Bottleneck potentials in Markov Random Fields

Ahmed Abbas

Abstract

We consider general discrete Markov Random Fields (MRFs) with additional bottleneck potentials which penalize the maximum (instead of the sum) over local potential value taken by the MRF-assignment. Bottleneck potentials or analogous constructions have been considered in (i) combinatorial optimization (e.g. bottleneck shortest path problem, the minimum bottleneck spanning tree problem, bottleneck function minimization in greedoids), (ii) inverse problems with $L^\infty$-norm regularization, and (iii) valued constraint satisfaction on the (min, max)-pre-semirings. Bottleneck potentials for general discrete MRFs are a natural generalization of the above direction of modeling work to Maximum-A-Posteriori (MAP) inference in MRFs. To this end, we propose MRFs whose objective consists of two parts: terms that factorize according to (i) (min, +), i.e. potentials as in plain MRFs, and (ii) (min, max), i.e. bottleneck potentials. To solve the ensuing inference problem, we propose high-quality relaxations and efficient algorithms for solving them. We empirically show efficacy of our approach on large scale seismic horizon tracking problems.

1. Introduction

In the field of computer vision MRFs have found many applications such as image segmentation, denoising, optical flow, 3D reconstruction and many more, see [16] for a non-exhaustive overview of problems and algorithms. The above application scenarios are modelled well as a sum of unary/pairwise/ternary/... potentials on the underlying graphical structure. More generally, whenever the error of fitting a model to data is captured by local terms, the objective can factorize into a sum of local error terms. However, such objectives are not always appropriate. In some computer vision applications, a single error can entail subsequent errors, rendering the solution useless and local terms are unable to properly penalize this. Prominent examples are tracking problems [41, 13, 25, 10], where making a single error and following a wrong track results in low accuracy nonetheless. In inverse problems, $L^\infty$-norm regularization penalizes the maximum deviation from the fitted model and is appropriate e.g. for some types of group sparsity [24, 22, 23]. In all of the above scenarios global potentials, which penalize the maximum value assignment w.r.t. a given set of local costs are the appropriate choice. We call this maximum value a bottleneck and the aim is to find a configuration such that its bottleneck has minimum cost. Formally, optimizing a bottleneck objective can be written as

$$\min_{x \in X} \left( \max_{i} \{ \psi_i \cdot x_i \} \right)$$

(1)

where $X \subset \mathbb{R}^n$ is the space of feasible elements of the optimization problem and $\psi \in \mathbb{R}^n$ is a real valued vector. Additionally, the bottleneck objective can be written as the infinity norm $\| \psi \odot x \|_\infty$, where $\odot$ is the Hadamard product.

2. Related work

Bottleneck-type objectives occur throughout many subfields of mathematical programming. The optimization problems are also often called min-max problems. Bottleneck potentials in MRFs. The case of pairwise binary MRFs with bottleneck potentials has been addressed in [29] and has been applied to image segmentation. As noted by the authors of [29], incorporating a bottleneck term gives better segment boundaries and resolves ‘small cuts’ (or shrinking bias) of graph cut [6]. In [8] the authors interpret $L^p$-norm regularization for discrete labeling problems and for $p \in [1, \infty)$ as MAP-inference in MRFs and propose approximating the bottleneck potential corresponding to $L^\infty$ via a high value of $p$.

Furthermore, labeling problems containing only bottleneck objectives were considered in [34, 30, 11, 26], where [11, 26] devised algorithms for special cases of the pure min-max labeling problem, and [34, 30] devised message passing schemes with application in parallel machine scheduling.

In contrast to previous works [8, 29, 34, 30, 11, 26] we investigate the mixed problem where ordinary and bottleneck potentials are both present in the same inference problem. Unlike [29] we allow arbitrary MRFs. Also unlike the algorithm in [8], our algorithms are based on a linear programming relaxation resulting in a more rigorous approach.

Combinatorial bottleneck optimization problems. Many classical optimization problems have bottleneck counter-
parts, e.g. bottleneck shortest paths [15], bottleneck spanning trees [12] and bottleneck function optimization in gredoids [21]. The above works consider the case of bottleneck potential only. More similar to our work is the mixed case of linear and bottleneck costs, as investigated for the shortest and bottleneck path problem in [28].

**$L_\infty$-norm regularization in inverse problems.** For inverse problems $\min_{x \in \mathbb{R}^n} \|Ax - b\|_p$ with uniform noise, the appropriate choice of norm is $p = \infty$ [5][Chapter 7.1.1]. Extensions of this basic $L_\infty$-norm are used in [24] for multi-task learning and in [22] for the multi-task lasso. More generally, mixed norms with $L_\infty$-norms are not useful and specialized methods are needed. For an in-depth study of the $(\min, max)$-pre-semiring we refer to [9]. Our case however does not fit into (pre-)semiring based constraint satisfaction (since classical arc consistency algorithms as discussed in [38, 39] are not applicable and specialized methods are needed). For an in-depth study of the $(\min, max)$-pre-semiring we refer to [9]. Our case however does not fit into (pre-)semiring based constraint satisfaction (since classical arc consistency algorithms as discussed in [38, 39] are not applicable and specialized methods are needed).

**Applications in horizon tracking.** The seismic horizon tracking problem, i.e. identifying borders between layers of various types of rock beds, has been addressed in [41, 13, 40, 36] from the computational perspective. The authors in [41] use a greedy method inspired by the minimum spanning tree problem to that end. In [13] the authors propose to solve a shortest path problem to track seismic horizons along a 2-D sections of the original 3-D volume. In [40, 36] the authors set up linear equations to solve the horizon tracking problem. Most similar to our work is the minimum spanning tree approach of [31]. In contrast to [31], we consider a rigorously defined optimization problem for which we develop a principled LP-based approach instead of greedily selecting solutions. Conceptually, the 2-D shortest path method [13] is also similar to ours, however it cannot be extended to the 3-D setting we are interested in. Moreover, we allow for a more sophisticated objective function than [13]. Methods of [40, 36] do not use optimization at all but solve linear systems and require more user intervention.

**Contribution & organization.** Section 3 introduces bottleneck potentials and their non-linear generalizations for general discrete MRFs. We also consider the mixed problem of MAP-inference for a combination of ordinary MRF-costs w.r.t. the $(\min, +)$-semiring and bottleneck potentials. In Section 4 we derive special algorithms to solve the problem for chain graphs via a dynamic shortest path method, and for graphs without pairwise interactions via an efficient enumerative procedure. Combining these two special cases, we derive a high-quality relaxation for the case of general graphs. To solve this relaxation, we propose an efficient dual decomposition algorithm. In Section 5 we show empirically that our approach results in a scalable algorithm that gives improved accuracy on seismic horizon tracking as compared to MAP-inference in ordinary MRFs and a state-of-the-art heuristic.

All code and datasets will be made publicly available upon acceptance of the paper.

**3. MRFs with bottleneck potentials**

First, we will review the classical problem of Maximum-A-Posteriori (MAP) inference in Markov Random Fields (MRF). Second, we will introduce the bottleneck labeling problem which extends MAP-MRF by additional bottleneck term that penalize the maximum value of potentials taken in an assignment (as opposed to the sum for MRFs).

### 3.1. Markov Random Fields:

A graph will be a tuple $G = (V, E)$ with undirected edges $E \subset \binom{V}{2}$. To each node $i \in V$ a label set $X_i$ is associated. To each node $i \in V$ a unary potential $\theta_i : X_i \rightarrow \mathbb{R}$ is associated, and to each edge $ij \in E$ a pairwise potential $\theta_{ij} : X_i \times X_j \rightarrow \mathbb{R}$. We will call $X_V = \prod_{i \in V} X_i$ the label space and $x \in X$ a labeling. For subsets $U \subset V$ we define $X_U = \prod_{i \in U} X_i$, analogously we refer to labels $x_U \in X_U$. In particular, $x_i$ refers to a node labeling and $x_{ij} = (x_i, x_j)$ to an edge labeling. A tuple $(G, X, \theta)$ consisting of a graph, corresponding label space and potentials is called a Markov Random Field (MRFs). The problem

$$
\min_{x \in X} \theta(x), \quad \theta(x) := \sum_{i \in V} \theta_i(x_i) + \sum_{ij \in E} \theta_{ij}(x_{ij})
$$

is called the Maximum-A-Posteriori (MAP) inference problem in MRFs.

**Local Polytope Relaxation:** We use the over-complete representation to obtain a linear programming relaxation of the optimization problem (2). For $i \in V$ we associate the $k$-th label from $X_i$ with one-hot encoding $e_k = (0, \ldots, 0, \frac{1}{k}, 0, \ldots, 0)$ which is a unit vector of length $|X_i|$
with a 1 at the $k$-th location. In other words, we can write $X_i = \{e_1, e_2, \ldots, e_{|X_i|}\}$ $\forall i \in V$. Analogously, we can write $X_{ij} = \{e_1, \ldots, e_{|X_{ij}|}\}$ $\forall ij \in E$. To obtain a convex relaxation, we define unary marginals as $\mu_i \in \text{conv } X_i$, $i \in V$, and pairwise marginals as $\mu_{ij} \in \text{conv } X_{ij}$, $ij \in E$. We couple unary and pairwise marginals together to obtain the local polytope relaxation [37].

$$A = \left\{ \mu \right\} = \left\{ \begin{array}{l} \mu_i(x_i) = \sum_{x_j \in X_j} \mu_{ij}(x_{ij}), \forall ij \in E, x_i \in X_i \\ \mu_f \in \text{conv } X_f, \forall f \in V \cup E \end{array} \right\}$$

(3)

With the local polytope we can relax the problem of MAP-inference in MRFs (2) as:

$$\min_{\mu \in A} \sum_{i \in V} (\theta_i, \mu_i) + \sum_{ij \in E} (\theta_{ij}, \mu_{ij})$$

(4)

Note that the relaxation (4) subject to constraint (3) is tight for some graphs such as trees and for special families of cost functions including different forms of submodularity [19].

3.2. Bottleneck labeling problem:

Given an MRF, we associate to it a second set of potentials, which we call bottleneck potentials. As opposed to MRF, however, the corresponding assignment cost is not given by the sum of individual potential values but by their maximum. The goal of inference in a pure bottleneck labeling problems (i.e. with all zero MRF potentials) is thus to find a labeling such that the maximum bottleneck potential value taken by the labeling is minimal.

Definition 1 (Bottleneck labeling problem). Let an MRF be given. Additionally, let unary bottleneck potentials $\phi_i : X_i \rightarrow \mathbb{R}$ $\forall i \in V$ and pairwise bottleneck potentials $\phi_{ij} : X_i \times X_j \rightarrow \mathbb{R}$ $\forall ij \in E$ are also given. We call the set of all possible values taken by bottleneck potentials as bottleneck values, i.e.

$$B = \{\phi_f(x_f) : f \in V \cup E, x_f \in X_f\}.$$  

(5)

Let bottleneck costs $\zeta : B \rightarrow \mathbb{R}$ be given. The bottleneck labeling problem is defined as

$$\min_{x \in X, b \in B} \theta(x) + \zeta(b)$$

(6a)

s.t. $\phi_f(x_f) \leq b$ $\forall f \in V \cup E$  

(6b)

and $\theta(x)$ is defined in (2).

It is straightforward to adapt our work to the case of multiple bottlenecks and triplet/quadruplet/… potentials. However we focus on models with only a single bottleneck and pairwise potentials for simplicity.

A special case of the problem (6) was considered by [29] where the authors only allow binary labels, special form of MRF costs and $\zeta(b) = b$. Additionally, heuristics for extending to the multi-label case were given in [8]. Below we propose exact algorithms for multi-label chain graphs and an LP-relaxation for general graphs.

Example. In the special case when $\zeta(b) = b$ and MRF costs are zero i.e. $\theta = 0$ then the problem reduces to a pure bottleneck labeling problem as:

$$\min_{x \in X} \max_{f \in V \cup E} (\phi_f(x_f))$$

(7)

Note that if we are know the optimal bottleneck value $b^*$ in $B$, then the bottleneck labeling problem can be reduced to the MAP-inference problem in MRFs. This reduction can be done by setting the unary and pairwise MRF costs to $\infty$ for the labelings which have bottleneck potentials greater than the optimal value $b^*$. i.e.

$$\theta_e(x_e) := \infty$ \quad \forall e, x_e : \phi_e(x_e) > b^*$$

(8)

Then, the constraints (6b) will be automatically satisfied by a feasible solution of the MAP-inference problem.

4. Algorithms:

In this section, algorithms for solving the bottleneck labeling problem are proposed. We first present efficient and exact algorithms for edge-free graphs and chain graphs. Later, we will use these two algorithms for solving the problem on general graphs using dual-decomposition. The algorithms for edge-free graphs and chains are designed with the end goal of using them inside dual decomposition for general graphs and therefore contain extra steps.

4.1. Bottleneck labeling with unary potentials:

Assume that the graphical model does not contain any edges i.e. $E = \emptyset$, so only unary potentials need to be considered. The problem (6) in this case can be efficiently solved with Algorithm unary_bottleneck.

Algorithm unary_bottleneck enumerates all bottleneck values in ascending order. For every bottleneck value $b$, an optimal node labeling can be found by choosing for each node $i$ the best label w.r.t. potentials $\theta_i$ that is feasible to bottleneck constraints (6b). The costs of such labelings are stored in $M$. Updating the best node labeling between consecutive bottleneck values can be done by only checking the nodes for which (6b) has changed the feasible set. Finally, the optimum bottleneck value can be computed by

$$(b^*, c^*) = \arg \min_{(b,c) \in M} [c + \zeta(b)]$$

(9)

Once the optimal bottleneck value $b^*$ is computed, the problem can be reduced to inference in MRF by disallowing the configurations which have bottleneck potentials greater than $b^*$ as mentioned in (8).
Algorithm \textbf{chain\_bottleneck} uses the constructed directed acyclic graph to solve the bottleneck labeling problem \eqref{eq:bottleneck} on chain MRF by an iterative procedure. For each bottleneck thresholds \( b \in B \), the shortest path is found such that only those edges are used whose bottleneck potentials are within the threshold. This way a matrix \( M \) of all possible pairs of bottleneck values \( b \) and shortest path costs \( c \) is computed. The optimal bottleneck value \( b^* \) can be then found by \eqref{eq:no_bottleneck}.

Naively implementing Algorithm \textbf{chain\_bottleneck} by computing a shortest path from scratch in each iteration of the loop will result in high complexity. Note that when iterating over bottleneck threshold values in ascending order, exactly one edge is added per iteration. Therefore, dynamic shortest path algorithms can be used for recomputing shortest paths more efficiently. Additionally, since the graph \( D \) is directed and acyclic, the shortest path in each iteration can be found in linear time by breadth-first traversal \cite{Fredman1984}. These improvements are detailed in Algorithm \textbf{dsp\_chain} in the Appendix.

**Proposition 2.** The worst case run-time of Algorithm \textbf{chain\_bottleneck} is \( \mathcal{O}(|A|^2) \) where, \( A \) is the arc-set in underlying graph \( D = (W, A) \).

While the worst-case runtime of \textbf{chain\_bottleneck} is quadratic in the number of edges of the underlying shortest path graph, the average case runtime is better, since not for every edge a new shortest path needs to be computed and often a shortest path computation can reuse previous results for speedup. Moreover, the sorting operation on line 3 can be re-used similar to Algorithm \textbf{unary\_bottleneck}.

**Remark 1.** If we have a pure bottleneck labeling problem without the MRF potentials \eqref{eq:no_bottleneck} then the problem can be solved in linear-time by sorting the bottleneck weights and model resp. with zero costs for modeling the shortest path problem. Algorithm \textbf{chain\_to\_dag} in the Appendix gives the general construction.

The shortest \( s,t \)-path \( P \) in graph \( D \) having cost \( \sigma(P) \) corresponds to an optimum labeling for \( \theta(x) \) from \eqref{eq:bottleneck} in the chain graphical model \( G \).
halving the number of edges by the median edge as mentioned in [15].

Remark 2. The chain bottleneck labeling problem can also be solved for linear bottleneck cost \( \zeta(x) = x \) by the method from [28]. However, the method from [28] is not polynomial w.r.t. bottleneck values \( B \) but requires them to be small integers to be efficient, making it unsuitable for our purposes.

Remark 3 (Bottleneck labeling on trees). The dynamic shortest path algorithm can be modified to work on trees as well, where we replace shortest path computations by belief propagation. For clarity of presentation we have chosen to restrict ourselves to chains.

4.3. Relaxation for the bottleneck labeling problem on arbitrary graphs:

Since MAP-MRF is NP-hard and the bottleneck labeling problem generalizes it, we cannot hope to obtain an efficient exact algorithm for the general case. As was done for MAP-MRF [37, 20, 31], we approach the general case with a Lagrangian decomposition into tractable subproblems. To this end, we decompose the underlying MRF problem into trees, which can be solved via dynamic programming. The bottleneck subproblem is decomposed into a number of bottleneck chain labeling problems. To account for the global bottleneck term, these bottleneck chain problems are connected through a unary bottleneck labeling problem defined on a higher level graph. We use Algorithms chain bottlenecks and unary bottleneck as subroutines to solve the bottleneck decomposition. An example of our decomposition can be seen in Figure 2.

To account for the MRF-inference problem, we cover the graph \( G \) by trees \( G_1 = (V_1, E_1), \ldots, G_k = (V_k, E_k) \). For the bottleneck labeling problem we cover the graph \( G \) by chains (trees) \( G_1 = (V_1, E_1), \ldots, G_k = (V_k, E_k) \). For the decomposition we introduce variables \( z' \) to specify the labeling of the MRF subproblem for graph \( G_i \), and variables \( y' \) for the chain bottleneck labeling subproblems defined for graph \( G_i \).

We propose the following overall decomposition:

\[
\min_{x, z', y'} \theta(x) + \zeta(b)
\]

s.t.

\[
x'_f = x_f \quad \forall f \in V_i \cup E_i, t \in [h] \quad (10b)
\]

\[
y'_f = y_f \quad \forall f \in V_i \cup E_i, l \in [k] \quad (10c)
\]

\[
\phi_f(y'_f) \leq b \quad \forall f \in V_i \cup E_i, l \in [k] \quad (10f)
\]

We constrain the variables for MRF tree subproblems and chain bottleneck subproblems to be consistent via (10b) and (10c). The labelings on chains \( y' \) are required to be feasible with respect to the bottleneck value \( b \) via (10f).

To obtain a tractable optimization problem, we dualize the constraints (10b) and (10c) using dual variables \( \lambda'_l \) and \( \eta'_l \) respectively. We keep rest of the constraints and denote the feasible variables \( y'_f \) for chain \( l \) w.r.t. constraints (10e), (10f) by the set \( Y'(l) \) as:

\[
Y'(l) = \{ y'_f \in \mathcal{X}_{V_i} \mid \phi_f(y'_f) \leq b, \forall f \in V_i \cup E_i \} \quad (11)
\]

The dual problem is:

\[
\max_{\lambda, \eta} \left[ \sum_{t \in [h]} E^t(\lambda'_t) + J(\eta) \right] \quad (12a)
\]

s.t.

\[
\sum_{t \in [h]} \lambda'_f + \sum_{f \in V_i \cup E_i} \eta'_f = \theta_f \quad \forall f \in V \cup E \quad (12b)
\]

Where \( E^t(\lambda') \) and \( J(\eta) \) are defined as:

\[
E^t(\lambda') := \min_{x' \in \mathcal{X}_{V_i}} \langle \lambda' : x' \rangle \quad (13a)
\]

\[
J(\eta) := \min_{b \in B} \left[ \zeta(b) + \sum_{l \in [k]} \min_{y' \in Y'(l)} \langle \eta' : y' \rangle \right] \quad (13b)
\]

Evaluating \( E^t(\cdot) \) amounts to solving a MAP-MRF problem on a tree. Evaluating \( J(\cdot) \) corresponds to computing for each bottleneck value \( b \) the corresponding minimal assignment from the set \( Y'(l) \) for all chains \( l \) in \([k] \), and then choosing a bottleneck value such that the sum over all chain subproblems is minimal.
The dual problem (12a) is a non-smooth concave maximization problem. Evaluating $E'(\cdot)$ and $J(\cdot)$ gives supergradients of (12a) which can be used in subgradient based solvers (e.g. subgradient descent, bundle methods).

**Finding** $E'(\lambda^*)$: Each MRF tree subproblem $E'(\lambda^*)$ can be solved independently using dynamic programming to get the optimal labeling $x^*$.  

**Finding** $J(\eta)$: $J(\eta)$ is solved by Algorithm `chain_decomp`. It proceeds as follows:

1. Build a higher level graph $H = ([k], \emptyset)$ and represent each chain in $[k]$ as a node in $H$. Populate the potentials $M_l$ for all nodes $l \in [k]$ by going through all $b \in B$ using Algorithm `unary_bottleneck` (lines 1-8).

2. Use Algorithm `unary_bottleneck` to find an optimal bottleneck value $b^*$ of $b \in B$ in graph $H$ (lines 9-10).

3. Find optimal labelings $\overline{\eta}$ for each chain subproblem by disallowing the configurations having bottleneck values more than $b^*$. (lines 11-18).

We optimize the Lagrange multipliers in the dual problem (12a) subject to constraints 12b with the proximal bundle method [32]. After solving the dual problem we have found a valid reparameterization of the MRF potentials (12b) which maximizes the lower bound. This lower bound can be used as a measure of the quality of the primal solution and helps in rounding a primal solution.

### 4.4. Primal rounding

For recovering the primal solution through rounding on MRF subproblems, we use the approach of [18] by treating the dual optimal values of $\lambda$ as MRF potentials. Additional details on how to best choose $\lambda$ and more details are given in Section 6.1 in the Appendix.

### 5. Experiments

We apply our proposed technique for tracking layers (horizons) in open source sub-surface volumetric data (a.k.a. seismic volumes). Accurate tracking of horizons is one of the most important problems in seismic interpretation for geophysical applications. See Fig. 3 for an illustration of a cross-section of a typical seismic volume.

Popular greedy approaches such as [41] rely on tracking horizons by establishing correspondences between nearby points that lie on the same horizon with high probability. As such they are prone to fall into local optima. On the other hand a natural option to track horizons is to
Table 1. Mean absolute deviation from the ground-truth for 11 horizons tracked using minimum spanning tree (MST) based heuristic [41], TRWS [18] solver for MRF, and our method with additional bottleneck terms on MRF (B-MRF). * marks the horizon surfaces which were used to train the CNN.

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>II*</th>
<th>III*</th>
<th>IV</th>
<th>V*</th>
<th>VI</th>
</tr>
</thead>
<tbody>
<tr>
<td>MST</td>
<td>1.6212</td>
<td>1.1245</td>
<td>1.6542</td>
<td>0.1283</td>
<td>0.1088</td>
<td>1.5319</td>
</tr>
<tr>
<td>MRF</td>
<td>7.1435</td>
<td>6.2105</td>
<td>3.2169</td>
<td>0.1540</td>
<td>0.0896</td>
<td>2.4498</td>
</tr>
<tr>
<td>B-MRF</td>
<td>0.8881</td>
<td>0.2644</td>
<td>0.0889</td>
<td>0.0463</td>
<td>0.0894</td>
<td>1.6049</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.4930</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.4596</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.0060</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.2759</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.9457</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.6108</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.5319</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.4786</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.9002</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.4073</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0221</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.4245</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.2388</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.8881</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.2644</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0889</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0463</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0894</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.6049</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.4887</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.4209</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.1855</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0116</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.4135</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.0377</td>
</tr>
</tbody>
</table>

We state the problem as MAP-inference in MRFs for which high-quality solvers exist that are less sensitive to local optimal. However, the cost structure of MRFs is not adequate for our problem. Specifically, the MRF energy is composed of a sum of local terms defined over nodes and edges of the graphical model that account for similarity between seed and adjacent nodes respectively. For the cost function it may be more advantageous to track a wrong horizon if it is more self-similar and pay a higher dissimilarity cost at a few locations rather than follow the correct horizon if it is harder to follow. This problem is exacerbated by the typically rather weak unary costs (see below). For an illustration of this behaviour we refer to Figure 3.

We argue that the bottleneck potential remedies this shortcoming of MAP-MRF, while preserving its advantages. In contrast to unary and pairwise potentials of MRFs, the bottleneck potential penalizes a single large discontinuity in the tracking that comes from jumping between layers more than multiple small discontinuities stemming from a hard to track horizon. As such it prefers to follow a rugged horizon with no big jump over a single large jump followed by an easy to track horizon.

Further computational challenges in horizon tracking are:

- Nearby rock layers look very similar to the layer in consideration, due to which tracking algorithms can easily jump to a wrong layer.
- Due to structural deformations in the subsurface environment, rock layers can have discontinuities.
- The appearance of a horizon can vary at different locations. Nearby horizon layers can bifurcate or disappear. This makes estimation of reliable cost terms for our model (6) difficult.
- The size of a seismic survey can be huge. Some open-source seismic volumes cover a 3000km² survey area. The corresponding input data has a size of 100GB.
- Lack of established open-source ground truth makes the evaluation procedure and learning of parameters difficult.

Seismic horizon tracking formulation. Our input are 3-D seismic volumes of size $N_1 \times N_2 \times D$, where $D$ corresponds to the depth axis and $N_1, N_2$ to x,y-axes resp. For each location $(x, y) \in [N_1] \times [N_2]$ in the volume we seek to assign a depth value $z \in [D]$. We formalize this as searching for a labeling function $z : [N_1] \times [N_2] \rightarrow [D]$.

For computational experiments we used publicly available seismic volumes [33]. Due to lack of available ground truth, we have selected three volumes (F3 Netherlands, Opunake-3D, Waka-3D) and tracked 11 horizons by hand with the commercial seismic interpretation soft-
ware [14]. Annotation took two days of work of an experienced seismic interpreter.

CNN-architecture for matching costs. In order to track horizons, established state-of-the-art tracking algorithms [41, 40, 13] rely on computing similarity scores between patches that might or might not lie on the same horizon. This is traditionally done using hand-crafted features that rely on cross-correlation or optical flow. We opt for a learning-based approach following the great success of CNN-based architectures for computing patch similarity [42, 35]. Specifically, we extend the architecture of the 2-D patch similarity CNN [42] to efficiently deal with the 3-D structure. We train two CNNs to compute matching probabilities between adjacent and non-adjacent nodes respectively:

\[ p_{ij}(z_{ij}) = \text{CNN}_{s|f}(\text{patch}_s(z_i), \text{patch}_f(z_j)) \in [0, 1] \]  
(14)

In (14) \( \text{patch}_k(z_k) \) is a two channel image of size 63 \times 63 centered around \((x_k, y_k, z_k)\).

Cost formulation. We rephrase computing a depth labeling as labeling nodes in a graph. Consider the grid graph \( G = (V, E) \) with nodes \( V = [N_1] \times [N_2] \) corresponding to points on \( x \) and \( y \) axes of the seismic volume. Edges \( E \) are given by a 4–neighborhood induced by the \((x, y)\)-grid. For each node \( v \in V \) the label space is \( X_v = [D] \). A depth labeling thus corresponds to a node labeling of \( V \). We compute unary \( \theta_v : [D] \rightarrow \mathbb{R}, v \in V \) and pairwise potentials \( \theta_{uv} : [D] \times [D] \rightarrow \mathbb{R}, uv \in E \) based on the patch matching costs computed in (14). For seed node \( s \), we set \( \theta_s(z_s) = \begin{cases} 0, & z_s = z_s^* \\ \infty, & \text{otherwise}. \end{cases} \)

The other unary potentials are computed as

\[ \theta_i(z_i) = -\log(\text{CNN}_f(\text{patch}_s(z_s^*), \text{patch}_f(z_i))) \]  
(15)

For the pairwise potentials we use additional terms. First, we compute an optical flow using the gradient structure ware [14]. Annotation took two days of work of an experienced seismic interpreter.

The authors from [41, 13] do not use the coherence estimate, making their cost functions less robust. The bottleneck pairwise potentials are:

\[ \phi_{ij}(z_{ij}) = |E| \theta_{ij}(z_{ij}) \]  
(17)

The scaling parameter \(|E|\) in (17) makes the bottleneck potential invariant to the grid size.

CNN training. We use six out of eleven labeled horizons for training the CNN adopted from [42] mentioned in Figure 6 in Appendix. To prevent the CNN from learning the whole ground truth for these six horizons only 10% of the possible patches are used. For data augmentation, translation is performed on non-matching patches as also done in [35] for stereo. The validation set contains 2% of the possible patches from each of the eleven horizons.

For training the CNN, in (16), edges in underlying MRF models are sampled for creating the patches. Training was done for 200 epochs and the model with the best validation accuracy (95%) was used for computing the potentials. For training the CNN used in (15), we use the same procedure as above except that the patches are sampled randomly and thus their respective nodes do not need to be adjacent. The best validation accuracy on the trained model was 83%.

Algorithms. We compare our method with a plain MRF, and a state-of-the-art heuristic for horizon tracking based on a variation of minimum spanning tree (MST) problem.

- MST: The horizon tracking approach of [41] is a greedy approach inspired by Borůvka’s algorithm for MST [4]. It starts by marking the nodes where the seed labels are given and iteratively marks assigns label to the adjacent node that contributes minimal cost w.r.t. unary and pairwise potentials.

- MRF: We solve MAP-MRF on \( G = (V, E) \) with unary and pairwise potentials \( \theta \) from above with TRWS [18].

- B-MRF: We additionally include a bottleneck term on pairwise bottleneck potentials (17). We solve the problem with the algorithm from Section 4.3, and round a primal solution with the procedure from Section 4.4.

Results. Table 1 lists the mean absolute deviation (MAD) of different methods for tracking horizon surfaces. MAD is computed as \( \frac{\|\tilde{d} - d\|_1}{N_1 N_2} \) i.e., the \( L_1 \) norm of the difference in tracked surface \( \tilde{d} \) and the ground-truth \( d \), normalized by the dimensions \( N_1, N_2 \) of the surface. From Table 1 we see that our method B-MRF outperforms MRF and MST by a wide margin on most problems. The resulting depth surfaces can be seen in Figure 4, and in the Appendix. Inspecting the results we see that MRF finds piece-wise continuous assignments not necessarily corresponding to the correct surface but preferring easy to track ones. On the other hand, MST often follows the horizon quite well but as soon as it marks a pixel wrongly, it often cannot recover due to it being greedy. Our method addresses these shortcomings to a large degree.
References


[41] Y. Yu, C. Kelley, and I. Mardanova. Automatic horizon picking in 3D seismic data using optical filters and minimum spanning tree (patent pending), pages 965–969. 2012. 1, 2, 6, 7, 8

6. Appendix

Lemma 1. The following relaxation over the local polytope \( A(4) \) is not tight for bottleneck labeling problem:

\[
\min_{\mu \in \mathcal{A}} \sum_{i \in V} \langle \theta_i, \mu_i \rangle + \sum_{ij \in E} \langle \theta_{ij}, \mu_{ij} \rangle + b \quad (18)
\]

s.t.
\[
b \geq \langle \phi_i, \mu_i \rangle, \quad \forall i \in V \quad (19a)
\]
\[
b \geq \langle \phi_{ij}, \mu_{ij} \rangle, \quad \forall ij \in E \quad (19b)
\]

Proof. We give a proof by example as in the Figure 5 with a 3 node binary graphical model only containing pairwise bottleneck potentials with \( \epsilon > 0 \). Both integer optimal solutions have cost: \( \max(a, a + \epsilon) = a + \epsilon \), whereas the optimal solution of the LP relaxation \( \mu_{uw}(0, 0) = \mu_{vw}(1, 1) = \mu_{vw}(0, 1) = 0.5 \) has cost \( a + \frac{\epsilon}{2} \).

Lemma 2. Shortest path computation (line 6) on directed acyclic graph \( D \) can be done in linear time in the number of arcs \( A \) using breadth-first traversal by Algorithm `dsp_chain`. However, shortest path update is performed every time after introducing each arc in \( A \) (lines 4-10), thus making the runtime \( \mathcal{O}(|A|^2) \) i.e. quadratic in the number of arcs in \( A \). The sorting operation adds an additional factor of \( |A| \log |A| \).

6.1. Primal rounding details

The function \( s(i) \) defines some ordering for all nodes \( i \) in \( V \). Based on this order, we sequentially fix labels of the respective node as in Algorithm `primal_rounding`.

For any given dual variables, there is a whole set of equivalent dual variables that give the same dual lower bound. However, when rounding with `primal_rounding`, the final solution is sensitive to the choice of dual equivalent dual variables. One of the reasons is that the rounding is done on the MRF subproblems only, so we want to ensure that the dual variables \( \lambda \) carry as much information as possible. Therefore, we propose a propagation mechanism that modifies the dual variables \( \eta \) of the bottleneck potentials

\[
\text{chain_to_dag: chain MRF to directed acyclic graph transformation}
\]

**Data:** Chain MRF: \( (G = (V, E), \mathcal{X}, \{\theta_f\}_{f \in V \cup E}) \), Bottleneck potentials: \( \{\phi_f(x_f)\}_{f \in V \cup E} \),

**Result:** Directed graph: \( D = (W, A) \)

Linear costs: \( \sigma(h, t) \quad \forall (h, t) \in A \)

Bottleneck costs: \( \omega(h, t) \quad \forall (h, t) \in A \)

Start and terminal nodes \( s, t \)

// Represent each label with two nodes and add source, sink:

1. \( W = \{s, t\} \cup \{x, \pi : i \in V, x \in \mathcal{X}_i\} \);

// Add arcs to represent potentials:

2. \( A = \{(x, \pi_i) : i \in V, x \in \mathcal{X}_i\} \cup \{\pi_i, x_{i+1} : j \in [n-1], x_{i+1} \in \mathcal{X}_{i+1}\} \cup \{\pi_n, t : t \in \mathcal{X}_n\} \)

// Unary potentials:

3. \( \sigma(\pi_i, x_i) = \theta_i(x_i) \quad \forall i \in V, x_i \in \mathcal{X}_i \);

// Pairwise potentials:

4. \( \sigma(\pi_i, x_{i+1}) = \theta_{i+1}(x_{i+1}) \)

// Connect \( s \) and \( t \):

5. \( A' = \{(s, x_1) : x_1 \in \mathcal{X}_1\} \cup \{\pi_n, t : t \in \mathcal{X}_n\} \)

**dsp_chain: dynamic shortest path on chains**

**Data:** Diagraph: \( (W, A) \)

Node distances: \( d(w), \quad \forall w \in W \)

Arc costs: \( \sigma(p, q), \quad \forall (p, q) \in A \)

Arc to insert: \( (u, v), \quad u, v \in W \)

**Result:** Updated node distance: \( d \)

Updated nodes: \( S \subseteq W \)

1. Initialize \( Q = \emptyset \);

2. Enqueue \((Q, (u, v))\);

3. while \( Q \neq \emptyset \) do

4. \((p, q) = \text{Dequeue}(Q)\);

5. if \( d(q) \geq c(p, q) + d(p) \) then

6. continue;

7. end

8. \( d(q) := c(p, q) + d(p) \);

9. \( S = S \cup \{q\} \);

// Enqueue outgoing arcs from \( q \) for possible distance update:

10. foreach \((q, r) \in A \) do

11. Enqueue \((Q, (u, v))\);

12. end

13. end
such that the overall lower bound is not diminished, while at the same time making the MRF dual variables as informative as possible. This procedure is described in Algorithm Min_Marginals_BMRF.

Schemes similar to Algorithm Min_Marginals_BMRF were also proposed in [27] for exchanging information between pure MRF subproblems. Intuitively, the goal in such schemes is to extract as much information out of a source subproblem (in our case bottleneck MRF subproblem) and send it to the target subproblem (in our case MRF subproblem) such that the overall lower bound does not decrease. Such a strategy helps the target subproblem in taking decisions which also comply with the source subproblem.

Algorithm Min_Marginals_BMRF computes min-marginals of nodes for a given bottleneck chain subproblem $u$. Proceeding in a similar fashion as Algorithm chain_bottleneck, min-marginals computation on a chain $u$ is done as follows:

1. As an input, the costs of solution in all other chains excluding $u$ are required, which can be obtained in the similar fashion as was done before in Algorithm chain_decomp (lines 1-10).
2. The algorithm maintains forward and backward shortest path distances $d_r, d_l$ (i.e., distances from source, sink resp.). This helps in finding the cost of minimum distance path passing through any given node in the directed acyclic graph of chain $u$.
3. Similar to Algorithm chain_bottleneck, bottleneck potentials are sorted and the bottleneck threshold is relaxed minimally on each arc addition.
4. On every arc addition, the set of nodes for which distance from source/sink got decreased are maintained in $S_r, S_l$ resp. Only this set of nodes will need to re-compute their min-marginals.

---

The above-mentioned Algorithm Min_Marginals_BMRF only calculates the min-marginals for nodes, the calculation of min-marginals for edges can be carried out in similar way by also updating those edges of the DAG whose adjacent node gets updated (lines 14-15). After computing the min-marginals, messages to MRF tree Lagrangians $\lambda$ can be computed by appropriate normalization.
Figure 6. Workflow to compute the patch matching probabilities (14). Two axis-aligned patches are extracted around each voxel in seismic volume to get 2-channel patch image. These two images are concatenated to get a 4-channel image which would be used as an input to the CNN [42] for computing patch matching probabilities by softmax activation at the output. $C(n, k, s)$ denotes a convolutional layer having $n$ filters of size $k \times k$ with stride $s$, $F(n)$ to a fully connected layer with $n$ outputs, and MaxPool($m$, $n$) to max-pooling with kernel size $m \times m$ and stride $n$.  

<table>
<thead>
<tr>
<th></th>
<th>F3-Netherlands</th>
<th>Opunake-3D</th>
<th>Waka-3D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>I</td>
<td>II*</td>
<td>III*</td>
</tr>
<tr>
<td>$</td>
<td>V</td>
<td>$</td>
<td>263231</td>
</tr>
<tr>
<td>$\sum_{i \in V}</td>
<td>X_i</td>
<td>$</td>
<td>3271662</td>
</tr>
</tbody>
</table>

Table 2. Instance sizes of all horizon surfaces used in the experiments, $|V|$ represents total number of nodes and $\sum_{i \in V} |X_i|$ is for total number of labels in the instance.

6.2. Results

Figures 7-17 contain the visual comparison of tracked horizon surfaces mentioned in Table 1. Similar to Figure 4, the color of the surface denotes depth.
Figure 7. F3-Netherlands-II

Figure 8. F3-Netherlands-III

Figure 9. F3-Netherlands-IV

Figure 10. F3-Netherlands-V
Figure 11. F3-Netherlands-VI

Figure 14. Waka-3D-I

Figure 15. Waka-3D-II

Figure 16. Waka-3D-III
Figure 17. Waka-3D-IV