Bottleneck potentials in Markov Random Fields

Ahmed Abbas [1,2]   Paul Swoboda[1]

[1]Max Planck Institute for Informatics,   [2]ZEISS

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_∞-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 [44, 13, 28, 10], where making a single error and following a wrong track results in low accuracy nonetheless. In inverse problems, L_∞-norm regularization penalizes the

maximum deviation from the fitted model and is appropriate e.g. for some types of group sparsity [27, 23, 24]. 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

$$\min_{x \in X} \left(\max_i \{\psi_i \cdot x_i\} \right) \tag{1}$$

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 [32] and has been applied to image segmentation. As noted by the authors of [32], 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_∞ via a high value of p. In [18] a more general class of MAP-inference with higher order costs were proposed which include the bottleneck potentials we investigate. In [25] the authors propose a model that can be specialized to bottleneck costs with linear distances and give a primal heuristic for solving it.

Furthermore, labeling problems containing only bottleneck objectives were considered in [37, 33, 11, 29], where [11, 29] devised algorithms for special cases of the pure min-max labeling problem, and [37, 33] devised message passing schemes with application in parallel machine scheduling.

In contrast to previous works [8, 32, 37, 33, 11, 29] we investigate the mixed problem where ordinary and bottleneck

potentials are both present in the same inference problem. Unlike [32] we allow arbitrary MRFs. Also unlike the algorithm in [8, 25], our algorithms are based on a linear programming relaxation resulting in a more rigorous approach. Lastly, in comparison to [18] we propose a stronger relaxation (theirs corresponds to 1) and an algorithm that scales to large problem sizes.

Combinatorial bottleneck optimization problems. Many classical optimization problems have bottleneck counterparts, e.g. bottleneck shortest paths [15], bottleneck spanning trees [12] and bottleneck function optimization in greedoids [22]. 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 [31].

L_∞**-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_∞-norm are used in [27] for multi-task learning and in [23] for the multi-task lasso. More generally, mixed norms with L_∞-norms are useful for problems where group sparsity is desirable. Proximal [27], block coordinate descent [23] as well as network flow techniques [24] have been proposed for numerical optimization of such problems.

Semiring-based constraint satisfaction. In semiring-based constraint satisfaction problems the goal is to compute $\bigoplus_{x_V \in X} \bigodot_{A \in E} f_A(x_A)$, where X is a space of labellings on a node set V, E is collection of subsets of V, and f_A are functions that depend only on nodes in the subset A [41, 42, 17, 3, 2]. Popular choices for the pair $(\bigoplus, \bigodot)$ is the $(\min, +)$-semiring which corresponds to MAP-inference and the $(+, \times)$-semiring which corresponds to computing the partition function. In contrast to the above semirings, the algebra corresponding to bottleneck potentials $(\min, \max)$ is only a pre-semiring (the distributive law does not hold) and hence classical arc consistency algorithms as discussed in [41, 42] 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 completely fit into (pre-)semiring based constraint satisfaction setting, since we are concerned with the mixed case in which both the $(\min, +)$-semiring and the $(\min, \max)$-pre-semiring occur together in a single optimization problem.

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 [44, 13, 43, 39] from the computational perspective. The authors in [44] 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 [43, 39] the authors set up linear equations to solve the horizon tracking problem.

Most similar to our work is the minimum spanning tree inspired approach of [44]. In contrast to [44], 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 [43, 39] 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.

Code and datasets are available at https://github.com/LPMP/LPMP.

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 $\mathcal{X}_i$ is associated. To each node $i \in V$ a *unary potential* $\theta_i : \mathcal{X}_i \to \mathbb{R}$ is associated, and to each edge $ij \in E$ a *pairwise potential* $\theta_{ij} : \mathcal{X}_i \times \mathcal{X}_j \to \mathbb{R}$. We will call $\mathcal{X}_V = \prod_{i \in V} \mathcal{X}_i$ the *label space* and $x \in \mathcal{X}$ a *labeling*. For subsets $U \subset V$ we define $\mathcal{X}_U = \prod_{i \in U} \mathcal{X}_i$, analogously we refer to labels $x_U \in \mathcal{X}_U$. In particular, x_i refers to a node labeling and $x_{ij} = (x_i, x_j)$ to an edge labeling. A tuple $(G, \mathcal{X}, \boldsymbol{\theta})$ consisting of a graph, corresponding label space and potentials is called a Markov Random Field (MRFs). The problem

$$\min_{\boldsymbol{x} \in \mathcal{X}} \boldsymbol{\theta}(\boldsymbol{x}), \quad \boldsymbol{\theta}(\boldsymbol{x}) := \sum_{i \in V} \theta_i(x_i) + \sum_{ij \in E} \theta_{ij}(x_{ij}) \quad (2)$$

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 $\mathcal{X}_i$ with one-hot encoding $e_k = (0, \ldots, 0, \underbrace{1}_{k}, 0, \ldots 0)$ which is a unit vector of length $|\mathcal{X}_i|$ with a 1 at the k-th location. In other words, we can write $\mathcal{X}_i = \{e_1, e_2, \ldots e_{|\mathcal{X}_i|}\} \ \forall i \in V$. Analogously, we can write $\mathcal{X}_{ij} = \{e_1, \ldots, e_{|\mathcal{X}_{ij}|}\} \ \forall ij \in E$. To obtain a convex relaxation, we define *unary marginals* as $\mu_i \in \text{conv} \, \mathcal{X}_i, \ i \in V$, and *pairwise marginals* as $\mu_{ij} \in \text{conv} \, \mathcal{X}_{ij}, \ ij \in E$. We couple unary and pairwise marginals together to obtain the *local polytope relaxation* [40].

$$\Lambda = \left\{ \mu \ \middle| \ \begin{array}{l} \mu_i(x_i) = \sum\limits_{x_j \in \mathcal{X}_j} \mu_{ij}(x_{ij}), \ \forall ij \in E, \ x_i \in \mathcal{X}_i \\ \mu_f \in \text{conv} \, \mathcal{X}_f, \ \forall f \in V \cup E \end{array} \right\} \tag{3}$$

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

$$\min_{\mu \in \Lambda} \sum_{i \in V} \langle \theta_i, \mu_i \rangle + \sum_{ij \in E} \langle \theta_{ij}, \mu_{ij} \rangle \tag{4}$$

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 [20].

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 $\phi_i : \mathcal{X}_i \to \mathbb{R} \ \forall i \in V$ *and* pairwise bottleneck potentials $\phi_{ij} : \mathcal{X}_i \times \mathcal{X}_j \to \mathbb{R} \ \forall ij \in E$ *are also given. We call the set of all possible values taken by bottleneck potentials as* bottleneck values*, i.e.*

$$B = \{\phi_f(x_f) : f \in V \cup E, x_f \in \mathcal{X}_f\}. \tag{5}$$

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

$$\min_{\boldsymbol{x} \in \mathcal{X}, b \in B} \boldsymbol{\theta}(\boldsymbol{x}) + \zeta(b) \tag{6a}$$

$$s.t. \quad \phi_f(x_f) \le b \quad \forall f \in V \cup E \tag{6b}$$

and $\boldsymbol{\theta}(\boldsymbol{x})$ *is defined in* (2).

It is straightforward to adapt our work to the case of multiple bottlenecks and triplet/quadrupelt/... 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 [32] 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. $\boldsymbol{\theta} = 0$ then the problem reduces to a pure bottleneck labeling problem as:

$$\min_{\boldsymbol{x} \in \mathcal{X}} \max_{f \in V \cup E} (\phi_f(x_f)) \tag{7}$$

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 ∞ for the labelings which have bottleneck potentials greater than the optimal value b^*. i.e.

$$\theta_e(x_e) := \infty \quad \forall e, x_e : \phi_e(x_e) > b^* \tag{8}$$

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. $E = \varnothing$, 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 θ_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

$$(b^*, c^*) = \underset{(b,c) \in M}{\arg\min} \, [c + \zeta(b)] \tag{9}$$

unary_bottleneck: bottleneck labeling on unary graph

Data:
MRF without edges: $(G = (V, \varnothing), \mathcal{X}, \{\theta_i\}_{i \in V})$,
Bottleneck potentials: $\{\phi_i\}_{i \in V}$.
Result: Costs of labelings:
$$M = \left\{ (b, c) : \begin{array}{l} c = \min_{\boldsymbol{x}} \sum_{i \in V} \theta_i(x_i), \\ \phi_i(x_i) \le b \quad \forall i \in V \end{array} \right\}.$$
// Merge labels of all nodes in Θ:
1   $\Theta = \{(i, x_i) : i \in V, x_i \in \mathcal{X}_i\}$;
2   Sort Θ according to $\phi_i(x_i)$;
3   $c = 0, l = \infty \in \mathbb{R}^{|V|}, S = \varnothing, M = \varnothing$;
4   **for** $(i, x_i) \in \Theta$ *in ascending order* **do**
5   **if** $i \notin S$ **then**
6         $S = S \cup \{i\}$;
7         $c = c + \theta_i(x_i)$;
8         $l_i = \theta_i(x_i)$;
9   **end**
10   **if** $\theta_i(x_i) < l_i$ **then**
11         $c = c - l_i + \theta_i(x_i)$;
12         $l_i = \theta_i(x_i)$;
13   **end**
14   **if** $S = V$ **then**
15         $M = M \cup \{(\phi_i(x_i), c)\}$;
16   **end**
17   **end**

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).

Proposition 1. *The runtime of Algorithm* **unary_bottleneck** *is $\mathcal{O}(L \log L + L)$ where $L = \sum_{i \in V} |\mathcal{X}_i|$*

The most expensive operation in Algorithm **unary_bottleneck** is sorting. However, the sorting can be reused when the algorithm is run multiple times with varying linear potentials θ, which is the case in our dual decomposition approach for general graphs.

4.2. Bottleneck labeling problem on chains:

Assume $V = [n]$ and $E = \{(1, 2), (2, 3), \ldots, (n-1, n)\}$ is a chain. Even though the relaxation over the local polytope for inference in chains is tight for pairwise MRFs, introducing the bottleneck potential (as also done in [18]) destroys this property which is demonstrated in the Appendix in Lemma 1. Therefore, we propose an efficient algorithm to solve the bottleneck labeling problem (6) exactly on chains. First, we note that the MAP-MRF problem (2) can be modeled through a shortest path problem in a directed acyclic graph. Figure 1 illustrates the construction for the case of $n = 3$. We treat each label x_i of a variable i in V in the

chain as a pair of nodes x_i, and $\overline{x}_i$ in the shortest path digraph $D = (W, A)$. Each unary cost $\theta_i(x_i)$ becomes an arc cost for $(x_i, \overline{x}_i)$. Each pairwise cost $\theta_{ij}(x_i, x_j)$ becomes an arc cost $(\overline{x}_i, x_j)$. The source and sink nodes s, t are connected with the labels of first and last node of the graphical model resp. with zero costs for modeling the shortest path problem. Algorithm **chain_to_dag** in the Appendix gives the general construction.

The shortest s, t-path P in graph D having cost $\boldsymbol{\sigma}(P)$ corresponds to an optimum labeling for $\boldsymbol{\theta}(\boldsymbol{x})$ from (6) in the chain graphical model G.
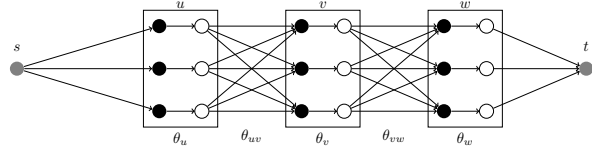

Figure 1. Shortest path network for a chain MRF with nodes $\{u, v, w\}$ and 3 labels for each node. The state x_i of each node $i \in V$ represented by ● has been duplicated to $\overline{x}_i$ represented by ○ to introduce unary potentials as arc costs. Text below the arcs represent their costs. Missing arcs have cost ∞.

Algorithm **chain_bottleneck** uses the constructed directed acyclic graph to solve the bottleneck labeling problem (6) 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 (9).

Naively implementing Algorithm **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 [7]. These improvements are detailed in Algorithm **dsp_chain** in the Appendix.

Proposition 2. *The worst case run-time of Algorithm* **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 **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 **unary_bottleneck**.

chain_bottleneck: bottleneck labeling on chain graphs

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

Result:
$$M = \left\{ (b, c) : \begin{array}{l} c = \min_{\boldsymbol{x}} \sum_{f \in F} \theta_f(x_f), \\ \phi_f(x_f) \leq b \quad \forall f \in V \cup E \end{array} \right\}$$

// Represent chain as DAG:

1 $(D = (W, A), \boldsymbol{\sigma}, \boldsymbol{\omega}, s, t) \leftarrow$ **chain_to_dag**(R, ϕ) ;

// Shortest path distances:

2 $d(s) := 0, \qquad d(w) := \infty, \quad \forall w \in W \setminus \{s\}$;

3 Sort A according to values $\boldsymbol{\omega}$;

4 **for** $(n_i, n_j) \in A$ *in ascending order* **do**

5 $\quad$ $A' = (n_i, n_j) \cup A'$;

$\quad$ // Update d w.r.t arc-costs $\boldsymbol{\sigma}$:

6 $\quad$ $d = $ **dsp_chain**$((W, A'), d, \boldsymbol{\sigma}, (n_i, n_j))$;

$\quad$ // Check for s-t path:

7 $\quad$ **if** $d(t) < \infty$ **then**

$\quad\quad$ // Store path costs:

8 $\quad\quad$ $M = M \cup \{(\omega(n_i, n_j), d(t))\}$;

9 $\quad$ **end**

10 **end**

Remark 1. *If we have a pure bottleneck labeling problem without the MRF potentials* (7) *then the problem can be solved in linear-time by sorting the bottleneck weights and halving the number of edges by the median edge as mentioned in* [15]. *The problem we consider is more general and needs to account for the MRF potentials.*

Remark 2. *The chain bottleneck labeling problem can also be solved for linear bottleneck cost* $\zeta(x) = x$ *by the method from* [31]. *However, the method from* [31] *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 [40, 21, 34], 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_bottleneck** 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 $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1), \ldots, (\mathcal{V}_h, \mathcal{E}_h)$. For the bottleneck labeling problem we cover the graph G by chains (trees) $\mathbb{G}_1 = (\mathbb{V}_1, \mathbb{E}_1), \ldots, \mathbb{G}_k = (\mathbb{V}_k, \mathbb{E}_k)$. For the decomposition we introduce variables $\boldsymbol{x}^t$ to specify the labeling of the MRF subproblem for graph $\mathcal{G}_t$, and variables $\boldsymbol{y}^l$ for the chain bottleneck labeling subproblems defined for graph $\mathbb{G}_l$. We propose the following overall decomposition:

$$\min_{\substack{\boldsymbol{x}, \{\boldsymbol{x}^t\}, \{\boldsymbol{y}^l\}, \\ b \in B}} \boldsymbol{\theta}(\boldsymbol{x}) + \zeta(b) \tag{10a}$$

$$\text{s.t.} \quad x_f^t = x_f \qquad \forall f \in \mathcal{V}_t \cup \mathcal{E}_t, t \in [h] \tag{10b}$$

$$y_f^l = x_f \qquad \forall f \in \mathbb{V}_l \cup \mathbb{E}_l, l \in [k] \tag{10c}$$

$$\boldsymbol{x}^t \in \mathcal{X}_{\mathcal{V}_t} \qquad \qquad \forall t \in [h] \tag{10d}$$

$$\boldsymbol{y}^l \in \mathcal{X}_{\mathbb{V}_l} \qquad \qquad \forall l \in [k] \tag{10e}$$

$$\phi_f(y_f^l) \leq b \qquad \forall f \in \mathbb{V}_l \cup \mathbb{E}_l, l \in [k] \tag{10f}$$

We constrain the variables for MRF tree subproblems and chain bottleneck subproblems to be consistent via (10b) and (10c). The labelings on chains ($\boldsymbol{y}$) 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 λ_e^t and η_e^l respectively. We keep rest of the constraints and denote the feasible variables $\boldsymbol{y}^l$ for chain l w.r.t constraints (10e), (10f) by the set $Y^l(b)$ as:

$$Y^l(b) = \left\{ \boldsymbol{y}^l \in \mathcal{X}_{\mathbb{V}_l} \, \middle| \, \phi_f(y_f^l) \leq b, \, \forall f \in \mathbb{V}_l \cup \mathbb{E}_l \right\} \tag{11}$$

The dual problem is:

$$\max_{\boldsymbol{\lambda}, \boldsymbol{\eta}} \left[\sum_{t \in [h]} E^t(\boldsymbol{\lambda}^t) + J(\boldsymbol{\eta}) \right] \tag{12a}$$

$$\text{s.t.} \sum_{\substack{t \in [h]: \\ f \in \mathcal{V}_t \cup \mathcal{E}_t}} \lambda_f^t + \sum_{\substack{l \in [k]: \\ f \in \mathbb{V}_l \cup \mathbb{E}_l}} \eta_f^l = \theta_f \quad \forall f \in V \cup E \tag{12b}$$

Where $E^t(\boldsymbol{\lambda}^t)$ and $J(\boldsymbol{\eta})$ are defined as:

$$E^t(\boldsymbol{\lambda}^t) := \min_{\boldsymbol{x}^t \in \mathcal{X}_{\mathcal{V}_t}} \langle \boldsymbol{\lambda}^t, \boldsymbol{x}^t \rangle \tag{13a}$$

$$J(\boldsymbol{\eta}) := \min_{b \in B} \left[\zeta(b) + \sum_{l \in [k]} \min_{\boldsymbol{y}^l \in Y^l(b)} \langle \boldsymbol{\eta}^l, \boldsymbol{y}^l \rangle \right] \tag{13b}$$

Evaluating $E^t(\cdot)$ amounts to solving a MAP-MRF problem on a tree. Evaluating $J(\cdot)$ corresponds to computing for each

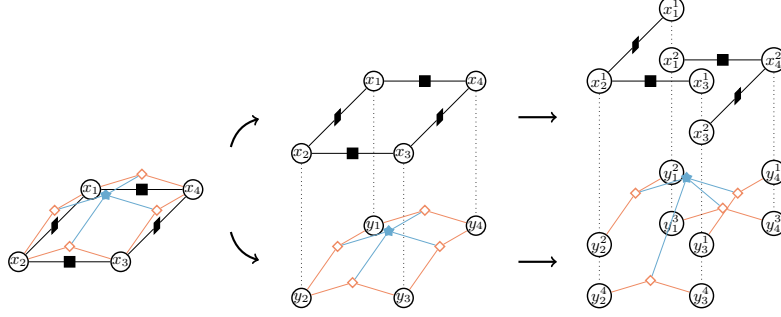Figure 2. Illustration of the decomposition (10) for the bottleneck labeling problem. Black squares ■ represent MRF-potentials θ_{ij}. Orange diamonds ◇ stand for bottleneck potentials ϕ_{ij}. The blue star ★ stands for bottleneck costs ζ. The problem is decomposed into a MRF part (upper layer) and a bottleneck part (lower layer). In turn, the MRF-layer is decomposed into trees and the bottleneck part into chains. Bottleneck chains are connected by a global bottleneck term ★.

bottleneck value b the corresponding minimal assignment from the set $Y^l(b)$ 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 $E^t(\cdot)$ and $J(\cdot)$ gives supergradients of (12a) which can be used in subgradient based solvers (e.g. subgradient descent, bundle methods).

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

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

1. Build a higher level graph $H = ([k], \varnothing)$ 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 **chain_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 $\overline{y}$ 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 [35]. 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 [19] by treating the dual optimal values of λ as MRF potentials. Additional details on how to best choose λ and more details are given in Section 6.2 in the Appendix.

We also considered using message passing instead of subgradient ascent by computing the min-marginals through the

---

chain_decomp: bottleneck labeling on chain decomposition of general graph

Data: Chain MRFs: $\{R_l = (\mathbb{G}_l, \mathcal{X}_l, \boldsymbol{\eta}^l)\}_{l \in [k]}$,
Bottleneck potentials on chains: $\{\boldsymbol{\phi}^l\}_{l \in [k]}$
Result: Optimal solution $\{\boldsymbol{y}^l\}_{l \in [k]}$ and b^* for $J(\boldsymbol{\eta})$

1   $H \leftarrow ([k], \varnothing)$ // higher level graph
2   **for** $l \in [k]$ **do**
     // Populate potentials for node l:
3      $M_l \leftarrow$ **chain_bottleneck**$(R_l, \boldsymbol{\phi}^l)$;
4   **for** $\forall (b, c) \in M_l$ **do**
5         $(\Phi_l(b), \Theta_l(b)) = (b, c)$ // Bott., MRF pots.
6         $\mathcal{Y}_l = \mathcal{Y}_l \cup b$ // Add b as a label for l
7   **end**
8   **end**
    // Solve graph H:
9   $\overline{M} \leftarrow$ **unary_bottleneck**$((H, \mathcal{Y}, \boldsymbol{\Theta}), \boldsymbol{\Phi})$;
10   $(b^*, c^*) = \arg\min_{(b,c) \in \overline{M}} \zeta(b) + c$;
    // Optimal labeling of chains:
11   **for** $l \in [k]$ **do**
12      $(\mathbb{D}_l = (W_l, \mathbb{A}_l), \boldsymbol{\sigma}^l, \boldsymbol{\omega}^l) \leftarrow$ **chain_to_dag**$(R_l, \boldsymbol{\phi}^l)$
        for $(n_i, n_j) \in \mathbb{A}_l$ **do**
13   **if** $\omega^l(n_i, n_j) > b^*$ **then**
14           $\sigma^l(n_i, n_j) \leftarrow \infty$ // mark as infeasible
15   **end**
16   **end**
17      $\overline{\boldsymbol{y}}^l = $ shortest s, t-path in $(W_l, A_l, \boldsymbol{\sigma}^l)$;
18   **end**

---

procedure used in primal rounding. However, our strong relaxation does not easily allow an efficient way for min-marginal computation. Specifically, for the subproblem $J(\boldsymbol{\eta})$ in (13b) it is hard to reuse computations to recalculate min-marginals efficiently for different variables, making any message passing approach slow. Rounding is only executed once at the end, hence very efficient min-marginal computation is of lesser concern.
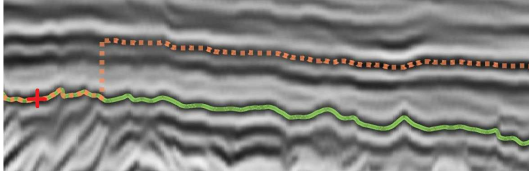
5. Experiments

Figure 3. Exemplary failure case for conventional MRF (- in dotted orange) in seismic horizon tracking problem as compared to MRF with an additional bottleneck potential (- in solid green). (+) indicates the seed. The MRF solution makes one local error with high cost and starts tracking another smoother layer leading to an overall lower cost solution. A bottleneck term penalizes such high cost errors and results in the correct track.

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 [44] 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 state the problem as MAP-inference in MRFs for which high-quality solvers exist that are less sensitive to local optimal.
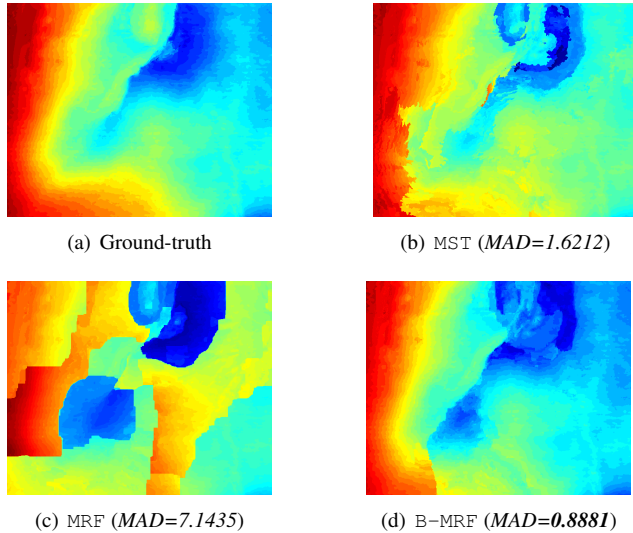

(a) Ground-truth

(b) MST (*MAD=1.6212*)

(c) MRF (*MAD=7.1435*)

(d) B-MRF (*MAD=0.8881*)

Figure 4. Comparison of results for horizon surface `F3-Netherlands-I`, colors indicate depth of the surface. Mean absolute deviation (*MAD*) scores are used as error metric. The `MRF` solution is piece-wise smooth but tracks wrong layers, `MST` is incorrect in the top-right and lower-left region, while our approach of `B-MRF` has least errors. Best viewed in color.

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 $3000km^2$ 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 [36]. 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 software [14]. Annotation took two days of work of an experienced seismic interpreter.

Matching costs: In order to track horizons, established state-of-the-art tracking algorithms [44, 43, 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 opti-

	F3-Netherlands						Opunake-3D		Waka-3D			
	I	II*	III*	IV	V*	VI	I*	II	I	II*	III*	IV
Instance sizes												
$|V|$	263K	153K	362K	153K	231K	101K	443K	2965K	614K	366K	601K	614K
$\sum_{i \in V} |\mathcal{X}_i|$	3271K	1768K	4394K	1422K	1896K	944K	406K	2630K	3524K	2020K	3110K	3646K
Mean absolute deviation from ground truth												
MST	1.6212	1.1245	1.6542	0.1283	0.1088	**1.5319**	0.4930	0.4596	2.0060	0.2759	0.9457	1.6108
MRF	7.1435	6.2105	3.2169	0.1540	0.0896	2.4498	**0.4786**	0.9002	1.4073	0.0221	0.4245	1.2388
B-MRF	**0.8881**	**0.2644**	**0.0889**	**0.0463**	**0.0894**	1.6049	0.4887	**0.4209**	**1.1855**	**0.0116**	**0.4135**	**1.0377**
Runtimes (seconds)												
MST	14	7	19	3	6	7	8	8	8	5	7	8
MRF	673	139	931	62	65	33	114	522	740	22	687	781
B-MRF	1820	1374	3108	1609	1554	1322	2979	2554	6548	2402	7738	8158

Table 1. **Top:** Instance sizes of horizon surfaces. $|V|$ represents total number of nodes and $\sum_{i \in V} |\mathcal{X}_i|$ the total number of labels in each instance. **Middle:** Mean absolute deviation from the ground-truth. **Bottom:** Runtimes using MST [44], TRWS [19] solver for MRF, and our method B-MRF. * marks the horizon surfaces used to train the CNN for computing patch similarity (14).

cal flow. We opt for a learning-based approach following the great success of CNN-based architectures for computing patch similarity [45, 38]. Specifically, we extend the architecture of the 2-D patch similarity CNN [45] to efficiently deal with the 3-D structure. CNN architecture and the procedure for training is mentioned in Figure 6 and Section 6.1 in Appendix. We train two CNNs to compute matching probabilities between depth labels $z_{ij} = z_i, z_j$ of nodes $i = (x_i, y_i), j = (x_j, y_j)$. The two neural networks CNN_n and CNN_f compute matching probabilities between adjacent and non-adjacent nodes respectively:

$$p_{ij}(z_{ij}) = CNN_{n|f}(I_i(z_i), I_j(z_j)) \in [0, 1] \qquad (14)$$

In (14) $I_k(z_k)$, is a two channel image of size 63×63 centered around (x_k, y_k, z_k).

Graph construction: 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 $\mathcal{X}_v = [D]$. A depth labeling thus corresponds to a node labeling of V. We compute unary $\theta_v : [D] \to \mathbb{R}, v \in V$ and pairwise potentials $\theta_{uv} : [D] \times [D] \to \mathbb{R}, uv \in E$ based on the patch matching costs computed in (14). Further details about unary and pairwise potentials can be found in Section 6.1 in Appendix.

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 [44] 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 θ from above with TRWS [19].

• B-MRF: We additionally include a bottleneck term on pairwise bottleneck potentials (19). 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 $\frac{\|d - \tilde{d}\|_1}{N_1 \cdot N_2}$ 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 the rest 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 its greedy nature. Our method addresses these shortcomings to a large degree. The resulting surfaces are smooth and track the correct surface to a larger extent than both MST and MRF by favoring piece-wise smooth regions due to the ordinary unary and pairwise MRF costs, yet following tracks that do not make incorrect jumps due to the additional bottleneck costs. Since we optimize a global energy by a convex relaxation method, we suffer less from poor local optimal than MST. On the other hand, the more expressive optimization model and advanced algorithms that allow B-MRF to obtain higher accuracy also lead to higher runtimes.

References

[1] Peter Bakker. *Image structure analysis for seismic interpretation*. PhD thesis, Delft University of Technology, 2002. 11, 12

[2] S. Bistarelli. *Semirings for Soft Constraint Solving and Programming*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004. 2

[3] S. Bistarelli, U. Montanari, F. Rossi, T. Schiex, G. Verfaillie, and H. Fargier. Semiring-based CSPs and valued CSPs: Frameworks, properties, and comparison. *Constraints*, 4(3):199–240, Sep 1999. 2

[4] Otakar Boruvka. O Jistém Problému Minimálním (About a Certain Minimal Problem) (in Czech, German summary). *Práce Mor. Prírodoved. Spol. v Brne III*, 3, 1926. 8

[5] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004. 2

[6] Y. Boykov and O. Veksler. *Graph Cuts in Vision and Graphics: Theories and Applications*, pages 79–96. Springer US, Boston, MA, 2006. 1

[7] Thomas T. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, MA, USA, 1990. 4

[8] C. Couprie, L. Grady, L. Najman, and H. Talbot. Power watershed: A unifying graph-based optimization framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(7):1384–1399, July 2011. 1, 2, 3

[9] R.A. Cuninghame-Green. *Minimax algebra*. Lecture notes in economics and mathematical systems. Springer-Verlag, 1979. 2

[10] Elena Fernandez, Robert Garfinkel, and Roman Arbiol. Mosaicking of aerial photographic maps via seams defined by bottleneck shortest paths. *Oper. Res.*, 46(3):293–304, Mar. 1998. 1

[11] Boris Flach and Michail I. Schlesinger. A class of solvable consistent labeling problems. In Francesc J. Ferri, José M. Iñesta, Adnan Amin, and Pavel Pudil, editors, *Advances in Pattern Recognition*, pages 462–471, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. 1

[12] Harold N Gabow and Robert E Tarjan. Algorithms for two bottleneck optimization problems. *Journal of Algorithms*, 9(3):411 – 417, 1988. 2

[13] Eliana L. Goldner, Cristina N. Vasconcelos, Pedro Mario Silva, and Marcelo Gattass. A shortest path algorithm for 2D seismic horizon tracking. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, SAC '15, pages 80–85, New York, NY, USA, 2015. ACM. 1, 2, 7, 12

[14] GVERSE Geophysics 2017.3. Lmkr. http://www.lmkr.com/gverse/gverse-geophysics/. 7

[15] Volker Kaibel and Matthias A. F. Peinhardt. On the bottleneck shortest path problem, 2006. 2, 5

[16] Jörg H. Kappes, Björn Andres, Fred A. Hamprecht, Christoph Schnörr, Sebastian Nowozin, Dhruv Batra, Sungwoong Kim, Bernhard X. Kausler, Thorben Kröger, Jan Lellmann, Nikos Komodakis, Bogdan Savchynskyy, and Carsten Rother. A comparative study of modern inference techniques for structured discrete energy minimization problems. *International Journal of Computer Vision*, 115(2):155–184, 2015. 1

[17] Jürg Kohlas and Nic Wilson. Semiring induced valuation algebras: Exact and approximate local computation algorithms. *Artif. Intell.*, 172(11):1360–1399, 2008. 2

[18] Pushmeet Kohli and M. Pawan Kumar. Energy minimization for linear envelope MRFs. In *CVPR*, 2010. 1, 2, 4

[19] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1568–1583, Oct 2006. 6, 8, 12

[20] Vladimir Kolmogorov, Johan Thapper, and Stanislav Źívnyź. The power of linear programming for general-valued CSPs. *SIAM J. Comput.*, 44(1):1–36, Feb. 2015. 3

[21] N. Komodakis, N. Paragios, and G. Tziritas. MRF energy minimization and beyond via dual decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):531–552, March 2011. 5

[22] B.H. Korte, L. Lovász, and R. Schrader. *Greedoids*. Algorithms and Combinatorics. Springer-Verlag, 1991. 2

[23] H. Liu, M. Palatucci, and J. Zhang. Blockwise coordinate descent procedures for the multi-task lasso, with applications to neural semantic basis discovery. *ICML*, pages 649–656, 2009. 1, 2

[24] Julien Mairal, Rodolphe Jenatton, Francis R. Bach, and Guillaume R Obozinski. Network flow algorithms for structured sparsity. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1558–1566. Curran Associates, Inc., 2010. 1, 2

[25] Pankaj Pansari and M. Pawan Kumar. Truncated max-of-convex models. In *CVPR*, 2017. 1, 2

[26] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017. 12

[27] Ariadna Quattoni, Xavier Carreras, Michael Collins, and Trevor Darrell. An efficient projection for l1,infinity regularization. In Andrea Pohoreckyj Danyluk, Lon Bottou, and Michael L. Littman, editors, *ICML*, volume 382 of *ACM International Conference Proceeding Series*, pages 857–864. ACM, 2009. 1, 2

[28] Fabian Rathke, Stefan Schmidt, and Christoph Schnörr. Probabilistic intra-retinal layer segmentation in 3-D OCT images using global shape regularization. *Medical image analysis*, 18(5):781–794, 2014. 1

[29] Michail I Schlesinger and Boris Flach. Some solvable subclasses of structural recognition problems. In *Czech Pattern Recognition Workshop*, volume 2000, pages 55–62, 2000. 1

[30] Alexander Shekhovtsov, Christian Reinbacher, Gottfried Graber, and Thomas Pock. Solving dense image matching in real-time using discrete-continuous optimization. In *Proceedings of the 21st Computer Vision Winter Workshop (CVWW)*, page 13, 2016. 12

[31] Tong-Wook Shinn and Tadao Takaoka. Combining the shortest paths and the bottleneck paths problems. In *Proceedings of the Thirty-Seventh Australasian Computer Science Conference - Volume 147*, ACSC '14, pages 13–18, Darlinghurst,

Australia, Australia, 2014. Australian Computer Society, Inc. 2, 5

[32] A. K. Sinop and L. Grady. A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8, Oct 2007. 1, 2, 3

[33] Christopher Srinivasa, Inmar Givoni, Siamak Ravanbakhsh, and Brendan J Frey. Min-max propagation. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5565–5573. Curran Associates, Inc., 2017. 1

[34] Geir Storvik and Geir Dahl. Lagrangian-based methods for finding map solutions for MRF models. *IEEE Transactions on Image Processing*, 9(3):469–479, 2000. 5

[35] P. Swoboda and V. Kolmogorov. MAP inference via Block-Coordinate Frank-Wolfe Algorithm. In *CVPR*, 2019. 6

[36] The Society of Exploration Geophysicists. Open data wiki. `https://wiki.seg.org/wiki/Open_data#3D_land_seismic_data`. 7

[37] M Vinyals, KS Macarthur, A Farinelli, SD Ramchurn, and NR Jennings. A message-passing approach to decentralized parallel machine scheduling. *COMPUTER JOURNAL*, 57:856–874, 2014. 1

[38] Jure Žbontar and Yann LeCun. Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research*, 17(65):1–32, 2016. 8, 12

[39] Ke Wang*, Kaihong Wei, Kevin Deal, and David Wilkinson. *3D Seismic horizon extraction with horizon patch constraints*, pages 1754–1758. 2015. 2

[40] Tomáš Werner. A linear programming approach to max-sum problem: A review. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 29(7):1165–1179, July 2007. 3, 5

[41] Tomáš Werner. Marginal consistency: Unifying constraint propagation on commutative semirings. In *Intl. Workshop on Preferences and Soft Constraints*, pages 43–57, September 2008. 2

[42] Tomáš Werner and Alexander Shekhovtsov. Unified framework for semiring-based arc consistency and relaxation labeling. In *12th Computer Vision Winter Workshop, St. Lambrecht, Austria*, pages 27–34. Graz University of Technology, February 2007. 2

[43] Xinming Wu and Dave Hale. Horizon volumes with interpreted constraints. *GEOPHYSICS*, 80(2):IM21–IM33, 2015. 2, 7

[44] Yingwei Yu, Cliff Kelley, and Irina Mardanova. *Automatic horizon picking in 3D seismic data using optical filters and minimum spanning tree (patent pending)*, pages 965–969. 2012. 1, 2, 7, 8, 12

[45] Sergey Zagoruyko and Nikos Komodakis. Learning to compare image patches via convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 8, 12, 14