Discrete optimization methods in computer vision
CSE-6389

Mostafa Parchami

Advanced Machine Learning

Slides By: N. Komodakis

N. Komodakis, N. Paragios, G. Tziritas
MRF Optimization via Dual Decomposition: Message-Passing Revisited
ICCV'07

Introduction:
Discrete optimization and convex relaxations

Introduction (1/2)

- Many problems in vision and pattern recognition can be formulated as discrete optimization problems:

$$\min_x f(x)$$ (optimize an objective function)

$$\text{s.t. } x \in \mathcal{C}$$ (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…

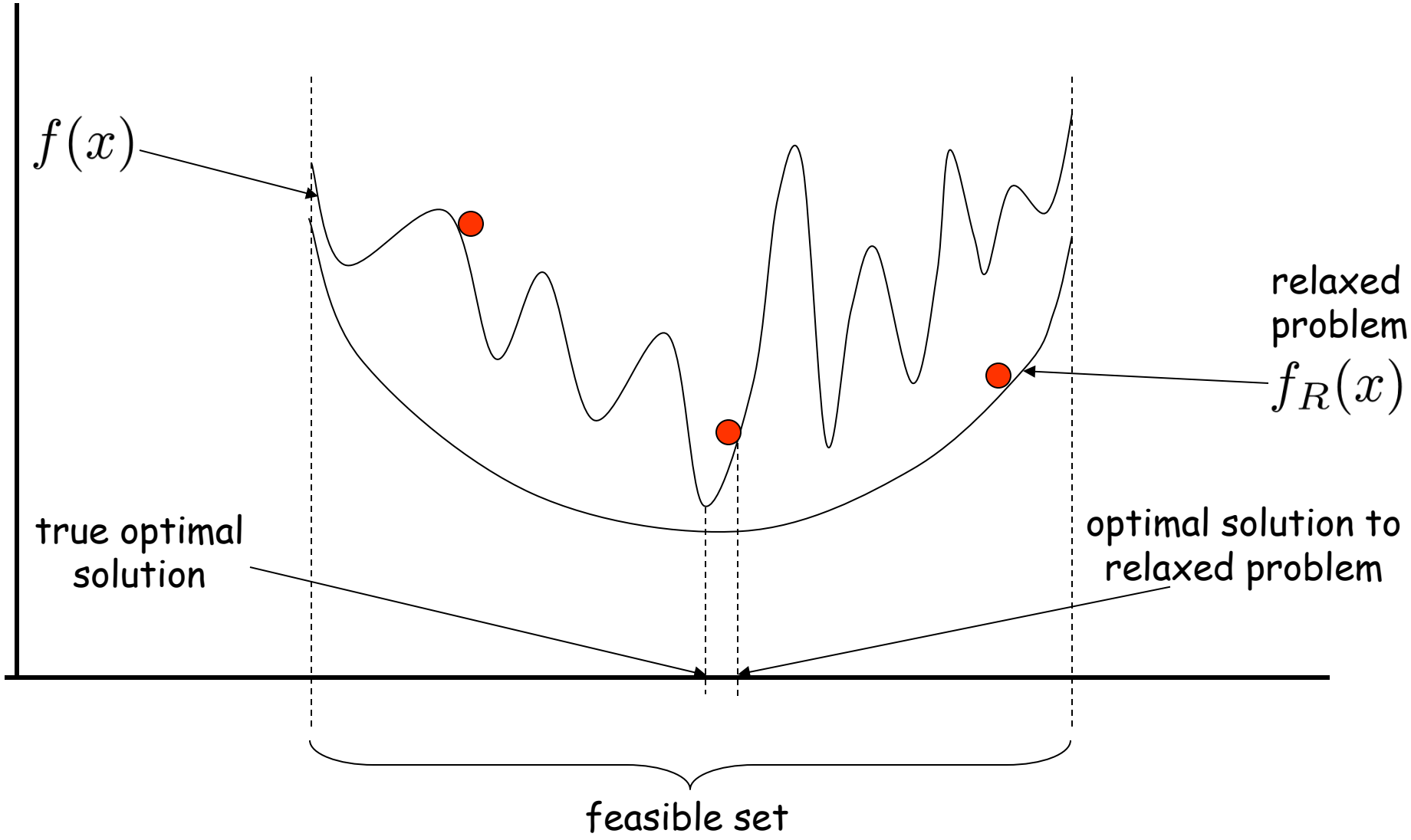
The relaxation technique (1/2)

- 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

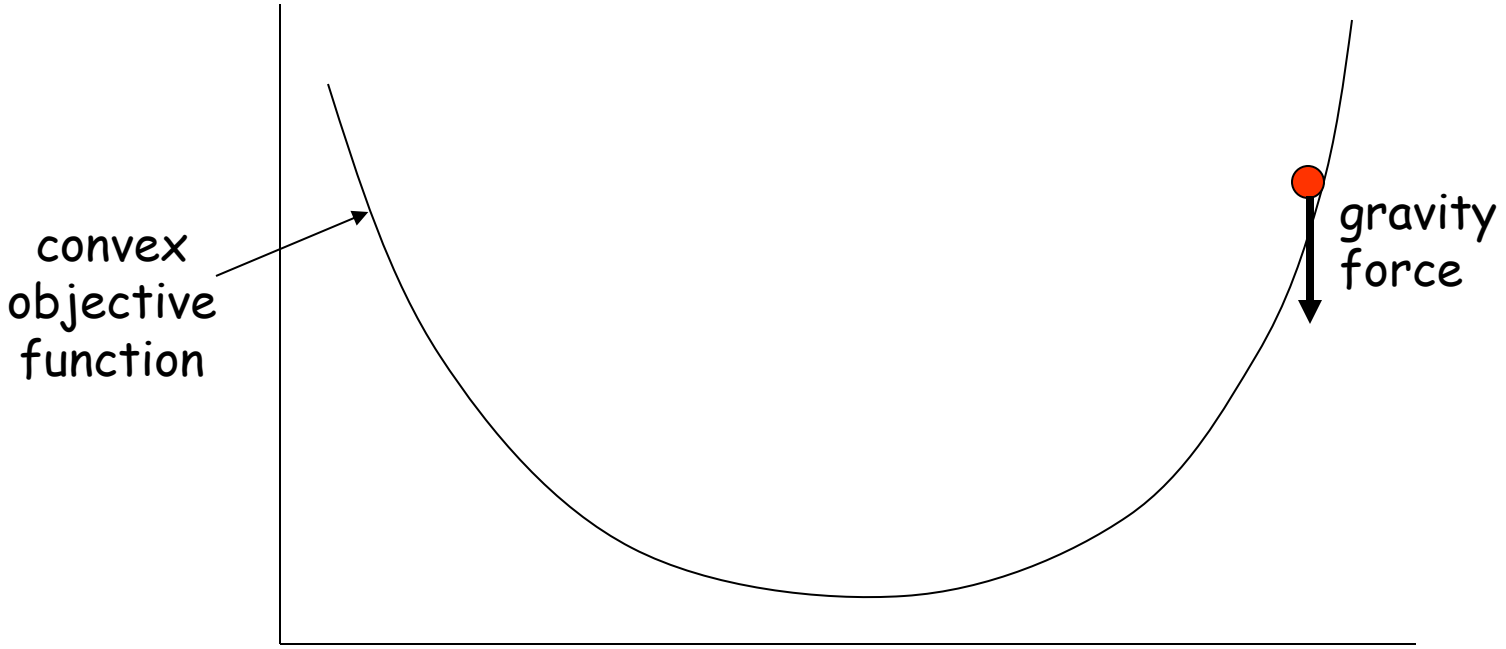
  "…in fact, the great watershed in optimization isn't between linearity and nonlinearity, but convexity and nonconvexity"

  – 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?

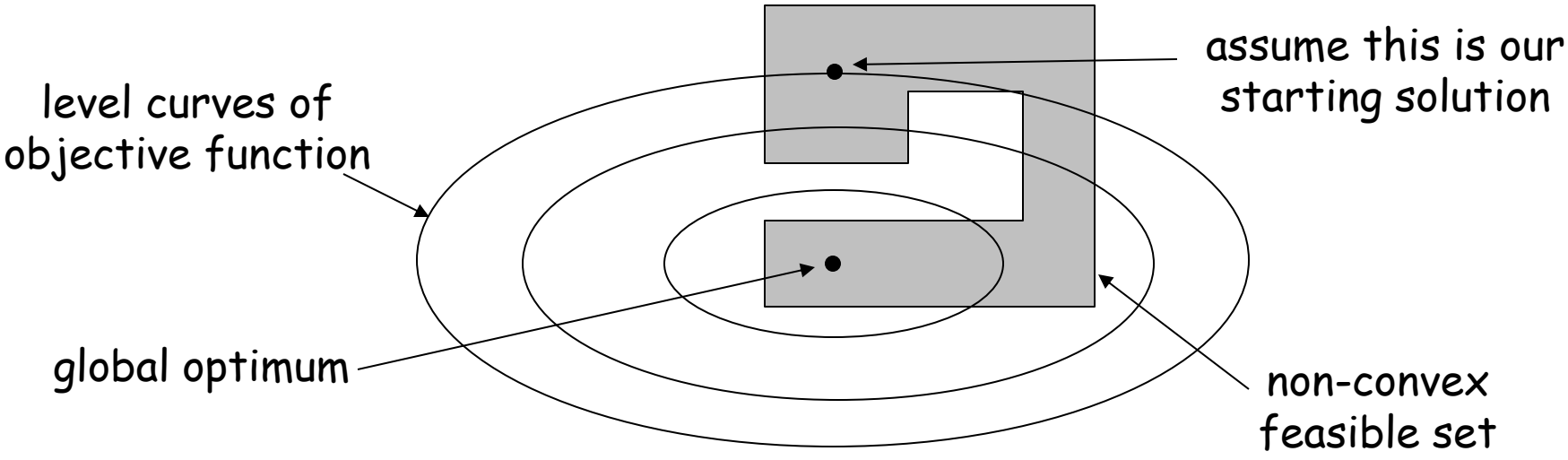
- Because we can simply let gravity do all the hard work for us



- 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.:

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

- 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?

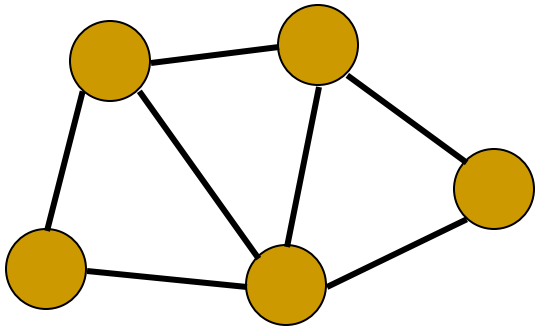
The "big picture" and the road ahead (2/2)

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

Part I:
MRF optimization via the primal-dual schema

The MRF optimization problem

- 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 **single node 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): $\min_{\{x_p\}} \sum_{p \in G} V_p(x_p) + \sum_{pq \in E} V_{pq}(x_p, x_q)$

MRF optimization in vision

- MRFs ubiquitous in vision and beyond

- Have been used in a wide range of problems:

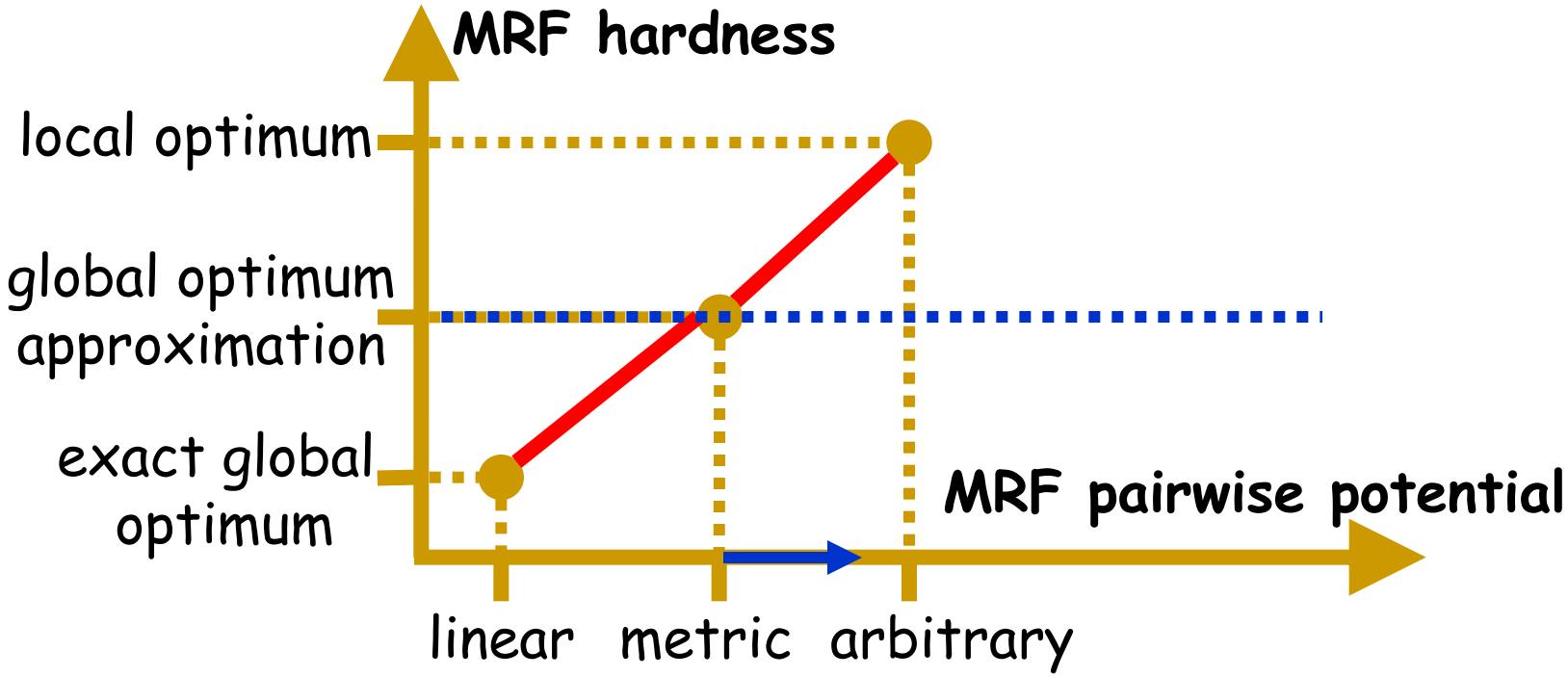
  segmentation — stereo matching
  optical flow — image restoration
  image completion — object detection & localization
  ...

- MRF optimization is thus a task of fundamental importance

- Yet, highly non-trivial, since almost all interesting MRFs are actually NP-hard to optimize

- Many proposed algorithms (e.g., [Boykov,Veksler,Zabih], [V. Kolmogorov], [Kohli,Torr], [Wainwright]...)

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)

- 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:

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

The primal-dual schema

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

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

(NP-hard problem)

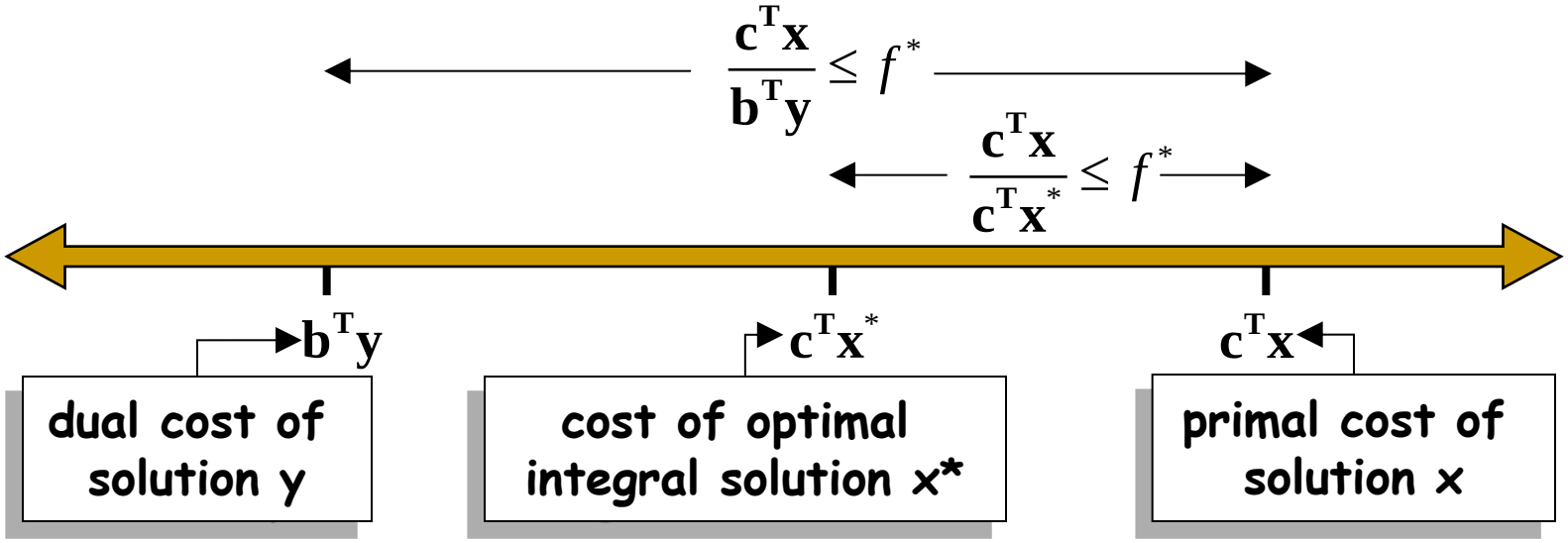
- To find an approximate solution, we first relax the integrality constraints to get a primal & a dual linear program:

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

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

The primal-dual schema

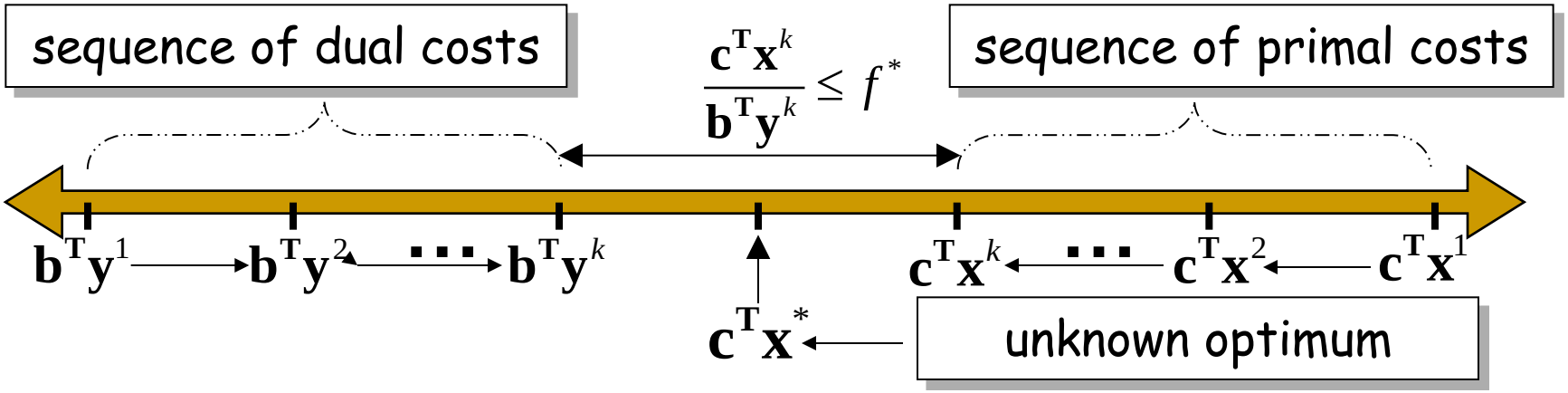
- Goal: find integral-primal solution x, feasible dual solution y such that their primal-dual costs are "close enough", e.g.,



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 $x_{pq,ab}$

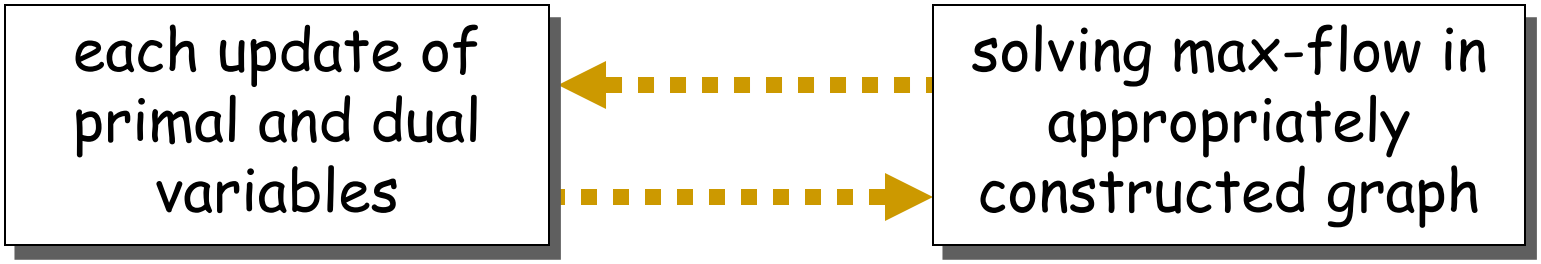
$$x_{p,a} \geq 0, \; x_{pq,ab} \geq 0$$

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:



- 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 a-expansion!**

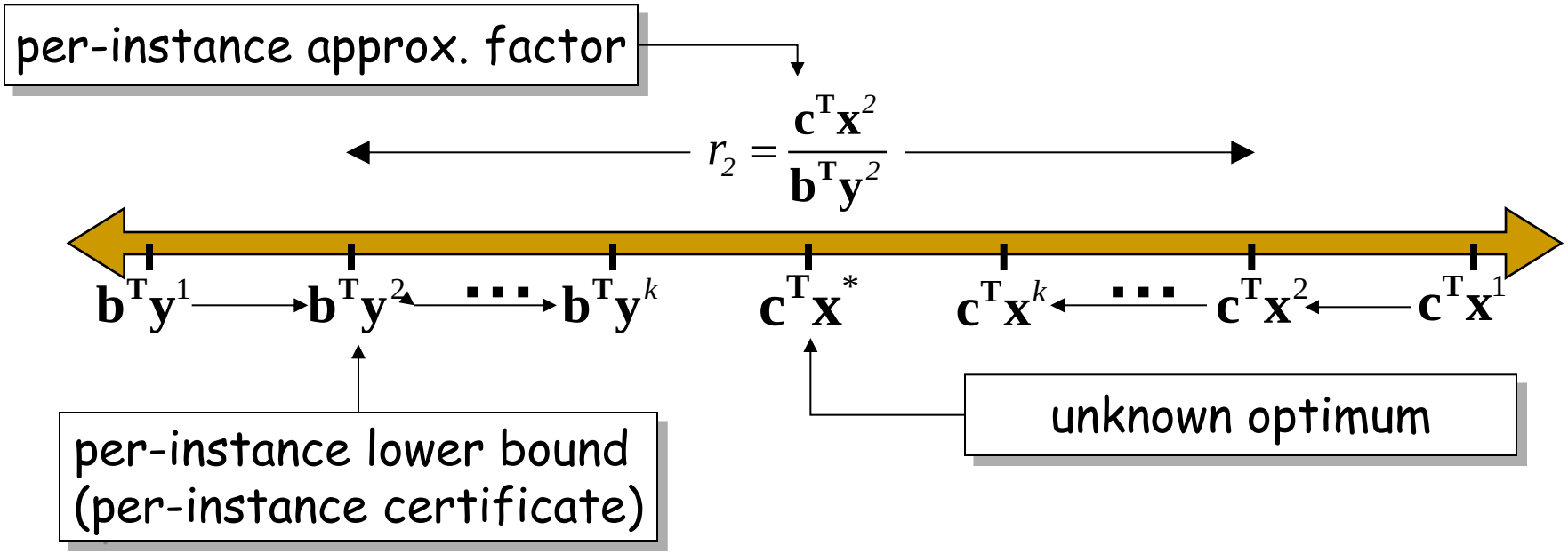
- PD-algorithms for non-metric potentials $V_{pq}(\cdot,\cdot)$ as well

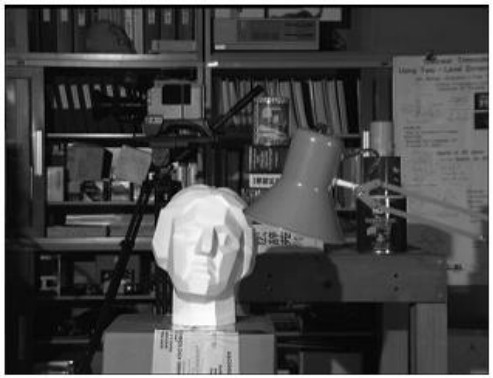
- **Theorem:** All derived PD-algorithms shown to satisfy certain relaxed complementary slackness conditions

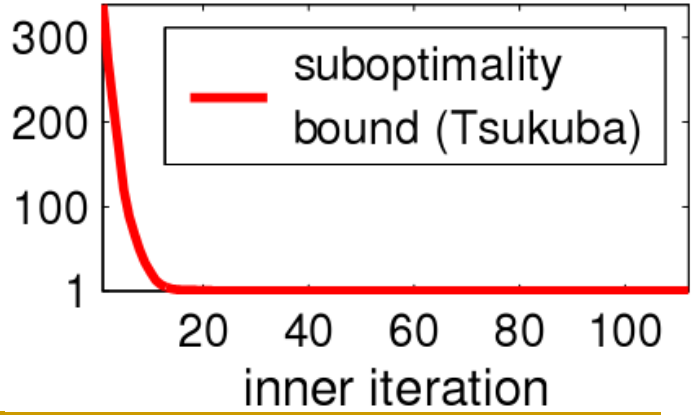
- Worst-case optimality properties are thus **guaranteed**

Per-instance optimality guarantees

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

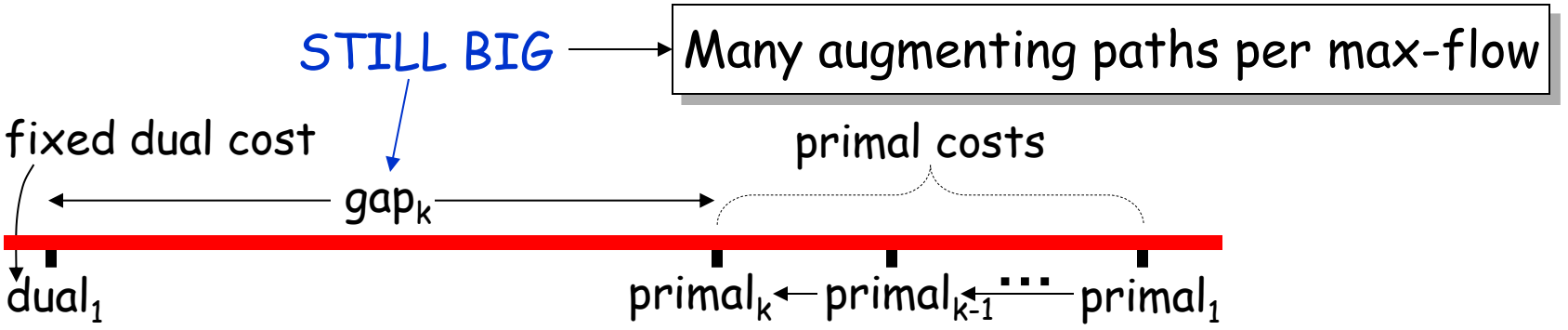




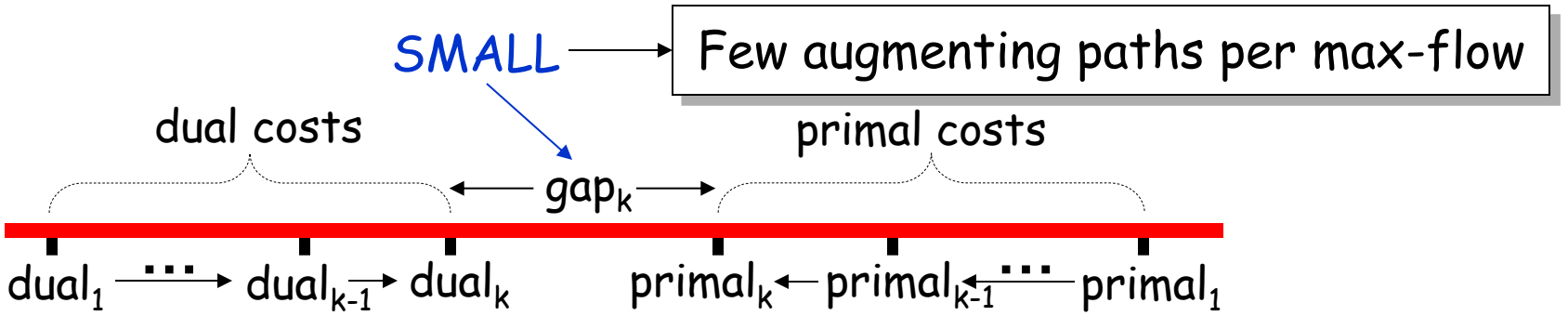


Computational efficiency (static MRFs)

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



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

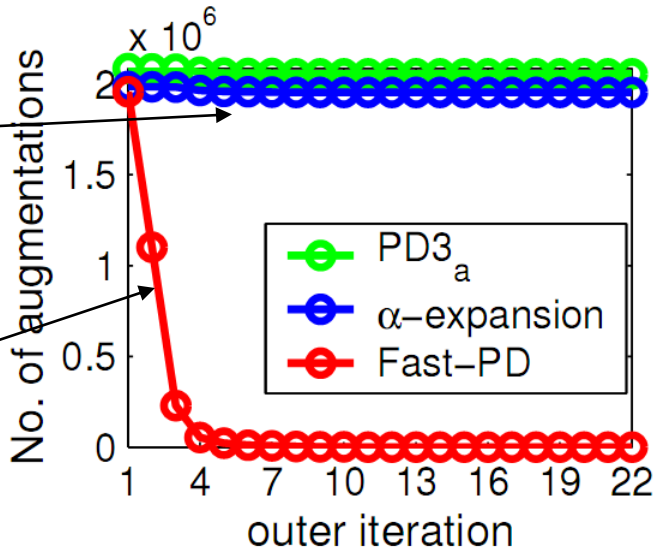


Theorem: primal-dual gap = upper-bound on #augmenting paths (i.e., primal-dual gap indicative of time per max-flow)

Computational efficiency (static MRFs)

noisy image

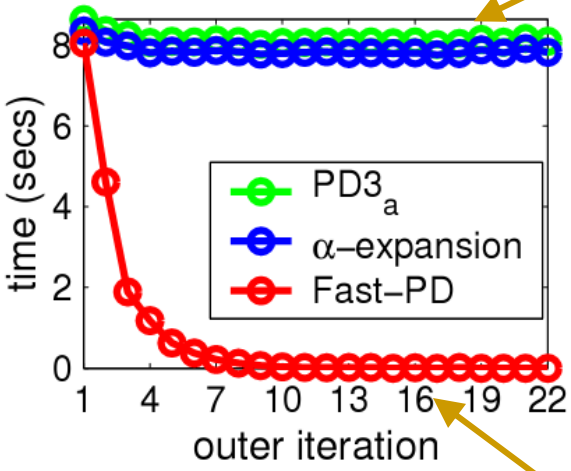
denoised image



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

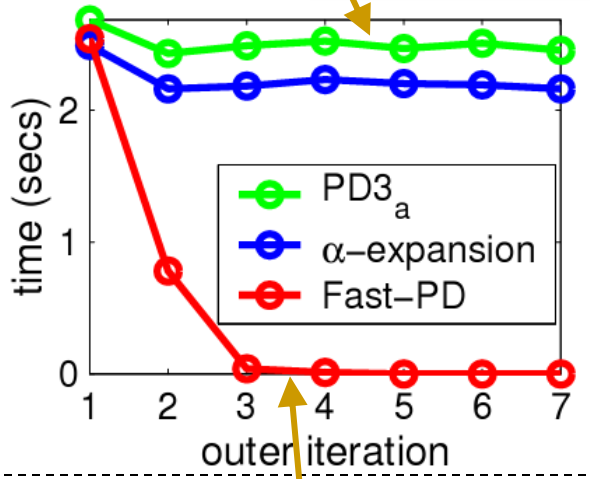
Computational efficiency (static MRFs)

penguin

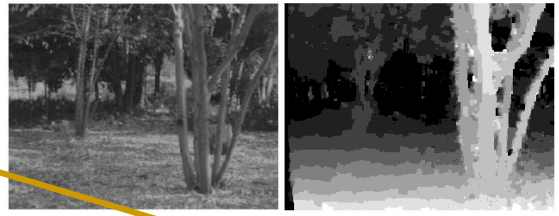


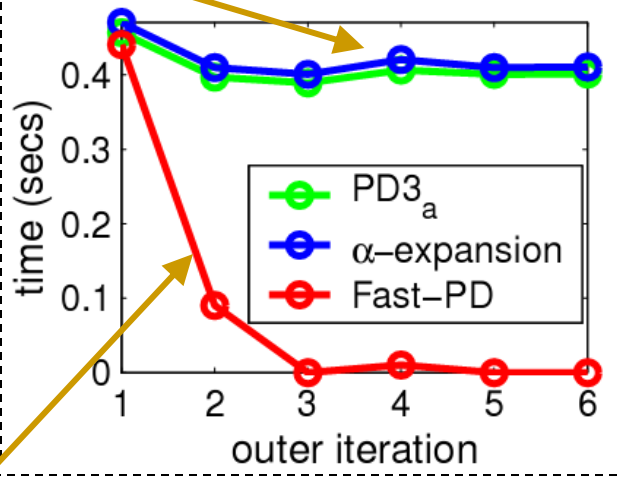
Tsukuba

almost constant



SRI-tree



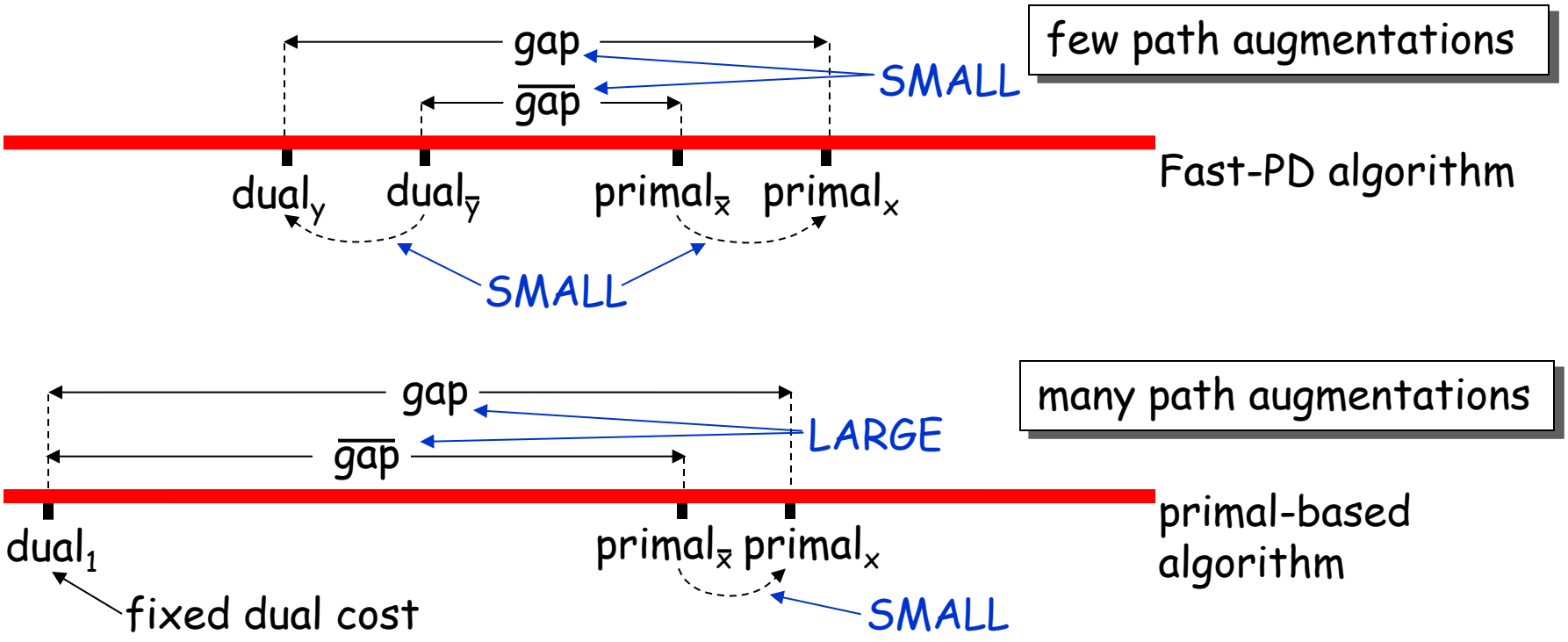


dramatic decrease

	total (secs)	Fast-PD	α-expansion
penguin		17.44	173.1
tsukuba		3.37	15.63
SRI tree		0.54	2.56

Computational efficiency (dynamic MRFs)

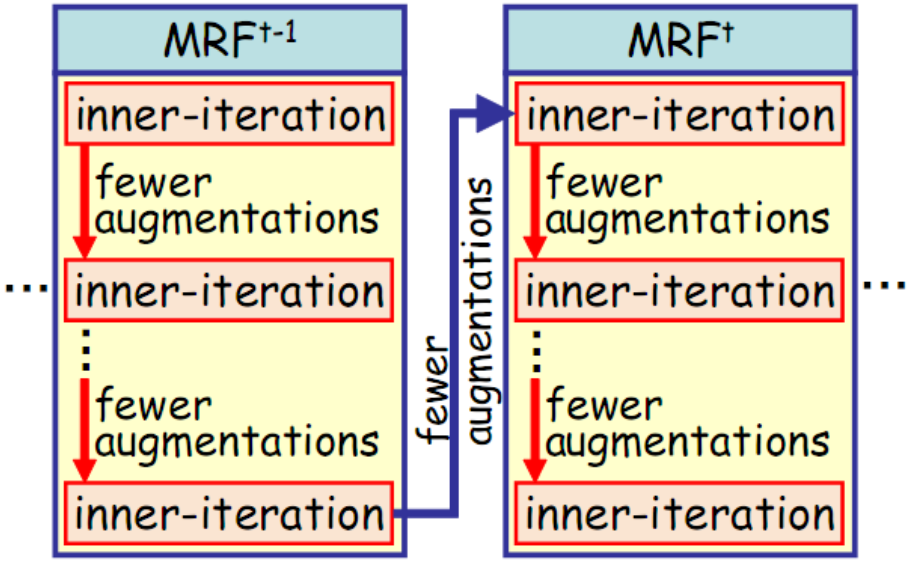
- Fast-PD can speed up dynamic MRFs [Kohli,Torr] as well (demonstrates the power and generality of this framework)



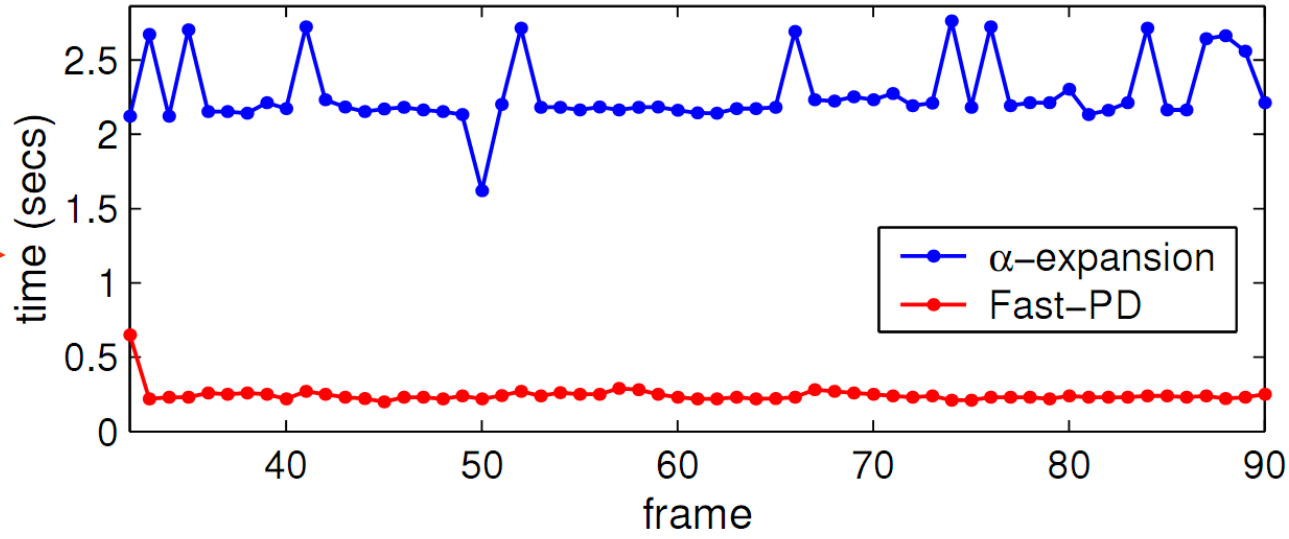
- This framework provides principled (and simple) way to update dual variables when switching between different MRFs

Computational efficiency (dynamic MRFs)

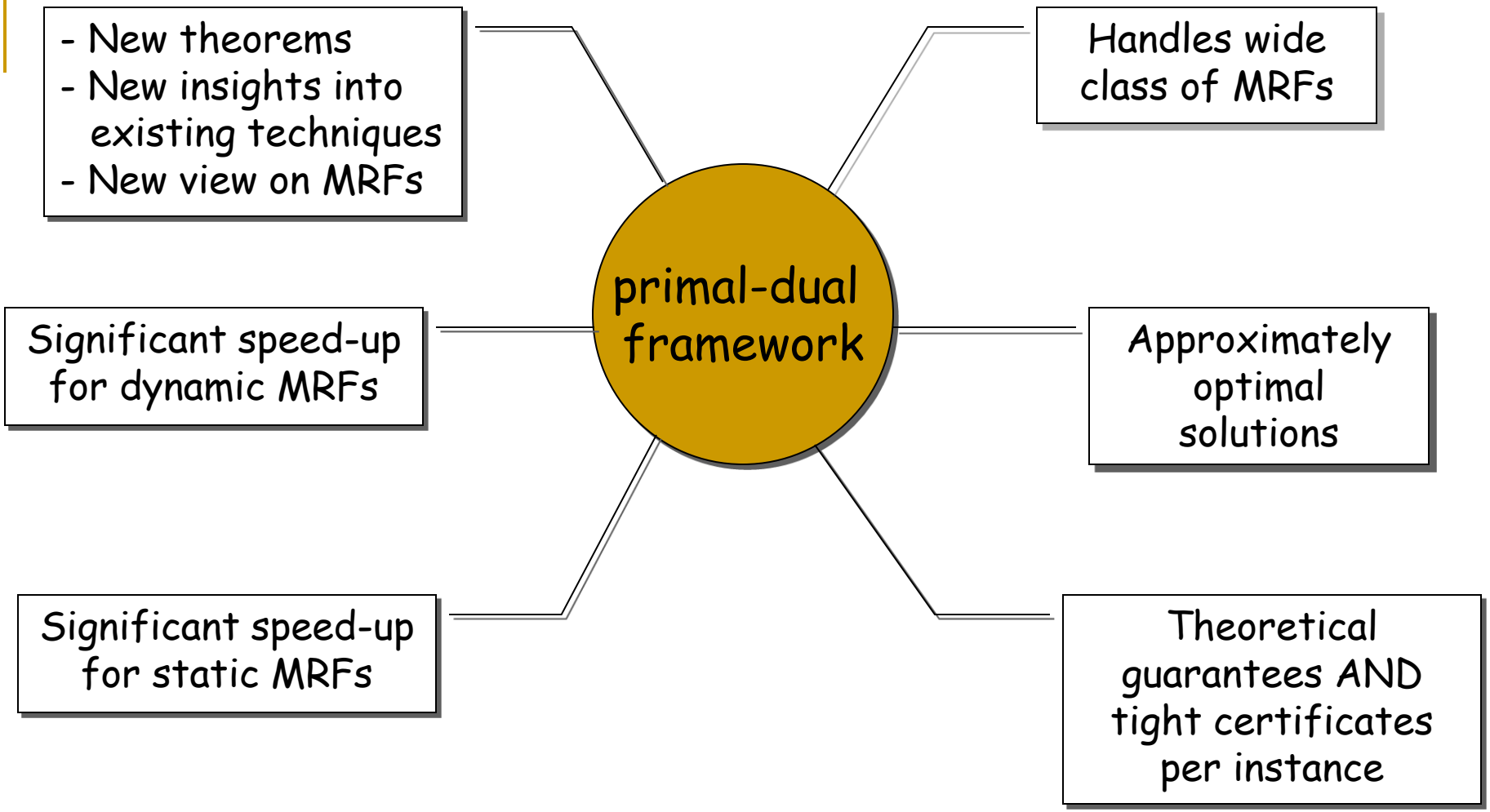
- Essentially, Fast-PD works along 2 different "axes"
  - reduces augmentations across different iterations of the same MRF
  - reduces augmentations across different MRFs



- Time per frame for SRI-tree stereo sequence



- Handles general (multi-label) dynamic MRFs


- New theorems
- New insights into existing techniques
- New view on MRFs
Handles wide class of MRFs
primal-dual framework
Significant speed-up for dynamic MRFs
Approximately optimal solutions
Significant speed-up for static MRFs
Theoretical guarantees AND tight certificates per instance

Part II:
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 via first solving its LP-relaxation.
- As we shall see, this will lead to some message passing methods for MRF optimization
- Actually, all resulting methods try to solve the dual to the LP-relaxation
  - but this is equivalent to solving the LP, as there is no duality gap due to convexity

Message-passing methods to the rescue

- Tree reweighted message-passing algorithms
  - [Vladimir, ICCV'07]
- MRF optimization via dual decomposition
  - [very brief sketch will be provided in this talk]
  - [see also work of Wainwright et al. on TRW methods]

MRF optimization via dual-decomposition

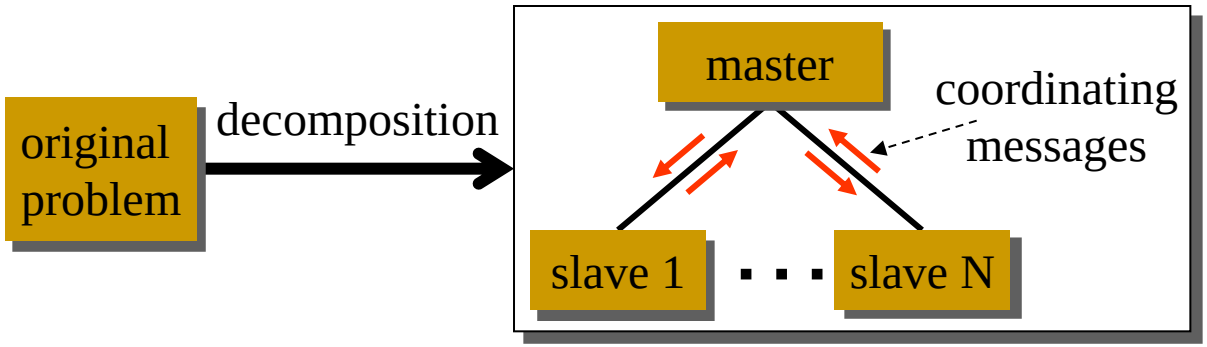
- New framework for understanding/designing message-passing algorithms
- Stronger theoretical properties than state-of-the-art
- New insights into existing message-passing techniques
- Reduces MRF optimization to a simple **projected subgradient** method
  (very well studied topic in optimization, i.e.,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

Dual decomposition (1/2)

- 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 (2/2)

- 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.

- Via auxiliary variables $\{x^i\}$, we 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:

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

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:

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

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:

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

$\nabla \equiv$ 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:

$$\min_{\mathbf{x}} \left[E(\theta, \mathbf{x}) = \sum_{p \in G} \sum_{a \in L} V_p(a) x_{p,a} + \sum_{p \in \mathcal{V}} \theta_p \cdot \mathbf{x}_p + \sum_{pq \in E} \sum_{a,b \in L} V_{pq}(a,b) x_{pq,ab} + \sum_{pq \in \mathcal{E}} \theta_{pq} \cdot \mathbf{x}_{pq} \right]$$

$$\text{s.t.} \sum_{a \in L} x_{p,a} = 1 \quad \longleftarrow \quad \text{(only one label assigned per vertex)}$$

$$\left.\begin{aligned} \sum_{a \in L} x_{pq,ab} &= x_{q,b} \\ \sum_{b \in L} x_{pq,ab} &= x_{p,a} \end{aligned}\right\} \quad \longleftarrow \quad \text{enforce consistency between variables } x_{p,a}, x_{q,b} \text{ and variable } x_{pq,ab}$$

$$x_{p,a},\ x_{pq,ab} \in \{0,1\}$$

θ: ...rs consisting of all

ur ...ctorized potentials.

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

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

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.

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):

$$\theta_p^T \mathrel{+}= \alpha_t \cdot \left(\bar{\mathbf{x}}_p^T - \frac{\sum_{T' \in \mathcal{T}(p)} \bar{\mathbf{x}}_p^{T'}}{|\mathcal{T}(p)|} \right), \quad \theta_{pq}^T \mathrel{+}= \alpha_t \cdot \left(\bar{\mathbf{x}}_{pq}^T - \frac{\sum_{T' \in \mathcal{T}(pq)} \bar{\mathbf{x}}_{pq}^{T'}}{|\mathcal{T}(pq)|} \right)$$

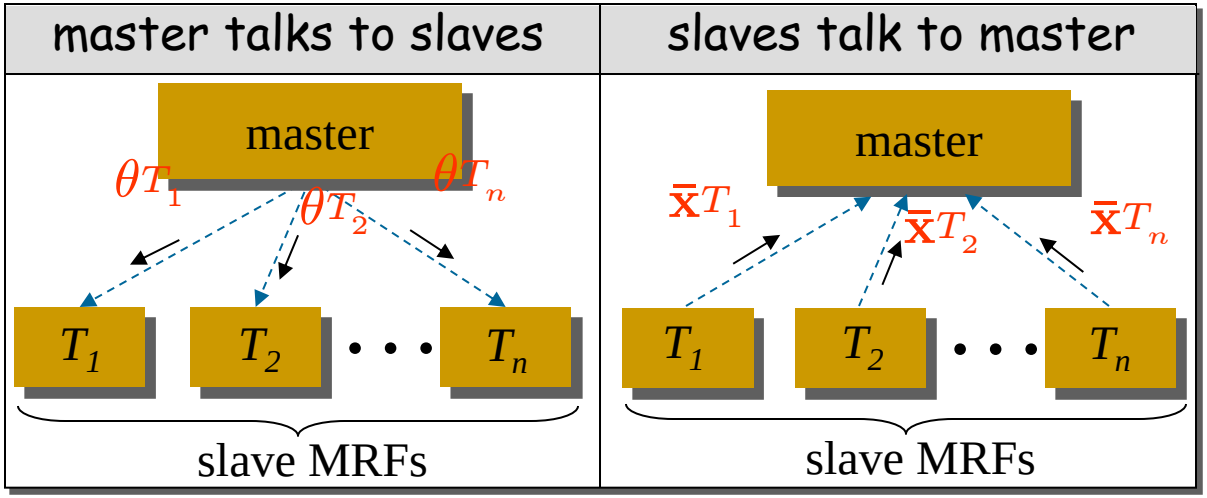
"What is it that you seek, Master?..."

- 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?..."



Theoretical properties (1/2)

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

Theoretical properties (2/2)

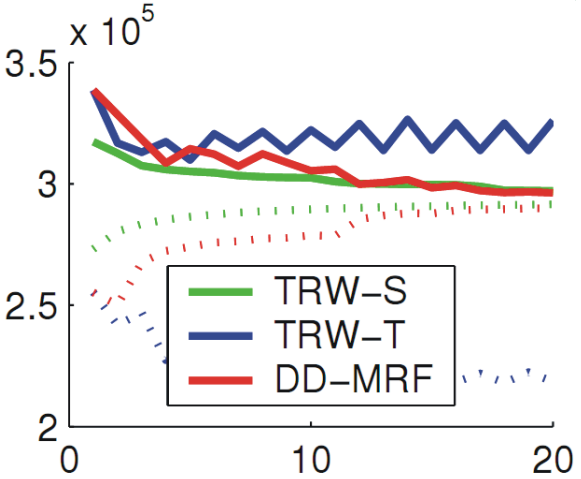
- Generalizes Weak Tree Agreement (WTA) condition introduced by V. Kolmogorov
  - Computes optimum for binary submodular MRFs
- Extremely general and flexible framework
  - Slave-MRFs need not be tree-structured (exactly the same framework still applies)

Experimental results (1/4)

- 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

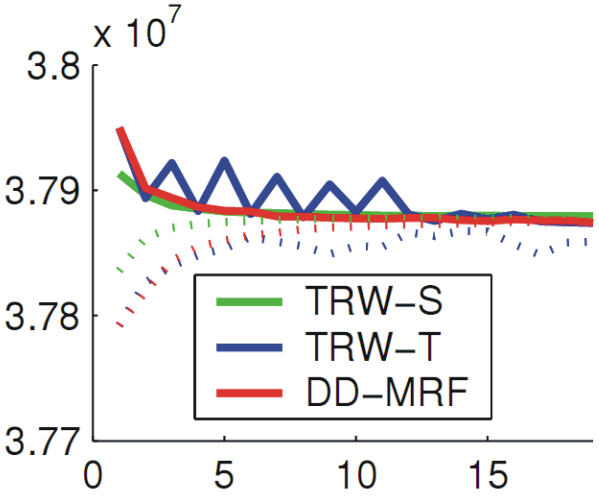
Experimental results (2/4)

estimated disparity for
Tsukuba stereo pair



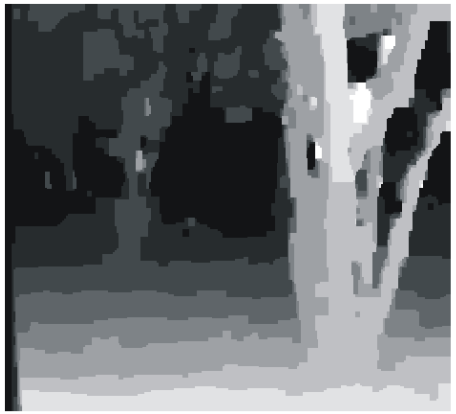
lower bounds (dual costs) and
MRF energies (primal costs)

estimated disparity for
Map stereo pair

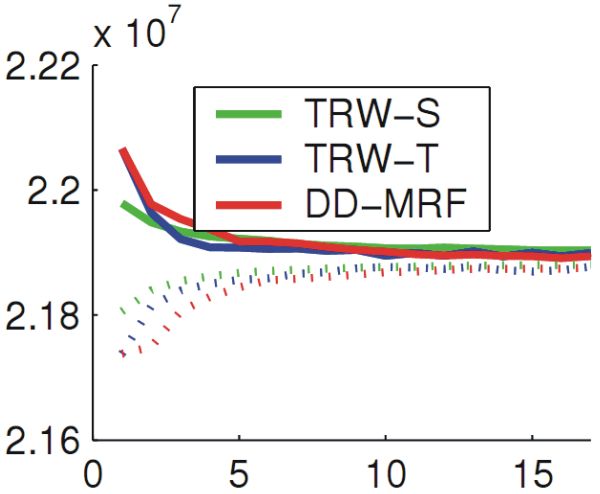


lower bounds (dual costs) and
MRF energies (primal costs)

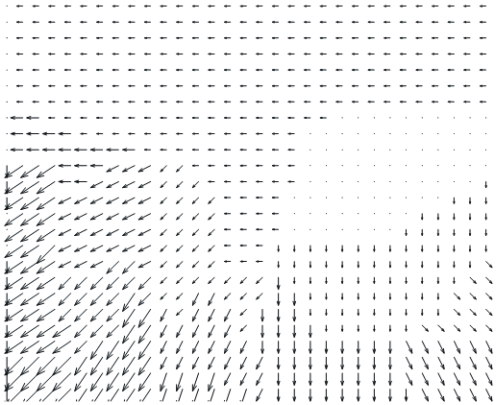
Experimental results (3/4)



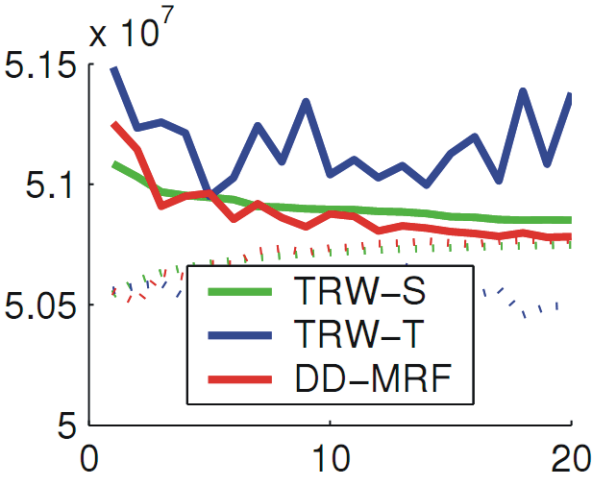
estimated disparity for SRI stereo pair



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

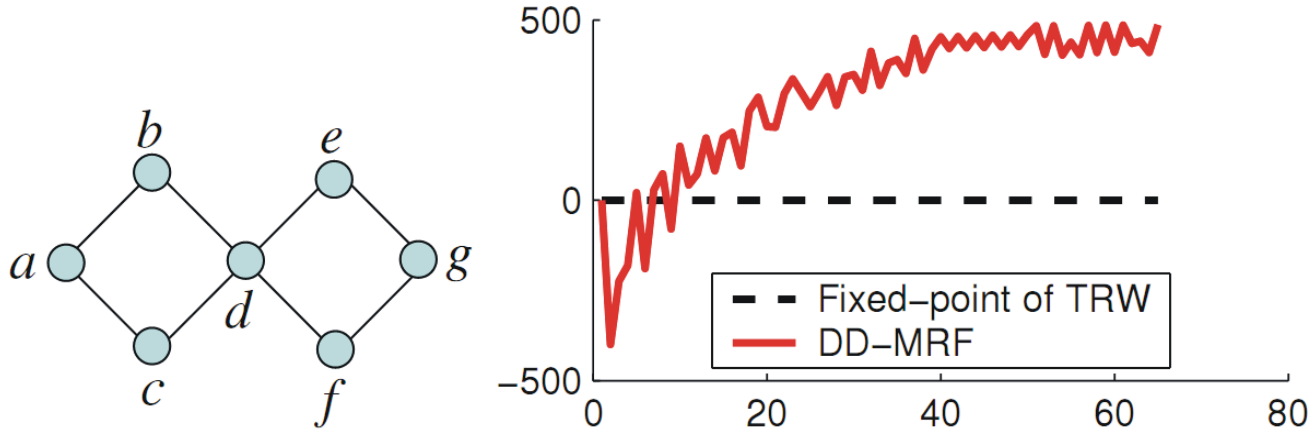


estimated optical flow for Yosemite sequence

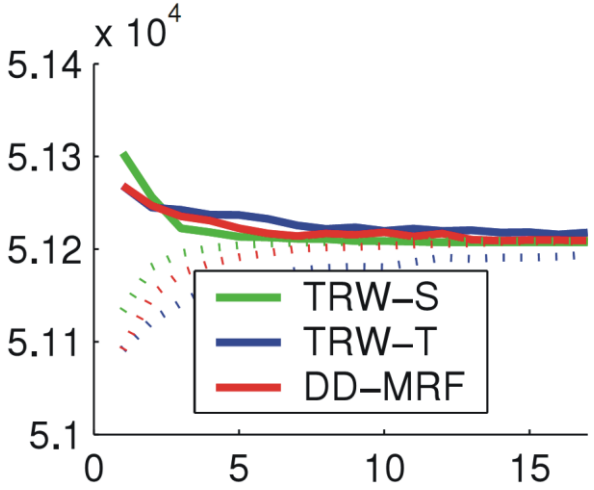


lower bounds (dual costs) and MRF energies (primal costs)

Experimental results (4/4)



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

Take home messages

1. Relaxing is always a good idea (just don't overdo it!)

2. Take advantage of duality, whenever you can

Thank you!