified by squeezing connections of factor nodes to fixed variable nodes, without changing the objective of the model. This squeezed model represent the problem under the observation given by v.

However, the factor graph corresponding to $J(x|v)$ can still be cyclic. To overcome this problem, we replace factors, which cause cycles. Let us first consider this problem from an abstract point of view, before we introduce the algorithmic method. We can iteratively select a factor f which causes a cycle. We replace f by a factor f', which depends on only one variable and underestimates f for all $x_{\mathrm{ne}(f)} \in \mathcal{X}_{\mathrm{ne}(f)}$, i.e.

$$\begin{array}{rl} f(x_{\mathrm{ne}(f)}) & \geq \min_{x_{\mathrm{ne}(f)\setminus\{a\}}} f(x_{\mathrm{ne}(f)}) =: f'(x_a) \\ f(x_{\mathrm{ne}(f)}|v) & \geq \min_{x_{\mathrm{ne}(f)\setminus\{a\}}} f(x_{\mathrm{ne}(f)}|v) =: f'(x_a|v) \end{array} \qquad a \in \max(\mathrm{ne}(f)). \tag{4.69}$$

If we repeat this procedure until no cycle is left, we get an acyclic factor graph model, which defines for all configurations in $\mathcal{X}(v)$ an energy which is lower or equal the energy of the cyclic model – see Figure 4.9.

Instead of searching for cycles and replacing factors to fix these cycles, we will predefine an acyclic substructure $G_T = (V, F_T, E_T)$ with $F_T \subset F$. We replace all factors $f \in F \setminus F_T$ of order larger than two by f' as defined in (4.69). Note, that factors of the order zero or one can not cause cycles. If we replace all these factors in the factor graph, we obtain a lower bound $\bar{J}(x|v)$ on $J(x|v)$ defined on a acyclic factor graph model.

$$\begin{aligned} \bar{J}(x|v) := & \underbrace{\sum_{f \in F^{\langle v \rangle}} f(v_{\mathrm{ne}(f)}) + \sum_{f \in F^{\langle v \rangle + 1} \setminus F^{\langle v \rangle}} f((v, x_{\langle v \rangle + 1})_{\mathrm{ne}(f)})}_{g(x_{\langle v \rangle + 1})} \\ & + \sum_{f \in F_T \setminus F^{\langle v \rangle + 1}} f(x_{\mathrm{ne}(f)}|v) + \sum_{f \in F \setminus (F^{\langle v \rangle + 1} \cup F_T)} f'(x_{\max(\mathrm{ne}(f))}|v). \end{aligned} \tag{4.70}$$

To avoid an additional parameter, which defines the acyclic substructure, we will use $G_T = (V, F_T, E_T)$ with

$$F_T = \left\{ f \in F \;\middle|\; \begin{array}{l} \mathrm{ne}(f) < 2 \text{ or} \\ \mathrm{ne}(f) = 2,\ \max(V) \in \mathrm{ne}(f) \end{array} \right\}, \tag{4.71}$$

$$E_T = \left\{ (a, f) \in V \times F_T \middle| a \in \mathrm{ne}(f) \right\} \tag{4.72}$$

as long as nothing else is stated.

The motivation for this choice is, that these edges correspond to nodes with high order. For full connected graphs, this choice leads to less or equal many factors which have to be approximated then for any other choice of G_T. One can think about selecting the edges including the most important information as substructure, in order to get better heuristics. However, even more problem specific choices of G_T have experimentally not lead to better behavior of the A^* search.

The lower bound on the energy function can be used to calculate an admissible heuristic for the minimal cost for the paths from u to s. Let us denote the parent node of u in G_S by v. An important observation is, that the first two terms in $\bar{J}(x|v)$ are equal to the cost of the path from s to $u = x_{\langle v \rangle + 1}$. Furthermore, it follows directly from (4.69) and (4.70), that for $v \in \mathrm{pa}(u)$

$$J(x|u) \geq \bar{J}(x|u) \geq \min_{x_{\langle u \rangle} \in \mathcal{X}_{\langle u \rangle}} \bar{J}(x|v) \qquad \forall x \in \mathcal{X}. \tag{4.73}$$

Theorem 4.5. *For $v \in pa(u)$ the energy $\bar{J}(x|v)$ defines an admissible heuristic by*

$$h(u) = \min_{x \in \mathcal{X}(u)} \bar{J}(x|v) - g(u) \tag{4.74}$$

Proof.

$$\begin{aligned} g(u) + h^*(u) &= \min_{x \in \mathcal{X}(u)} J(x|u) \geq \min_{x \in \mathcal{X}(u)} \bar{J}(x|u) \geq \min_{x \in \mathcal{X}(u)} \bar{J}(x|v) \\ &= g(u) + \min_{x \in \mathcal{X}(u)} \bar{J}(x|v) - g(u) = g(u) + h(u). \end{aligned}$$

□

For all $u \in \mathrm{ch}(v)$ we can calculate a lower bound on the cost of the path from s to t over u by calculating the min marginal $\bar{J}^{\min}(x_{\langle u \rangle}|v)$ since

$$g(u) + h(u) = \min_{x \in \mathcal{X}(u)} \bar{J}(x|v) = \bar{J}^{\min}(x_{\langle u \rangle}|v). \tag{4.75}$$

Therefore, when we expand v, the lower bounds for the expanded nodes $u \in \mathrm{ch}(v)$ can be calculated by dynamic programming on the acyclic factor graph.

We sum up the method in Algorithm 4.7. The set B includes all pairs of nodes and their calculated lower bound, which wait to be expanded. We start with $B = \{(s, 0)\}$ and iteratively expand the pair with the lowest bound. For fast calculation of the pair with the lowest bound, we use a heap-data-structure [30] for B. The method popBest(B) returns the current best candidate used for expansion. The algorithm terminates if a node u of the level $|V|$ will be expanded, because this expansion cause the expansion of t in the next step since the costs from s to t over $u \in \mathcal{X}$ are equal to the cost from s to u.

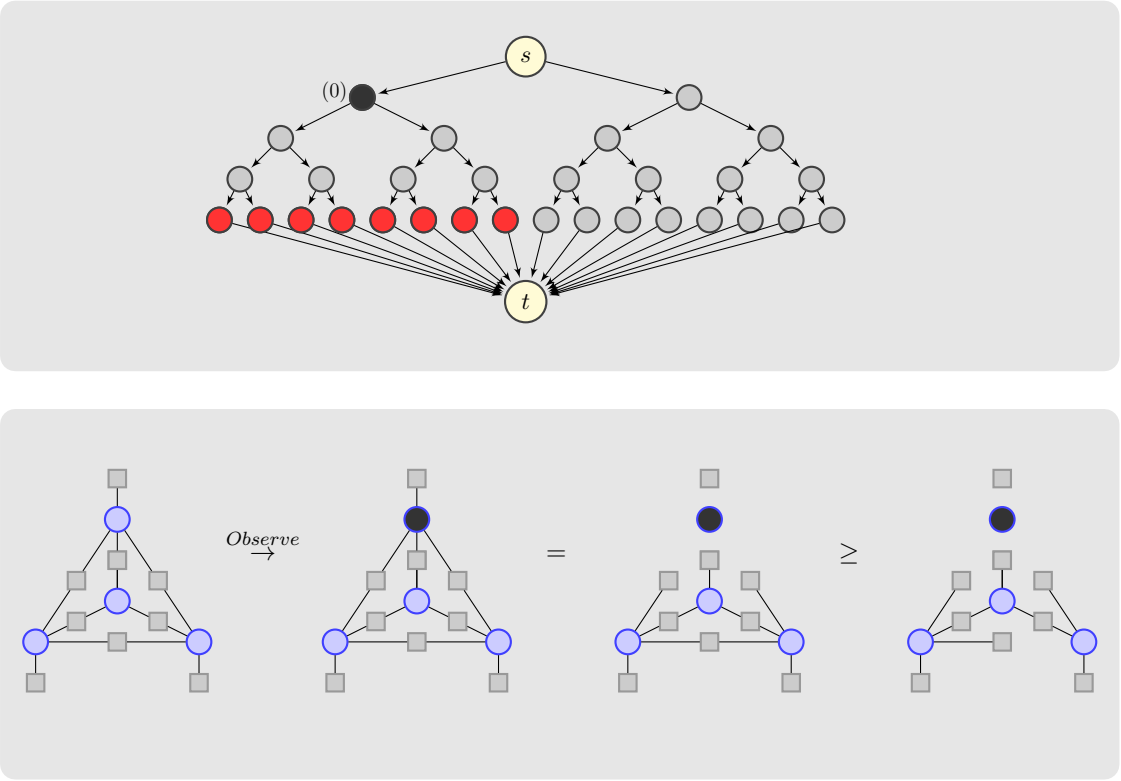


Figure 4.9.: Illustration of the idea of the tree-based lower bound. Assume that our model is given by a factor graph shown in the lower left. The corresponding search graph is illustrated at the top. Consider the node (0) in G_S, marked completely black, should be expanded. The minimal cost of the path from s through (0) to t is equivalent to the minimal energy of the configurations corresponding to the red node. To calculate a lower bound we return to the factor graph model. In a first step we fix the states for the known variables, denoted by darker nodes. We can replace factors which depend on this variables by simpler factors and get an equivalent but simpler model. This model still contains a cycle; we eliminate this cycle by replacing a factor of this cycle by a function that only depends on one variable, but is still a lower bound for the original factor for all settings. On this acyclic model dynamic programming is used to calculate lower bounds on the energy for the children of (0).

Algorithm 4.7 The A^*-algorithm for the min-sum-problem calculates the configuration with the minimal energy. During the search, the set B includes the boundary of the visited search graph nodes. The operations on B in the lines 6-13 can be efficiently implemented by a heap data-structure. G_T is an acyclic subgraph of G.

Require: $G = (V, F, E)$, $G_T = (V, F_T \subset F, E_T)$
Ensure: $x^* \in \arg\min_{x \in \mathcal{X}} \sum_{f \in F} f(x_{\mathrm{ne}(f)})$
1: $B = \{(s, 0)\}$
2: $(v, b) = \mathrm{popBest}(B)$
3: **while** $\langle v \rangle < |V|$ **do**
4: $n = \langle y \rangle + 1$
5: Calculate the min-marginals $\bar{J}^{\min}(x_n|v)$ according to G_T using dynamic programming
6: **for** $i \in \mathcal{X}_n$ **do**
7: $u = (v, i)$
8: $B = B \cup \{(u, \bar{J}^{\min}(x_n = i|v))\}$
9: **end for**
10: **if** B is too large **then**
11: Drop the pairs in B with the highest bound
12: **end if**
13: $(v, b) = \mathrm{popBest}(B)$
14: **end while**
15: $x^* = v$,

Implementation Details

To speed up the calculation of the bounds we pre-calculate the approximations for the non-tree factors. Furthermore, we implement a memory efficient algorithm for the calculation of the min-marginals by dynamic programming for second order models.

The proposed algorithm can be applied on max-product problems straight forward. We introduced the method in its min-sum-version, since this is more related to the usual application of A^*. A further notable feature of this approach is, that it easily can be upgraded to calculate the n-best configurations. Continuing the algorithm will return the next best configuration.

For larger problems the algorithm can get into trouble by memory limitations. Furthermore, a large heap increases the costs for all operations on the heap. Thus, it is useful to limit the size of the heap. It is possible to use a fixed upper bound on the energy or calculate upper bounds on the optimal value of the set online. Both can be used to drop sub-configurations with high lower bounds. Another option is to simply remove the sub-configurations with the highest bounds from the heap if the heap becomes too large. If we store the lowest bound for removed sub-configuration, it is possible to check after all if all removed sub-configuration are worse than $J(x^*)$ and consequently x^* is an optimal solution.

4.2.6. Mixed Integer Programming

The MAP-problem can also be formulated as a mixed integer problem (MIP) or more precisely as a mixed integer linear problem (MILP) defined by

$$\min_{\mu \in \mathbb{R}^{|\mathcal{I}(\mathcal{G})|}} \langle -\theta, \mu \rangle \tag{4.76}$$

$$s.t. \sum_{y_{\mathrm{ne}(f)} \in \mathcal{X}_{\mathrm{ne}(f)},\, y_a = x_a} \mu_{f;y_{\mathrm{ne}(f)}} = \mu_{a,x_a} \qquad \forall\, f \in F,\ a \in \mathrm{ne}(f) \tag{4.77}$$

$$\sum_{x_a \in \mathcal{X}_a} \mu_{a;x_a} = 1 \qquad \forall\, a \in V \tag{4.78}$$

$$\mu_{a,i} \in \{0,1\} \qquad \forall\, a \in V,\ i \in \mathcal{X}_a. \tag{4.79}$$

The integer constraints (4.79) guarantee that the solution is an integer solution. Furthermore, (4.78) ensures that exactly one entry of the sub-vector $\mu_{a;*}$ is one and the others are zero. Finally, (4.77) enforces the consistency of node and factor arguments.

We use CPLEX (version 12.1) to solve this MILP without further investigation in this topic. Since CPLEX is a commercial software, the algorithm used is not known in detail and we use it as a black box. The general method used by the CPLEX-solver for MILP is a branch and bound algorithm [35, 105]. The lower bounds are calculated by solving a relaxed version of the problem without integer constraints by the simplex method. In the branching step a fractal variable is selected and the problem is divided into two subproblems in which this fractal variable is forced to be 0 and 1, respectively. For the two subproblems then again a lower bound is calculated via a warm start with the simplex algorithm, and the procedure repeated iteratively until the best subproblem has an integer solution. Indeed this description of the CPLEX-solver is very sketchy and it includes also several more complex steps and heuristics for fast performance, which are not public.

The main differences to our A^*-based approach are, that the bound is calculated by linear programming and the branching is performed much more generally and can include a subset of states for one random variable in one branch of the search-tree. The main advantage of this further degree in branching is that branching can be adapted to the problem at runtime.

In the computer vision literature MIP is often doomed to be not applicable for computer vision problems. Even for solving the LP with relaxed integer constraints, specialized algorithms are used. While we first also consider that MIP as not applicable for our problems, we decide at the end to apply the CPLEX-solver, too. The results are surprisingly, as long the problems are small enough, many MAP-problems for computer vision application can be solved by MIP standard solvers. However, on larger problems, even solving the relaxed LP becomes intractable due to the high memory requirements of the problem.

For a detailed discussion of the performance of MILP-solvers on the MAP-problem compared to the other methods see Section 4.4.

4.3. Variational Inference, Relaxations and Convex Optimization

4.3.1. Motivation

Instead of solving the inference problems by combinatorial algorithms, we make use of the theory of exponential families, transform the problems, into convex optimization problems

and try to solve those.

The MAP-problem written as an integer program (IP) reads

$$\max_{x\in\mathcal{X}} \exp\left(\langle\theta,\phi(x)\rangle - A(\theta)\right), \tag{4.80}$$

and can be reformulated as linear program (LP) given by

$$\max_{\mu\in\mathcal{M}(G)} \langle\theta,\mu\rangle, \tag{4.81}$$

where, according to Section 2.5, the set of affine constraints is represented by the marginal polytope

$$\mathcal{M}(G) = \mathrm{conv}(\{\phi(x)|x\in\mathcal{X}\}). \tag{4.82}$$

The equivalence of (4.80) and (4.81) is based on the fact, that $\mathcal{M}(G)$ is the convex hull of the overcomplete representation of the integer solutions. Since $A(\theta)$ does not depend on x and μ, respectively, and for any θ at least one vertex of $\mathcal{M}(G)$ is an optimal solution of (4.81), the equivalence follows directly.

The problem of calculating the value of the log-partition function $A(\theta)$ for a given exponential parameter θ can be formulated in a combinatorial form

$$A(\theta) = \log \sum_{x\in\mathcal{X}} \exp\langle\theta,\phi(x)\rangle \tag{4.83}$$

or in its variational form

$$A(\theta) = \sup_{\mu\in\mathcal{M}(G)} \langle\theta,\mu\rangle - A^*(\mu). \tag{4.84}$$

The argument μ, that optimizes (4.81) and (4.84), is the mode and marginal, respectively. An interesting relation between (4.81) and (4.84), can be seen if we consider the zero-temperature limit of $A(\theta)$, known from statistic mechanics. If we rescale the canonic parameter θ by $\beta > 0$ this will put, in a relative sense, more mass into regions of the sample space $\mathcal{X}$ where $\langle\theta,\phi(x)\rangle$ is large. Ultimately, if $\beta\to+\infty$ probability mass only remain on configurations $x^*\in X^*$. As a particular consequence of the general variational principle, we have

$$\lim_{\beta\to\infty}\frac{A(\beta\theta)}{\beta} = \lim_{\beta\to\infty}\frac{1}{\beta}\sup_{\mu\in\mathcal{M}(G)}\langle\beta\theta,\mu\rangle - A^*(\mu) \tag{4.85}$$

$$= \lim_{\beta\to\infty}\sup_{\mu\in\mathcal{M}(G)}\langle\theta,\mu\rangle - \frac{1}{\beta}A^*(\mu) \tag{4.86}$$

The last step, which exchanges the limit over β and the supremum over μ requires $A^*(\mu)$ to be convex. Formally, we can obtain this result by concepts of convex analysis, c.f. Theorem 13.3 in [134].

Theorem 4.6. *For all $\theta\in\Omega$,* (4.81) *has the following alternative representation:*

$$\max_{x\in\mathcal{X}}\langle\theta,\phi(x)\rangle = \lim_{\beta\to\infty}\frac{A(\beta\theta)}{\beta} \tag{4.87}$$

Proof. See Theorem 8.1 in [173] which uses concepts introduced in [134]. □

Solving the integer program for the MAP-problem and the evaluation of the log partition function are in general NP-hard. The equivalence of the combinatorial and convex formulations underline the inherent complexity of the marginal polytopes. Note, that this transformation from an IP into a LP over the convex hull, is also used in the area of combinatorial optimization and polyhedral combinatorics, e.g. [14, 63, 144]. However, solving the convex programs with standard solvers is in general computationally infeasible, due to the complexity of $\mathcal{M}(G)$. Nevertheless these problems can be a starting point for relaxations and approximations. Contrary to approximations, relaxations guarantee upper or lower bounds on the objective which is maximized or minimized. The formal definition of a relaxation of a minimization problem is given in Definition A.9.

A common relaxation of the marginal polytope is to replaced it by a set of local marginalization constraints, which define the local polytope $\mathcal{L}(G)$. For the undirected graphical model $G = (V, E)$, the local polytope is defined by

$$\mathcal{L}(G) = \left\{ \mu \;\middle|\; \begin{array}{ll} \mu \in [0,1]^{|\mathcal{I}(G)|}, & \\ \forall a \in V: & \sum_{x_a \in \mathcal{X}_a} \mu_{a;x_a} = 1, \\ \forall ab \in E,\, x_a \in \mathcal{X}_a: & \sum_{x_b \in \mathcal{X}_b} \mu_{ab;x_a x_b} = \mu_{a;x_a} \end{array} \right\} \tag{4.88}$$

$$= [0,1]^{|\mathcal{I}(G)|} \cap \operatorname{aff}(\{\phi(x) | x \in \mathcal{X}\}) \tag{4.89}$$

The affine hull $\operatorname{aff}(S)$ is the smallest affine set containing S. Its formal definition is given in Definition A.4. For a discussion of the relation between the local polytope and the affine hull, we refer to [181]. $\mathcal{M}(G)$ is a subset of $\mathcal{L}(G)$. If G is acyclic, equivalence holds. However, for cyclic models, $\mathcal{L}(G)$ is a strict superset of $\mathcal{M}(G)$ [173]. Furthermore, all vertices of $\mathcal{M}(G)$ are vertices of $\mathcal{L}(G)$, but $\mathcal{L}(G)$ has some further vertices which correspond to fractal solutions. Contrary to $\mathcal{M}(G)$ for which the number of affine inequalities grows rapidly with the graph size, the local polytope can be represented by $|V| + |V| \cdot L + |E| \cdot L$ constraints. The relation between the local and marginal polytope is illustrated highly idealized in Figure 4.10. While this picture might suggest, that $\mathcal{L}(G)$ has more facets then $\mathcal{M}(G)$, $\mathcal{L}(G)$ has less but this is difficult to convey in a 2D-representation. Furthermore, it is shown that $\mathcal{L}(G)$ includes additional to the integer vertices of $\mathcal{M}(G)$ some further vertices, which are fractal and not in $\mathcal{M}(G)$.

The set of affine constraints of the marginal polytope includes constraints which ensure non negativity

$$\mu_{a;x_a} \geq 0 \qquad \forall a \in V, x_a \in \mathcal{X}_a, \tag{4.90}$$

the node wise marginalization property

$$\sum_{x_a \in \mathcal{X}_a} \mu_{a;x_a} = 1 \qquad \forall a \in V, \tag{4.91}$$

and the local consistency marginals

$$\sum_{x_b \in \mathcal{X}_b} \mu_{ab;x_a x_b} = \mu_{a;x_a} \qquad \forall ab \in E,\; x_a \in \mathcal{X}_a. \tag{4.92}$$

Solving the MAP-problem by a LP-relaxation over the local polytope using a general linear programming algorithm, such as simplex or interior point method, is possible for problems

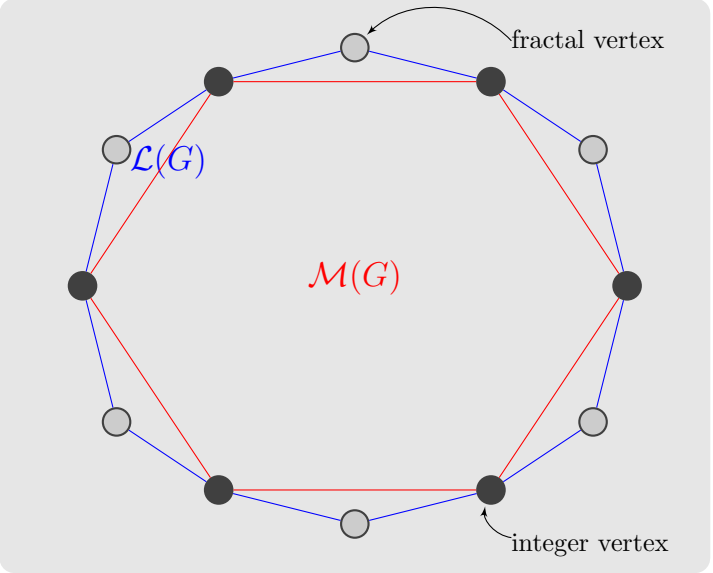


Figure 4.10.: Highly idealized illustration of the local polytope $\mathcal{L}(G)$ as outer relaxation of the marginal polytope $\mathcal{M}(G)$. $\mathcal{L}(G)$ is always an outer bound on $\mathcal{M}(G)$ and a strict superset if G is cyclic. Both polytopes can be represented by their vertices or the set of facets. Fractal vertices are ploted in gray and integer vertices in black.

of moderate size, but inefficient and virtually impossible for large instances which occur in many computer vision applications. Solving the LPs with standard solvers fails in this setting, because they are implemented for general purpose and require a large amount of memory.

To do better, problem specific solvers are used, which utilizing the structure of the graphical models. For an overview of work of Schlesinger and colleagues, see the review of Werner [179] and for an overview including the work of the last decade, we refer to the book of Wainwright [173]. Although, the works of Schlesinger and Wainwright are very similar, they had been developed independently. We will discuss the work of Wainwright and colleagues in the following. They minimize the dual problem of (4.81) by a message passing algorithms, which we will sketch in Section 4.3.3. In this context we will also discuss how the entropy term $A^*(\mu)$ can be bounded for a relaxation of (4.84).

A current line of research in computer vision focus on relaxations of $\mathcal{M}(G)$ which are tighter than $\mathcal{L}(G)$. The motivation for this tighter relaxations is, that the standard LP relaxation are often not tight enough in many real-world problems and integer optimality can often not guaranteed. Consequently, often LP-solvers finds only fractal solutions. To get an integer solutions these fractal solutions are rounded without any guarantees of optimality if the objective values of the fractal and integer solution differ.

To overcome this problem, cutting plane approaches have been used. The idea is to add iteratively additional constraints to the constraint set, see [154, 180] for examples. These additional constraints leads to tighter relaxations, which may have integer solutions.

Another class of algorithms make use of Lagrangian decomposition also known as dual decomposition in computer vision. The primary motivation for the decomposition of graphical models is twofold. Firstly, an approximation of the intractable inference problem can be formulated in terms of a two-level optimization procedure, where at the lower level infer-

ence on tractable substructures is carried out, while the master program at the upper level combines these partial solutions via dual variables. Secondly, the resulting objective value at the upper level yields a bound to the original (intractable) objective function, whose optimization through dual variables possibly meets the value of some locally computed optimum, thus providing a certificate that this optimum is indeed a global one.

Algorithms which use this dual decomposition technique, differ in i) the chosen decomposition of the problem, ii) the methods which are used to solve this subproblems, and iii) the optimization scheme which is used to optimize the dual problem. TRW-algorithms can also be seen as dual decomposition into tree-structured subproblems. Optimization in done by a block coordinate descent like method, which can get stuck in local fixed points. Another class of algorithms perform a projective subgradient descent method on the dual problem. We will investigate in the later one, which converge to the optima of the relaxation under some technical conditions. Our contributions in this context are besides of an alternative view on this framework by exponential families, a new class of subproblems which are easy to solve, while leading to significant tighter relaxations for highly connected models than standard approaches.

4.3.2. LBP Revisited

We have already introduced the loopy belief propagation algorithm (LBP) and showed its convergence properties and correctness for acyclic graphs. While there is no barrier to apply it on cyclic graphs, since all updates have a local form, this was done by many researchers. However, the underlying objective is not clear at first sight. In general, LBP has no convergence guarantees on cyclic graphs. Yedidia et al. [186, 187] show that fix points of the sum-prod version of the LBP correspond for second order models to stationary points of the Bethe free energy.

Let us consider (4.84) and relax $\mathcal{M}(G)$ by $\mathcal{L}(G)$. For general graphs $A^*(\mu)$ typically lacks a closed form. However, for acyclic models we can rewrite the negative entropy $A^*(\mu)$ as a sum of local entropy terms associated with edges and nodes.

$$
\begin{aligned}
-A^*(\mu) &= H(\mu) && (4.93)\\
&= \mathbb{E}_{p_\mu}(-\log p_\mu(X)) && (4.94)\\
&= \sum_{a\in V} \underbrace{-\sum_{a_a\in\mathcal{X}_a} \mu_{a,x_a}\log\mu_a}_{:=H_a(\mu_a)} - \sum_{ab\in E} \underbrace{\sum_{x_{ab}\in\mathcal{X}_{ab}} \mu_{ab;x_{ab}} \log\frac{\mu_{ab;x_{ab}}}{\mu_{a;x_a}\mu_{b;x_b}}}_{:=I_{ab}(\mu_{ab})} && (4.95)
\end{aligned}
$$

The terms $H_a(\mu_a)$ are the singleton entropy for each random variable X_a with $a \in V$ and for each edge $ab \in E$ the term $I_{ab}(\mu_{ab})$ is the mutual information of the random variable pair X_a and X_b.

Consequently, for acyclic models the dual function $A^*(\mu)$ can be expressed in an explicit and easily computable function in terms of μ. The assumption, that the decomposition of the entropy for tree structured models in (4.95) is approximately valid for cyclic graphs, yields to the Bethe entropy approximation.

$$
-A^*(\tau) \approx H_{\text{Bethe}}(\tau) := \sum_{a\in V} H_a(\tau_a) + \sum_{ab\in E} I_{ab}(\tau_{ab}) \tag{4.96}
$$

The change of notation form μ to τ is deliberate and highlight an important fact. The function (4.96) can be evaluated for all $\tau \in \mathcal{L}(G)$, this fact is central in the derivation of the sum-prod LBP.

Note, that Yedidia et al. use an alternative representation of the Bethe entropy approximation, which can be obtained by the relation $I_{ab}(\tau_{ab}) = H_a(\tau_a) + H_b(\tau_b) - H_{ab}(\tau_{ab})$, where $H_{ab}(\tau_{ab})$ is the joint entropy of X_{ab}. Performing some algebraic manipulation, yields to the alternative form

$$H_{\text{Bethe}}(\tau) := \sum_{a \in V} (1 - |\text{ne}(a)|) H_a(\tau_a) + \sum_{ab \in E} H_{ab}(\tau_{ab}) \tag{4.97}$$

used by Yedidia et al. .

We now have the two components, which are required to construct a Bethe approximation to (4.84). Firstly, the set of local consistency mean parameters $\tau \in \mathcal{L}(G)$, which is an outer bound of the marginal polytope $\mathcal{M}(G)$, and secondly, an approximation of $A^*(\tau)$ by the Bethe entropy approximation $H_{\text{Bethe}}(\tau)$. Combining both, we obtain the Bethe variational problem

$$\max_{\tau \in \mathcal{L}(G)} \langle \theta, \tau \rangle + H_{\text{Bethe}}(\tau). \tag{4.98}$$

This problem is much easier than the problem in (4.84) since the cost function is given in a closed form and is differentiable, and the number of affine constraints on τ is polynomial in $|V| + |E|$ and much smaller then in (4.84).

Sum-Product Loopy Belief Propagation As we will see in the following, the LBP algorithms turns out to be a simple scheme to solve (4.98).

In order to develop this relation, let λ_a be the Lagrangian multiplier associated with the normalizing constraint $C_a(\tau) = 0$, where

$$C_a(\tau) := 1 - \sum_{x_a \in \mathcal{X}_a} \tau_{a;x_a}. \tag{4.99}$$

Moreover, for each direction $a \to b$ of each edge and each $x_b \in \mathcal{X}_b$, we associate the Lagrangian multiplier $\lambda_{ab;x_b}$ with the constraint $C_{ab;x_b}(\tau) = 0$, where

$$C_{ab;x_b}(\tau) := \tau_{b,x_b} - \sum_{x_a \in \mathcal{X}_a} \tau_{ab;x_a,x_b}. \tag{4.100}$$

These Lagrangian multipliers turn out to be related to the messages of the sum-prod LBP by

$$M_{ab}(x_b) \propto \exp(\lambda_{ab;x_b}). \tag{4.101}$$

Furthermore, the Lagrangian multiplies λ_a correspond to the normalization constants in the LBP algorithm, see [173, 187] for proofs and details.

It should be noted, that the connection between the Bethe problem and the sum-product-LBP does not provide any guarantee on the convergence of the sum-product updates of LBP. In fact, the convergence properties of the algorithm depends on the topology of the graph and the potential functions. Several authors [72, 78, 118, 135] have investigated the convergence of LBP and introduce condition on the objective, which guarantee convergence

and optimality for LBP. In a parallel line of research [74, 178, 188] convergent variants of LBP have been explored, albeit with the price of increasing computational costs. However, with the exception of some special cases [73, 115, 188] including tree structured models, the Bethe variational problem is not convex. Consequently, the optimization problem (4.98) contains several local optima, such that even a convergent version of LBP can converge to non-global optima. To reduce oscillating messages, we use a damping method suggested amongst others in [119, 128, 169].

A multitude of variants of LBP was investigated in the last decades. Some authors [114, 40, 157, 159] suggest different message schedules in contrast to parallel updates. In principle, the accuracy of the Bethe variational principle can be improved by using tighter outer bounds on the marginal polytope and better approximations of the entropy. Yedidia et al. [186, 187] proposed a higher order relaxation based on Kikuchi approximations and hypertree-based methods.

Max-Product Loopy Belief Propagation In analogy to the relation between the sum-product LBP and the Bethe variational problem (4.98), one might expect that fix points of the max-product LBP correspond to fixed points of $\max_{\tau \in \mathcal{L}(G)} \langle \theta, \tau \rangle$ by applying the idea of Theorem 4.6. While this is true for acyclic models, in general it is not the case. An essential condition for the proof of Theorem 4.6 is the convexity of the log partition function. In the Bethe approximation the dual of the log partition function is approximated by the negative Bethe entropy approximation. Since $-H_{\text{Bethe}}(\mu)$ is in general not convex, we cannot apply methods from convex analysis to build any general connection. However, for special classes of problems, the negative Bethe entropy approximation is convex, e.g. for acyclic models. This discloses an alternative way to prove the correctness of the max-prod LBP for tree structured models, see [173]. Furthermore, it can be shown by counterexamples, that fix points of the max-product messages of LBP do not correspond to optimal solutions of the LP over the local polytope, see [98, 169].

Weiss and Freeman [176] have shown, that for a fix point M^+ of the max-product message passing, we can calculated x^+, such that changing the settings of x^+_A will not lead to a better probability, as long as the subset A of nodes in G consists of disconnected combinations of trees and single loops. This neighborhood is called Single Loops and Trees (SLT) neighborhood.

4.3.3. Tree Reweighted Message Passing

From the variational point of view, the main drawback of LBP or the Bethe approximated problem is the lack of convexity of the approximation of $A^*(\mu)$. Since the exact variational principle is convex, it is natural to consider variational approximations which are convex, too. We will discuss a convexified Bethe variational problem introduced by Wainwright [173] next, which ends up in a message passing algorithm similar to LBP. The idea behind these convexified algorithms is to use approximations based on convex decompositions of tractable distributions. We will use tree-structured distributions, for which the Bethe approximation is exact, to decompose the objective. We will again restrict ourselves to models of second order, given by an undirected model $(X, G = (V, E))$. In the end we will present an alternative approach, also introduced by Wainwright [169], which shows the relation to the dual decomposition method we will discuss in Section 4.3.4.

For a given graph $G = (V, E)$ we consider an exponential family with the sufficient statistic $\phi(x) = (\phi_\alpha(x))_{\alpha \in \mathcal{I}(G)}$ and the exponential parameter vector $\theta = (\theta_\alpha))_{\alpha \in \mathcal{I}(G)}$. Calculating the log partition function $A(\theta)$ or the mean parameters μ for an arbitrary $\theta \in \Omega$ is intractable, in general. However, for certain choices of exponential parameters such computations are tractable. If we can find an acyclic graph $T = (V, E')$ with $E' \subset E$, such that $\theta_{ab,x_{ab}} = 0$ for all $ab \in E \setminus E'$ and $x_{ab} \in \mathcal{X}_{ab}$, this member of the exponential family is quasi acyclic. Quasi acyclic means that if we remove all edges which are non-active, i.e. they have no influence on the objective, we get an acyclic graph. Moreover, we can use this observation to obtain a convex lower bound on the dual function $A^*(\mu)$.

Let us now consider the details to define an lower bound on $A^*(\mu)$. To this end, we define a coordinate projective mapping $[\cdot]_{\mathcal{I}(T)}$ from the full space $\mathcal{I}(G)$ to the subspace $\mathcal{I}(T)$ of indexes associated with T. We denote with $[\mu]_{\mathcal{I}(T)}$ the mapping of a mean parameter $\mu \in \mathcal{M}(G)$ to $\mathcal{M}(T)$ and with $[\theta]_{\mathcal{I}(T)}$ the mapping from the parameter space corresponding to $\mathcal{I}(G)$ into the space which corresponds to $\mathcal{I}(T)$. This projection guarantees, that for all $\alpha \in \mathcal{I}(T)$, we have the equivalences $\mu_\alpha = ([\mu]_{\mathcal{I}(T)})_\alpha$ and $\theta_\alpha = ([\theta]_{\mathcal{I}(T)})_\alpha$.

This technical definitions are used to obtain a lower bound on the dual function $A^*(\mu)$

$$\begin{aligned} A^*(\mu) &= \sup_{\theta \in \mathbb{R}^{|\mathcal{I}(G)|}} \langle \theta, \mu \rangle - A(\theta) && (4.102) \\ &\geq \sup_{\theta \in \mathbb{R}^{|\mathcal{I}(G)|},\, \theta_\alpha = 0 \text{ if } \alpha \notin \mathcal{I}(T)} \langle \theta, \mu \rangle - A(\theta) && (4.103) \\ &= \sup_{[\theta]_{\mathcal{I}(T)} \in \mathbb{R}^{|\mathcal{I}(T)|}} \langle [\theta]_{\mathcal{I}(T)}, [\mu]_{\mathcal{I}(T)} \rangle - A([\theta]_{\mathcal{I}(T)}) && (4.104) \\ &= A^*([\mu]_{\mathcal{I}(T)}). && (4.105) \end{aligned}$$

One drawback of the lower bound in (4.105) is, that it is not strictly convex on $\mu \in \mathcal{M}(G)$. A change in the mean parameters $(\mu_\alpha)_{\alpha \in \mathcal{I}(G) \setminus \mathcal{I}(T)}$ has no influence on $A^*([\mu]_{\mathcal{I}(T)})$. To obtain a strict convex lower bound, Wainwright [173] suggests to use a convex combination of spanning trees, such that all edges of G are contained at least in one tree $T \in \mathcal{T}$.

For a given probability distribution over the trees in $\mathcal{T}$, denoted by $\rho \in [0, 1]^{|\mathcal{T}|}$ such that $\sum_{T \in \mathcal{T}} \rho_T = 1$, we obtain the lower bound

$$B^*(\mu; \mathcal{T}, \rho) := \sum_{T \in \mathcal{T}} \rho_T A^*([\mu]_{\mathcal{I}(T)}) \leq A^*(\mu). \tag{4.106}$$

We can simplify the left hand side. For each single tree T we know that

$$A^*([\mu]_{\mathcal{I}(T)}) = -\left(\sum_{a \in V} H_a(\mu_a) + \sum_{ab \in E_T} I_{ab}(\mu_{ab}) \right). \tag{4.107}$$

If we insert (4.107) in (4.106) and make use of the fact that all trees are spanning trees, i.e. each node is contained in each tree, we can rewrite the left hand side of (4.106) by

$$B^*(\mu; \mathcal{T}, \rho) = -\left(\sum_{a \in V} H_a(\mu_a) + \rho_{st} \sum_{ab \in E_T} I_{ab}(\mu_{ab}) \right), \tag{4.108}$$

where

$$\rho_{ab} = \sum_{T \in \mathcal{T},\, ab \in E_T} \rho_T. \tag{4.109}$$

It is not obvious that this lower bound is strictly convex. Although the formal proof is given in Theorem 7.2 in [173], we will sketch the idea. Without loss of generality, let us assume that the exponential family is minimal – if the problem is formulated in a overcomplete representation it can be transformed into a minimal form, see [173]. As shown in Theorem 2.8, the log partition function is strictly convex for minimal representations. Since $A(\theta)$ and $A^*(\mu)$ are lower semi-continues, the dual $A^*(\mu)$ is strictly convex if and only if $A(\theta)$ is strictly convex, see Theorem 26.3 in [134]. Furthermore, for each $T \in \mathcal{T}$ the dual $A^*([\mu]_{\mathcal{I}(T)})$ is strictly convex for $(\mu_\alpha)_{\alpha \in \mathcal{I}(T)}$. By definition of $\mathcal{T}$, for each $\alpha \in \mathcal{I}(G)$ there exists at least one tree $T \in \mathcal{T}$ such that $\alpha \in \mathcal{I}(T)$. Thus, the convex combination $\sum_{T \in \mathcal{T}} \rho_T A^*([\mu]_{\mathcal{I}(T)})$ is strictly convex.

If we now choose an outer bound relaxation on the marginal polytope $\mathcal{M}(G)$, it has to be ensured that $B^*(\mu; \mathcal{T}, \rho)$ is well defined on this set. The largest set of mean parameters, which fulfills this property is

$$\mathcal{M}(\mathcal{T}, \rho) = \left\{ \tau \in \mathbb{R}^{|\mathcal{I}(G)|} \middle| \, \forall T \in \mathcal{T} : [\tau]_{\mathcal{I}(T)} \in \mathcal{M}(T) \right\}. \tag{4.110}$$

For each tree $T \in \mathcal{T}$, the marginal polytope $\mathcal{M}(T)$ is equivalent to the local polytope $\mathcal{L}(T)$. $\mathcal{L}(T)$ enforces that τ is non-negative, normalized in each vertex $a \in V$ and marginalized across each edge in the tree. Enforcing these constraints for all trees $T \in \mathcal{T}$ is equivalent to enforce non-negativity, normalization in each vertex $a \in V$, and marginalization across each edge for the original graph G. Finally, we concluded that

$$\mathcal{M}(\mathcal{T}, \rho) = \mathcal{L}(G), \tag{4.111}$$

if

$$\forall T \in \mathcal{T} : \ \rho_T > 0 \qquad \text{and} \qquad \bigcup_{T \in \mathcal{T}} E_T = E. \tag{4.112}$$

Note that, (4.111) holds for any set of trees for which (4.112) is satisfied. We will use a fix set of spanning trees and consequently a fixed vector ρ. However, it is possible to optimize over the parameter ρ in order to obtain the best bound on $A^*(\mu)$, as considered in [170].

So far, we constructed the tree reweighted Bethe variational problem

$$B(\theta; \mathcal{T}, \rho) = \max_{\tau \in \mathcal{L}(G)} \langle \theta, \tau \rangle + \sum_{a \in V} H_a(\tau_a) - \rho_{st} \sum_{ab \in E_T} I_{ab}(\tau_{ab}), \tag{4.113}$$

which yields for a given parameter θ an upper bound on $A(\theta)$.

Sum-Product Tree Reweighted Belief Propagation Similar to the Bethe variational problem, we can obtain message updates for tree reweighted version by

$$M_{ab}(x_b) = \kappa \sum_{x'_a \in \mathcal{X}_a} \exp(\theta_{a,x'_a} + \frac{1}{\rho_{ab}} \theta_{ab;x'_a x_b}) \frac{\prod_{c \in \mathrm{ne}(a) \setminus \{b\}} [M_{ca}(x'_a)]^{\rho_{ca}}}{[M_{ba}(x'_a)]^{1-\rho_{ba}}}. \tag{4.114}$$

Again, we can build a connection between fix points of (4.114) and the problem (4.113). The idea is similar to the proof of the Bethe approximation and shows, that fixed points of the message update satisfy the conditions of being a stationary point of the Lagrangian associated with the constrained optimization problem (4.113). Since the relaxation on the

Algorithm 4.8 Tree Reweighted Belief Propagation

Require: Undirected Graphical Model $(X, G = (V, E))$ with $p(x|\theta) = \exp(\langle\theta, \phi(x)\rangle - A(\theta))$ and a damping parameter $\alpha \in [0, 1]$.

$\forall ab \in E: \quad M^0_{b\leftarrow a}(x_b) = 1$
$t = 0$
repeat
  $t = t + 1$
  for all $ab \in E$ **do**
    $M^t_{ab}(x_b) = \kappa_{ab} \sum_{x_a \in \mathcal{X}_a} \exp(\frac{1}{\rho_{ab}}\theta_{ab;x_a,x_b} + \theta_{a,x_a}) \frac{\prod_{c\in \mathrm{ne}(a)\setminus\{b\}} \left[M^{(t-1)}_{ca}(x_a)\right]^{\rho_{ca}}}{\left[M^{(t-1)}_{ba}(x_a)\right]^{1-\rho_{ba}}}$
    $M^t_{ab}(x_b) = M^t_{ab}(x_b)^{1-\alpha} \cdot M^{(t-1)}_{ab}(x_a)^{\alpha}$
  end for
until $t > t_{max}$ or $\|M^{(t-1)} - M^t\|_\infty \leq \epsilon$
for all $a \in V$ **do**
  $b_a(x_a)$ by (4.115)
end for
for all $ab \in E$ **do**
  $b_{ab}(x_a x_b)$ by (4.116)
end for

dual function is strictly convex, this update equation has a unique fix point. The proof is given in Appendix C in [170].

Furthermore, equations for the calculation of singleton and pairwise pseudo-marginals from the fix point messages M^* [173] are given by

$$\tau^*_{a,x_a} = \kappa_a \exp(\theta_{a,x_a}) \prod_{b\in \mathrm{ne}(a)} [M^*_{ba}(x_a)]^{\rho_{ba}} \tag{4.115}$$

$$\begin{aligned}\tau^*_{ab,x_a x_b} = \kappa_{ab} \exp(\theta_{ab,x_a x_b} + \theta_{a,x_a} + \theta_{b,x_b}) \cdot \\ \frac{\prod_{c\in \mathrm{ne}(a)\setminus\{b\}} [M^*_{ca}(x_a)]^{\rho_{ca}}}{\left[M^*_{ba}(x_a)\right]^{1-\rho_{ba}}} \frac{\prod_{c\in \mathrm{ne}(b)\setminus\{a\}} [M^*_{cb}(x_b)]^{\rho_{cb}}}{\left[M^*_{ab}(x_b)\right]^{1-\rho_{ab}}}.\end{aligned} \tag{4.116}$$

The mean parameter τ^* is called a pseudo-marginal because is lies in the local polytope but not mandatory in the marginal polytope. If $\tau^* \in \mathcal{M}(G)$ than the pseudo-marginals represent the exact marginal distribution.

Roosta et al. [135] provide a sufficient condition for convergence. In practical terms, the updates (4.114) are done in parallel and convergence is not guaranteed. However, it can be empirically shown, that if the messages as sufficiently damped, the message update converge in most cases. This kind of message updating is known as TRBP, see Algorithm 4.8.

Max-Product Tree Reweighted Belief Propagation The convexity of the bound on the dual function has another important effect. We can apply the zero-temperature limit in the same way as we did for the original problem. Since the approximation $B(\theta; \mathcal{T}, \rho)$ of the log partition function $A(\theta)$ is convex we can apply again exchange the limit over β with the supremum. Consequently, the zero-temperature limit of $B(\theta; \mathcal{T}, \rho)$ is equivalent

to

$$\max_{\tau \in \mathcal{L}(G)} \langle \theta, \tau \rangle. \tag{4.117}$$

In analogy to the sum-prod-TRBP updates, Wainwright [169] defines max-prod-TRBP updates, which have the form

$$M_{ab}(x_b) = \kappa \max_{x'_a \in \mathcal{X}_a} \exp(\theta_{a,x'_a} + \frac{1}{\rho_{ab}} \theta_{ab;x'_a x_b}) \frac{\prod_{c \in \mathrm{ne}(a) \setminus \{b\}} [M_{ca}(x'_a)]^{\rho_{ca}}}{[M_{ba}(x'_a)]^{1-\rho_{ba}}}. \tag{4.118}$$

Corresponding to the reweighted sum-product messages, the max-product messages define a selection of pseudo-max-marginals of the form

$$\nu^*_{a,x_a} = \kappa_a \exp(\theta_{a,x_a}) \prod_{b \in \mathrm{ne}(a)} [M^*_{ba}(x_a)]^{\rho_{ba}} \tag{4.119}$$

$$\begin{aligned} \nu^*_{ab,x_a x_b} = \kappa_{ab} \exp(\theta_{ab,x_a x_b} + \theta_{a,x_a} + \theta_{b,x_b}) \frac{\prod_{c \in \mathrm{ne}(a) \setminus \{b\}} [M^*_{ca}(x_a)]^{\rho_{ca}}}{\left[M^*_{ba}(x_a)\right]^{1-\rho_{ba}}} \\ \frac{\prod_{c \in \mathrm{ne}(b) \setminus \{a\}} [M^*_{cb}(x_b)]^{\rho_{cb}}}{\left[M^*_{ab}(x_b)\right]^{1-\rho_{ab}}} \end{aligned} \tag{4.120}$$

This leads to max-product TRBP algorithm, see Algorithm 4.9. Wainwrigth et al. [169] define the strong tree agreement (STA) condition, and show that fix points (M^*, ν^*) which satisfy the STA condition define a dual solution of the LP relaxation (4.117).

Definition 4.4. If there is a $\hat{x} \in \mathcal{X}$, such that $\hat{x}$ is optimal for all pseudo-max-marginals ν, i.e.

$$\hat{x}_a \in \arg\max_{x_a \in \mathcal{X}_a} \nu_a(x_a) \qquad \forall a \in V, \text{ and} \tag{4.121}$$

$$\hat{x}_{ab} \in \arg\max_{x_{ab} \in \mathcal{X}_{ab}} \nu_{ab}(x_{ab}) \qquad \forall ab \in E, \tag{4.122}$$

ν satisfies the strong tree agreement (STA) condition.

In interesting question is, if any fix-point (M^*, ν^*) is an optimal solution of the dual LP relaxation (4.117). While this question was left open in [169], Kolmogorov [87] sequentially provides a counterexample. For this counterexample with non-binary variables, Kolmogorov shows that a fix point of the max-product-TRBP algorithm exists, which does not correspond to a dual optimal solution. In a follow up work, Wainwright and Kolmogorov [90] show that for second order models with binary random variables, fix-points of max-product-TRBP correspond to a dual optimal solution.

Kolmogorov introduces in [87] a sequential schedule of TRBP message updates, for which convergence guarantees can be given. Furthermore, he shows empirically, that this sequential version, called TRW-S, converges faster than the parallel standard updates. However, a technical condition for TRW-S is, that an order on the node set V exists, such that each path between two nodes in the used spanning tree is monotonic. While this condition can be fulfilled for grid graph models, which are commonly used in computer vision, such an order does not exist for arbitrary graph and spanning-tree-decomposition. Therefore, we will use the parallel update scheme, and try to obtain convergence by sufficient damping.

Algorithm 4.9 Max-Product Tree Reweighted Belief Propagation

Require: Undirected Graphical Model $(X, G = (V, E))$ with $p(x|\theta) = \exp(\langle\theta, \phi(x)\rangle - A(\theta))$ and a damping parameter $\alpha \in [0, 1]$.

$\forall ab \in E: \quad M^0_{b \leftarrow a}(x_b) = 1$
$t = 0$
repeat
  $t = t + 1$
  for all $ab \in E$ **do**
    $M^t_{ab}(x_b) = \kappa_{ab} \max_{x_a \in \mathcal{X}_a} \exp(\frac{1}{\rho_{ab}} \theta_{ab;x_a,x_b} + \theta_{a,x_a}) \frac{\prod_{c \in \mathrm{ne}(a) \setminus \{b\}} \left[M^{(t-1)}_{ca}(x_a)\right]^{\rho_{ca}}}{\left[M^{(t-1)}_{ba}(x_a)\right]^{1-\rho_{ba}}}$
    $M^t_{ab}(x_b) = M^t_{ab}(x_b)^{1-\alpha} \cdot M^{(t-1)}_{ab}(x_a)^{\alpha}$
  end for
until $t > t_{max}$ or $\|M^{(t-1)} - M^t\|_\infty \leq \epsilon$
for all $a \in V$ **do**
  $b_a(x_a)$ by (4.119)
end for
for all $ab \in E$ **do**
  $b_{ab}(x_a x_b)$ by (4.120)
end for

An alternative view of this framework is used in [169, 87] by Wainwright and Kolmogorov. They show, that the dual problem of the relaxed LP can be interpreted as a minimizing over a set of valid reparameterizations.

To illustrate this, let us again decompose the graph $G = (V, E)$ into a set of trees $\mathcal{T}$, such that $\bigcup_{T \in \mathcal{T}} E_T = E$ and $V_T = V$. For a given tree probability ρ with $\rho_T > 0$ and $\sum_{T \in \mathcal{T}} \rho_T = 1$, we define a decomposition of θ into $\sum_{T \in \mathcal{T}} \rho_T \cdot \theta(T)$ such that $\theta_\alpha = 0$ if $\alpha \in \mathcal{I}(G) \setminus \mathcal{I}(T)$. Jensen's inequality yields the upper bound

$$\max_{x \in \mathcal{X}} \langle\theta, \phi(x)\rangle = \max_{x \in \mathcal{X}} \sum_{T \in \mathcal{T}} \rho_T \langle\theta(T), \phi(x)\rangle \leq \sum_{T \in \mathcal{T}} \rho_T \max_{x \in \mathcal{X}} \langle\theta(T), \phi(x)\rangle. \tag{4.123}$$

Minimizing this upper bound over the set of possible parameters $\{\theta(T) | T \in \mathcal{T}\}$ leads to the convex optimization problem

$$\min_{\{\theta(T)\}_{T \in \mathcal{T}}} \sum_{T \in \mathcal{T}} \rho_T \max_{x \in \mathcal{X}} \langle\theta(T), \phi(x)\rangle \tag{4.124}$$

$$s.t. \sum_{T \in \mathcal{T}} \rho_T \theta(T) \equiv \theta \tag{4.125}$$

$$\theta^T_\alpha = 0, \quad \text{if } \alpha \in \mathcal{I}(G) \setminus \mathcal{I}(T). \tag{4.126}$$

Wainwright [169] shows that the dual of this problem is the LP-relaxation over the local polytope. Thus, messages or Lagrangian duals can be seen as a reweighting of the sub-problems. Another important observation is, that the objective in (4.124) is convex but not necessarily smooth. Consequently, coordinate descent algorithms might get stuck in non-optimal points, as sketched in Figure 4.11.

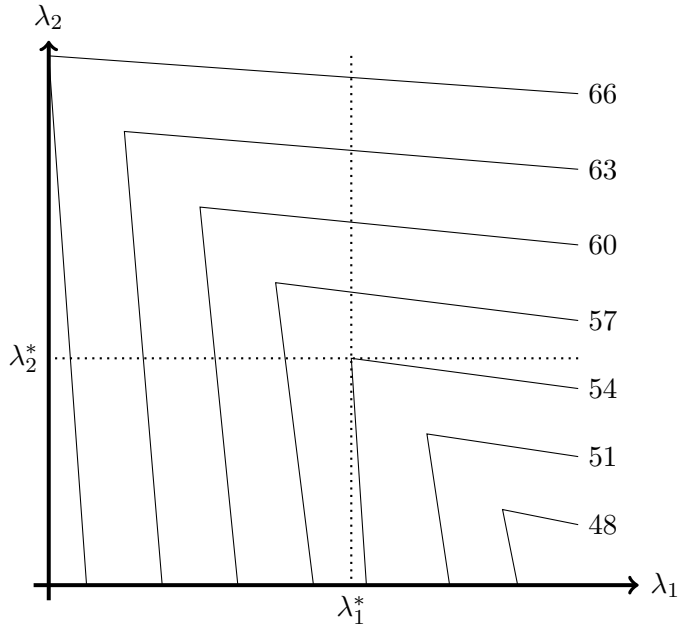


Figure 4.11.: The figure shows the contours of a non-smooth convex function, $f(\lambda_1, \lambda_2)$. Coordinate descent did not find the global minimum, since point $(\lambda_1^*, \lambda_2^*)$ is globally minimal for each coordinate separately but not for both simultaneously.

4.3.4. Lagrangian Decomposition

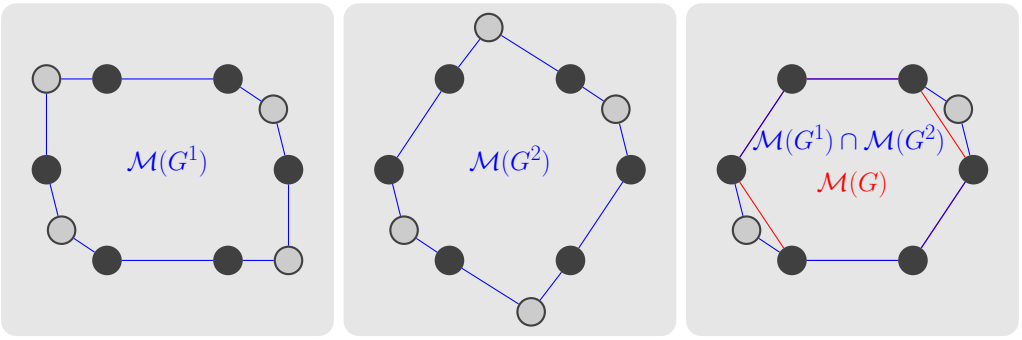
Let us finally consider a dual decomposition approach for solving the MAP-problem. Starting with the exact problem formulation

$$\max_{\mu \in \mathcal{M}(G)} \langle \theta, \mu \rangle, \tag{4.127}$$

we will introduce a decomposition into several subproblems similar to (4.124) and minimize an upper bound on (4.127). Contrary to TRBP-approaches, we will use subgradient decent methods for minimizing the dual problem. These methods can deal with the non-smoothness of the problem and guarantee to converge to an optima of the relaxed problem. Furthermore, we show that this framework enables to enforce relaxations that are tighter than the local polytope relaxation. With this tighter relaxation, integer optimality can be obtained and verified more often.

Using the technique of Lagrangian decomposition [65, 64, 14, 13], also known as dual decomposition in the field of computer vision, is not new and was used by several authors [95, 93, 181]. Similar to our work, Werner [181] motivates his work on the basis of exponential families. From the viewpoint of optimization, the most related works are [95, 93] which also apply sub-gradient methods.

It will be convenient to distinguish between original parameter vectors θ, θ^i and parameter vectors $\hat{\theta}^i$ defined by the problem decomposition – cf. (4.129) below. Starting with the convex optimization problem (4.127), we decompose it as follows. Given a set of graphs



(a) (b) (c)

Figure 4.12.: Highly idealized illustration of the relaxation of the marginal polytope by a decomposition into sub-polytopes. The two polytopes of the subproblems define outer bounds on the marginal polytope. The intersection of both defines a quite tight relaxation of the marginal polytope. Note that $\mathcal{M}(G)$ and $\mathcal{M}(G^i)$ are not defined in a equal space. **(a)** and **(b)** show all $\mu \in \mathbb{R}^{|\mathcal{I}(G)|}$ with $[\mu]_{\mathcal{I}(G^i)} \in \mathcal{M}(G^i)$.

$\{G^1, \dots, G^n\}$, with $G^i = (V, E^i)$ such that $E^i \subset E$ and $\bigcup_{i=1}^n E^i = E$, we define $\theta^i \in \mathbb{R}^{\mathcal{I}(G)}$:

$$\theta^i_{a;j} := \begin{cases} 0 & \text{if } a \notin V \cup E^i, \\ \theta_{a;j}/n & \text{if } a \in V, \\ \theta_{a;j}/\#a & \text{if } a \in E^i. \end{cases} \tag{4.128}$$

Here, $\#a$ denotes the number of edge-sets containing a. Note that the decomposition ensures $\theta = \sum_i \theta^i$. For each subproblem, we define another smaller exponential parameter vector

$$\hat{\theta}^i := [\theta^i]_{\mathcal{I}(G^i)} \tag{4.129}$$

called the *projection* of θ^i with respect to $\mathcal{I}(G^i)$ and reformulate problem (4.127):

$$\max_{\mu \in \mathcal{M}(G)} \langle \theta, \mu \rangle = \max_{\mu \in \mathcal{M}(G)} \sum_i \langle \theta^i, \mu \rangle \tag{4.130a}$$

$$= \max_{\substack{\mu \in \mathcal{M}(G) \\ \forall i: \mu^i \in \mathcal{M}(G) \\ \forall i: \mu^i = \mu}} \sum_i \langle \theta^i, \mu^i \rangle \overset{\text{eqn. (4.128)}}{=} \max_{\substack{\mu \in \mathcal{M}(G) \\ \forall i: \mu^i \in \mathcal{M}(G^i) \\ \forall i: \mu^i = [\mu]_{\mathcal{I}(G^i)}}} \sum_i \langle \hat{\theta}^i, \mu^i \rangle \tag{4.130b}$$

$$\leq \max_{\substack{\mu \in \mathbb{R}^{\mathcal{I}(G)} \\ \forall i: \mu^i \in \mathcal{M}(G^i) \\ \forall i: \mu^i = [\mu]_{\mathcal{I}(G^i)}}} \sum_i \langle \hat{\theta}^i, \mu^i \rangle \overset{\text{eqn. (4.128)}}{=} \max_{\substack{\mu \in \mathbb{R}^{\mathcal{I}(G)} \\ \forall i: [\mu]_{\mathcal{I}(G^i)} \in \mathcal{M}(G^i)}} \langle \theta, \mu \rangle \tag{4.130c}$$

In the relaxation (4.130c) of the original problem, we replace the marginal polytope by a intersection of simpler polytopes. Since for all $\mu \in \mathcal{M}(G)$ the projection to $\mathcal{I}(G^i)$ lies in the marginal polytope corresponding to the subgraph $[\mu]_{\mathcal{I}(G^i)} \in \mathcal{M}(G^i)$, we obtain

$$\mathcal{M}(G) \subset \left\{ \mu \in \mathbb{R} \middle| \, \forall G^i : \, [\mu]_{\mathcal{I}(G^i)} \in \mathcal{M}(G^i) \right\}. \tag{4.131}$$

Each of these marginal polytopes correspond to a set of affine constraints $A^i\mu \leq b^i$. If all affine constraints of the marginal polytope $\mathcal{M}(G)$ are implied by these affine constraints,

$$A\mu \leq b \Leftrightarrow \begin{pmatrix} A^1 \\ \vdots \\ A^n \end{pmatrix} \mu \leq \begin{pmatrix} b^1 \\ \vdots \\ b^n \end{pmatrix}, \tag{4.132}$$

the relaxation is tight. A highly idealized illustration is shown in Figure 4.12.

The decomposition (4.130) has the properties, that i) if all subgraphs are trees, the relaxation is equivalent to the standard relaxation over the local polytope [173] and ii) if the subproblems include cycles, we get tighter relaxations than the local polytope relaxation which also take into account higher-order constraints.

Because problem (4.130c) is still difficult to solve, we focus on its dual by adding Lagrangian multipliers for the equality constraints, yielding the dual function

$$g(\lambda^1, \ldots, \lambda^n) := \max_{\substack{\mu \in \mathbb{R}^{\mathcal{I}(G)} \\ \forall i: \mu^i \in \mathcal{M}(G^i)}} \sum_i \langle \hat{\theta}^i, \mu^i \rangle + \sum_i \sum_{\alpha \in \mathcal{I}(G^i)} \lambda^i_\alpha (\mu^i_\alpha - \mu_\alpha). \tag{4.133}$$

Since μ is unconstrained, this vector is determined by the corresponding partial derivatives of the right-hand side of (4.133). This yields the condition

$$(\lambda^1, \ldots, \lambda^n) \in \Lambda := \left\{ (\lambda^1, \ldots, \lambda^n) \;\middle|\; \forall \alpha \in \mathcal{I}(G) : \sum_{i \in \{j | \alpha \in \mathcal{I}(G^j)\}} \lambda^i_\alpha = 0 \right\}, \tag{4.134}$$

and by insertion into (4.133) the dual problem of the relaxed LP (4.130c)

$$\inf_{(\lambda^1, \ldots, \lambda^n) \in \Lambda} \sum_i \underbrace{\max_{\mu^i \in \mathcal{M}(G^i)} \langle (\hat{\theta}^i + \lambda^i), \mu^i \rangle}_{g^i(\lambda^i)}. \tag{4.135}$$

Since the feasible set of the primal problem (4.130c) includes a strict feasible point, Slater's condition [18] holds and guarantees that the duality gap between (4.130c) and its dual problem (4.135) is zero, i.e.

$$U^* := \inf_{(\lambda^1, \ldots, \lambda^n) \in \Lambda} g(\lambda^i, \ldots, \lambda^n) \tag{4.136}$$

$$\| \quad$$

$$L^* := \max_{\substack{\mu \in \mathbb{R}^{\mathcal{I}(G)} \\ \forall i: [\mu]_{\mathcal{I}(G^i)} \in \mathcal{M}(G^i)}} \langle \theta, \mu \rangle. \tag{4.137}$$

Instead of solving the relaxed primal problem (4.130c), which is still fairly complex, we can now solve the dual problem (4.135) by projected sub-gradient descent [14, 139], taking advantage of the problem decomposition into tractable subproblems. To this end, we have to optimize each subproblem

$$\max_{\mu^i \in \mathcal{M}(G^i)} \left\langle (\hat{\theta}^i + \lambda^i), \mu^i \right\rangle \tag{4.138}$$

for a given λ^i. Rather than solving the LP in (4.138) directly, we solve the corresponding integer programming problem instead. This pose no loss of generality because vertices

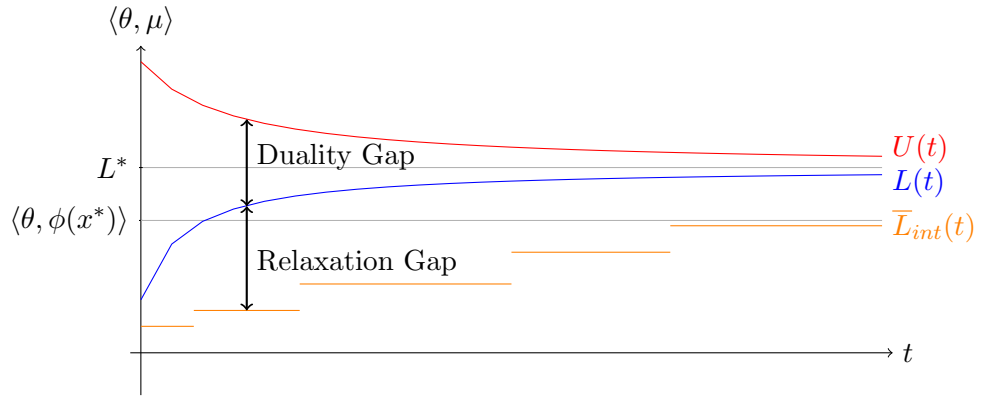


Figure 4.13.: This figure displays an idealized progressions of the bounds. Gray lines mark optimal values for the original primal and the relaxed primal/dual problem. Note that $L^* = U^*$ (zero duality gap for the relaxed problems). The red line shows the current upper bound $U(t)$ of the dual relaxed problem. The blue line marks the current lower bound $L(t)$ of the primal relaxed problem. Since we optimize only the dual problem we do not know the values of $L(t)$. Instead we calculate a lower bound $\overline{L}_{int}$ (marked in orange) of the original primal problem. The observed gap $\overline{L}_{int}(t) - U(t)$ includes the current duality gap as well as the current relaxation gap. We can not infer how they split up the total gap. However, we know that the duality gap will become zero. As a consequence, the final gap is only due to the relaxation.

of the polytopes $\mathcal{M}(G^i)$ correspond to integer configurations. Accordingly, if the decomposition has been chosen properly, these integer problems can be solved efficiently. As a by-product, we obtain a lower bound $L_{int}(t)$ of the original objective function (4.127), where the subscript *int* denotes that $L_{int}(t)$ is a lower bound on the integer problem. We denote the best lower integer bound obtained so far by $\overline{L}_{int}(t)$:

$$L_{int}(t) := \max_{i=1,\dots,n} \langle \theta, [(\mu^i)^{(t)}]_{\mathcal{I}(G)} \rangle \tag{4.139}$$

$$\overline{L}_{int}(t) := \max_{t'=1,\dots,t} L_{int}(t') \tag{4.140}$$

The main trick in (4.139) is, that firstly, $(\mu^i)^{(t)}$ is integer and secondly, both graphs (G and G^i) have the same node set V. Consequently, the projection $[\cdot]_{\mathcal{I}(G)}$ is well defined. Without the technical constraint that each subproblem contains the full vertex set, a good lower bound cannot be obtained so simple.

Furthermore, we can also calculate an upper bound $U(t)$ at iteration t. Again, we denote the best upper bound obtained so far by $\overline{U}(t)$, formally defined by

$$U(t) := \sum_{i=1,\dots,n} \langle \hat{\theta}^i + (\lambda^i)^{(t)}, (\mu^i)^{(t)} \rangle \tag{4.141}$$

$$\overline{U}(t) := \min_{t'=1,\dots,t} U(t') \tag{4.142}$$

The bounds crucially depend on the problem decomposition and thus reflect the quality of the relaxation. Figure 4.13 further explains and illustrates the relation between the different bounds and optima.

Algorithm 4.10 Projected Sub-gradient Method

Require: Graphical model $(X, G = (V, E))$ and a set of graphs $\{G^i = (V, E^i)\}$ with $\bigcup_i E^i = E$. Furthermore, a step size sequence defined by $\overline{\tau} \in \mathbb{R}^+$ and $\alpha \in \mathbb{R}^+$

Ensure: Calculate upper and lower bound on (4.127)

1: $t = 0$
2: $\lambda^{(0)} = 0 \in \Lambda$
3: **repeat**
4: $\quad \tau^{(t)} = \overline{\tau} \frac{1}{1+\alpha t}$
5: $\quad s \in \partial g((\lambda)^{(t)})$
6: $\quad \left(x^1, \ldots, x^n\right) \leftarrow s$
7: $\quad L_{int}(t) \ = \max_{i=1,\ldots,n} \langle \theta, \phi(x^i) \rangle$
8: $\quad U(t) \quad = \sum_{i=1,\ldots,n} \langle \hat{\theta}^i + (\lambda^i)^{(t)}, [\phi(x^i)]_{\mathcal{I}(G^i)} \rangle$
9: $\quad (\lambda)^{(t+1)} = \left[(\lambda)^{(t)} - \tau^{(t)} \cdot s\right]_\Lambda$
10: $\quad t = t + 1$
11: **until** $\|\overline{U}(t-1) - \overline{L}_{int}(t-1)\| \leq 10^{-6}$ or $t > t_{\max}$

Solving the Dual Problem The dual problem (4.135) is a non-smooth, convex minimization problem with linear constraints. The main difference between most inference algorithms based on dual decomposition [80, 93, 173, 181], besides the decomposition itself, concerns the choice and the computation of updates of λ in each step. A standard solver for such problems is the Projected Sub-Gradient Method (PSGM) [14, 127, 147] that requires to compute a subgradient of g at λ. The set of all subgradients at λ is called the subdifferential at λ and is denoted by $\partial g(\lambda)$. We perform inference with respect to all subproblems and select a subgradient from the set

$$\partial g^i(\lambda^i) = \partial \left(\max_{\mu^i \in \mathcal{M}(G^i)} \langle \hat{\theta}^i + \lambda^i, \mu^i \rangle \right) \tag{4.143}$$

$$= \left\{ \nabla \langle \hat{\theta}^i + \lambda^i, \mu^* \rangle \,\middle|\, \mu^* \in \operatorname*{arg\,max}_{\mu^i \in \mathcal{M}(G^i)} \langle \hat{\theta}^i + \lambda^i, \mu^i \rangle \right\} \tag{4.144}$$

$$= \left\{ \mu^* \,\middle|\, \mu^* \in \operatorname*{arg\,max}_{\mu^i \in \mathcal{M}(G^i)} \langle \hat{\theta}^i + \lambda^i, \mu^i \rangle \right\}. \tag{4.145}$$

The subgradient method iteratively updates λ by

$$\lambda^{(t+1)} = \lambda^{(t)} - \tau^{(t)} \cdot s, \tag{4.146}$$

where s is a subgradient of $g(\lambda^{(t)})$ and $\tau^{(t)} > 0$ the step size. Since we have some additional constraints on λ, we perform an additional projection and obtain

$$\lambda^{(t+1)} = \left[\lambda^{(t)} - \tau^{(t)} \cdot s\right]_\Lambda. \tag{4.147}$$

If the step size $\tau^{(t)}$ is chosen such that the diminishing step size rules

$$\lim_{t \to \infty} \tau^{(t)} = 0, \qquad \sum_{t=1}^{\infty} \tau^{(t)} = \infty \tag{4.148}$$

hold, the projective subgradient descent converges to the optimum [139]. Due to the non-smoothness of the problem, there is no guarantee that the objective decrease monotonic

Algorithm 4.11 Smoothed Projected Sub-gradient Method

Require: Graphical model $(X, G = (V, E))$ and a set of graphs $\{G^i = (V, E^i)\}$ with $\bigcup_i E^i = E$. Furthermore, a step size sequence defined by $\overline{\tau} \in (0, 1]$ and $\alpha \in \mathbb{R}^+$ and a smoothing parameter $\overline{\rho} \in (0, 1]$.

Ensure: Calculate upper and lower bound on (4.127)

$t = 0$
$\lambda^{(0)} = 0 \in \Lambda$
repeat
  $s \in \partial g((\lambda)^{(t)})$
  $\tau^{(t)} = \overline{\tau}\frac{1}{1+\alpha t}$
  $\rho^{(t)} = \beta\tau^{(t)}$
  if $t == 0$ **then**
    $\zeta^{(t)} = s$
  else
    $\zeta^{(t)} = \zeta^{(t-1)} + \rho^{(t)}(s - \zeta^{(t-1)})$
  end if
  $L_{int}(t) = \max_{i=1,\ldots,n}\langle\theta, \phi(x^i)\rangle$
  $U(t) = \sum_{i=1,\ldots,n}\langle\hat{\theta}^i + (\lambda^i)^{(t)}, [\phi(x^i)]_{\mathcal{I}(G^i)}\rangle$
  $(\lambda)^{(t+1)} = \left[(\lambda)^{(t)} - \tau^{(t)} \cdot \zeta^{(t)}\right]_\Lambda$
  $t = t + 1$
until $\|\overline{U}(t-1) - \overline{L}_{int}(t-1)\| \leq 10^{-6}$ or $t > t_{\max}$

with $\lambda^{(t)}$. A sufficient choice for a diminishing step size sequence is

$$\tau^{(t)} = \overline{\tau}\frac{1}{1+\alpha t} \tag{4.149}$$

which we will use. It includes 2 degrees of freedom. The parameter $\overline{\tau}$ defines the initial step size and α adjusts the speed of decreasing the step size. We iteratively repeat this update until a maximal number of iterations is obtained or the gap becomes less than 10^{-6}, see Algorithm 4.10.

The problem of Algorithm 4.10 is to choose of the step size parameters properly. To large step sizes lead to oscillation, where to small step sizes will cause slow converge and may yield oscillation. To cope with oscillations a modified version which smooths the subgradients over several iterations can be used. This modified version, also known as *heavy ball method* [24, 153, 161], does not step into the direction of the last subgradient, but rather into the direction of a convex combination of the subgradients observed so far. For $\rho^{(t)} = 1$, we obtain the standard PSGM, and for the constant sequence $\rho^{(t)} \in (0, 1)$ a smoothed version, with corresponding update equations

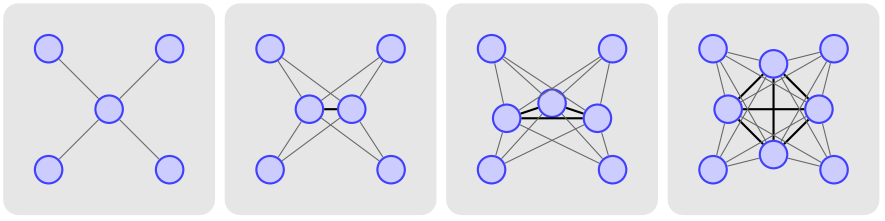
$$\lambda^{(t+1)} = [\lambda^{(t)} - \tau^{(t)} z^{(t)}]_\Lambda \tag{4.150}$$

$$z^{(t+1)} = z^{(t)} + \rho^{(t)}(s^{(t+1)} - z^{(t)}). \tag{4.151}$$

Converges to an optimum is guarantee if

$$\lim_{t\to\infty} \tau^{(t)} = 0, \qquad \sum_{t=0}^{\infty} \tau^{(t)} = \infty, \qquad \lim_{t\to\infty} \frac{\tau^{(t)}}{\rho^{(t)}} = 0. \tag{4.152}$$

For $\tau^{(t)} = \overline{\tau}\frac{1}{1+\alpha\cdot t}$ conditions (4.152) is satisfied for any constant sequence $\rho^{(t)}$.



(a) $G^{1,5}_{\text{fan}}$ (b) $G^{2,6}_{\text{fan}}$ (c) $G^{3,7}_{\text{fan}}$ (d) $G^{4,8}_{\text{fan}}$

Figure 4.14.: Examples for fan graphs. Inner nodes are connected to each other. Outer nodes are connected to all inner nodes, but not among each other.

The speed of convergence highly depends on the choice of the sequence $\tau^{(t)}$, which is determined offline by grid-search on the parameter space for a particular problem class and the corresponding graphical model. However, a good choice of $\rho^{(t)}$ also depends on the current value of $\tau^{(t)}$. Ruszczynski [139] suggests a damping sequence $\rho^{(t)} = \beta\tau^{(t)}$ and shows that for this sequence Algorithm 4.11 converges if

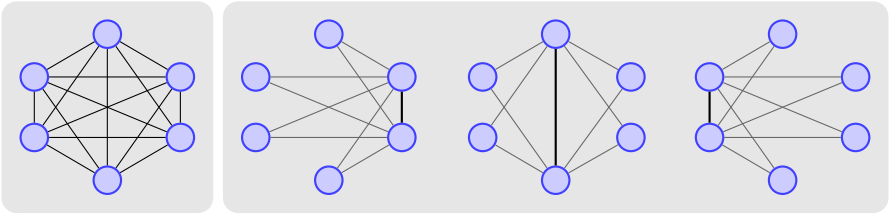
$$\beta, \tau^{(t)} \in (0, 1], \qquad \lim_{t\to\infty} \tau^{(t)} = 0, \qquad \sum_{t=0}^{\infty} \tau^{(t)} = \infty, \qquad \sum_{t=0}^{\infty} (\tau^{(t)})^2 < \infty. \tag{4.153}$$

Our choice $\tau^{(t)} = \overline{\tau}\frac{1}{1+\alpha\cdot t}$ fulfills this condition for any $\overline{\tau} \in (0, 1]$. The pseudo code to this smoothed method is given in Algorithm 4.11. However, even if this method leads to less oscillations and faster convergence, it introduces another parameter which has to be estimated. Convergence behavior is very sensitive to changes in this parameters. Choosing the right parameters often leads to significant faster convergence compared to a conservative parameter choice. This fact is very unsatisfying and has to be investigated in further work. In the remainder of this work, we empirically adjust the parameters by hand.

Decomposition by k-Fan Substructures In the last years, several different subproblem-structures have been suggested. The most naive one decomposes the cyclic model into a set of trees. In connection with the projected subgradient method, this was suggested by Komodakis et al. [94]. In the sequential work [93], they expand this approach to simple cyclic subproblems, for which the tree width of the triangulated graph is two and therefore inference is still feasible. Recentlyly, Betra et al. [7] used outer-planar graphs as subproblems for decomposition. Strandmark and Kahl [155] use this decomposition technique to decompose a huge problem, which can be theoretical solved by graph cuts, into several smaller problems. These smaller problems are solved in parallel on a GPU.

We will concern another class of decompositions, with the goal to obtain tighter relaxations. Therefore, we decompose our graph $G = (V, E)$ into a set of so-called k-Fans. As illustrated in Figure 4.14, the defining property of k-fans is that a acyclic graph is obtained by replacing all inner nodes with a single node and merging the resulting multiple edges. We will use the shorthand $G^{k,n}_{fan}$ for a fan graph with n nodes and k inner nodes. If n is given by the context, we call $G^{k,n}_{fan}$ k-fan.

Since the tree width of a k-fan is bounded by k, we could use the junction tree algorithm to perform inference. However, the asymptotic complexity for the junction tree algorithm on a $G^{k,n}_{\text{fan}}$-structured problem is $O((n-k)\cdot L^{(k+1)})$. For a faster optimization of the



(a) G (b) Decomposition in three 2-fans

Figure 4.15.: A decomposition of **(a)** a full connected graph with 6 nodes in **(b)** three 2-fan structured graphs.

subproblems, we will use the A^* method, presented in Section 4.2.5, which has the same asymptotic worst case runtime complexity for fan graphs, but performs faster on average. To guarantee the same worst case complexity, we have to select the order of the nodes in the A^*-algorithm such that the inner nodes of the k-fan are expanded first. As a consequence, the heuristic used in the A^*-search is tight for all levels that are greater or equal k. We gain performance since, contrary to the junction tree algorithm where all inner nodes are merged to one super node, the A^*-search avoids searching over the complete domain of inner variables by an admissible heuristic. Furthermore, it is much easier to implement, since we do not have to deal with super nodes and index projections. For the synthetic data in Section 4.4.4, the use of A^* decreases the dominant term of the average complexity from L^{k+1} to approximately $L^{0.5\cdot(k+1)}$, which is significant in practice.

When decomposing a graph $G = (V, E)$ into a set of k-fans, we divide $V = \{1, \dots, N\}$ in $\lceil N/k \rceil$ subsets $S_i = \{1+(b-1 \pmod k) | b \in [(i-1)*k+1, i*k] \cup \mathbb{N}\}$, where $S_1 = \{1, \dots, k\}$, $S_2 = \{k+1, \dots, 2k\}$ and so forth. If k divides N, the last subset is $S_{N/k} = \{N-k+1, \dots, N\}$. Otherwise we add nodes such that $|S_{N/k}| = k$. An example for a decomposition of a fully connected graph with 6 nodes into three 2-fans is shown in Figure 4.15. For numerical evaluations and discussion of the experiments see Section 4.4.

4.4. Empirical Comparison of MAP-Inference Algorithms

To compare the proposed inference algorithms, we conclude this chapter with an empirical evaluation of the presented inference algorithms for the MAP-problem. More precisely, we will consider the problem of minimizing an energy function over a discrete set of configurations

$$\min_{x \in \mathcal{X}} J(x). \tag{4.154}$$

Note, that this problem corresponds to the MAP problem given by

$$\min_{x \in \mathcal{X}} J(x) = -\max_{x \in \mathcal{X}} \langle \theta, \phi(x) \rangle \propto -\max_{x \in \mathcal{X}} p(x|y). \tag{4.155}$$

The integer solution x, produced by the inference algorithms defines, an upper bound on the minimal energy, i.e.

$$J(x) \geq J(x^*) \qquad \forall x \in \mathcal{X},\ x^* \in \mathcal{X}^*. \tag{4.156}$$

Additionally, keeping in mind that since some of the algorithms calculate a lower bound on the optimal energy, we can provide guarantees about the precession of the solution for

these algorithms. If this lower bound is equal to the energy of the integer solution, global optimality of the integer solution is guaranteed. For numerical reasons, we will assume that the integer solution is optimal if the gap is less than 10^{-6}. For LBP we obtain no lower bound. For TRBP theoretical a bound can be computed, if the number of spanning trees is large this becomes non-trivial. Thus, we will not consider the evaluation of a lower bound for TRBP.

We will compare the following algorithms:

- Damped Loopy Belief Propagation **(LBP)**
- Damped Tree Reweighted Belief Propagation **(TRBP)**
- A^*-search with a 1-fan for the heuristic **(A^*)**
- Mixed Integer Program solved by CPLEX **(MIP)**
- Linear Program with the local polytope relaxation solved by CPLEX with node based rounding **(LP)**
- Dual Decomposition with a set of $\lceil |V|/k \rceil$ k-fans solved by projected subgradient methods **(DD-k-fan)**

Graph Cut methods can not be applied to our models, due to the general form of our energy terms, which is neither submodular nor defined by a semimetric. Integer solutions for LBP and TRBP are obtained by an algorithm similar to Algorithm 4.2. We start with an arbitrary node and select the state with the lowest min-marginal at this node. Then, we proceed iteratively with all min-marginals, defined over higher order factors, for which at least one variable is fixed already. We fix the states of unknown variables neighbored to the factor, by setting the state to the optimal state of the conditioned min-marginal. While for acyclic graphs this method guarantees that we do not mix up two modes, for cyclic graph it is only a heuristic without any guarantee for better rounding. However, this method tracks a mode at least locally. For the LP solution τ, we apply simple node based rounding

$$x_a^* = \arg\max_{x_a \in \mathcal{X}} \tau_{a,x_a}. \tag{4.157}$$

All of these algorithms are implemented in a C++ library for inference on graphical models called openGM [3] developed at the HCI as part of my thesis in collaboration with Björn Andres. The internal implementation of the openGM library is much more general than required for the problem discussed here. It can deal with arbitrary commutative semirings and factor models with discrete factors of arbitrary order. For a current published evaluation of higher order models we refer to [3]. We will further concentrate on second order models as used in our visual object detection model.

We will evaluate the four measurements for real world and synthetic data.

- **Energy:** Mean energy over all test data – standard deviation is given in brackets
- **Optimality:** Number of optimal integer solutions in percentage
- **Run-time:** Mean run time for optimization over all test data (setup time for the solvers is not taken into account) – standard deviation is given in brackets
- **gap $\leq 10^{-6}$:** Number of integer solutions in percentage for which the algorithm guarantees optimality within a precision of 10^{-6}

The parameters for the algorithms used for our evaluation are tuned by hand, where we stick to conservative selection rules. The evaluation concerning the run times of the algorithms have to be treated with caution for several reasons: Firstly, for different algorithms different stopping criteria are used, which are not comparable directly. For LBP and TRBP we use a distance measure on the messages as stopping criteria. For the LP-solver and the Dual Decomposition, the primal dual gap is used to determine termination. Furthermore, the speed of convergence of LBP and TRBP depends on the choice of damping. Since we chose this conservatively, we would expect to obtain faster convergence results for smaller damping values as long as this causes no significant oscillation. Another option to speed up LBP and TRBP would be to use special data structures for second order models. The current implementation of factors of arbitrary order cause a lot of computational overhead, which can be omitted for second order models. We would expect a speed up of 10 to 100 for specialized implementation.

For the DD-method, the stopping criteria is defined on the gap, which contains the duality and relaxation gap. Consequently, the current implementation runs until the maximal number of iterations for non-tight decompositions is reached. Furthermore, the step size sequence has a significant influence on the convergence behavior of DD. Tuning this parameters by hand is very cumbersome and is the major draw back of this method. We tune the parameters manually or start with a sufficient large step size. The method which additionally smooths the subgradients over sequence of iterations are only applied on the Human Eva dataset.

All experiments are performed on Pentium Dual 2.00 GHz machines with a 64-bit Ubuntu 10.04.1 LTS operating system. All algorithms are evaluated on a single CPU-core.

We intentionally consider only the optimization problem and do not focus on the evaluation of the computer vision problem, since we clearly want to distinguish between modeling and optimization. One might argue that even a suboptimal solution in terms of energies might lead to similar or better results if evaluation is restricted to the computer vision problem. We will show exemplary on the Human Pascal dataset that this is not the case. Approximative methods lead to significant worse results than optimal solvers from the viewpoint of optimization and application.

However, tracing the reason for poor results in the individual parts, is difficult when evaluating the whole approach exclusively. Optimal inference methods guarantee that at least the optimization of our approach is exact and further insufficiency is caused by the model.

4.4.1. Faces

The Face dataset, introduced in Section 3.4.1, includes 285 test-images. Each face contains 5 parts for which in average 75 candidates exist. For combining the local feature functions of the face model, we use the heuristic model parameters. The learned parameters might be biased to LBP and to approximations using the local polytope, as reported in [168].

We apply LBP and TRBP for maximal 1000 iterations with a damping value of 0.3. For TRBP we use all spanning trees, such that $\rho_e = 2/5$ for all $e \in E$. For our dual decomposition approach, we use two 3-fans with the inner node sets $\{1, 2, 3\}$ and $\{4, 5, 1\}$.

The Face dataset is rather simple. For over 95% of the images the gap between the energy of the rounded integer solution and the original solution of the LP is less than 10^{-6}. In

Algorithm	Energy	Optimality	Run-time in sec.	gap$\leq 10^{-6}$
LBP	0.78 (2.77)	96.49%	0.432 (1.181)	-
TRBP	0.70 (1.40)	96.84%	1.059 (0.837)	-
A^*	0.62 (0.18)	100.00%	0.005 (0.005)	100.00%
MIP	0.62 (0.18)	100.00%	1.381 (1.032)	100.00%
LP	3.05 (17.45)	96.14%	1.274 (0.582)	95.79%
DD-3-fan	0.62 (0.18)	100.00%	0.088 (0.470)	100.00%

Table 4.1.: Empirical results for the Face dataset. The A^*-search, MIP-solver and the dual decomposition method with two 3-fan subproblems determines for all problems the global optimum. The methods based on local polytope relaxations obtain the optimum in more than 95%. The rounding of the fractal solutions of the LP leads in 6 cases to integer solutions with high energies, which is indicated by the high standard deviation for the energy of LP solutions, given in brackets. However, LP can guarantee global optimality for 273 of 285 images.

the cases in which the LP does not reveal a tight bound, at least two parts are labeled as occluded in the global optimal configuration and rounding of the fractal LP solution fails. The number of results where LBP and TRBP ends up with optimal integer solutions is nearly equal. Furthermore, rounding on the min-marginal distributions seems to be more robust as the node based rounding used for LP. This is indicated by the standard deviation of the energies, which is for LBP and TRBP much lower than for the LP. The A^*-search, MIP-solver, and the DD method with two 3-fans, determines global optimal integer solution for all models and guarantees the global optimality. The A^*-method is significantly faster than any other method for this data. For a complete overview, see Table 4.1.

4.4.2. Human Pascal

The Human Pascal dataset contains 441 test-images with human bodies in highly cluttered scenes. Each human body is represented by 13 parts, see Section 3.4.2 for a detailed model description. Again, we use heuristically estimated model parameters to avoid bias. The average number of candidates per part is 22. The absolute number of candidates depends on the clutter in the images and the choice of the threshold parameters for candidate selection and varies between different parts.

We apply LBP and TRBP for maximal 2000 iterations with a damping value of 0.8 on the Human Pascal dataset. For TRBP, we use all spanning trees such that $\rho_e = 2/13$ for all edges $e \in E$. For our dual decomposition approach, we use five 3-fans with the inner node sets $\{1, 2, 3\}$, $\{4, 5, 6\}$, $\{7, 8, 9\}$, $\{10, 11, 12\}$, and $\{13, 1, 2\}$.

The graphical models of the Human Pascal dataset are much more challenging than those obtained for Face dataset. On the one hand, the models contain more nodes and on the other hand it is more likely that parts are occluded. This complexity is also reflected by the fact that the LP-solver reveals a gap less than 10^{-6} between energies of rounded integer and fractal solution in less than 50% of the images. Again, LBP and TRBP show better rounding behavior. Especially, LBP solutions give very good results in terms of energy. A^* and the MIP-solver obtain the optimal solution for all problems. A^* requires approximately twice the time LBP needed, but guarantee optimality. The commercial MIP-solver is two times slower than our A^*-implementation. The dual decomposition approach guarantees

Algorithm	Energy	Optimality	Run-time in sec.	gap$\leq 10^{-6}$
LBP	2.40 (1.10)	79.14%	2.201 (5.492)	-
TRBP	5.33 (15.99)	56.92%	13.510 (10.682)	-
A^*	2.35 (0.12)	100.00%	4.535 (18.299)	100.00%
MIP	2.35 (0.12)	100.00%	8.867 (20.888)	100.00%
LP	6.58 (22.98)	53.97%	41.649 (75.871)	49.21%
DD-3-fan	2.45 (1.56)	88.66%	151.271 (495.491)	85.26%

Table 4.2.: Empirical results for the Human Pascal dataset: While A^* and MIP determine the optimum in all 441 images, DD-bounds with five 3-fans can guarantee optimality only for 376 and LP-bounds for 217 images. The rounding to integer solutions fails quite often for LP and TRBP. For the pseudo marginals of LBP, rounding works much better. Concerning the runtime, LBP is the fastest, while A^* guarantees global optimality at the cost of a runtime factor 2.

global optimality for 85.26% of the models. The huge run time is cause by non-optimal step size parameters and may be reduced by further tuning. Furthermore, a more sophisticated stopping criteria could avoid processing the maximal number of steps if a relaxation gap exists. Table 4.2 shows the mean energy and run times for the different algorithms.

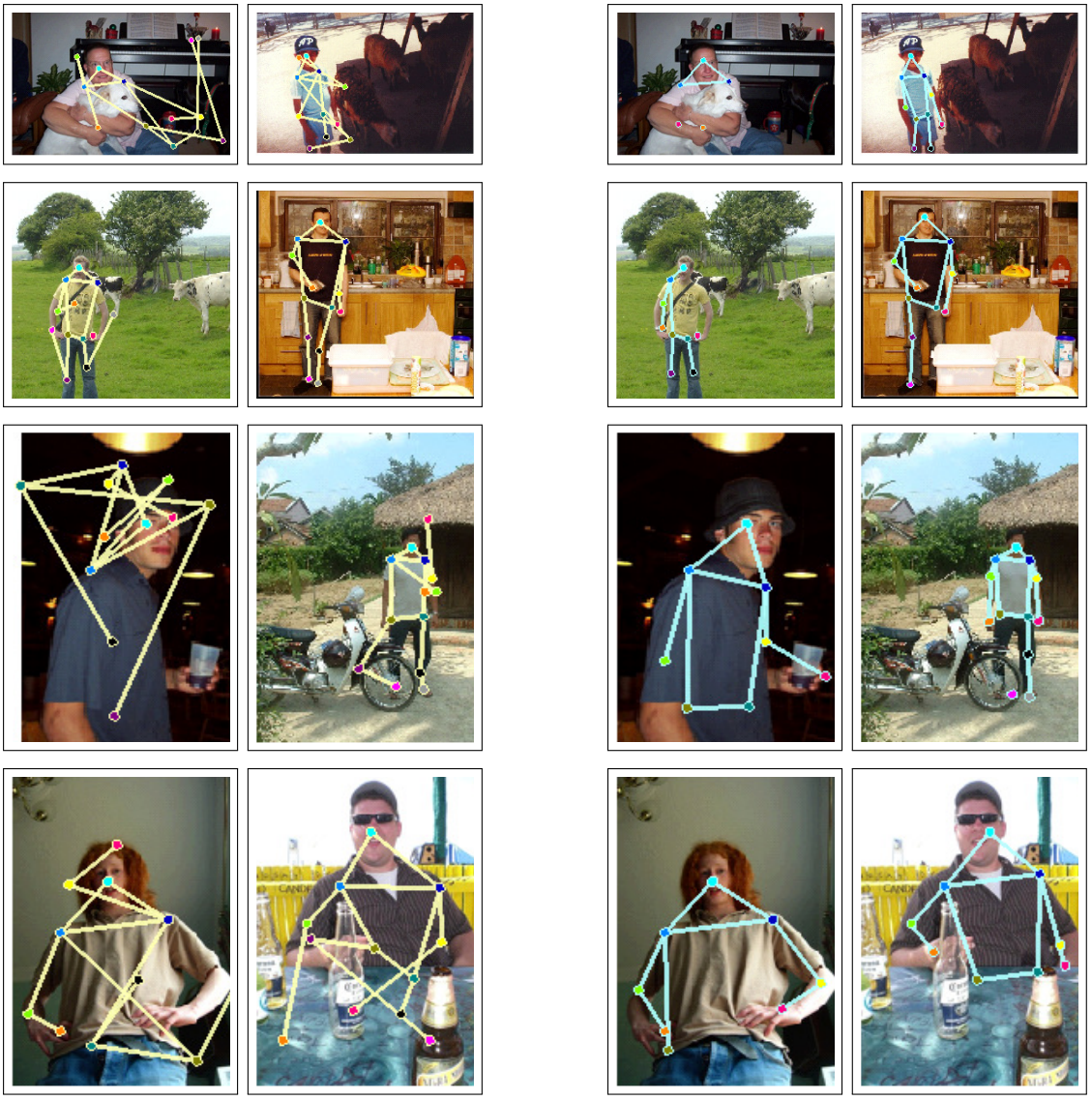
Effects of Approximating Solutions on Visual Object Detection While approximating methods are able to determine the optimal solution in over 50% of the models, they fail for over 20% of the models. As the models typically include candidates which are located next to each other and causes configurations with similar energy, one might expect that suboptimality is primarily due to that fact and for applications those solutions reveal comparable description of the objects, i.e. describe a similar pose. Our experimental results show that this is not the case and many suboptimal configurations lead to impossible descriptions of the pose.

Figure 4.16 shows exemplary results for non-optimal solutions of local polytope based methods compared to the optimal configurations for these models. If the heuristic parameters are used, solutions of approximating solvers tend to occlude less parts and the rounding procedure mix up several configuration with comparable energy. While wrong solutions in Figure 4.16(a) could be caused by insufficient model parameters or bad approximate solutions of the optimization method, any insufficient description of pose in Figure 4.16(b) is caused by the model since the inference problem is solved to optimality.

For models using the learned model parameters, the situation is similar. However, approximating methods tend to occluded to many parts. This behavior is caused by another arrangement of the energy terms for occlusion compared to the models using heuristic parameters. The rounding procedures prefer to select the occluded states when mixing solutions with comparable energy. Figure 4.17 shows exemplary results for these cases and the corresponding optimal configurations calculated with A^*.

4.4.3. Human Eva

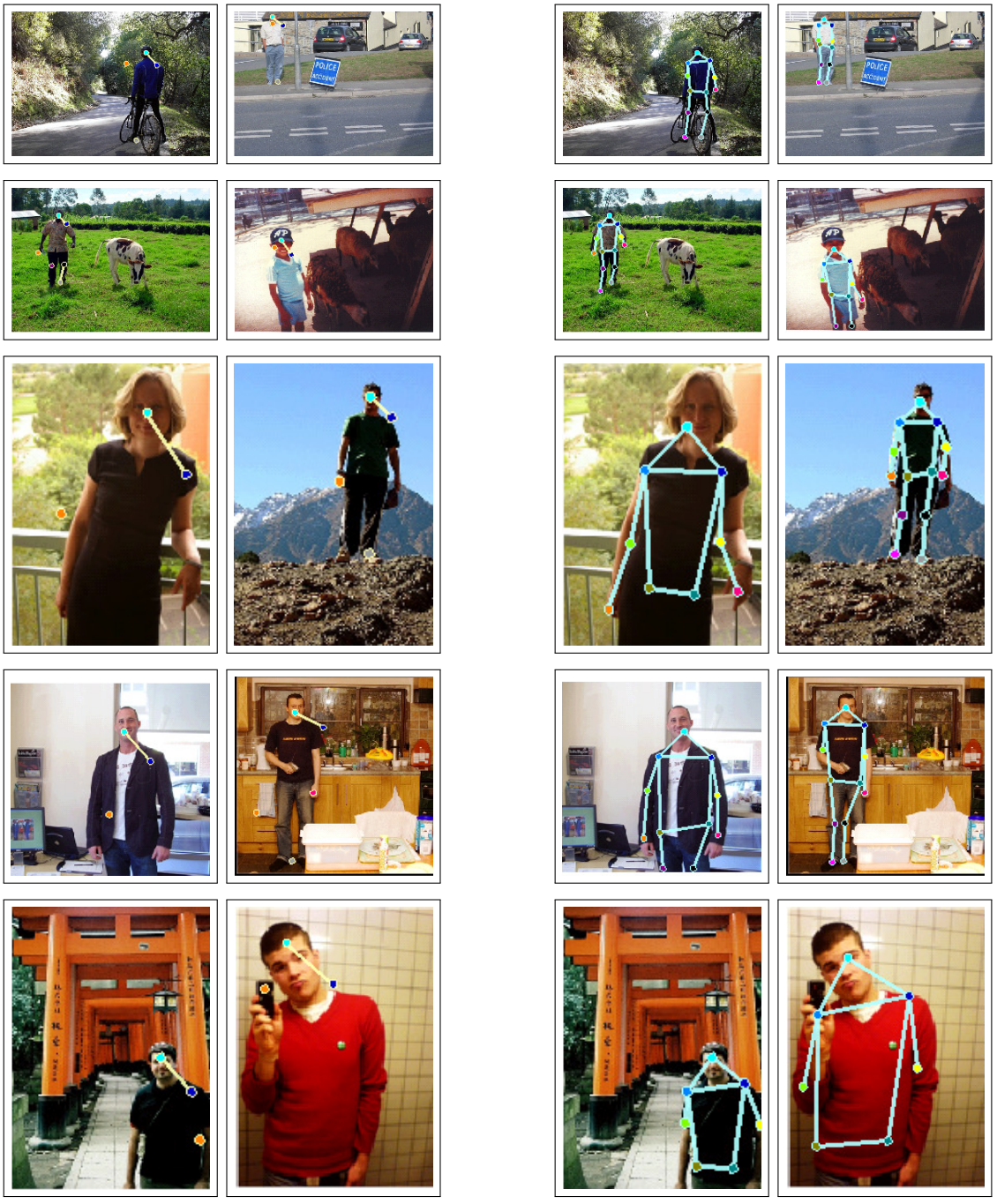
From the Human Eva dataset we use 103 sets of 4 simultaneously captured images. For each single image the model comprises of 15 parts. Overall, each of the 103 models has 60 variables with 19 candidates per part in average. A detailed description of these models is



(a) Solutions of LBP, TRBP or LP

(b) Optimal Solution

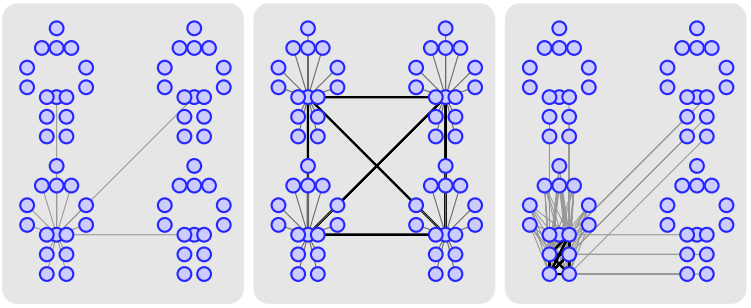
Figure 4.16.: Images where approximation based on the local polytope do not obtain the optimal integer solution for heuristic parameters. **(a)** shows the result for LBP, TRBP or LP and **(b)** the global optimum computed with A^*. When rounding fails, approximative methods typically miss to label some parts as occluded. Instead, the rounding procedure mixes configurations with comparable energies and avoids occlusion. While this leads to impossible poses, the optimal solution to these models correspond to meaningful body poses, which include occluded parts.



(a) Solutions for LBP, TRBP or LP

(b) Exact Solution

Figure 4.17.: Images where approximation based on the local polytope do not found the optimal integer solution for models with parameters learned with CRF-LBP. **(a)** shows the result for LBP, TRBP or LP and **(b)** the global optimum computed with A^*. When rounding fails, these algorithms have a problem with handling occlusion. Contrary to the models using heuristic parameters, the models with trained parameters trend to label parts occluded. Local rounding procedure mixes up solutions with comparable energy and preferably select the occluded candidates.



(a) 1-fan (b) 4-fan (c) 5-fan

Figure 4.18.: The three graphs above sketch the structure of subproblems corresponding to three decompositions of the graphical model used for the HumanEva data. The 4-fan subgraphs include all epipolar constraints between corresponding parts. The 5-fan decomposes the 15 nodes in each single view into three 5-fan substructures.

given in Section 3.4.4. For these models the A^*-method cannot be applied, as the simple branching strategy of our A^*-approach, which enables fast calculation of bounds, is not able to exclude enough subconfigurations in low levels in the search tree. This causes a large quantity of nodes which have to be explored. Surprisingly, commercial MIP-solvers can deal with this problem. We suppose that this behavior is caused by the fact that the MIP-solver is based on more educated and dynamic branches compared to our simple and static branches in the A^*-approach and the LP relaxation provides tighter bounds than our tree approximations in these cases.

However, since we formerly have assumed that neither LP nor MIP solvers are applicable to this problem, we applied the introduced dual decomposition scheme with k-fans. We tested three different k-fan structures on our problem, which are visualized in Figure 4.18. The simplest decompositions contains 60 1-fans as sketched in Figure 4.18a. To obtain tighter relaxations, we tested also a decomposition into 12 5-fans as sketched in Figure 4.18c. Surprisingly, this does not lead to better results, rather the results for comparable runtime become worse.

The additional constraints enforced by the 5-fans, seem to be unimportant. Furthermore, the complex subproblems reduce the number of iterations which can be done in a fixed amount of time. We select another set of k-fans which includes only four inner nodes. Those nodes correspond to a body parts in all four views. Overall this decomposition contains 15 4-fans as sketched in Figure 4.18b. It turns out that exactly this additional constraint, which enforce global consistency of the mean parameters corresponding to all epipolar edges of the body parts, leads to significant tighter bounds. Figure 4.19 shows the changes of upper and lower bounds over time for a set of images where a relaxation gap remains. We run the projected subgradient method for each of these decompositions for 1 hour. While the 4-fan decomposition leads to a small gap after a few minutes, the gap of the 1- and 5-fan decomposition is still large after 1 hour.

In fact, it takes quite long until the subgradient method converges and we observed considerable oscillation. This motivates the use of a smoothed subgradient method as suggested in Section 4.3.4. While this includes an additional parameter which has to been tuned, we obtain for all three decompositions better convergence behavior. In Table 4.3 we mark the results for our dual decomposition approach where smoothing is used by an asterisk.

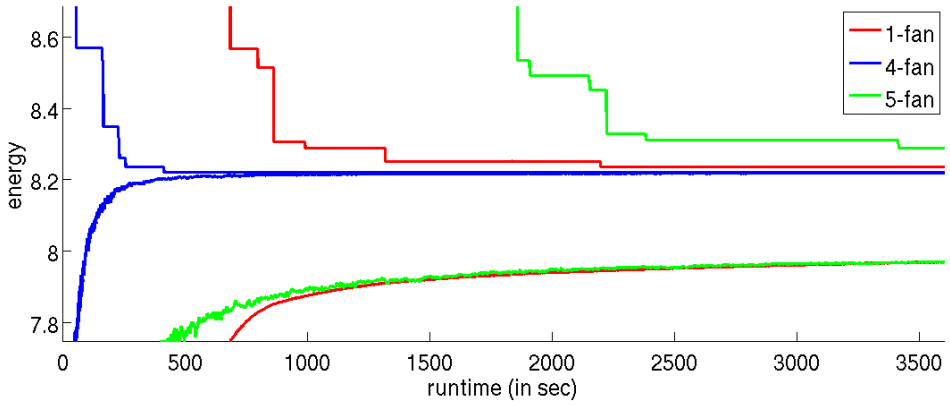


Figure 4.19.: The plot shows the progression of lower and upper bounds as a function of runtime. While both 1-fan and 5-fan decompositions restricted to single views perform similarly, the 4-fan decomposition enforcing epipolar consistency generates significantly tighter bounds and better integer solutions. The decompositions are sketched in Figure 4.18.

We also compare this results to the standard message passing methods. We run LBP and TRBP with a damping of 0.8 (LBP) and 0.4 (TRBP) for at most 2000 iterations. For edges between nodes of a single view we set $\rho_e = 2/15$ and for epipolar edges we set $\rho_e = 1/30$. The idea behind this choice is that we consider each view independently. This ends up in 4 disconnected trees. We now can add 3 of the $6 \cdot 15$ epipolar edges to build a spanning tree. Of course, some combinations will not define a tree, but for reasons of symmetry this can be ignored. For TRBP, the convergence is rather slow and even after 2000 iterations the progress of messages updates has not converged. We supposed that the small values of ρ, especial for the epipolar edges, cause this behavior. LBP converge very fast if it does not start to oscillate.

Due to the lack of an optimal integer solution, we decide to try MIP-solvers. We select the probably current fastest MIP solver, CPLEX [1]. Surprisingly, the CPLEX mixed integer solver solves the optimization problems for the HumanEva dataset extremely fast and significantly faster than it is solved by the relaxed LP.

A further observation is that the standard LP solver ends up with tight lower bounds than the dual decomposition approaches more often. Since convergence rates of subgradient methods are rather poor, this is not surprising. However, the k-fan decompositions find an nearly optimal solution in all cases, see Table 4.3, while at LP with node based rounding fails often, indicated by the high standard deviation of it energy value.

4.4.4. Synthetic Data

Let us now investigate the possible causes that render the inference problem hard for different methods. To this end, we will consider second order, fully connected models, with independently sampled factor functions. The main advantage of this synthetic data is that we can generate arbitrary many models with arbitrary parameters. The main disadvantage is that the assumption that all factor functions are independent does not hold in real world scenarios in general. However, to gain the flexibility to generate arbitrary models, we have to deal with this approximation of the reality. We will parametrize the used models by,

Algorithm	Energy	Optimality	Run-time in sec.	gap$\leq 10^{-6}$
LBP	10.29 (8.03)	72.82%	38.920 (46.409)	-
TRBP	12.05 (11.62)	61.17%	183.047 (5.158)	-
MIP	8.50 (0.82)	100.00%	7.983 (3.557)	100.00%
LP	14.96 (20.69)	65.05%	211.756 (46.451)	62.14%
DD-1-fan	8.50 (0.83)	95.15%	3195.583 (826.848)	21.36%
DD*-1-fan	8.50 (0.83)	96.12%	3005.632 (916.892)	28.16%
DD-4-fan	8.50 (0.82)	100.00%	2451.643 (1084.023)	60.19%
DD*-4-fan	8.50 (0.82)	100.00%	1779.061 (1170.113)	73.79%
DD-5-fan	8.50 (0.83)	89.32%	3464.223 (419.755)	11.65%
DD*-5-fan	8.50 (0.83)	90.29%	3237.776 (752.950)	23.30%

Table 4.3.: Empirical results for 103 models taken from the HumanEva dataset. A^* can not be applied to this model. The DD-methods are interrupted after 1 hour runtime. The commercial MIP-solver performs overall best. Dual decomposition approaches provide good integer solutions but converge slowly. Methods based on a local polytope relaxation suffer from bad rounding schemes if the solution of the LP is fractal.

- the number of nodes ($|V|$)
- the number of labels per variables ($|\mathcal{X}_a|$)
- the coupling strength of the pairwise terms relative to the unary terms (α)

and define a energy function

$$J(x) = (1-\alpha)\sum_{a\in V} f_a(x_a) + (\alpha)\sum_{ab\in E} f_{ab}(x_a, x_b) \tag{4.158}$$

for all $x \in \mathcal{X}$. For factors, values v are sampled uniformly from the interval (0, 1], and the values of factors set to -log(v), i.e.

$$f_C(x_C) = \log(v) \qquad v \sim (0,1] \qquad\qquad \forall C \in V \cup E,\ x_C \in \mathcal{X}_C. \tag{4.159}$$

We use the coupling strength to adjust the relative importance of the pairwise factors and the unary factors. For small α, the objective is dominated by the unary terms while for larger α the problem becomes less locally. We use the following parameters: For LBP and TRBP we use a damping of 0.8 and at most 2000 iterations. For TRBP we use all spanning trees. The maximal number of iterations for DD is also set to 2000, the step size parameters are selected conservatively and equal for all models.

Influence of the Number of Random Variables

Let us first analyze the impact of the number of nodes to the empirical complexity. Therefore, we fix the number of labels to 20 and set $\alpha = 0.5$. We vary the number of variables from 3 up to 18 and generate for each model size 10 models. Figure 4.20 shows the average results in terms of energy and runtime. The MIP-solver and A^* have exponential complexity in $|V|$. For this synthetic data, A^* is able to deal with larger models than the MIP-solver. For the DD approach it is for larger $|V|$ more likely that a relaxation gap remains and the maximal number of iteration is required. Using more sophisticated stopping

criteria and proper step size parameters would decrease these run times. Concerning the energies, LBP performs best, when exact methods become to time consuming. However, LBP provides no criteria to check optimality.

Influence of the Number of Labels

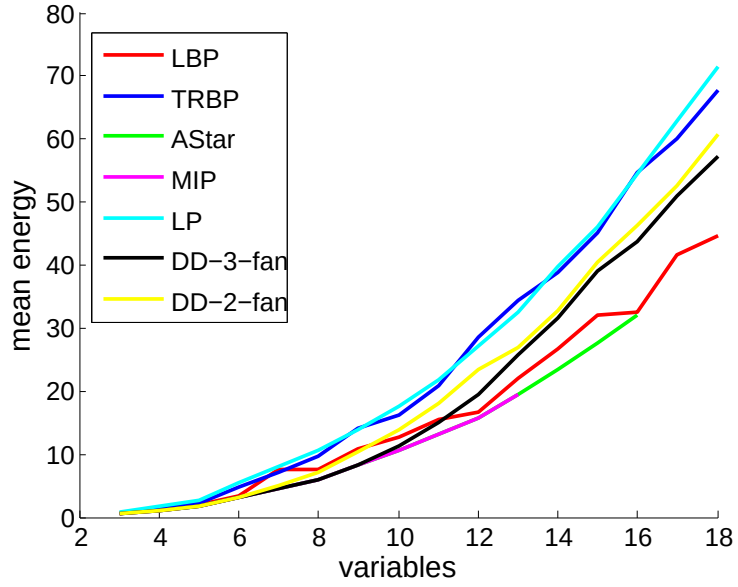
For the analysis of the impact of the number of labels, we fix the number of nodes to 6 and set the coupling strength to 0.5. We evaluate in each case 10 models for $|\mathcal{X}_a| \in \{2, 4, 8, 16, 32, 64, 128, 265\}$. Concerning run time and energy, the A^*-search is for all settings the best. For larger quantity of labels, the performance of the commercial MIP-solver breaks down extremely. We assume that the reason for this is that the LP becomes large and the internal heuristics of the MIP-solver gain no profit from the arbitrary factor functions. The solutions of DD-3-fan are also very promising, but the runtime increases fast. All methods based on local polytope relaxations end up with insufficient results.

Influence of the Coupling Strength

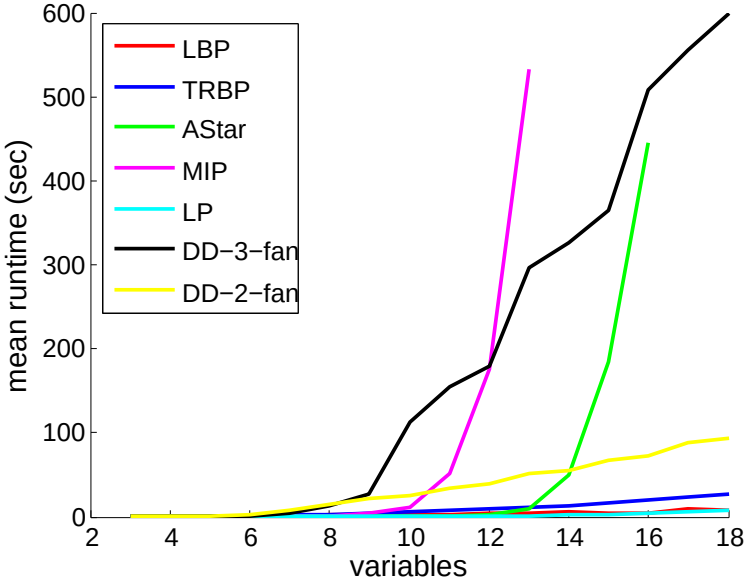
We vary the coupling strength from 0 to 1 and fix $|V| = 6$ and $|\mathcal{X}_a| = 8$. For each coupling strength we sample 10 models. For larger coupling strength the runtime increases by trend for all algorithms. While MIP, DD-3-fan and A^* always end up in the global optimum, the remaining methods fail to find the global optimum. With increasing coupling strength, the rounding methods which are based on local decisions, lead to mixing of configurations and suboptimal integer solutions more often. LBP seems to suffer less from this. TRBP performs slightly better than the LP-solver, which may be caused by the inspection of marginal distributions over edges.

Influence of the Number of Inner Nodes of k-Fan-Subproblems

Finally, we analyze the impact of k for k-fan decomposition. For this, we generate a full connected, second order model with 12 variables and 10 labels per node. We solve this model by the dual decomposition approach with a 1-, 2-, 3-, 4-, 6-, and 12-fan decomposition. The parameters for the step sizes of the subgradient methods are tuned by hand. We run these algorithms for at most 2 minutes and stop if the gap between the optimal integer solution and the bound is below 10^{-6}. Optimality of the solution is guaranteed by the 12-fan decomposition, which requires only one round of the A^* algorithm, and the 6-fan decomposition. For all other decompositions there is a gap remaining after 2 minutes. With larger k, we obtain tighter bounds and also better integer solutions. The progress of the bounds for the different decompositions is illustrated in Figure 4.23.

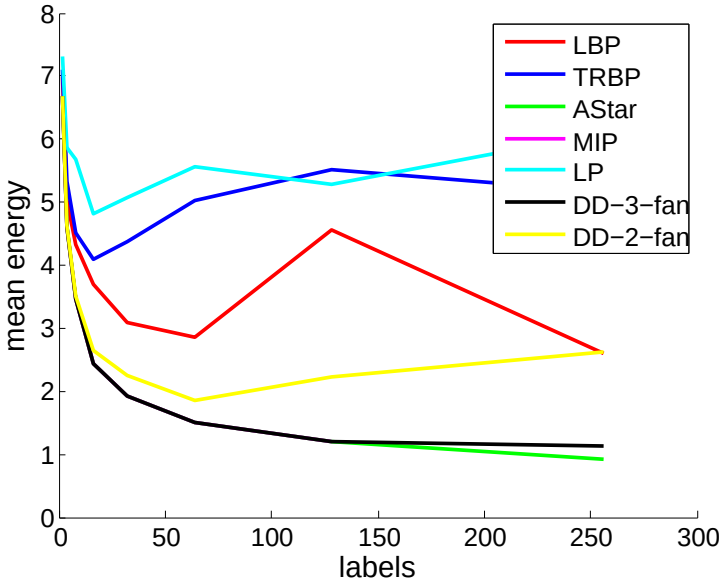


(a) Mean Energies

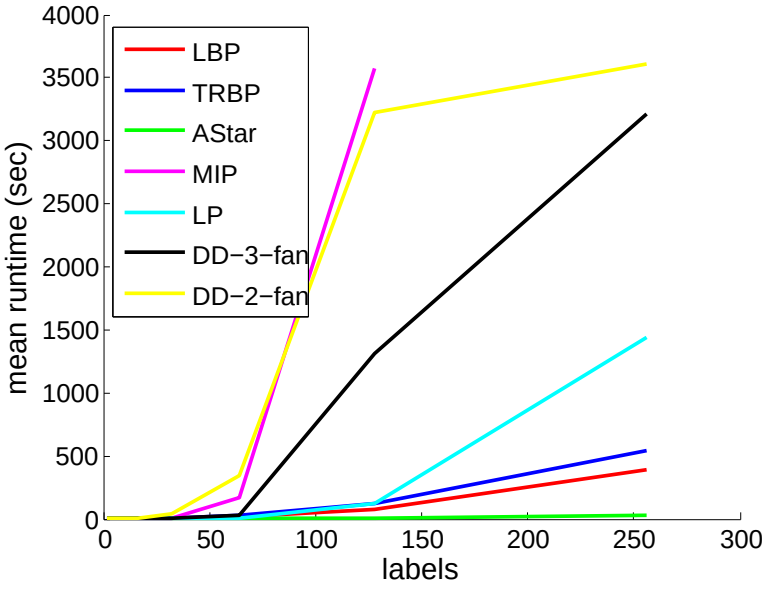


(b) Mean Run-Times

Figure 4.20.: Results for $|\mathcal{X}_a| = 20$, $\alpha = 0.5$ for arbitrary number of variables. For each size 10 models are evaluated. For MIP and A^* we limit the evaluation to manageable sizes. Both, MIP-solver and A^* show exponential complexity in $|V|$, but A^* is for this data more manageable. Among the other approaches, LBP performs empirically best.

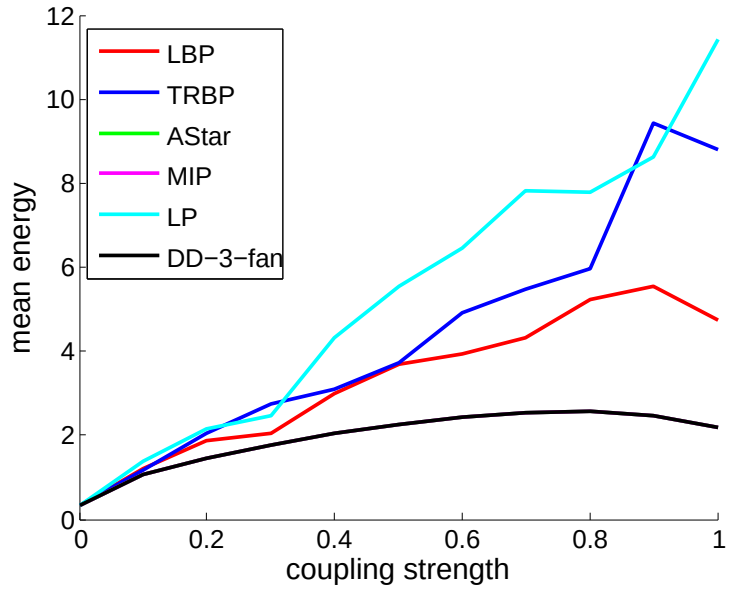


(a) Mean Energies

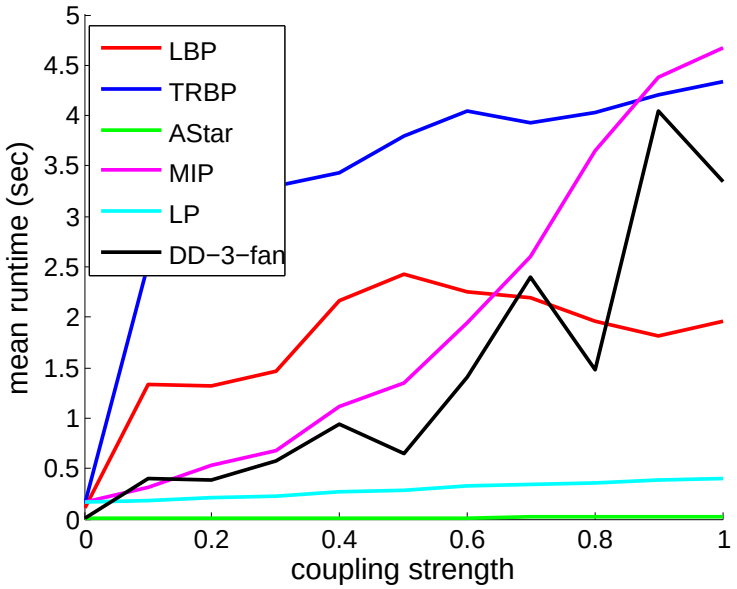


(b) Mean Run-Times

Figure 4.21.: Results for $|V| = 6$, $\alpha = 0.5$ for arbitrary number of labels. For each size 10 models are evaluated. For larger numbers of labels the commercial MIP-solver performs worse than our A^*-approach. Also the 3-fan decomposition given good results in terms of energies.

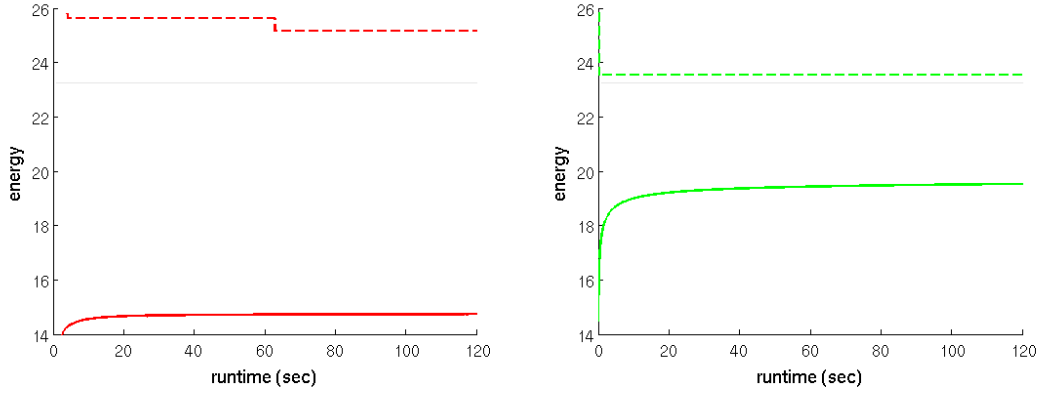


(a) Mean Energies



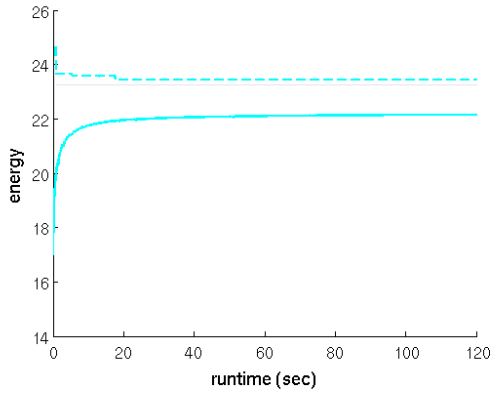
(b) Mean Run-Times

Figure 4.22.: Results for $|V| = 6$, $|\mathcal{X}_a| = 8$ for arbitrary coupling strength. For each coupling strength 10 models are evaluated. With increasing coupling strength, all algorithms require by trend more run time. Overall, A^* perform best. The optimal solution is found by MIP, DD-3-fan and A^* for all models.

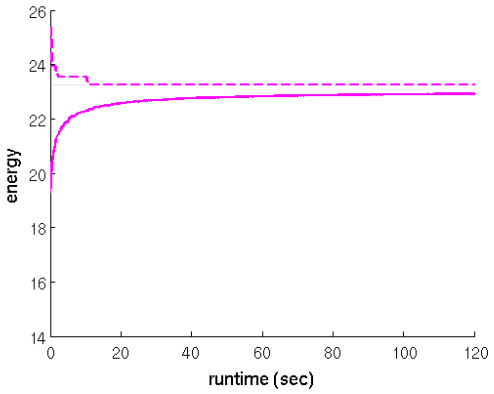


(a) 1-fan

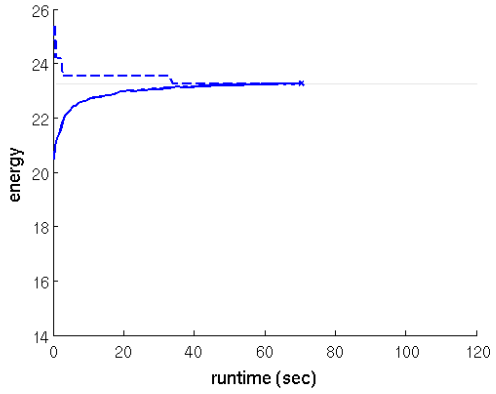
(b) 2-fan



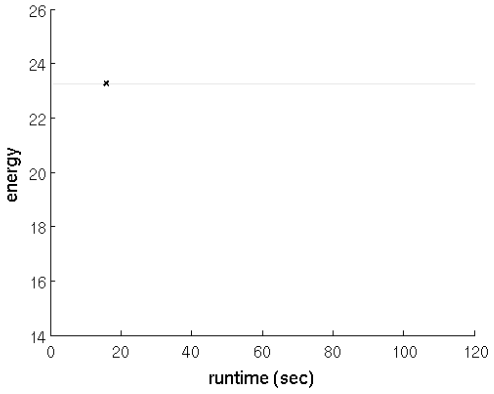
(c) 3-fan



(d) 4-fan



(e) 6-fan



(f) 12-fan

Figure 4.23.: Solving synthetic models with k-fan decompositions. With increasing k we get tighter relaxations. The 12-fan decomposition is equal to one round of the A^* algorithm. Beside this decomposition only the 6-fan decomposition stops with a gap less than 10^{-6}.

CHAPTER 5

CONCLUSION

In this thesis we presented a practicable and expandable probabilistic framework for part-based object detection. It is based on highly connected graphical models, in combination with efficient optimization algorithm, which allow exact MAP inference. In contrast to inexact algorithms such as Loopy Belief Propagation, we can rule out approximate inference as a source of error. Furthermore, the proposed A^*-method proved to be computationally more efficient for the class of graphical models studied, than approximative standard methods.

In Chapter 2 we introduced the basic principles of graphical models, which constitute the mathematical basis of our approach. Although we focus on part-based models for visual object detection, the results of this thesis – especially those concerning optimization – can be employed in many other applications as well.

A major issue in connection with object detection is the large variability of object appearance, especially for the human body data. We have tackled this by using discriminative classifiers and by incorporating them into a probabilistic conditional random field (CRF) framework. Each single classifier is defined locally, thus it has to cope with a smaller local variability only. We assume no special form of the input features. Hence, the presented framework can easily be expanded to other collections of features. While having more feature functions makes CRF-learning certainly more computationally expensive, there is no impact on the complexity of MAP-inference, because these feature functions are combined into a single potential for each node and edge before inference. The use of completely connected graphs for model representation introduces redundancies enabling to average out noise of individual features when inferring difficult object configurations and allows for efficient handling of occlusion. The inclusion of additional features is straightforward if they only depend on the configuration of two parts. The extension to features using triples or more parts is also conceivable. However, the computational complexity increases rapidly with the number of parts.

A configuration of an object is completely defined by the location of its parts, which in the case of humans are the joints as opposed to parametrization of the limbs. However, even using this compact representation requires a reduction of the problem complexity. Our bottom-up process ensures that early stages keep the number of candidates small, thus

allowing for the computation of structural information in form of pairwise appearance features without suffering from the quadratic complexity usually involved. Nevertheless, the evaluation of the feature functions is currently the most time consuming part of the overall model. On the other hand, meaningful appearance terms are essential for achieving a reasonable accurate method. The geometric prior alone does not suffice to estimate unusual poses when the accuracy for the classifier is low.

Optimization issues of the involved inference problems were discussed in Chapter 4. While acyclic models inference problems are efficiently solvable by dynamic programming, for cyclic models approximative methods have mostly been employed so far. We investigated exact solvers for the MAP problem and came up with a shortest path based problem transformation. This was solved by A^*-search together with an admissible heuristic based on an acyclic problem relaxation. This method exploits that local feature functions themselves already exclude large sets of configurations. This "online-pruning" reduces the number of configurations which have to be explored and makes exact inference computationally feasible on standard PCs for fully connected model-graphs up to a few tens of nodes and a large set of labels. However, for larger models the search becomes computational infeasible even with the A^*-search. A main limitation of the A^*-approach is the fixed branching strategy, which allows only a fixed sequence of branches but enables fast calculation of bounds. Dynamic branching strategies together with LP-based bounds, as used in standard mixed integer program (MIP) solvers, overcome this problem, but do not scale to large problems, too. However, somewhat surprisingly it turned out that standard MIP-solvers are applicable to many medium sized computer vision problems, for which approximative solvers have been used so far.

We also considered an alternative line of research and investigated convex relaxations and corresponding optimization methods for inference problems. Contrary to search based methods, they are applicable to large-scale problems and provide bounds on the optimal objective value. Using the theory of exponential families we obtained a linear program equivalent to the MAP problem. The number of affine constraints of the LP grows in general exponentially with the number of variables. Consequently, solving this LP is computationally infeasible for general graphs. While approximations over the local polytope is the common way to relax this problem, our approach considerably improves the tightness of the relaxation by decomposing the original problem into cyclic substructures. Provided the MAP problems with such substructures are efficiently solvable, projective subgradient methods can be applied to solve the dual problem. The main drawbacks of this approach is that subgradient methods converge more slowly and lack a good stopping criterion. On the other hand, this approach can be used to solve the LP with the local polytope constraints or tighter relaxations, without need to set up the affine constraints explicitly. This makes this approach applicable to large scale data. Furthermore, it empirically outperforms other scalable methods in terms of the energy of computed integer solutions.

Further Work

While this thesis presents a snapshot of the ongoing work, several interesting questions are still open or are raised by the results of the work presented here.

Concerning the modeling aspect, further improvement in accuracy involve better local descriptors and feature functions. In some cases, however, we would like to increase detection speed by removing redundant computations in the detection phase. Applying variations of the CRF learning algorithm by the inclusion of additional prior terms on the

model parameters that favor sparse solutions and also penalize costly computations could help in such settings. Moreover, expansion to third and higher order models would in principle allow to design scale- and rotation-invariant geometric priors which are useful for many applications. However, these applications would also require features robust against these transformations. From the view point of optimization such models are slightly harder but still practicable, as we have reported currently in [3].

We have shown that the MAP problem can be solved by search based algorithms for moderate problem sizes. For larger problem sizes neither A^*-search nor commercial MIP solvers are computationally feasible any more. Even solving the LP with the local polytope relaxation will suffer from large memory requirements when using standard solvers. However, in principle we do not see any reason why standard methods from convex analysis should not be applicable, as long as the affine constraints of the LP can be presented in an implicit form, making use of efficient data structures.

APPENDIX A

APPENDIX

A.1. Definitions

Definition A.1. A ***monoid*** is a set, Ω, together with a binary operation $\oplus$ that satisfies the following three axioms:

1. $\forall a, b \in \Omega: \quad a \oplus b \in \Omega$
2. $\forall a, b, c \in \Omega: \quad (a \oplus b) \oplus c = a \oplus (b \oplus c)$
3. $\exists \mathbf{1}_{\oplus} \in \Omega: \quad \forall a \in \Omega: \ \mathbf{1}_{\oplus} \oplus a = a$

A monoid is called ***commutative monoid*** if the operation $\oplus$ is commutative, that is

4. $\forall a, b \in \Omega: \quad a \oplus b = b \oplus a$

Definition A.2. A commutative semiring $(\Omega, \odot, \oplus)$ is a set Ω, together with two binary operations called $\oplus$ and $\odot$, which satisfy the following three axioms:

1. The set Ω together with the operation $\oplus$ is a commutative monoid.
2. The set Ω together with the operation $\odot$ is a commutative monoid.
3. $\forall a, b, c \in \Omega: \quad (a \oplus b) \odot (a \oplus c) = a \oplus (b \odot c)$

Definition A.3. ***Convex Hull***

$$\text{conv}(S) = \left\{ \sum_{i=1}^{N} a_i \cdot x_i | S = \{x_1, \ldots, x_N\},\ a_i \in \mathbb{R},\ a_i \geq 0,\ \sum_{i=1}^{N} a_i = 1 \right\}$$

Definition A.4. ***Affine Hull***

$$\text{aff}(S) = \left\{ \sum_{i=1}^{N} a_i \cdot x_i | S = \{x_1, \ldots, x_N\},\ a_i \in \mathbb{R},\ \sum_{i=1}^{N} a_i = 1 \right\}$$

Definition A.5. ***Expectation Value***
For a random variable or vector its expectation value is defined as

$$\mathbb{E}(X) := \sum_{x \in \mathcal{X}} p(x)\, x, \quad \mathbb{E}(X) := \int_{\mathcal{X}} p(x)\, x\, dx \tag{A.1}$$

The expectation value for a function $f(x)$ with respect to the distribution $p(x)$ is defined by

$$\mathbb{E}_p[f(x)] := \int_{\mathcal{X}} p(x)\, f(x) dx \tag{A.2}$$

Definition A.6. ***Variance***
For a real values random variable its variance is defined as

$$\mathrm{Var}(X) := \mathbb{E}((X - \mathbb{E}(X))^2) = \mathbb{E}(X^2) - \mathbb{E}(X)^2 \tag{A.3}$$

$$\mathrm{Var}(X) = \sum_{x \in \mathcal{X}} p(x)\, (x - \mathbb{E}(X))^2, \quad \mathrm{Var}(X) = \int_{\mathcal{X}} p(x)\, (x - \mathbb{E}(X))^2\, dx \tag{A.4}$$

Definition A.7. ***Covariance Matrix***
For a random variable or vector its covariance matrix value is defined as

$$\mathrm{Cov}(X) := \mathbb{E}[(X - \mathbb{E}(X))(X - \mathbb{E}(X))^\top] = \mathbb{E}(XX^\top) - \mathbb{E}(X)\mathbb{E}(X)^\top \tag{A.5}$$

Definition A.8. ***Entropy***
For a random variable or vector its entropy is defined as

$$H(X) := -\sum_{x \in \mathcal{X}} p(x)\, \log(p(x)), \quad H(X) := -\int_{\mathcal{X}} p(x)\, \log(p(x))\, dx \tag{A.6}$$

Definition A.9. Given two problems

$$(A) \qquad \min_{x \in U}\ f(x) \tag{A.7}$$

$$(B) \qquad \min_{x \in W}\ g(x), \tag{A.8}$$

with the same decision variable x. We say problem (A) is a relaxation of the problem (B), if and only if

(i) $W \subset U$, and

(ii) $\forall x \in W : f(x) \leq g(x)$.

A.2. Duality

Consider a optimization problem given in the standard form

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f_0(x) \\ \text{s.t.} \quad & f_i(x) \leq 0 \qquad i = 1 \dots m \\ & g_i(x) = 0 \qquad i = 1 \dots p \end{aligned} \tag{A.9}$$

with the domain $\mathcal{D} = \bigcap_{i=1}^m \mathrm{dom}(f_i) \cap \bigcap_{i=1}^p \mathrm{dom}(g_i)$. We assume that $\mathcal{D}$ is not empty and denote the optimal value of (A.9) by p^*.

Lagrangian The basic idea of Lagrangian Duality is to include the constraints in (A.9) into the objective function by a sum of weighted constraint functions. We define the Lagrangian function L as a function from $\mathbb{R}^n \times \mathbb{R}^m_+ \times \mathbb{R}^p$ to $\mathbb{R}$ as

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i g_i(x) \tag{A.10}$$

with the domain $\text{dom}(L) = \mathcal{D} \times \mathbb{R}^m_+ \times \mathbb{R}^p$. The vectors λ and ν are called dual variables or Lagrangian multiplier vectors and are associated with the inequality respectively the equality constraints.

Lagrangian dual function The Lagrangian dual function (or just dual function) is defined as a function from the space of dual variables to the minimum value over x of the corresponding Lagrangian.

$$g(\lambda, \nu) = \inf_{x \in \mathcal{D}} L(x, \lambda, \nu) \tag{A.11}$$

The dual function has two important properties. Firstly, since the dual function is the point-wise infimum of a family of affine functions of (λ, ν), it is concave, even when the problem A.9 is not convex. And secondly, the dual function is for any $\lambda \geq 0$ and any ν a lower bound on the optimal value p^* of the problem A.9.

$$g(\lambda, \nu) \leq p^* \qquad \text{for } \lambda \geq 0 \tag{A.12}$$

Consequently the dual function takes the value $-\infty$ if the Lagrangian is unbounded below in x.

The proof for (A.12) is quite simple. Suppose it is given the feasible point $\tilde{x} \in \mathcal{D}$ and $\lambda \geq 0$. By definition of $\mathcal{D}$ we know that $f_i(x) \leq 0$ and $g_i(x) = 0$ holds and therefore

$$g(\lambda, \nu) = \inf_{x \in \mathcal{D}} L(x, \lambda, \nu) \leq L(\tilde{x}, \lambda, \nu) \leq f_0(\tilde{x}) \tag{A.13}$$

Since this holds for any feasible point $\tilde{x} \in \mathcal{D}$, the inequality A.12 follows.

A pair (λ, ν) is in the domain of the dual function $\text{dom}(g)$ if the dual function is an none-trivial bound on p^*, that is $g(\lambda, \nu) > -\infty$.

Lagrange dual problem For each pair (λ, ν) with $\lambda \geq 0$ the dual function gives a lower bound on p^*, see (A.12). A natural question is: "Which pair produce the best bound?" This leads to the optimization problem

$$\max_{(\lambda,\nu) \in \mathbb{R}^m_+ \times \mathbb{R}^p} \quad g(\lambda, \nu) \tag{A.14}$$

This problem is called the Lagrange dual problem. The pair (λ, ν) is feasible if it fulfill the constraints $(\lambda \geq 0)$ an the objective is feasible $(g(\lambda, \nu) > -\infty)$. The Lagrange dual problem is always a convex optimization problem, since the objective to be maximized is concave and the constraints are convex.

We denote the optimal value of the dual problem with d^*, which is by definition the best lower bound on p^* we can obtain from the Lagrangian dual function. The inequality

$$d^* \leq p^* \tag{A.15}$$

is always fulfilled and this property is called weak duality. We refer to difference of p^* and d^* as the optimal duality gap. If the duality gap becomes zero the optimum of the dual is also the optimum of the primal problem. If the inequality becomes a equality such that

$$d^* = p^* \tag{A.16}$$

we say that strong duality holds.

For convex problems of the form

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f_0(x) \\ \text{s.t.} \quad & f_i(x) \leq 0 \qquad i = 1 \dots m \\ & Ax = b \end{aligned} \tag{A.17}$$

we usually, but not always, have strong duality. There are many conditions on problems which guarantee strong duality [134] . These conditions are called constraint qualifications. One simple constraint qualifications is Slater's condition

Theorem A.1. *An optimization problem of the form*

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f_0(x) \\ s.t. \quad & f_i(x) \leq 0 \qquad i = 1 \dots m \\ & Ax = b \end{aligned} \tag{A.18}$$

fulfills Slater's condition if there exists $x \in ri(\mathcal{D})$ such that

$$f_i(x) < 0 \; \forall i = 1 \dots m, \qquad Ax = b.$$

Such points are also called strict feasible, since the inequality constraints hold with strict inequality. Affine inequality constraints do not need to fulfill strict inequality for Slater's condition.

If the problem is convex and Slater's condition holds, than strong duality holds.

Proof. See Section 5.2.3 in [18]. □

A.3. Proofs

Theorem A.2. *Given a acyclic factor graph model $(\oplus, (V, F, E))$ together with a commutative semiring $(\Omega, \odot, op)$. For all $a \in V$, $f \in F$ with $af \in E$ the recursive definition of $M_{a \leftarrow f}(x_a)$ and $M_{f \leftarrow a}(x_a)$ in (4.29)-(4.30) are equivalent to the explicit terms in (4.31)-(4.32).*

Proof. Proof by a complete induction over the $|(V \cup F)_{n \leftarrow m}|$ with $nm \in E$:

- *Basis* $|(V \cup F)_{n \leftarrow m}| = 1$

i) If $a = n \in V$ and $f = m \in F$ than $|(V \cup F)_{a \leftarrow f}| = 1$ implies that the only

neighbour of f is a.

$$M_{a\leftarrow f}(x_a) = \bigodot_{x_{\mathrm{ne}(f)\setminus\{a\}}\in\mathcal{X}_{\mathrm{ne}(f)\setminus\{a\}}} \left(f(x_{\mathrm{ne}(f)}) \oplus \bigoplus_{b\in\mathrm{ne}(f)\setminus\{a\}} M_{f\leftarrow b}(x_b) \right) \tag{A.19}$$

$$= f(x_a) \tag{A.20}$$

$$= \bigodot_{x_{V_{a\leftarrow f}}\in\mathcal{X}_{V_{a\leftarrow f}}} \bigoplus_{g\in F_{a\leftarrow f}} g(x_{\mathrm{ne}(g)}) \tag{A.21}$$

ii) If $f = n \in F$ and $a = m \in V$ than $|(V \cup F)_{f\leftarrow a}| = 1$ implies that the only neighbor of a is f.

$$M_{f\leftarrow a}(x_a) = \bigoplus_{g\in\mathrm{ne}(a)\setminus\{f\}} M_{a\leftarrow g}(x_a) \tag{A.22}$$

$$= \mathbf{1} \tag{A.23}$$

$$= \bigodot_{x_{V_{f\leftarrow a}\setminus\{a\}}\in X_{V_{f\leftarrow a}\setminus\{a\}}} \bigoplus_{g\in F_{f\leftarrow a}} g(x_{\mathrm{ne}(g)}) \tag{A.24}$$

- *Inductive hypothesis:*

  For $a \in V$, $f \in F$ with $af \in E$ (4.32) is the explicit form of (4.32) if $|(V \cup F)_{a\leftarrow f}| \leq N$ and (4.31) is the explicit form of (4.29) if $|(V \cup F)_{f\leftarrow a}| \leq N$.

- *Inductive step:* $|(V \cup F)_{a\leftarrow b}| = N \rightarrow |(V \cup F)_{a\leftarrow b}| = N + 1$

i) If $a = n \in V$ and $f = m \in F$ and $|(V \cup F)_{a\leftarrow f}| = N + 1$

$$M_{a\leftarrow f}(x_a) = \bigodot_{x_{\mathrm{ne}(f)\setminus\{a\}}\in\mathcal{X}_{\mathrm{ne}(f)\setminus\{a\}}} \left(f(x_{\mathrm{ne}(f)}) \oplus \bigoplus_{b\in\mathrm{ne}(f)\setminus\{a\}} M_{f\leftarrow b}(x_b) \right) \tag{A.25}$$

$$\overset{IH}{=} \bigodot_{x_{\mathrm{ne}(f)\setminus\{a\}}\in\mathcal{X}_{\mathrm{ne}(f)\setminus\{a\}}} \left(f(x_{\mathrm{ne}(f)}) \oplus \bigoplus_{b\in\mathrm{ne}(f)\setminus\{a\}} \left(\bigodot_{x_{V_{f\leftarrow b}\setminus\{b\}}\in X_{V_{f\leftarrow b}\setminus\{b\}}} \bigoplus_{g\in F_{f\leftarrow b}} g(x_{\mathrm{ne}(g)}) \right) \right) \tag{A.26}$$

$$= \bigodot_{x_{V_{a\leftarrow f}}\in\mathcal{X}_{V_{a\leftarrow f}}} \bigoplus_{g\in F_{a\leftarrow f}} g(x_{\mathrm{ne}(g)}) \tag{A.27}$$

ii) If $f = n \in F$ and $a = m \in V$ and $|(V \cup F)_{f\leftarrow a}| = N + 1$

$$M_{f\leftarrow a}(x_a) = \bigoplus_{g\in\mathrm{ne}(a)\setminus\{f\}} M_{a\leftarrow g}(x_a) \tag{A.28}$$

$$\overset{IH}{=} \bigoplus_{g\in\mathrm{ne}(a)\setminus\{f\}} \left(\bigoplus_{x_{Va\leftarrow g}\in\mathcal{X}_{Va\leftarrow g}} \bigoplus_{h\in F_{a\leftarrow g}} h(x_{\mathrm{ne}(h)}) \right) \tag{A.29}$$

$$= \bigodot_{x_{V_{f\leftarrow a}\setminus\{a\}}\in X_{V_{f\leftarrow a}\setminus\{a\}}} \bigoplus_{g\in F_{f\leftarrow a}} g(x_{\mathrm{ne}(g)}) \tag{A.30}$$

□

Proof of Theorem 2.8

Proof.

a) Due to the dominated convergence theorem [23], the differential can be relocated into the integral:

$$\begin{aligned}
\frac{\partial A}{\partial \theta_\alpha}(\theta) &= \frac{\int_{\mathcal{X}} \frac{\partial}{\partial \theta_\alpha} \exp\left(\langle \theta,\, \phi(x)\rangle \nu(dx)\right)}{\int_{\mathcal{X}} \exp\left(\langle \theta,\, \phi(x)\rangle \nu(dx)\right)} \\
&= \frac{\int_{\mathcal{X}} \phi_\alpha(x) \exp\left(\langle \theta,\, \phi(x)\rangle \nu(dx)\right)}{\int_{\mathcal{X}} \exp\left(\langle \theta,\, \phi(x)\rangle \nu(dx)\right)} \\
&= \int_{\mathcal{X}} \phi_\alpha(x) p(x|\theta) \nu(dx) \\
&= \mathbb{E}_\theta[\phi_\alpha(X)]
\end{aligned}$$

$$\begin{aligned}
\frac{\partial^2 A}{\partial \theta_\alpha \partial \theta_\beta}(\theta) &= \frac{\partial}{\partial \theta_\alpha} \frac{\partial A}{\partial \theta_\beta}(\theta) \\
&= \frac{\partial}{\partial \theta_\alpha} \left(\int_{\mathcal{X}} \phi_\beta(x) \exp\left(\langle \theta,\, \phi(x)\rangle \nu(dx)\right) \right) \left(\int_{\mathcal{X}} \exp\left(\langle \theta,\, \phi(x)\rangle \nu(dx)\right) \right)^{-1} \\
&= \frac{\int_{\mathcal{X}} \phi_\alpha(x) \phi_\beta(x) \exp\left(\langle \theta,\, \phi(x)\rangle \nu(dx)\right)}{\int_{\mathcal{X}} \exp\left(\langle \theta,\, \phi(x)\rangle \nu(dx)\right)} \\
&\quad - \frac{\int_{\mathcal{X}} \phi_\alpha(x) \exp\left(\langle \theta,\, \phi(x)\rangle \nu(dx)\right) \int_{\mathcal{X}} \phi_\beta(x) \exp\left(\langle \theta,\, \phi(x)\rangle \nu(dx)\right)}{\left(\int_{\mathcal{X}} \exp\left(\langle \theta,\, \phi(x)\rangle \nu(dx)\right)\right)^2} \\
&= \mathbb{E}_\theta[\phi_\alpha(X)\phi_\beta(X)] - \mathbb{E}_\theta[\phi_\alpha(X)]\mathbb{E}_\theta[\phi_\beta(X)]
\end{aligned}$$

b) To show that $A(\theta)$ is a convex function we have to show that the Hessian of $A(\theta)$ is positive semi-definite.

$$\nabla^2 A(\theta) = \mathbb{E}_\theta[\phi(X)\phi(X)^\top] - \mathbb{E}_\theta[\phi(X)]\mathbb{E}_\theta[\phi(X)]^\top = \mathrm{Cov}_\theta(\phi(X))$$

Since each covariance matrix is positive semi-definite, $A(\theta)$ is a convex function.

If the exponential family is minimal, then there is no vector $a \in \mathbb{R}^{|\mathcal{I}|}$ and constant $b \in \mathbb{R}$ such that $\langle a, \phi(x)\rangle = b$ almost everywhere with respect to the base measure. This implies that the variance of $\langle a, \phi(X)\rangle$ is positive for all $a \in \mathbb{R}^{|\mathcal{I}|}$ and by simple reformulations we get the strict convexity of $A(\theta)$.

$$\begin{aligned}
0 &< \mathrm{Var}_\theta[\langle a, \phi(X)\rangle] \\
&= \mathbb{E}_\theta[(\langle a, \phi(X)\rangle)^2] - (\mathbb{E}_\theta[(\langle a, \phi(X)\rangle)])^2 \\
&= a^\top \mathbb{E}_\theta[\phi(X)\phi(X)^\top] a - (\langle a, \mathbb{E}_\theta[\phi(X)]\rangle)])^2 \\
&= a^\top \mathbb{E}_\theta[\phi(X)\phi(X)^\top] a - a^\top \mathbb{E}_\theta[\phi(X)]\mathbb{E}_\theta[\phi(X)]^\top a \\
&= a^\top \mathrm{Cov}_\theta[\phi(X)] a \\
&= a^\top \nabla^2 A(\theta) a
\end{aligned}$$

□

BIBLIOGRAPHY

[1] Cplex 12.1. `www.cplex.com/`.

[2] Aji, S., and McEliece, R. The generalized distributive law. *Information Theory, IEEE Transactions on 46*, 2 (mar 2000), 325 –343.

[3] Andres, B., Kappes, J. H., Köthe, U., Schnörr, C., and Hamprecht, F. An empirical comparison of inference algorithms for graphical models with higher order factors using OpenGM. In *Pattern Recognition, Proc. 32th DAGM Symposium* (2010).

[4] Andriluka, M., Roth, S., and Schiele, B. People-tracking-by-detection and people-detection-by-tracking. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on* (2008), pp. 1–8.

[5] Andriluka, M., Roth, S., and Schiele, B. Pictorial structures revisited: People detection and articulated pose estimation. In *CVPR* (2009), pp. 1014–1021.

[6] Balan, A., Black, M., Haussecker, H., and Sigal, L. Shining a light on human pose: On shadows, shading and the estimation of pose and shape. In *ICCV* (2007).

[7] Batra, D., Gallagher, A., Parikh, D., and Chen, T. Beyond trees: Mrf inference via outer-planar decomposition. In *Conference on Computer Vision and Pattern Recognition* (2010).

[8] Bayes, T. An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London 53* (1763).

[9] Becker, F. Matrix-valued filters as convex programs. Master's thesis, CVGPR group, University of Mannheim, 2004.

[10] Bellman, R. *Dynamic Programming*. Princeton University Press, Princeton, New Jersey, 1957.

[11] Bergtholdt, M., Kappes, J. H., Schmidt, S., and Schnörr, C. A study of parts-based object class detection using complete graphs. *Int. J. Comp. Vision 87*, 1-2 (2010), 93–117.

[12] Bergtholdt, M., Kappes, J. H., and Schnörr, C. Learning of graphical models and efficient inference for object class recognition. In *Ann. Symp. German Assoc. for Patt. Recog.* (September 2006).

[13] Bertsekas, D. *Nonlinear Programming*, 2nd ed. Athena Scientific, Belmont, Mass., 1999.

[14] Bertsimas, D., and Tsitsiklis, J. *Introduction to Linear Optimization.* Athena Scientific, 1997.

[15] Bethe, H. A. Statistical theory of superlattices. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences 150*, 871 (1935), 552–575.

[16] Borenstein, E., and Ullman, S. Combined top-down/bottom-up segmentation. *IEEE Trans. Pattern Anal. Mach. Intell. 30*, 12 (2008), 2109–2125.

[17] Bourdev, L., and Brandt, J. Robust object detection via soft cascade. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 236–243.

[18] Boyd, S., and Vandenberghe, L. *Convex Optimization.* Cambridge University Press, New York, NY, USA, 2004.

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

[20] Boykov, Y., Veksler, O., and Zabih, R. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell. 23*, 11 (2001), 1222–1239.

[21] Bray, M., Kohli, P., and Torr, P. Posecut: Simultaneous segmentation and 3d pose estimation of humans using dynamic graph-cuts. In *ECCV* (2006), pp. 642–655.

[22] Brogefors, G. Hierarchical chamfer matching: A parametric edge matching algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence 10* (1988), 849–865.

[23] Brown, L. D. *Fundamentals of statistical exponential families: with applications in statistical decision theory.* Institute of Mathematical Statistics, Hayworth, CA, USA, 1986.

[24] Butnariu, D., and Resmerita, E. Averaged subgradient methods for constrained convex optimization and nash equilibria computation. *Optimization 51* (2002), 863–888.

[25] Carreira, J., and Sminchisescu, C. Constrained Parametric Min-Cuts for Automatic Object Segmentation. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR* (June 2010). description of our winning PASCAL VOC 2009 segmentation entry.

[26] Castillo, E., Gutiérrez, J. M., and Hadi, A. S. *Expert systems and probabilistic network models.* Springer-Verlag, 1997.

[27] Cheng, S. Y., and Trivedi, M. M. Articulated human body pose inference from voxel data using a kinematically constrained gaussian mixture model, 2006. In CVPR EHuM2: 2-nd Workshop on Evaluation of Articulated Human Motion and Pose Estimation.

[28] Cohen, J. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement 20* (1960), 37–46.

[29] Cooper, G. F. The computational complexity of probabilistic inference using bayesian belief networks. *Artif. Intell. 42*, 2-3 (1990), 393–405.

[30] Cormen, T. H., Stein, C., Rivest, R. L., and Leiserson, C. E. *Introduction to Algorithms*. The MIT Press, 2009.

[31] Coughlan, J., and Shen, H. Shape matching with belief propagation: Using dynamic quantization to accomodate occlusion and clutter. In *CVPR Workshop* (Washington, DC, USA, 2004), IEEE Computer Society, p. 180.

[32] Coughlan, J., and Yuille, A. Bayesian A^* tree search with expected O(N) node expansions: applications to road tracking. *Neural Computation 14*, 8 (2002), 1929–1958.

[33] Coughlan, J. M., and Ferreira, S. J. Finding deformable shapes using loopy belief propagation. In *ECCV* (London, UK, 2002), Springer-Verlag, pp. 453–468.

[34] Cowell, R. G., Dawid, A. P., Lauritzen, S. L., and Spiegelhalter, D. J. *Probabilistic Networks and Expert Systems: Exact Computational Methods for Bayesian Networks*. Springer Publishing Company, Incorporated, 2007.

[35] Dakin, R. J. A tree-search algorithm for mixed integer programming problems. *Computer Journal 8* (1965), 250–255.

[36] Dalal, N., and Triggs, B. Histograms of oriented gradients for human detection. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 886–893.

[37] Dance, C., Willamowski, J., Fan, L., Bray, C., and Csurka, G. Visual categorization with bags of keypoints. In *ECCV International Workshop on Statistical Learning in Computer Vision* (2004).

[38] Elad, M., Hel-Or, Y., and Keshet, R. Pattern detection using a maximal rejection classifier. *Pattern Recognition Letters 23* (2001), 1459–1471.

[39] Elias, P., Feinstein, A., and Shannon, C. E. A note on the maximum flow through a network. *IRE Transactions on Information Theory (later IEEE Transactions on Information Theory) IT-2* (December 1956), 117 – 199.

[40] Elidan, G. Residual belief propagation: Informed scheduling for asynchronous message passing. In *Twenty-second Conference on Uncertainty in AI (UAI* (2006).

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

[42] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. The PASCAL Visual Object Classes Challenge 2009 (VOC2009) Results. http://www.pascal-network.org/challenges/VOC/voc2009/workshop/index.html.

[43] Everingham, M., Zisserman, A., Williams, C. K. I., and Van Gool, L. The PASCAL Visual Object Classes Challenge 2006 (VOC2006) Results. http://www.pascal-network.org/challenges/VOC/voc2006/results.pdf.

[44] Fawcett, T. ROC graphs: Notes and practical considerations for researchers, Mar. 16 2004.

[45] Felzenszwalb, P., Girshick, R., and McAllester, D. Cascade object detection with deformable part models. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)* (San Fransisco, California, USA, 2010).

[46] Felzenszwalb, P., and Huttenlocher, D. Pictorial structures for object recognition. *IJCV 61*, 1 (2005), 55–79.

[47] Fergus, R., Perona, P., and Zisserman, A. Object class recognition by unsupervised scale-invariant learning. In *CVPR* (June 2003), vol. 2, pp. 264–271.

[48] Fergus, R., Perona, P., and Zisserman, A. A sparse object category model for efficient learning and exhaustive recognition. In *CVPR* (2005).

[49] Fergus, R., Perona, P., and Zisserman, A. Weakly supervised scale-invariant learning of models for visual recognition. *IJCV 71*, 3 (2007), 273–303.

[50] Fergus, R., Weber, M., and Perona, P. Efficient methods for object recognition using the constellation model. Tech. rep., California Institute of Technology, 2001.

[51] Ferrari, V., Marin-Jimenez, M., and Zisserman, A. Progressive search space reduction for human pose estimation. In *ICCV* (June 2008).

[52] Fischler, M. A., and Elschlager, R. A. The representation and matching of pictorial structures. *IEEE Tr. Computers 22*, 1 (January 1973), 67–92.

[53] Fleuret, F., and Geman, D. Coarse-to-fine face detection. *Int. J. Comput. Vision 41*, 1-2 (2001), 85–107.

[54] Ford, L. R., and Fulkerson, D. R. Maximal flow through a network. *Canadian Journal of Mathematics 8* (1956), 399 – 404.

[55] Ford, L. R., and Fulkerson, D. R. *Flows in Network*. Princeton University Press, Princeton, 1962.

[56] Frey, B., and Jojic, N. A comparison of algorithms for inference and learning in probabilistic graphical models. *IEEE PAMI 27*, 9 (2005), 1392–1416.

[57] Gangaputra, S., and Geman, D. A design principle for coarse-to-fine classification. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (Washington, DC, USA, 2006), IEEE Computer Society, pp. 1877–1884.

[58] Geurts, P., Ernst, D., and Wehenkel, L. Extremely randomized trees. *Mach. Learn. 36*, 1 (2006), 3–42.

[59] Gibbs, J. W. *Elemantary principles of statsistical mechanics*. Yale University Press, 1902.

[60] Goldberg, A. V., and Tarjan, R. E. A new approach to the maximum flow problem. In *STOC '86: Proceedings of the eighteenth annual ACM symposium on Theory of computing* (New York, NY, USA, 1986), ACM, pp. 136–146.

[61] Gonfaus, J. M., Boix, X., van de Weijer, J., Bagdanov, A. D., Serrat, J., and Gonzàlez, J. Harmony potentials for joint classification and segmentation. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)* (San Fransisco, California, USA, 2010), pp. 1–8.

[62] Greig, D. M., Porteous, B. T., and Seheult, A. H. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society. Series B (Methodological)* (1989), 271–279.

[63] Grötschel, M., Lovász, L., and Schrijver, A. *Geometric Algorithms and Combinatorial Optimization*, second corrected edition ed., vol. 2 of *Algorithms and Combinatorics*. Springer, 1993.

[64] Guignard, M. Lagrangean relaxation. *TOP: Sociedad de Estatistica e Investigacion Operational 1*, 2 (2003), 151–228.

[65] Guignard, M., and Kim, S. Lagrangean decomposition: A model yielding stronger lagrangean bounds. *Math. Program. 39*, 2 (1987), 215–228.

[66] Gupta, A., Mittal, A., and Davis, L. S. Constraint integration for efficient multiview pose estimation with self-occlusions. *IEEE PAMI 30*, 3 (2008), 493–506.

[67] Hammer, P. L., Hansen, P., and Simeone, B. Roof duality, complementation and persistency in quadratic 0-1 optimization. *Mathematical Programming 28* (1984), 121–155.

[68] Hammersley, J., and Clifford, P. E. Markov fields on finite graphs and lattices. Unpublished manuscript, 1971.

[69] Hart, P. E., Nilsson, N. J., and Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Tr. Syst. Sci. Cybernetics 4* (1968), 100–107.

[70] Hartley, R. I. Estimation of relative camera positions for uncalibrated cameras. In *Lect. Notes Comp. Sci.* (1992), vol. 588, ECCV, Springer-Verlag, pp. 589–587.

[71] Hastie, T., Tibshirani, R., and Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*, 2nd ed. 2009. corr. 3rd printing ed. Springer Series in Statistics. Springer, 2010.

[72] Heskes, T. On the uniqueness of loopy belief propagation fixed points. *Neural Comput. 16*, 11 (2004), 2379–2413.

[73] Heskes, T. Convexity arguments for efficient minimization of the bethe and kikuchi free energies. *J. Artif. Int. Res. 26*, 1 (2006), 153–190.

[74] Heskes, T., Albers, K., and Kappen, B. Approximate inference and constrained optimization. In *In Uncertainty in Artificial Intelligence* (2003), Morgan Kaufmann Publishers, pp. 313–320.

[75] Hinton, G. E. Training products of experts by minimizing contrastive divergence. *Neural Computation 14*, 8 (2002), 1771–1800.

[76] Howe, N. R. Recognition-based motion capture and the humaneva ii test data, 2007. In CVPR EHuM2: 2-nd Workshop on Evaluation of Articulated Human Motion and Pose Estimation.

[77] Huffman, W. C., and Pless, V. *Fundamentals of error-correcting codes*. Cambridge Univ. Press, 2003.

[78] Ihler, A. T., Fischer III, J. W., and Willsky, A. S. Loopy belief propagation: Convergence and effects of message errors. *J. Mach. Learn. Res. 6* (2005), 905–936.

[79] Jiang, H., and Martin, D. R. Global pose estimation using non-tree models. In *CVPR* (2008), pp. 1–8.

[80] Johnson, J. K., Malioutov, D., and Willsky, A. S. Lagrangian relaxation for MAP estimation in graphical models. In *45th Annual Allerton Conference on Communication, Control and Computing* (September 2007).

[81] Jolliffe, I. T. *Principal Component Analysis*, second ed. Springer, October 2002.

[82] Kappes, J. H., Schmidt, S., and Schnörr, C. MRF inference by k-fan decomposition and tight lagrangian relaxation. In *European Conference on Computer Vision (ECCV)* (2010).

[83] Kappes, J. H., and Schnörr, C. MAP-inference for highly-connected graphs with DC-programming. In *Pattern Recognition – 30th DAGM Symposium* (2008), vol. 5096 of *lncs*, Springer Verlag.

[84] Karim, R., Bergtholdt, M., Kappes, J. H., and Schnörr, C. Greedy-based design of sparse two-stage SVMs for fast classification. In *Pattern Recognition – 29th DAGM Symposium* (2007), vol. 4713 of *LCNS*, Springer, pp. 395–404.

[85] Kohli, P., Kumar, M. P., and Torr, P. H. S. P3 & beyond: Move making algorithms for solving higher order functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence 31*, 9 (2009), 1645–1656.

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

[87] Kolmogorov, V. Convergent tree-reweighted message passing for energy minimization. *IEEE Trans. Patt. Anal. Mach. Intell. 28*, 10 (2006), 1568–1583.

[88] Kolmogorov, V. A note on the primal-dual method for the semi-metric labeling problem, June 2007.

[89] Kolmogorov, V., and Rother, C. Minimizing nonsubmodular functions with graph cuts-a review. *IEEE Trans. Pattern Anal. Mach. Intell. 29*, 7 (2007), 1274–1279.

[90] Kolmogorov, V., and Wainwright, M. On the optimality of tree-reweighted max-product message passing. In *21st Conference on Uncertainty in artificial Intelligence* (2005), pp. 316–322.

[91] Kolmogorov, V., and Zabin, R. What energy functions can be minimized via graph cuts? *Pattern Analysis and Machine Intelligence, IEEE Transactions on 26*, 2 (2004), 147–159.

[92] Komodakis, N., and Paragios, N. Beyond loose LP-relaxations: Optimizing MRFs by repairing cycles. In *ECCV08* (2008), pp. III: 806–820.

[93] Komodakis, N., and Paragios, N. Beyond Pairwise Energies: Efficient Optimization for Higher-order MRFs. In *CVPR* (June 2009).

[94] Komodakis, N., Paragios, N., and Tziritas, G. MRF optimization via dual decomposition: Message-passing revisited. In *IEEE 11th International Conference on Computer Vision, ICCV 2007* (2007), IEEE, pp. 1–8.

[95] Komodakis, N., and Tziritas, G. Approximate labeling via graph cuts based on linear programming. *IEEE Trans. Pattern Anal. Mach. Intell. 29*, 8 (2007), 1436–1453.

[96] Komodakis, N., Tziritas, G., and Paragios, N. Fast, approximately optimal solutions for single and dynamic MRFs. In *IEEE Computer Society Conference on*

Computer Vision and Pattern Recognition (CVPR 2007) (Minneapolis, Minnesota, USA, 2007), IEEE Computer Society.

[97] Kschischang, F., Frey, B. J., and Loeliger, H.-A. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory 47* (2001), 498–519.

[98] Kulesza, A., and Pereira, F. Structured learning with approximate inference. In *Advances in Neural Information Processing Systems 20*, J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds. MIT Press, Cambridge, MA, 2008, pp. 785–792.

[99] Kullback, S., and Leibler, R. A. On information and sufficiency. *The Annals of Mathematical Statistics 22*, 1 (1951), 79–86.

[100] Kumar, M. P., Kolmogorov, V., and Torr, P. H. S. An analysis of convex relaxations for MAP estimation. In *Proceedings of Advances in Neural Information Processing Systems* (2007).

[101] Kumar, M. P., Torr, P. H. S., and Zisserman, A. Solving markov random fields using second order cone programming relaxations. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (Washington, DC, USA, 2006), IEEE Computer Society, pp. 1045–1052.

[102] Kumar, M. P., Zisserman, A., and Torr, P. H. Efficient discriminative learning of parts-based models. In *Proceedings of the IEEE International Conference on Computer Vision* (2009).

[103] Kumar, S., and Hebert, M. Discriminative random fields. *IJCV 68*, 2 (2006), 179–201.

[104] Lafferty, J. D., McCallum, A., and Pereira, F. C. N. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning* (San Francisco, CA, USA, 2001), Morgan Kaufmann Publishers Inc., pp. 282–289.

[105] Land, A. H., and Doig, A. G. An automatic method of solving discrete programming problems. *Econometrica 28*, 3 (1960), 497–520.

[106] Lauritzen, S. L. *Graphical Models*. Oxford University Press, 1996.

[107] Lauritzen, S. L., and Sheehan, N. A. Graphical models for genetic analyses. *Statistical Science 18* (2003), 489–514.

[108] Lee, M.-W., and Cohen, I. A model-based approach for estimating human 3D poses in static images. *IEEE PAMI 28*, 6 (2006), 905–916.

[109] Leibe, B., Seemann, E., and Schiele, B. Pedestrian detection in crowded scenes. In *In CVPR* (2005), pp. 878–885.

[110] Lepetit, V., and Fua, P. Keypoint recognition using randomized trees. *IEEE PAMI 28*, 9 (2006), 1465–1479.

[111] Levin, A., and Weiss, Y. Learning to combine bottom-up and top-down segmentation. In *ECCV* (2006), pp. IV: 581–594.

[112] Loeliger, H. A. An introduction to factor graphs. *Signal Processing Magazine, IEEE 21*, 1 (2004), 28–41.

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

[114] McEliece, R. J., Mackay, D. J. C., and Cheng, J.-f. Turbo decoding as an instance of pearls belief propagation algorithm. *IEEE Journal on Selected Areas in Communications 16* (1998), 140–152.

[115] McEliece, R. J., and Yildirimt, M. Belief propagation on partially ordered sets. In *In Mathematical Theory of Systems and Networks* (2002), pp. 275–300.

[116] Mikolajczyk, K., Leibe, B., and Schiele, B. Multiple object class detection with a generative model. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (Washington, DC, USA, 2006), IEEE Computer Society, pp. 26–36.

[117] Mikolajczyk, K., Schmid, C., and Zisserman, A. Human detection based on a probabilistic assembly of robust part detectors. In *ECCV* (May 2004), Springer.

[118] Mooij, J. M. Sufficient conditions for convergence of loopy belief propagation. In *in UAI* (2005), AUAI Press, pp. 396–403.

[119] Murphy, K. P., Weiss, Y., and Jordan, M. I. Loopy belief propagation for approximate inference: An empirical study. In *In Proceedings of Uncertainty in AI* (1999), pp. 467–475.

[120] Orlin, J. B. A faster strongly polynomial time algorithm for submodular function minimization. In *IPCO '07: Proceedings of the 12th international conference on Integer Programming and Combinatorial Optimization* (Berlin, Heidelberg, 2007), Springer-Verlag, pp. 240–251.

[121] Parnas, M., Ron, D., and Rubinfeld, R. On testing convexity and submodularity. *SIAM J. Comput. 32*, 5 (2003), 1158–1184.

[122] Pearl, J. *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Addison-Wesley, 1984.

[123] Pearl, J. *Probabilistic reasoning in intelligent systems: networks of plausible inference.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[124] Pearl, J. *Causality: Models, Reasoning, and Inference.* Cambridge University Press, March 2000.

[125] Pearl, J., and Paz, A. Graphoids: A graph-based logic for reasning about relevancy relations. In *Advances in Artificial Intelligence-II* (1987), B. D. Boulay, Ed., NORTHHOLLAND.

[126] Pham, T., and Smeulders, A. Object recognition with uncertain geometry and uncertain part detection. *CVIU 99*, 2 (August 2005), 241–258.

[127] Polyak, B. A general method for solving extremum problems. *Soviet Math. 8* (1966), 593–597.

[128] Pretti, M. A message-passing algorithm with damping. *Journal of Statistical Mechanics: Theory and Experiment 2005*, 11 (2005), P11008.

[129] Quattoni, A., Collins, M., and Darrell, T. Conditional random fields for object recognition. In *NIPS* (2004).

[130] Ramalingam, S., Kohli, P., Alahari, K., and Torr, P. Exact inference in multi-label crfs with higher order cliques. In *CVPR* (2008), pp. 1–8.

[131] Ramanan, D., Forsyth, D.-A., and Zisserman, A. Tracking people by learning their appearance. *IEEE PAMI 29*, 1 (2007), 65–81.

[132] Ravikumar, P., and Lafferty, J. Quadratic programming relaxations for metric labeling and markov random field MAP estimation. In *ICML '06: Proceedings of the 23rd international conference on Machine learning* (New York, NY, USA, 2006), ACM Press, pp. 737–744.

[133] Ren, X., Berg, A. C., and Malik, J. Recovering human body configurations using pairwise constraints between parts. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 824–831.

[134] Rockafellar, R. T. *Convex Analysis*. Princeton University Press, 1970.

[135] Roosta, T., Wainwright, M. J., and Sastry, S. S. Convergence analysis of reweighted sum-product algorithms. *IEEE Transactions on Signal Processing 56*, 9 (2008), 4293–4305.

[136] Rosenhahn, B., Brox, T., and Weickert, J. Three-dimensional shape knowledge for joint image segmentation and pose tracking. *IJCV 73*, 3 (2007), 243–262.

[137] Rother, C., Kolmogorov, V., Lempitsky, V. S., and Szummer, M. Optimizing binary MRFs via extended roof duality. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)* (Minneapolis, Minnesota, USA, 2007), IEEE Computer Society.

[138] Russell, S. J., and Norvig, P. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.

[139] Ruszczynski, A. A merit function approach to the subgradient method with averaging. *Optimization Methods Software 23*, 1 (2008), 161–172.

[140] Schellewald, C., and Schnörr, C. Subgraph matching with semidefinite programming. In *Electronic Notes in Discrete Mathematics* (2003), V. D. G. Alberto Del Lungo and A. Kuba, Eds., vol. 12, Elsevier Science Publishers.

[141] Schlesinger, D. Exact solution of permuted submodular minsum problems. In *Energy Minimization Methods in Computer Vision and Pattern Recognition, 6th International Conference, EMMCVPR 2007* (2007), A. L. Yuille, S. C. Zhu, D. Cremers, and Y. Wang, Eds., vol. 4679 of *Lecture Notes in Computer Science*, Springer, pp. 28–38.

[142] Schlesinger, D. General search algorithms for energy minimization problems. In *Energy Minimization Methods in Computer Vision and Pattern Recognition* (2009), D. Cremers, Y. Boykov, A. Blake, and F. R. Schmidt, Eds., vol. 5681 of *Lecture Notes in Computer Science*, Springer, pp. 84–97.

[143] Schmidt, S., Kappes, J. H., Bergtholdt, M., Pekar, V., Dries, S., Bystrov, D., and Schnörr, C. Spine detection and labeling using a parts-based graphical model. In *Information Processing in Medical Imaging* (July 2007), N. Karssemeijer and B. Lelieveldt, Eds., no. 4584 in Lect. Notes Comp. Sci., Springer, pp. 122–133.

[144] Schrijver, A. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer-Verlag, 2003.

[145] Seemann, E., Leibe, B., Mikolajczyk, K., and Schiele, B. An evaluation of local shape-based features for pedestrian detection. In *In Proc. BMVC* (2005).

[146] Shimony, S. E. Finding MAPs for belief networks is NP-hard. *Artificial Intelligence 68*, 2 (1994), 399 – 410.

[147] Shor, N. Z., Kiwiel, K. C., and Ruszcayǹski, A. *Minimization methods for non-differentiable functions*. Springer-Verlag New York, Inc., New York, NY, USA, 1985.

[148] Sigal, L., and Black, M. Measure locally, reason globally: Occlusion-sensitive articulated pose estimation. In *CVPR* (2006), vol. 2.

[149] Sigal, L., and Black, M. J. HumanEva: Synchronized video and motion capture dataset for evaluation of articulated human motion. Tech. Rep. CS-06-08, Brown University, Department of Computer Science, Providence, RI, September 2006.

[150] Sivic, J., Russell, B., Efros, A., Zisserman, A., and Freeman, W. Discovering objects and their locations in images. In *ICCV*. IEEE, 2005.

[151] Sminchisescu, C., Kanaujia, A., and Metaxas, D. Bm^{3}e: Discriminative density propagation for visual tracking. *IEEE PAMI 29*, 11 (2007), 2030–2044.

[152] Sochman, J., and Matas, J. Waldboost " learning for time constrained sequential detection. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 150–156.

[153] Solodov, M. V., and Zavries, S. K. Error stability properties of generalized gradient-type algorithms. *J. Optim. Theory Appl. 98*, 3 (1998), 663–680.

[154] Sontag, D., Meltzer, T., Globerson, A., Jaakkola, T., and Weiss, Y. Tightening LP relaxations for MAP using message passing. In *UAI* (2008), D. A. McAllester and P. Myllymäki, Eds., AUAI Press, pp. 503–510.

[155] Strandmark, P., and Kahl, F. Parallel and distributed graph cuts by dual decomposition. In *Conference on Computer Vision and Pattern Recognition* (2010).

[156] Sudderth, E., Ihler, A., Freeman, W., and Willsky, A. Nonparametric belief propagation. In *CVPR* (2003).

[157] Sutton, C., and McCallum, A. Improved dynamic schedules for belief propagation. In *Conference on Uncertainty in Artificial Intelligence (UAI)* (2007).

[158] Sutton, C., McCallum, A., and Rohanimanesh, K. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *J. Mach. Learn. Res. 8* (March 2007), 693–723.

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

[160] Tian, T.-P., and Sclaroff, S. Fast Globally Optimal 2D Human Detection with Loopy Graph Models. In *CVPR* (2010).

[161] Tseng, P. An incremental gradient(-projection) method with momentum term and adaptive stepsize rule. *SIAM J. on Optimization 8*, 2 (1998), 506–531.

[162] Vedaldi, A., Gulshan, V., Varma, M., and Zisserman, A. Multiple kernels for object detection. In *Proceedings of the International Conference on Computer Vision (ICCV)* (2009).

[163] Veksler, O. Star shape prior for graph-cut image segmentation. In *ECCV* (2008), pp. III: 454–467.

[164] Viola, P., and Jones, M. Rapid object detection using a boosted cascade of simple features. *Proc. CVPR 1* (2001), 511–518.

[165] Viola, P., Jones, M. J., and Snow, D. Detecting pedestrians using patterns of motion and appearance. *Int. J. Comput. Vision 63*, 2 (2005), 153–161.

[166] Wainwright, M. Estimating the wrong markov random field: Benefits in the computation-limited setting. In *Adv. in Neur. Inf. Proc. Sys.*, Y. Weiss, B. Schölkopf, and J. Platt, Eds. MIT Press, Cambridge, MA, 2006, pp. 1425–1432.

[167] Wainwright, M., Jaakkola, T., and Willsky, A. Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE Trans. Inform. Theory 49*, 5 (2003), 1120–1146.

[168] Wainwright, M. J. Estimating the "wrong" graphical model: Benefits in the computation-limited setting. *Journal of Machine Learning Research 7* (2006), 1829–1859.

[169] Wainwright, M. J., Jaakkola, T., and Willsky, A. S. MAP estimation via agreement on trees: message-passing and linear programming. *IEEE Transactions on Information Theory 51*, 11 (2005), 3697–3717.

[170] Wainwright, M. J., Jaakkola, T., and Willsky, A. S. A new class of upper bounds on the log partition function. *IEEE Transactions on Information Theory 51*, 7 (2005), 2313–2335.

[171] Wainwright, M. J., and Jordan, M. I. Semidefinite relaxations for approximate inference on graphs with cycles, 2003.

[172] Wainwright, M. J., and Jordan, M. I. Variational inference in graphical models: The view from the marginal polytope. Forty-first Annual Allerton Conference on Communication, Control, and Computing, Urbana-Champaign, IL, 2004., 2003.

[173] Wainwright, M. J., and Jordan, M. I. *Graphical Models, Exponential Families, and Variational Inference*. Now Publishers Inc., Hanover, MA, USA, 2008.

[174] Wang, Y., and Mori, G. Multiple tree models for occlusion and spatial constraints in human pose estimation. In *ECCV '08: Proceedings of the 10th European Conference on Computer Vision* (Berlin, Heidelberg, 2008), Springer-Verlag, pp. 710–724.

[175] Weber, M., Welling, M., and Perona, P. Unsupervised learning of models for recognition. In *ECCV* (2000), pp. 18–32.

[176] Weiss, Y., and Freeman, W. On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. *IEEE Tr. Information Theory 47* (2001).

[177] Welk, M., Weickert, J., Becker, F., Schnörr, C., Feddern, C., and Burgeth, B. Median and related local filters for tensor-valued images. *Signal Process. 87*, 2 (2007), 291–308.

[178] Welling, M., and Teh, Y. W. Belief optimization for binary networks: a stable alternative to loopy belief propagation. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence* (2001), vol. 17.

[179] Werner, T. A linear programming approach to max-sum problem: A review. *IEEE Trans. Patt. Anal. Mach. Intell. 29*, 7 (2007), 1165–1179.

[180] Werner, T. High-arity interactions, polyhedral relaxations, and cutting plane algorithm for soft constraint optimisation (map-mrf). In *CVPR* (2008), pp. 1–8.

[181] Werner, T. Revisiting the decomposition approach to inference in exponential families and graphical models. Tech. Rep. Research report CTU-CMP-2009-06, Center for Machine Perception, Czech Technical University, May 2009.

[182] Werner, T. Fixed points of loopy belief propagation as zero gradients of a function of reparameterizations. Tech. Rep. Research report CTU-CMP-2010-05, Center for Machine Perception, Czech Technical University, February 2010.

[183] Werner, T. Primal view on belief propagation. In *Conf. on Uncertainty in Artificial Intelligence (UAI)* (2010).

[184] Yang, Y., Hallman, S., Ramanan, D., and C., F. Layered object detection for multi-class segmentation. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)* (San Fransisco, California, USA, 2010).

[185] Yanover, C., Meltzer, T., Weiss, Y., Bennett, P., and Parrado-Hernandez, E. Linear programming relaxations and belief propagation – an empirical study. *Journal of Machine Learning Research 7* (2006), 2006.

[186] Yedidia, J. S., Freeman, W. T., and Weiss, Y. Generalized belief propagation. In *IN NIPS 13* (2000), MIT Press, pp. 689–695.

[187] Yedidia, J. S., Freeman, W. T., and Weiss, Y. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory 51*, 7 (2005), 2282–2312.

[188] Yuille, A. L. CCCP algorithms to minimize the bethe and kikuchi free energies: convergent alternatives to belief propagation. *Neural Comput. 14*, 7 (2002), 1691–1722.