Discrete Optimization in Vision and Graphics

Nikos Komodakis
University of Crete
http://www.csd.uoc.gr/~komod
komod@csd.uoc.gr

EUSIPCO 2008
Lausanne, Switzerland, August 2008

Talk outline

MRF optimization via message passing

MRF optimization based on linear programming

- Discrete optimization and convex relaxations
- MRFs and the primal-dual schema
- MRFs and dual decomposition

Message passing algorithms for MRF optimization

Message-passing algorithms

- Central concept: messages
- These methods work by propagating messages across the MRF graph
- Widely used algorithms in many areas

Message-passing algorithms

- But how do messages relate to optimizing the energy?
- Let's look at a simple example first: we will examine the case where the MRF graph is a chain

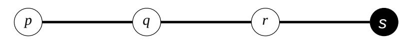
Message-passing on chains

MRF graph

Message-passing on chains

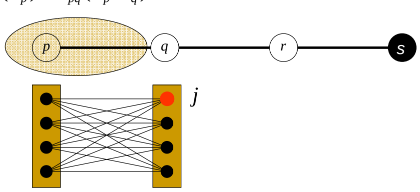
- Global minimum in linear time
- Optimization proceeds in two passes:
  - Forward pass (dynamic programming)
  - Backward pass

Message-passing on chains



Forward pass (dynamic programming)

$\theta_p(x_p) + \theta_{pq}(x_p, x_q)$

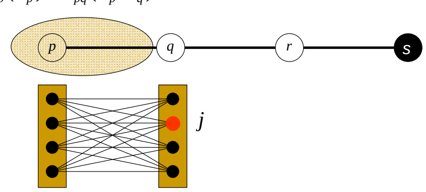


$M_{pq}(j) = \min_i \{ \theta_p(i) + \theta_{pq}(i, j) \}$

Slide credit: V. Kolmogorov

Forward pass (dynamic programming)

$\theta_p(x_p) + \theta_{pq}(x_p, x_q)$

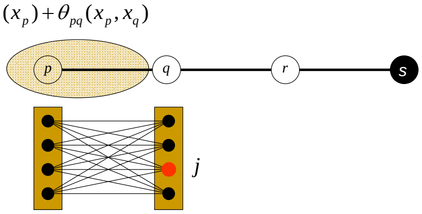


$M_{pq}(j) = \min_i \{ \theta_p(i) + \theta_{pq}(i, j) \}$

Slide credit: V. Kolmogorov

Forward pass (dynamic programming)

$\theta_p(x_p) + \theta_{pq}(x_p, x_q)$

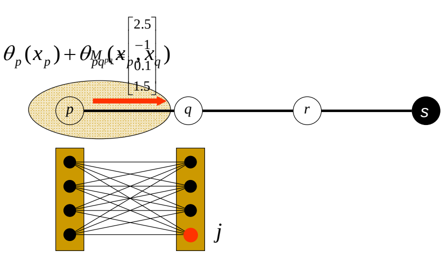


$M_{pq}(j) = \min_i \{ \theta_p(i) + \theta_{pq}(i, j) \}$

Slide credit: V. Kolmogorov

Forward pass (dynamic programming)

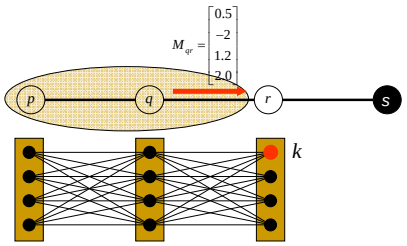
$\theta_p(x_p) + \theta_{pq}(x_p, x_q)$



$M_{pq}(j) = \min_i \{ \theta_p(i) + \theta_{pq}(i, j) \}$

Slide credit: V. Kolmogorov

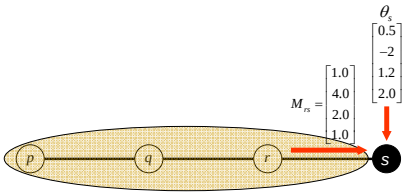
Forward pass (dynamic programming)



$$M_{qr}(k) = \min_j \left\{ \left(\theta_q(j) + M_{pq}(j) \right) + \theta_{qr}(j,k) \right\}$$

Slide credit: V. Kolmogorov

Forward pass (dynamic programming)



Min-marginal for node s and label j:

$$\min_{\mathbf{x}} \left\{ E(\mathbf{x}) \mid x_S = j \right\}$$

Slide credit: V. Kolmogorov

Backward pass

$$M_{qr}(k) = \min_j \left\{ \left(\theta_q(j) + M_{pq}(j) \right) + \theta_{qr}(j,k) \right\}$$

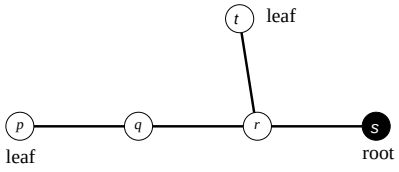
$$x_q^*(x_r) = \arg\min_j \left\{ \left(\theta_q(j) + M_{pq}(j) \right) + \theta_{qr}(j,x_r) \right\}$$

$$x_r^* = \arg\min_k \left\{ \left(\theta_r(k) + M_{qr}(k) \right) + \theta_{rs}(k,x_s) \right\}$$

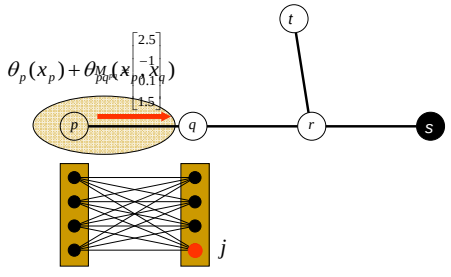
Message-passing on trees

- We can apply the same idea to tree-structured graphs
- Slight generalization from chains (still gives global optimum)
- Resulting algorithm called: **belief propagation [Pearl '88]** (independently invented many times in different fields)

Message-passing on trees



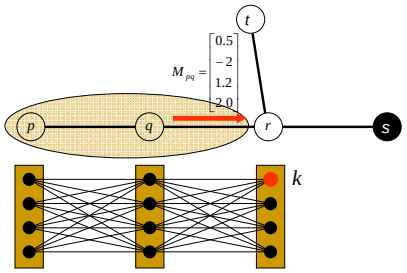
Forward pass (dynamic programming)



$$M_{pq}(j) = \min_i \left\{ \theta_p(i) + \theta_{pq}(i,j) \right\}$$

Slide credit: V. Kolmogorov

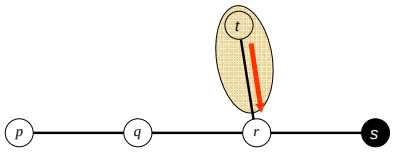
Forward pass (dynamic programming)



$$M_{qr}(k) = \min_j \{ (\theta_q(j) + M_{pq}(j)) + \theta_{qr}(j,k) \}$$

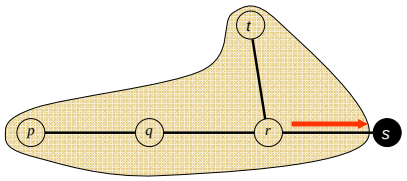
Slide credit: V. Kolmogorov

Forward pass (dynamic programming)



Slide credit: V. Kolmogorov

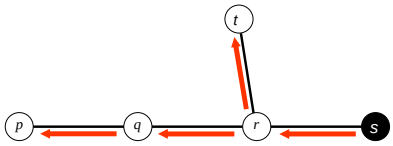
Forward pass (dynamic programming)



$$M_{rs}(k) = \min_j \{ (\theta_r(j) + M_{qr}(j) + M_{tr}(j)) + \theta_{rs}(j,k) \}$$

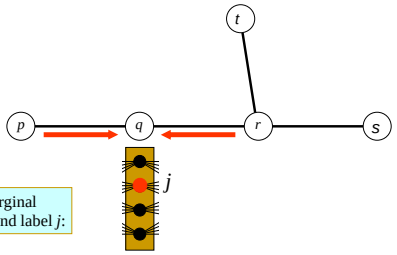
Slide credit: V. Kolmogorov

Backward pass



Slide credit: V. Kolmogorov

BP on a tree: min-marginals

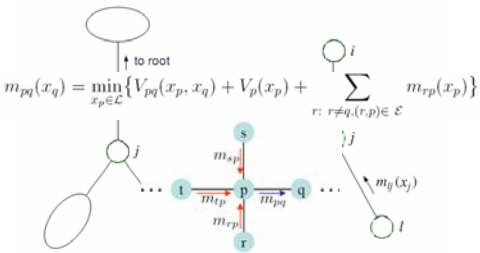


Min-marginal for node q and label j:

$$\min_{\mathbf{x}} \{ E(\mathbf{x}) \mid x_q = j \} = \theta_q(j) + M_{pq}(j) + M_{rq}(j)$$

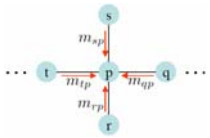
Slide credit: V. Kolmogorov

Belief propagation on a tree



Belief propagation on a tree

min-marginals = sum of all messages + unary



Message-passing as dynamic programming

- Essentially, message passing on trees is dynamic programming
- And essentially dynamic programming = reuse of computations

Message-passing as dynamic programming

$$\min_{\{x_i\}} \sum_{i=1}^{N} \theta_{i,i+1}(x_i, x_{i+1}) = \min_{x_N} \dots \min_{x_1} \left[\theta_{N-1,N}(x_{N-1}, x_N) + \dots + \theta_{1,2}(x_1, x_2)\right] =$$

Message-passing as dynamic programming

$$\min_{\{x_i\}} \sum_{i=1}^{N} \theta_{i,i+1}(x_i, x_{i+1}) = \min_{x_N} \dots \min_{x_1} \left[\theta_{N-1,N}(x_{N-1}, x_N) + \dots + \theta_{1,2}(x_1, x_2)\right] =$$

$$= \min_{x_N} \left[\theta_{N-1,N}(x_{N-1}, x_N) + \left[\dots \min_{x_1} \theta_{1,2}(x_1, x_2) \right] \right] =$$

message M_{12}

Message-passing as dynamic programming

$$\min_{\{x_i\}} \sum_{i=1}^{N} \theta_{i,i+1}(x_i, x_{i+1}) = \min_{x_N} \dots \min_{x_1} \left[\theta_{N-1,N}(x_{N-1}, x_N) + \dots + \theta_{1,2}(x_1, x_2)\right] =$$

$$= \min_{x_N} \left[\theta_{N-1,N}(x_{N-1}, x_N) + \left[\dots \min_{x_1} \theta_{1,2}(x_1, x_2) \right] \right] =$$

$$= \min_{x_N} \left[\theta_{N-1,N}(x_{N-1}, x_N) + \left[\dots \min_{x_2} \left[\theta_{2,3}(x_2, x_3) + M_{1,2}(x_2)\right] \right] \right] =$$

message M_{23}

Message-passing as dynamic programming

$$\min_{\{x_i\}} \sum_{i=1}^{N} \theta_{i,i+1}(x_i, x_{i+1}) = \min_{x_N} \dots \min_{x_1} \left[\theta_{N-1,N}(x_{N-1}, x_N) + \dots + \theta_{1,2}(x_1, x_2)\right] =$$

$$= \min_{x_N} \left[\theta_{N-1,N}(x_{N-1}, x_N) + \left[\dots \min_{x_1} \theta_{1,2}(x_1, x_2) \right] \right] =$$

$$= \min_{x_N} \left[\theta_{N-1,N}(x_{N-1}, x_N) + \left[\dots \min_{x_2} \left[\theta_{2,3}(x_2, x_3) + M_{1,2}(x_2)\right] \right] \right] =$$

$$= \dots$$

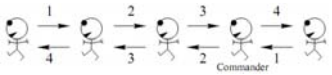
$$= \min_{x_N} \left[\theta_{N-1,N}(x_{N-1}, x_N) + M_{N-2,N-1}(x_{N-1})\right]$$

Generalizing belief propagation

- Key property: min(a+b,a+c) = a+min(b,c)
- BP can be generalized to any operators satisfying the above property
- E.g., instead of (min,+), we could have:
  - (max,*)
    Resulting algorithm called max-product.
    What does it compute?
  - (+,*)
    Resulting algorithm called sum-product.
    What does it compute?

Belief propagation as a distributive algorithm

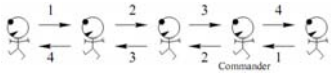
- BP works distributively
  (as a result, it can be parallelized)
- Essentially BP is a decentralized algorithm
- Global results through local exchange of information
- Simple example to illustrate this: counting soldiers



Counting soldiers in a line

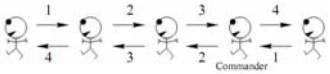
(From David MacKay's book "Information Theory, Inference, and Learning")

- Can you think of a distributive algorithm for the commander to count its soldiers?



Counting soldiers in a line

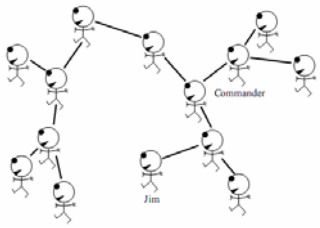
1. If you are the front soldier in the line, say the number 'one' to the soldier behind you.
2. If you are the rearmost soldier in the line, say the number 'one' to the soldier in front of you.
3. If a soldier ahead of or behind you says a number to you, add one to it, and say the new number to the soldier on the other side.



	the number said to him by the soldier in front of him,	(which equals the total number of soldiers in front)
+	the number said to the commander by the soldier behind him,	(which is the number behind)
+	one	(to count the commander himself).

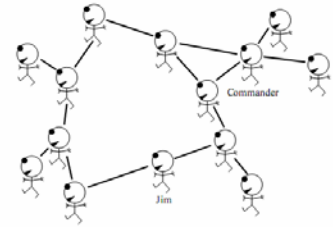
Counting soldiers in a tree

- Can we do the same for this case?



Graphs with loops

- How about counting these soldiers?



- Hmmm…overcounting?

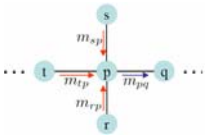
BP on graphs with loops

- What if the graph contains loops?
- Well…just pretend it is a tree and keep passing messages until convergence.
- Resulting algorithm called **Loopy Belief Propagation**

Loopy belief propagation (LBP)

- Messages from node p to q form a set $\{m_{pq}(x_q)\}_{x_q \in \mathcal{L}}$ with:

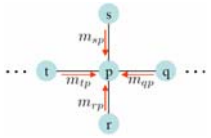
$$m_{pq}(x_q) = \min_{x_p \in \mathcal{L}} \{V_{pq}(x_p, x_q) + V_p(x_p) + \sum_{r:\ r \neq q, (r,p) \in \mathcal{E}} m_{rp}(x_p)\}$$



- Messages are circulated around the network until they stabilize (fixed-point)

Loopy belief propagation (LBP)

- After convergence, compute **pseudo min-marginals** for each node by summing up all incoming messages to that node



- To each node, assign the label whose pseudo min-marginal is the smallest
(**Question:** even if the graph is a tree, is this guaranteed to give you the optimal labeling?)

Loopy belief propagation (LBP)

- No guarantee that LBP computes the optimum
  - In some cases, it may not even converge
- Pseudo min-marginals
- Empirically, it works well in many cases
- Message-passing schedule
  - Parallel
  - Sequential

Extensions/generalizations

- BP for MRFs with higher order clicques
  - Factor graphs
- Exact optimum for loopy graphs
  - Junction tree algorithm
- Fast messages for specific class of MRFs [Felzenszwalb and Huttenlocher]
  - Time $O(|L|)$ instead of $O(|L|^2)$ ($|L|$ denotes the number of labels)
- many others… e.g., see [Wainwright], [Kolmogorov], [Yedidia et al.], [Weiss] etc.

Loopy belief propagation (LBP)

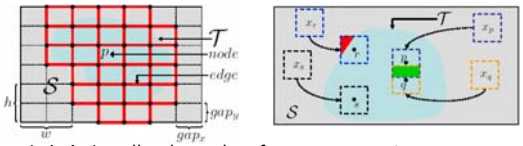
- Before proceeding let us very briefly look at a fancy application of BP on a challenging problem
- We will consider a specific case from vision and graphics.
- It is called the **"image completion problem"**

The Image Completion Problem

- Based only on the observed part of an incomplete image, fill its missing part in a visually plausible way

- We want to be able to handle:
  - complex natural images
  - with (possibly) large missing regions
  - in an automatic way (i.e. without user intervention)
- Many applications: photo editing, film post-production, object removal, text removal, image repairing etc.

Image Completion as a Discrete MRF Optimization Problem [Komodakis '06]



- **Labels L** = all wxh patches from source region S
- **MRF nodes** = all lattice points whose neighborhood intersects target region T
- **potential $V_p(x_p)$** = how well source patch x_p agrees with source region around p
- **potential $V_{pq}(x_p,x_q)$** = how well source patches x_p, x_q agree on their overlapping region

Image Completion as a Discrete MRF Optimization Problem

- Image completion reduces to finding labeling $\hat{x}$ with minimum total energy:

$$\mathcal{F}(\hat{x}) = \sum_{p \in \mathcal{V}} V_p(\hat{x}_p) + \sum_{(p,q) \in \mathcal{E}} V_{pq}(\hat{x}_p, \hat{x}_q)$$

- Intuitively, it's like assembling a huge jigsaw puzzle
- This formulation encompasses texture synthesis as well
  - E.g. to extend input texture T_0 to larger region T_1:
    set source region = T_0
    set target region = T_1-T_0

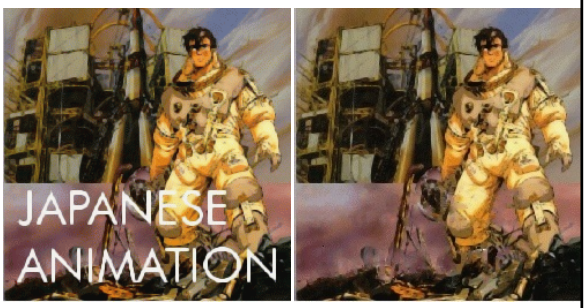
Priority-BP [Komodakis '06]

- In this case BP has an intolerable computational cost:
  - Just the basic operation of updating messages from node p to node q takes $O(|\mathcal{L}|^2)$ time
  - $|\mathcal{L}|^2$ SSD calculations between patches thus needed **(recall that |L| is huge in our case!)**
- Two extensions over standard-BP to reduce computation cost:
  - **"Dynamic label pruning"** and
  - **"Priority-based message scheduling"**

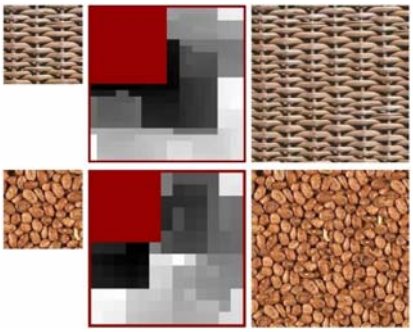




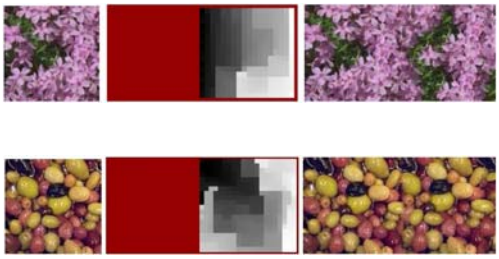




Texture synthesis results



Texture synthesis results



MRF optimization based on linear programming

MRF optimization based on linear programming

- We will now look at MRF optimization from a broader perspective
- To this end, we will rely on tools from convex programming and, in particular, linear programming
- Using such "tools", we will be able to generalize both:
  - Graph-cut based techniques
  - Message-passing techniques
- But first, a brief introduction into some basic, but very useful, concepts

Discrete optimization and convex relaxations

Introduction (1/2)

- As you saw, many problems in vision and pattern recognition can be formulated as discrete optimization problems:

$$\min_x f(x) \qquad \text{(optimize an objective function)}$$

$$\text{s.t. } x \in \mathcal{C} \qquad \text{(subject to some constraints)}$$

this is the so called **feasible set**, containing all x satisfying the constraints

- Typically x lives on a very high dimensional space

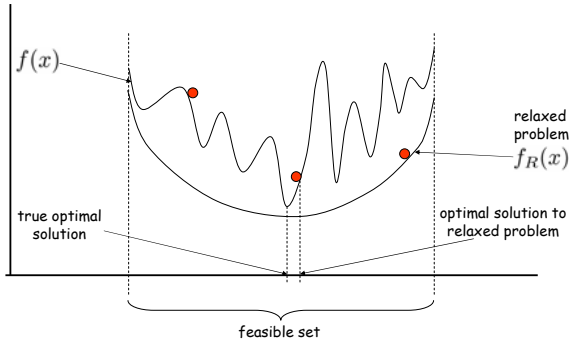
Introduction (2/2)

- Unfortunately, the resulting optimization problems are very often extremely hard (a.k.a. NP-hard)
  - E.g., feasible set or objective function highly non-convex
- So what do we do in this case?
  - Is there a principled way of dealing with this situation?
- Well, first of all, we don't need to panic. Instead, we have to stay calm and **RELAX!**
- Actually, this idea of relaxing turns out not to be such a bad idea after all…

The relaxation technique (1/2)

- Very successful technique for dealing with difficult optimization problems
- It is based on the following simple idea:
  - try to approximate your original difficult problem with another one (the so called **relaxed problem**) which is easier to solve
- Practical assumptions:
  - Relaxed problem must always be easier to solve
  - Relaxed problem must be related to the original one

The relaxation technique (2/2)



How do we find easy problems?

- Convex optimization to the rescue

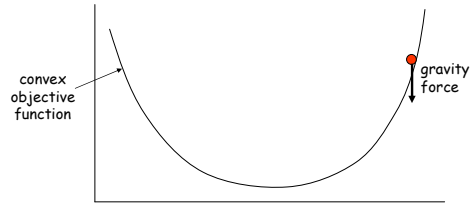
  "…in fact, the great watershed in optimization isn't between linearity and nonlinearity, but convexity and nonconvexity"

  - R. Tyrrell Rockafellar, in SIAM Review, 1993

- Two conditions must be met for an optimization problem to be convex:
  - its objective function must be convex
  - its feasible set must also be convex

Why is convex optimization easy?

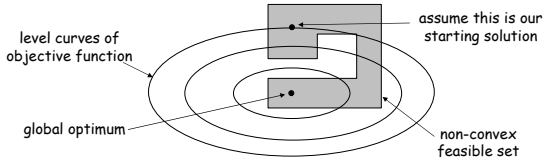
- Because we can simply let gravity do all the hard work for us



- More formally, we can let gradient descent do all the hard work for us

Why do we need the feasible set to be convex as well?

- Because, otherwise we may get stuck in a local optimum if we simply "follow" gravity



How do we get a convex relaxation?

- By dropping some constraints (so that the enlarged feasible set is convex)
- By modifying the objective function (so that the new function is convex)
- By combining both of the above

Linear programming (LP) relaxations

- Optimize a linear function subject to linear constraints, i.e.:

$$\begin{aligned} &\min \mathbf{c}^T\mathbf{x} \\ &\text{s.t. } \mathbf{A}\mathbf{x} = \mathbf{b} \end{aligned}$$

- Very common form of a convex relaxation
- Typically leads to very efficient algorithms
- Also often leads to combinatorial algorithms
- This is the kind of relaxation we will use for the case of MRF optimization

The "big picture" and the road ahead (1/2)

- As we shall see, MRF can be cast as a linear integer program (very hard to solve)
- We will thus approximate it with a LP relaxation (much easier problem)
- **Critical question:** How do we use the LP relaxation to solve the original MRF problem?

The "big picture" and the road ahead (2/2)

- We will next describe two general techniques for that:
  - **Primal-dual schema**
    doesn't try to solve LP-relaxation exactly
    (leads to **graph-cut** based algorithms)
  - **Rounding**
    - Tries to solve LP-relaxation exactly **(dual decomposition)**
    - Leads to **message-passing** algorithms

MRF optimization via dual-decomposition

Revisiting our strategy to MRF optimization

- We will now follow a different strategy: we will try to optimize an MRF by first solving its LP-relaxation.
- As we shall see, this will lead to a message passing method for MRF optimization
- Actually, resulting method solves the dual to the LP-relaxation
  - but this is equivalent to solving the LP, as there is no duality gap due to convexity
- Maximization of this dual LP is also the driving force behind all tree-reweighted message passing methods [Wainwright05][Kolmogorov06]

  (however, TRW methods cannot guarantee that the maximum is attained)

MRF optimization via dual-decomposition

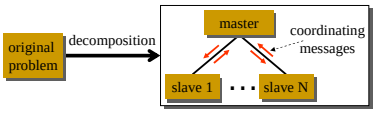
- New framework for understanding/designing message-passing algorithms [Komodakis et al. '07]
- Stronger theoretical properties than state-of-the-art
- New insights into existing message-passing techniques
- Reduces MRF optimization to a simple **projected subgradient** method
  - very well studied topic in optimization, i.e., with a vast literature devoted to it (see also [Schlesinger & Giginyak07])
- Its theoretical setting rests on the very powerful technique of **Dual Decomposition** and thus offers extreme generality and flexibility .

Decomposition

- Very successful and widely used technique in optimization.
- The underlying idea behind this technique is surprisingly simple (and yet extremely powerful):
  - decompose your difficult optimization problem into easier subproblems (these are called the **slaves**)
  - extract a solution by cleverly combining the solutions from these subproblems (this is done by a so called **master** program)

Dual decomposition

- The role of the master is simply to coordinate the slaves via messages



- Depending on whether the primal or a Lagrangian dual problem is decomposed, we talk about **primal** or **dual** decomposition respectively

An illustrating toy example (1/4)

- For instance, consider the following optimization problem (where **x** denotes a vector):

$$\begin{array}{ll}\min_x & \sum_i f^i(x) \\ \text{s.t.} & x \in \mathcal{C}\end{array}$$

- We assume minimizing each $f^i(\cdot)$ separately is easy, but minimizing their sum $\sum_i f^i(\cdot)$ is hard.
- To apply dual decomposition, we will use multiple copies $\mathbf{x}^i$ of the original variables **x**
- Via these auxiliary variables $\{x^i\}$, we will thus transform our problem into:

$$\begin{array}{ll}\min_{\{x^i\},x} & \sum_i f^i(x^i) \\ \text{s.t.} & x^i \in \mathcal{C},\ \boxed{x^i = x}\end{array}$$

An illustrating toy example (2/4)

- If coupling constraints $x^i = x$ were absent, problem would decouple. We thus relax them (via Lagrange multipliers $\{\lambda^i\}$) and form the following Lagrangian dual function:

$$g(\{\lambda^i\}) = \min_{\{x^i \in \mathcal{C}\},x} \sum_i f^i(x^i) + \sum_i \lambda^i \cdot (x^i - x)$$

Last equality assumes $\{\lambda^i\} \in \Lambda = \{\{\lambda^i\} | \sum_i \lambda^i = 0\}$ because otherwise it holds $g(\{\lambda^i\}) = -\infty$

- The resulting dual problem (i.e., the maximization of the Lagrangian) is now decoupled! Hence, the decomposition principle can be applied to it!

An illustrating toy example (3/4)

- The **i-th slave problem** obviously reduces to:

$$g^i(\lambda^i) = \min_{x^i \in \mathcal{C}} f^i(x^i) + \lambda^i \cdot x^i$$

Easily solved by assumption. Responsible for updating only x^i, set equal to $\bar{x}^i(\lambda^i) \equiv$ minimizer of i-th slave problem for given λ^i

- The **master problem** thus reduces to:

$$\max_{\{\lambda^i\} \in \Lambda} g(\{\lambda^i\}) = \sum_i g^i(\lambda^i)$$

This is the Lagrangian dual problem, responsible to update $\{\lambda^i\}$
Always convex, hence solvable by projected subgradient method:

$$\lambda^i \leftarrow [\lambda^i + \alpha_t \nabla g^i(\lambda^i)]_\Lambda$$

$\nabla \equiv$ subgradient w.r.t. λ^i
$[\cdot]_\Lambda \equiv$ projection on feasible set Λ

In this case, it is easy to check that: $\nabla g^i(\lambda^i) = \bar{x}^i(\lambda^i)$

An illustrating toy example (4/4)

- **The master-slaves communication then proceeds as follows:**

  1. Master sends current $\{\lambda^i\}$ to the slaves

  2. Slaves respond to the master by solving their easy problems and sending back to him the resulting minimizers $\bar{x}^i(\lambda^i)$

  3. Master updates each λ^i by setting $\lambda^i \leftarrow \left[\lambda^i + \alpha_t \bar{x}^i(\lambda^i)\right]_\Lambda$

  (Steps 1, 2, 3 are repeated until convergence)

Optimizing MRFs via dual decomposition

We can apply a similar idea to the problem of MRF optimization, which can be cast as a linear integer program:

$$\min_{\mathbf{x}} \; E(\boldsymbol{\theta}) \cdot \mathbf{x} = \sum_{a \in \mathcal{L}} \theta_p(a) x_p(a) + \sum_{a,b \in \mathcal{L}} \theta_{pq}(a,b) x_{pq}(a,b)$$

$$\text{s.t.} \quad \mathbf{x} \in \mathcal{X}$$

$$\text{s.t.} \quad \sum_{a \in \mathcal{L}} x_p(a) = 1, \quad \forall p \in \mathcal{V} \qquad \text{(only one label assigned per vertex)}$$

$$\sum_{b \in \mathcal{L}} x_{pq}(a,b) = x_p(b), \quad \forall pq \in \mathcal{E}, \forall b \in \mathcal{L}$$

$$\sum_{a \in \mathcal{L}} x_{pq}(a,b) = x_q(b), \quad \forall pq \in \mathcal{E}, \forall b \in \mathcal{L}$$

enforce consistency between variables x_p, x_q and x_{pq}

$\theta = \{\{\theta_p\}, \{\theta_{pq}\}\}$ is the vector of MRF-parameters consisting of all unary $\theta_p = \{\theta_p(\cdot)\}$ and pairwise $\theta_{pq} = \{\theta_{pq}(\cdot,\cdot)\}$ vectorized potentials.

$\mathbf{x} = \{\{x_p\}, \{x_{pq}\}\}$ is the vector of binary MRF-variables consisting of unary subvectors $\mathbf{x}_p = \{x_p(\cdot)\}$ and pairwise subvectors $\mathbf{x}_{pq} = \{x_{pq}(\cdot,\cdot)\}$

Binary variables

$x_p(a) = 1 \Leftrightarrow$ label a is assigned to node p

$x_{pq}(a,b) = 1 \Leftrightarrow$ labels a, b are assigned to p, q

Constraints enforce consistency between variables $\{\mathbf{x}_p\}$ and $\{\mathbf{x}_{pq}\}$

Optimizing MRFs via dual decomposition

- We will again introduce multiple copies of the original variables (one copy per subgraph, e.g., per tree)
  - $\mathbf{x}^T$ denotes the variables associated to subgraph T
- We will split the original potentials into the subgraphs, i.e.:
  $\sum_{T \in \mathcal{T}(p)} \theta_p^T = \theta_p, \qquad \sum_{T \in \mathcal{T}(pq)} \theta_{pq}^T = \theta_{pq},$
  (Here θ^T denotes the potentials associated to subgraph T)
  (While $\mathcal{T}(p), \mathcal{T}(pq)$ denote all subgraphs in $\mathcal{T}$ containing respectively p and pq)
- MRF problem then transforms into:
  $$\min_{\{\mathbf{x}^T\}, \mathbf{x}} \; \sum_{T \in \mathcal{T}} E(\boldsymbol{\theta}^T, \mathbf{x}^T)$$
  $$\text{s.t.} \quad \mathbf{x}^T \in \mathcal{X}^T, \quad \forall T \in \mathcal{T}$$
  $$\mathbf{x}^T = \mathbf{x}_{|T}, \quad \forall T \in \mathcal{T}$$
  (this problem is thus similar to the toy example shown earlier)

So, who are the slaves?

- One possible choice is that the slave problems are tree-structured MRFs.
- To each tree T from a set of trees $\mathcal{T}$, we can associate a slave MRF with parameters θ^T
- These parameters must initially satisfy:
  $\sum_{T \in \mathcal{T}(p)} \theta_p^T = \theta_p, \qquad \sum_{T \in \mathcal{T}(pq)} \theta_{pq}^T = \theta_{pq},$
  (Here $\mathcal{T}(p), \mathcal{T}(pq)$ denote all trees in $\mathcal{T}$ containing respectively p and pq)
- Note that the slave-MRFs are easy problems to solve, e.g., via max-product.

And who is the master?

- In this case the master problem can be shown to coincide with the LP relaxation considered earlier.
- To be more precise, the master tries to optimize the dual to that LP relaxation (which is the same thing)
- In fact, the role of the master is to simply adjust the parameters of all slave-MRFs such that this dual is optimized (i.e., maximized).

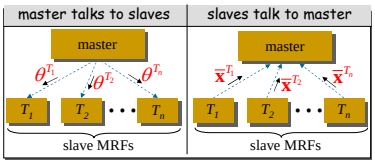
"I am at you service, Sir…" (or how are the slaves to be supervised?)

- The coordination of the slaves by the master turns out to proceed as follows:
  - Master sends current parameters $\{\theta^T\}$ to slave-MRFs and requests the slaves to "optimize" themselves based on the MRF-parameters that he had sent.
  - Slaves "obey" to the master by minimizing their energy and sending back to him the new tree-minimizers $\{\bar{\mathbf{x}}^T\}$
  - Based on all collected minimizers, master readjusts the parameters of each slave MRF (i.e., of each tree T):
    $$\theta_p^T \mathrel{+}= \alpha_t \cdot \left(\bar{\mathbf{x}}_p^T - \frac{\sum_{T' \in \mathcal{T}(p)} \bar{\mathbf{x}}_p^{T'}}{|\mathcal{T}(p)|} \right), \quad \theta_{pq}^T \mathrel{+}= \alpha_t \cdot \left(\bar{\mathbf{x}}_{pq}^T - \frac{\sum_{T' \in \mathcal{T}(pq)} \bar{\mathbf{x}}_{pq}^{T'}}{|\mathcal{T}(pq)|} \right)$$

"What is it that you seek, Master?..."

- Master updates the parameters of the slave-MRFs by "averaging" the solutions returned by the slaves.
- Essentially, he tries to achieve consensus among all slave-MRFs
  - This means that tree-minimizers should agree with each other, i.e., assign same labels to common nodes
- For instance, if a node is already assigned the same label by all tree-minimizers, the master does not touch the MRF potentials of that node.

"What is it that you seek, Master?..."



Economic interpretation:

- Think of $\{\bar{\mathbf{x}}^T\}$ as amount of resources consumed by slave-MRFs
- Think of $\{\theta^T\}$ as corresponding prices
- Master naturally adjusts prices as follows:
  - prices for **overutilized** resources are **increased**
  - prices for **underutilized** resources are **decreased**

Algorithmic properties

- Guaranteed convergence
- Provably optimizes LP-relaxation (unlike existing tree-reweighted message passing algorithms)
  - In fact, distance to optimum is guaranteed to decrease per iteration

Algorithmic properties

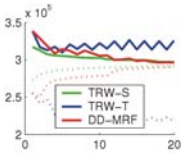
- Generalizes Weak Tree Agreement (WTA) condition introduced by V. Kolmogorov
  - Computes optimum for binary submodular MRFs
- Extremely general and flexible framework
  - Slave-MRFs need not be tree-structured (exactly the same framework still applies)

Results

- Resulting algorithm is called DD-MRF
- It has been applied to:
  - stereo matching
  - optical flow
  - binary segmentation
  - synthetic problems
- Lower bounds produced by the master certify that solutions are almost optimal

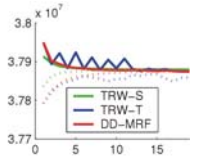
Results

estimated disparity for Tsukuba stereo pair



lower bounds (dual costs) and MRF energies (primal costs)

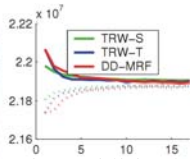
estimated disparity for Map stereo pair



lower bounds (dual costs) and MRF energies (primal costs)

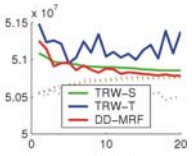
Results

estimated disparity for SRI stereo pair



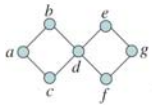
lower bounds (dual costs) and MRF energies (primal costs)

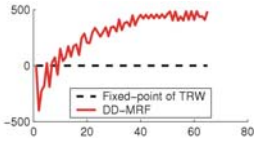
estimated optical flow for Yosemite sequence



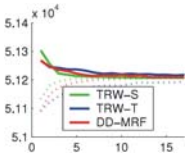
lower bounds (dual costs) and MRF energies (primal costs)

Results





a simple synthetic example illustrating that TRW methods are not able to maximize the dual lower bound, whereas DD-MRF can.

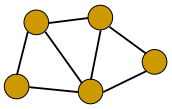


lower bounds (dual costs) and MRF energies (primal costs) for binary segmenation

MRF optimization via the primal-dual schema

The MRF problem (a recap)

- vertices G = set of objects
  edges E = object relationships
  set L = discrete set of labels

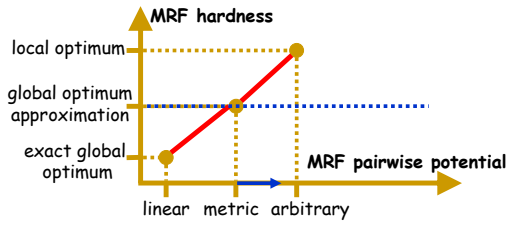


- $V_p(x_p)$ = cost of assigning label x_p to vertex p
  (also called **unary potential**)
- $V_{pq}(x_p,x_q)$ = cost of assigning labels (x_p,x_q) to neighboring vertices (p,q) (also called **pairwise potential**)
- Find labels that minimize the MRF energy (i.e., the sum of all potentials): $\min_{\{x_p\}} \sum_{p\in G} V_p(x_p) + \sum_{pq\in E} V_{pq}(x_p, x_q)$

The MRF problem (a recap)

- MRFs ubiquitous in vision and beyond
- Have been used in a wide range of problems:
  segmentation — stereo matching
  optical flow — image restoration
  image completion — object detection & localization
  ...
- MRF optimization is thus a task of fundamental importance but very **hard** to solve in general.

MRF hardness



- Move right in the horizontal axis,
  and remain low in the vertical axis
  (i.e., still be able to provide approximately optimal solutions)
- But we want to be able to do that efficiently, i.e. fast

Contributions to MRF optimization

General framework for optimizing MRFs based on duality theory of Linear Programming (the Primal-Dual schema) [Komodakis et al. '05, '07]

- Can handle a very wide class of MRFs
- Can guarantee approximately optimal solutions (worst-case theoretical guarantees)
- Can provide tight certificates of optimality per-instance (per-instance guarantees)
- Provides significant speed-up for static MRFs
- Provides significant speed-up for dynamic MRFs

→ **Fast-PD**

The primal-dual schema

- Highly successful technique for exact algorithms. Yielded exact algorithms for cornerstone combinatorial problems:

matching	network flow
minimum spanning tree	minimum branching
shortest path	...

- Soon realized that it's also an extremely powerful tool for deriving approximation algorithms [Vazirani]:

set cover	steiner tree
steiner network	feedback vertex set
scheduling	...

The primal-dual schema

- **Conjecture:**

Any approximation algorithm can be derived using the primal-dual schema

(the above conjecture has not been disproved yet)

The primal-dual schema

- Say we seek an optimal solution x^* to the following integer program (this is our **primal** problem):

$$\min \mathbf{c}^T\mathbf{x} \quad \text{s.t.} \quad \mathbf{Ax} = \mathbf{b},\ \mathbf{x} \in \mathbb{N}$$

← (NP-hard problem)

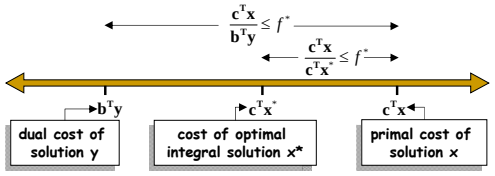
- To find an approximate solution, we first relax the integrality constraints to get a primal & a dual linear program:

primal LP: $\min \mathbf{c}^T\mathbf{x}$ s.t. $\mathbf{Ax} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0}$

dual LP: $\max \mathbf{b}^T\mathbf{y}$ s.t. $\mathbf{A}^T\mathbf{y} \leq \mathbf{c}$

The primal-dual schema

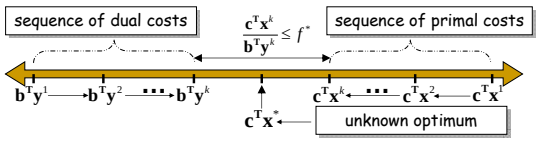
- Goal: find integral-primal solution x, feasible dual solution y such that their primal-dual costs are "close enough", e.g.,



Then x is an f^*-approximation to optimal solution x^*

The primal-dual schema

- The primal-dual schema works iteratively



- Global effects, through local improvements!
- Instead of working directly with costs (usually not easy), use RELAXED complementary slackness conditions (easier)
- Different relaxations of complementary slackness
  Different approximation algorithms!!!

The primal-dual schema for MRFs

$$\min \left[\sum_{p \in G} \sum_{a \in L} V_p(a) x_{p,a} + \sum_{pq \in E} \sum_{a,b \in L} V_{pq}(a,b) x_{pq,ab} \right]$$

$$\text{s.t.} \sum_{a \in L} x_{p,a} = 1$$

(only one label assigned per vertex)

$$\sum_{a \in L} x_{pq,ab} = x_{q,b}$$
$$\sum_{b \in L} x_{pq,ab} = x_{p,a}$$

enforce consistency between variables $x_{p,a}$, $x_{q,b}$ and variable $x_{pq,ab}$

$$x_{p,a} \geq 0, \; x_{pq,ab} \geq 0$$

Binary variables:

- $x_{p,a}=1 \Leftrightarrow$ label a is assigned to node p
- $x_{pq,ab}=1 \Leftrightarrow$ labels a, b are assigned to nodes p, q

The primal-dual schema for MRFs

- During the PD schema for MRFs, it turns out that:
  - each update of primal and dual variables ⟷ solving max-flow in appropriately constructed graph
- Resulting flows tell us how to update both:
  - the dual variables, as well as
  - the primal variables

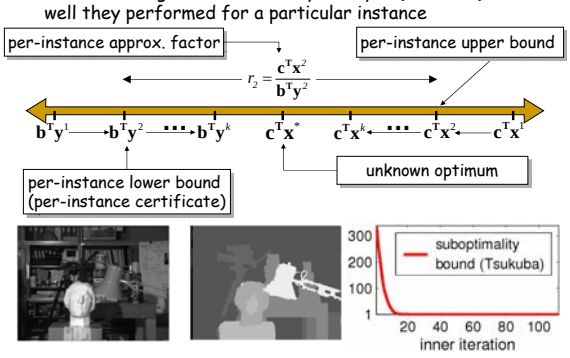
  ← for each iteration of primal-dual schema
- Max-flow graph defined from current primal-dual pair (x^k,y^k)
  - (x^k,y^k) defines **connectivity** of max-flow graph
  - (x^k,y^k) defines **capacities** of max-flow graph
- Max-flow graph is thus continuously updated

The primal-dual schema for MRFs

- Very general framework. Different PD-algorithms by RELAXING complementary slackness conditions differently.
- E.g., simply by using a particular relaxation of complementary slackness conditions (and assuming $V_{pq}(\cdot,\cdot)$ is a metric) THEN **resulting algorithm shown equivalent to α-expansion! [Boykov, Veksler, Zabih]**
- PD-algorithms for non-metric potentials $V_{pq}(\cdot,\cdot)$ as well
- **Theorem:** All derived PD-algorithms shown to satisfy certain relaxed complementary slackness conditions
- Worst-case optimality properties are thus **guaranteed**

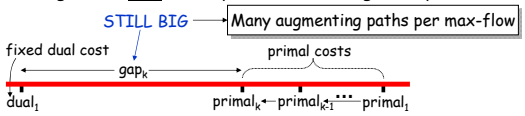
Per-instance optimality guarantees

- Primal-dual algorithms can always tell you **(for free)** how well they performed for a particular instance

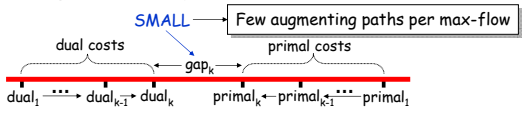


Computational efficiency (static MRFs)

- MRF algorithm only in the primal domain (e.g., α-expansion)

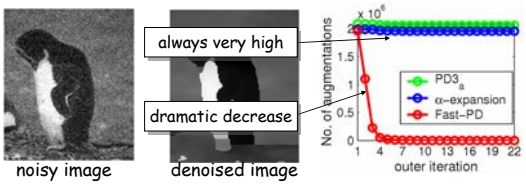


- MRF algorithm in the primal-dual domain (Fast-PD)



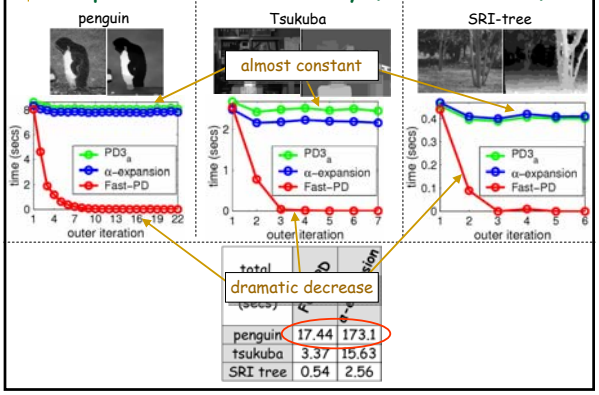
Theorem: primal-dual gap = upper-bound on #augmenting paths (i.e., primal-dual gap indicative of time per max-flow)

Computational efficiency (static MRFs)



- Incremental construction of max-flow graphs (recall that max-flow graph changes per iteration)
- This is possible only because we keep both **primal** and **dual** information
- Our framework provides a principled way of doing this incremental graph construction for general MRFs

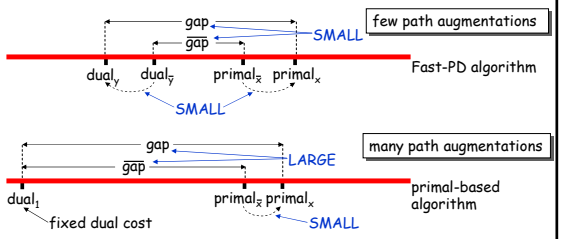
Computational efficiency (static MRFs)



	total	PD3a	α-expansion
penguin	17.44	173.1	
tsukuba	3.37	15.63	
SRI tree	0.54	2.56	

Computational efficiency (dynamic MRFs)

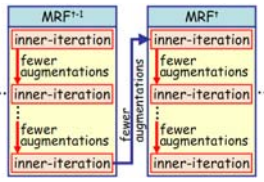
- Fast-PD can speed up dynamic MRFs [Kohli,Torr] as well (demonstrates the power and generality of our framework)



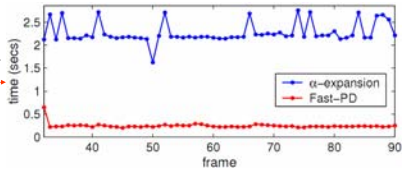
- It provides principled (and simple) way to update dual variables when switching between different MRFs

Computational efficiency (dynamic MRFs)

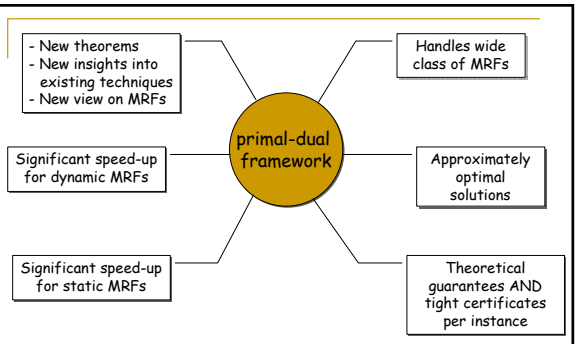
- Essentially, Fast-PD works along 2 different "axes"
  - reduces augmentations across different iterations of the same MRF
  - reduces augmentations across different MRFs



- Time per frame for SRI-tree stereo sequence



- Handles general (multi-label) dynamic MRFs



Take home messages

1. **Convex relaxations provide a principled way for tackling the MRF optimization problem**

2. **Duality provides a very valuable tool in this case**

Thank you!