Approximate Labeling via Graph Cuts Based on Linear Programming

Nikos Komodakis and Georgios Tziritas, *Senior Member*, *IEEE*

Abstract—A new framework is presented for both understanding and developing graph-cut-based combinatorial algorithms suitable for the approximate optimization of a very wide class of Markov Random Fields (MRFs) that are frequently encountered in computer vision. The proposed framework utilizes tools from the duality theory of linear programming in order to provide an alternative and more general view of state-of-the-art techniques like the α-expansion algorithm, which is included merely as a special case. Moreover, contrary to α-expansion, the derived algorithms generate solutions with guaranteed optimality properties for a much wider class of problems, for example, even for MRFs with nonmetric potentials. In addition, they are capable of providing per-instance suboptimality bounds in all occasions, including discrete MRFs with an arbitrary potential function. These bounds prove to be very tight in practice (that is, very close to 1), which means that the resulting solutions are almost optimal. Our algorithms' effectiveness is demonstrated by presenting experimental results on a variety of low-level vision tasks, such as stereo matching, image restoration, image completion, and optical flow estimation, as well as on synthetic problems.

Index Terms—Global optimization, graph-theoretic methods, linear programming, Markov Random Fields, pixel classification, graph labeling, graph algorithms, early vision, stereo, motion, image restoration.

1 Introduction

A large variety of important tasks in low-level vision, image analysis, and pattern recognition can be formulated as labeling problems where one seeks to optimize some measure related to the quality of the labeling [1]. For example, such is the case in optical flow estimation, stereo matching, and image restoration, to mention only a few of them. Therefore, an issue of paramount importance, which has attracted a significant amount of computer vision research over the past years, is how to solve this class of labeling problems efficiently and accurately.

The Metric Labeling (ML) problem, which was recently introduced by Kleinberg and Tardos [2], can capture a broad range of these classification problems that arise in early vision. According to that problem's definition, the task is to classify a set $\mathcal{V}$ of n objects by assigning to each object a label from a given set $\mathcal{L}$ of labels. To this end, we are also given a weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$, where the set of edges $\mathcal{E}$ represents the pairwise relationships between the objects, with the weight w_{pq} of an edge pq representing the strength of the relationship between objects p and q. Each labeling of the objects in $\mathcal{V}$ is represented by a function $f : \mathcal{V} \to \mathcal{L}$ and is also associated with a certain cost, which can be decomposed into terms of two kinds.

On one hand, for each $p \in V$, there is a *label cost* $\boldsymbol{c}_p(a) \geq 0$ for assigning label $a = f_p$ to p. Intuitively, the label costs express the likelihood of assigning labels to objects. On the other hand, for each pair of objects p and q that are related (that is, connected by an edge in the graph $\mathcal{G}$), there is a so-called *separation cost* for assigning labels $a = f_p$ and $b = f_q$ to them. This separation cost is equal to $w_{pq}d(a, b)$, where, as already mentioned, the edge weight w_{pq} represents the strength of the relationship between p and q, and $d(a, b)$ is a distance function between labels, measuring how similar two labels are. The intuition behind this definition of the separation cost is that objects that are strongly related to each other should be assigned similar labels. This helps in preserving the spatial coherence of the final labeling. To simplify the notation, we assume that all edges share a common distance $d(a, b)$, but, in fact, each edge pq could have its own unique distance $d_{pq}(a, b)$. Also, in the original formulation of ML, the distance $d(a, b)$ was assumed to be a metric, that is, $d(a, b) = 0 \Leftrightarrow a = b$, $d(a, b) = d(b, a) \geq 0$, and $d(a, b) \leq d(a, c) + d(c, b)$, but, here, we will relax this assumption. Based on these definitions, the total cost of a labeling f equals

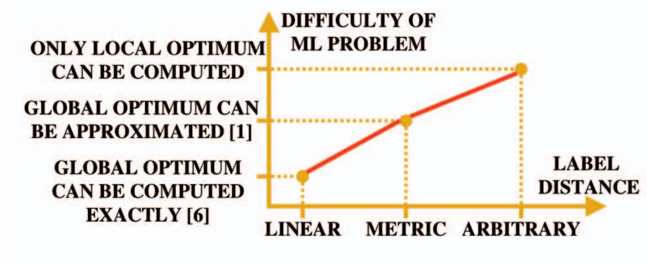
$$\mathrm{COST}(f) = \sum\nolimits_{p \in \mathcal{V}} \boldsymbol{c}_p(f_p) + \sum\nolimits_{(p,q) \in \mathcal{E}} w_{pq} d(f_p, f_q)$$

and the goal is to find a labeling with the minimum total cost.

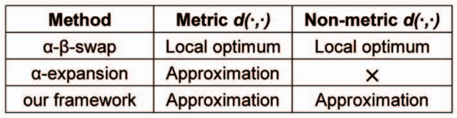
The ML problem is directly connected to the theory of Markov Random Fields (MRFs). In fact, optimizing the cost in the ML problem is essentially equivalent to minimizing the energy of a discrete MRF, with the potential function of the MRF to be now replaced by the distance function between labels [2]. Due to this connection to MRFs, solving the ML problem is (in general) NP-hard and, therefore, one can only hope for methods that provide approximate solutions. To this end, two main classes of methods have been proposed so far: those based on combinatorial optimization [1], [3], [4], [5], [6] and those based on linear programming (LP) [2], [7], [8]. Methods of the first class are efficient and have been applied with great success to many problems in vision. However, up to now, they have been interpreted only as greedy local search

- *The authors are with the Computer Science Department, University of Crete, PO Box 2208, 71409, Heraklion, Greece. E-mail: {komod, tziritas}@csd.uoc.gr.*

Manuscript received 5 Dec. 2005; revised 5 July 2006; accepted 11 Oct. 2006; published online 18 Jan. 2007.
Recommended for acceptance by R. Zabih.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0681-1205.
Digital Object Identifier no. 10.1109/TPAMI.2007.1061.



(a)



Method	Metric $d(\cdot,\cdot)$	Non-metric $d(\cdot,\cdot)$
α-β-swap	Local optimum	Local optimum
α-expansion	Approximation	×
our framework	Approximation	Approximation

(b)

Fig. 1. (a) The difficulty of the ML problem critically depends on the type of chosen label distance $d(\cdot,\cdot)$. (b) A comparison of our framework with respect to state-of-the-art optimization methods that are based on graph cuts.

techniques. On the other hand, methods of the second class possess good theoretical properties, but their main drawback is the intolerable computational cost due to the fact that they formulate ML as an equivalent integer program with a very large number of variables. For example, one such formulation, introduced in [7], is the following:

$$\min \sum_{p\in\mathcal{V}}\sum_{a\in\mathcal{L}} \boldsymbol{c}_p(a)x_p(a) + \sum_{(p,q)\in\mathcal{E}} w_{pq}\sum_{a,b\in\mathcal{L}} d(a,b)x_{pq}(a,b), \quad (1)$$

$$\text{s.t.} \sum\nolimits_a x_p(a) = 1 \qquad \forall\, p\in V, \quad (2)$$

$$\sum\nolimits_a x_{pq}(a,b) = x_q(b) \qquad \forall\, b\in L,\ (p,q)\in E, \quad (3)$$

$$\sum\nolimits_b x_{pq}(a,b) = x_p(a) \qquad \forall\, a\in L,\ (p,q)\in E, \quad (4)$$

$$x_p(\cdot),\ x_{pq}(\cdot,\cdot)\in\{0,1\}.$$

The $\{0,1\}$-variable $x_p(a)$ indicates that vertex p is assigned label a, whereas the $\{0,1\}$-variable $x_{pq}(a,b)$ indicates that vertices p and q are assigned labels a and b, respectively. The variables $x_{pq}(a,b)$ and $x_{qp}(b,a)$ therefore indicate the same thing. So, in order to eliminate one of them and reduce the number of variables, we assume (without loss of generality) that only one of (p,q) and (q,p) belongs to $\mathcal{E}$ for any neighbors p and q. The notation "$p\sim q$" will hereafter denote that p and q are neighbors, that is, "either only $(p,q)\in\mathcal{E}$ or only $(q,p)\in\mathcal{E}$." The first constraint (2) simply expresses the fact that each vertex must receive exactly one label, whereas (3) and (4) maintain consistency between variables $x_p(\cdot)$, $x_q(\cdot)$, and $x_{pq}(\cdot,\cdot)$ in the sense that, if $x_p(a)=1$ and $x_q(b)=1$ hold true, then these constraints force $x_{pq}(a,b)=1$ to hold true as well.

To overcome the limitations of current state-of-the-art methods, a new framework [9], [10] is proposed in this paper which provides novel global minimization algorithms for the approximate optimization of the ML problem (and, thus, of a very wide class of MRFs frequently encountered in computer vision). It makes use of the primal-dual schema of LP in order to derive efficient (that is, combinatorial) approximation techniques with guaranteed optimality properties, thus bridging the gap between the two classes of approximation algorithms mentioned above. The major contributions of the proposed framework are the following:

1. It turns out that the difficulty of the ML problem critically depends on the type of the chosen distance $d(\cdot,\cdot)$ between labels (see Fig. 1a). Up to now, one limitation of the state-of-the-art α-expansion method was that it had to assume that this distance was a metric, that is, it satisfied the triangle inequality. However, this case often does not hold in practice, thus limiting the applicability of the α-expansion method. On the contrary, the algorithms derived in the proposed framework only require a nonmetric distance function that satisfies $d(a,b)=0\Leftrightarrow a=b$ and $d(a,b)=d(b,a)\geq 0$, which is a weaker assumption.[1]

   This opens the way for applying our techniques to a wider class of MRFs with more general energy functions. Given that MRFs are ubiquitous in computer vision, this also implies that these algorithms can handle many more instances of a large variety of computer vision tasks (including stereo matching, image restoration, image completion, optical flow estimation, and so forth). For all these problems, the use of more sophisticated MRF priors is allowed based on our framework, thus leading to a better modeling of the problem at hand. This is important since it is well known that the choice of the prior plays a very significant role in the quality of the generated solutions.
2. Furthermore, the quality of these solutions also critically depends on how close they are to the true optimum of the MRF energy function. Another contribution of our framework is that, even in the case of a nonmetric distance, it can still guarantee that the generated solution will always be within a known factor of the global optimum, that is, a worst-case suboptimality bound can be provided in this case (see Fig. 1b). This is in contrast to local MRF optimization methods, such as the iterated conditional mode (ICM) algorithm or the Highest-Confidence-First method, for which no such theoretical guarantees (that is, no such analysis) can be provided. We should also note that, although any algorithm (for example, the α-expansion) can be converted to handle nonmetric distances without a loss in the worst-case bounds (for example, by replacing nonmetric terms with Potts terms, see [1]), this completely misses any structure of the nonmetric distance function. On the contrary, our method can handle both metric and nonmetric costs naturally.
3. In fact, in practice, the resulting solutions are much closer to the true optimum than what the worst-case approximation factors predict, that is, they are nearly

1. In fact, the assumption of a symmetric distance is not used by any of the theorems in this paper and, so, our algorithms can handle any distance for which $d(a,b)=0\Leftrightarrow a=b$, $d(a,b)\geq 0$. The term "nonmetric" will thus refer just to these conditions hereafter. Furthermore, our framework can be easily extended to even handle certain distances for which $d(a,b)=0\Leftrightarrow a=b$ is not true.

optimal. This can be verified thanks to our algorithms' ability to also provide per-instance suboptimality bounds, a property common to any other primal-dual or LP-rounding-based algorithm as well [2], [7], [11]. Moreover, in our case, these bounds can be derived without having to solve large linear programs to optimality and they also prove to be very tight (that is, close to 1) in practice. They can therefore be used to access the optimality of the generated solutions and, thus, are very useful in deciding the ability of the chosen MRF to model the problem under consideration (for example, the existence of a nearly optimal solution that does not look intuitively good implies that a different MRF should be chosen). Moreover, since these per-instance bounds are updated throughout the algorithm's execution, they can also be used in assessing its convergence, thus possibly reducing the total running time.

4. The generality and power of our framework is exhibited by presenting various algorithms, just one of which is proven to be equivalent to the α-expansion graph-cut technique (that is, a method that is currently considered state-of-the-art). Our framework therefore provides an alternative and more general view of these very successful graph-cut techniques, which can now be interpreted not merely as greedy local search, but also in terms of principles drawn from duality theory of LP, thus shedding further light on their essence (for example, a connection between α-expansion and tree-reweighted max-product belief propagation (BP) [11], which also tries to solve exactly the same dual LP relaxation, can thus be established). This is an important advance that, we believe, may open the way for new related research and can thus lead to even better MRF optimization algorithms in the future. Moreover, the primal-dual schema, a powerful optimization tool that was already known to people in combinatorial optimization (since it was already used for tackling many LP problems [12], for example, for providing an alternate way to derive Dijkstra's algorithm, Ford-Fulkerson's algorithm, the Hungarian method, and so forth), is now also introduced to the field of computer vision, which can prove to be a great benefit as well.

The rest of the paper is organized as follows: We review related work in Section 2. In Section 3, the primal-dual schema is presented, which will guide the design of all of our approximation algorithms. These algorithms are described in Sections 4, 5, and 6. More specifically, to handle the various cases of the distance function, we will progressively present three different families of primal-dual algorithms, which are thus named PD1, PD2, and PD3, respectively. Algorithm PD1 forms the base for deriving and understanding the other two types of algorithms and, so, the main points of that algorithm are described thoroughly in Section 4. In Section 5, we derive $\mathrm{PD2}_\mu$ (based on PD1), which is the second family of primal-dual algorithms and are parameterized by a variable μ. Unlike algorithm PD1, all algorithms in this family can be applied only to metric MRFs. Furthermore, we show that the well-known α-expansion technique is equivalent to just one member of this family of algorithms. In particular, α-expansion arises if we simply set $\mu = 1$, that is, it is equivalent to algorithm $\mathrm{PD2}_{\mu=1}$. In Section 6, we present PD3 algorithms, which make up the third family of our primal-dual methods. These algorithms manage to extend, as well as generalize, the α-expansion method (that is, algorithm $\mathrm{PD2}_{\mu=1}$) to the case of nonmetric MRFs. In addition, despite this generalization, these algorithms manage to maintain the theoretical approximation guarantees of the $\mathrm{PD2}_{\mu=1}$ algorithm. Experimental results are shown in Section 7 and we conclude in Section 8. We note that, for reasons of clarity (as well as space), not all technical proofs of the theorems are presented here, but they can all be found in [10].

2 Related Work

There is a vast amount of computer vision methods on how MRFs can be optimized. Such methods include, for example, the ICM algorithm, the Highest-Confidence-First heuristic, multiscale MRFs, relaxation labeling, graduated nonconvexity, and mean field annealing, to mention just a few of them. However, all of the abovementioned methods, as well as the great majority of the methods in the literature, are only able to provide a local minimum that can be arbitrarily far away from the true optimum, thus giving no guarantees about the quality of the resulting solutions (that is, how close these are to the true optimum). Most closely related to our work are those (few) approaches that do provide such guarantees about the optimality of their solutions.

One such class of approximation algorithms [2], [7], [8] is based on formulating MRF optimization as a natural integer program. An LP relaxation of that integer program is then solved, and a randomized rounding technique is being used to extract a near-the-optimum-integer solution. Different authors choose different linear programs or rounding techniques for that purpose. Although these algorithms appear to have good theoretical properties, they are still impractical to use in problems of early vision since, in that case, the linear program to be solved becomes extremely large. Moreover, in order to provide any guarantees about the suboptimality of their solutions, they usually need to further assume that the MRF potential function is a metric.

Another class of approximation algorithms is based on combinatorial optimization. Out of these algorithms, a very popular one is the α-expansion graph-cut method [1], [3]. This can be interpreted as an iterative local search technique, which, at each iteration, tries to extract a better solution (that is, one with lower energy) by finding the minimum cut in a suitable graph. This state-of-the-art method has been proven to be very efficient in practice and has been applied with great success to many problems in computer vision [13], [14]. Its drawback, however, is that it is only applicable to MRFs with a metric potential function. In fact, for some of these metrics, graph-cut techniques with better optimality properties seem to exist as well [5].

Also related to α-expansion is the α-β-swap algorithm [1]. Although this is a more general method, as it applies to nonmetric potentials as well, it does not seem to be as effective as α-expansion. This mainly has to do with the fact that it provides no guarantees about the optimality of its solutions and, thus, may very well get stuck to a bad local minimum. Finally, we should note that, for a certain class of MRFs, there also exist graph-cut-based methods that are capable of extracting the exact global optimum [4], [6]. These, however,

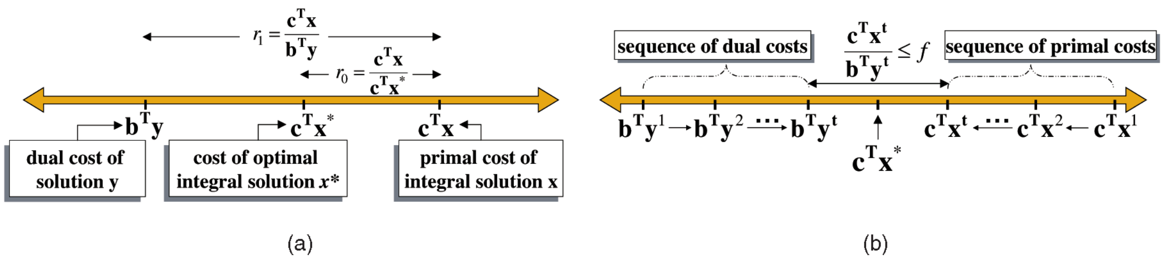

$r_1 = \frac{c^T x}{b^T y}$
$r_0 = \frac{c^T x}{c^T x^*}$
$b^T y$
$c^T x^*$
$c^T x$
dual cost of solution y
cost of optimal integral solution x*
primal cost of integral solution x
(a)
sequence of dual costs
$\frac{c^T x^t}{b^T y^t} \le f$
sequence of primal costs
$b^T y^1 \to b^T y^2 \to \cdots \to b^T y^t$
$c^T x^*$
$c^T x^t \leftarrow \cdots c^T x^2 \leftarrow c^T x^1$
(b)

Fig. 2. (a) By weak duality, the optimal cost $\mathbf{c}^T\mathbf{x}^*$ will lie between the costs $\mathbf{b}^T\mathbf{y}$ and $\mathbf{c}^T\mathbf{x}$ of any pair $(\mathbf{x}, \mathbf{y})$ of integral-primal and dual feasible solutions. Therefore, if $\mathbf{b}^T\mathbf{y}$ and $\mathbf{c}^T\mathbf{x}$ are close enough (for example, their ratio r_1 is $\le f$), so are $\mathbf{c}^T\mathbf{x}^*$ and $\mathbf{c}^T\mathbf{x}$ (for example, their ratio r_0 is $\le f$ as well), thus proving that $\mathbf{x}$ is an f-approximation to $\mathbf{x}^*$. (b) According to the primal-dual schema, dual and integral-primal feasible solutions make local improvements to each other until the final costs $\mathbf{b}^T\mathbf{y}^t$ and $\mathbf{c}^T\mathbf{x}^t$ are close enough (for example, their ratio is $\le f$). We can then apply the Primal-Dual Principle (as in (a)) and, thus, conclude that $\mathbf{x}^t$ is an f-approximation to $\mathbf{x}^*$.

require the potential function to be convex and the labels to be 1D, a fact that restricts their applicability.

Finally, we should also mention that there also exist those optimization algorithms that are based on BP [15]. Although they impose no restrictions on the type of the MRF potential function to be chosen, their theoretical optimality and convergence properties are not yet well understood. However, significant progress has been made with respect to this issue over the last years. In particular, the tree-reweighted max-product BP algorithm [11] can be implemented in a way that will provably converge [16] and can also be used to obtain bounds on the optimal solution. In fact, it was recently shown that, for certain instances of the stereo problem, it can even find the global minimum [17].

3 The Primal-Dual Schema

Let us consider the following pair of primal and dual linear programs:

$$\text{Primal}: \quad \min \mathbf{c}^T\mathbf{x} \qquad\qquad \text{Dual}: \quad \max \mathbf{b}^T\mathbf{y}$$
$$\text{s.t.} \quad \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \ge \mathbf{0} \qquad\qquad \text{s.t.} \quad \mathbf{A}^T\mathbf{y} \le \mathbf{c}.$$

Here, $\mathbf{A} = [a_{ij}]$ represents an $m \times n$ rectangular matrix, whereas $\mathbf{b}$ and $\mathbf{c}$ are column vectors of size m and n, respectively. We would like to find an optimal solution to the primal program under the additional constraint that its components are integer numbers. Due to this integrality requirement, this problem is, in general, NP-hard and, so, we need to settle with estimating approximate solutions. A primal-dual f-approximation algorithm achieves that by using of the following principle:

Primal-Dual Principle. *If* $\mathbf{x}$ *and* $\mathbf{y}$ *are integral-primal and dual feasible solutions satisfying*

$$\mathbf{c}^T\mathbf{x} \le f \cdot \mathbf{b}^T\mathbf{y}, \tag{5}$$

then $\mathbf{x}$ *is an* f*-approximation to the optimal integral solution* $\mathbf{x}^*$*; that is,* $\mathbf{c}^T\mathbf{x}^* \le \mathbf{c}^T\mathbf{x} \le f \cdot \mathbf{c}^T\mathbf{x}^*$.

The reason that this principle holds true is rather simple and is illustrated graphically in Fig. 2a: In particular, due to weak duality, it will hold that the cost $\mathbf{c}^T\mathbf{x}^*$ of the optimal integral solution will always lie between the dual cost $\mathbf{b}^T\mathbf{y}$ and the primal cost $\mathbf{c}^T\mathbf{x}$, that is, $\mathbf{b}^T\mathbf{y} \le \mathbf{c}^T\mathbf{x}^* \le \mathbf{c}^T\mathbf{x}$. If we therefore manage to bring the two quantities $\mathbf{b}^T\mathbf{y}$ and $\mathbf{c}^T\mathbf{x}$ close to each other (for example, by making their ratio $r_1 = \mathbf{c}^T\mathbf{x}/\mathbf{b}^T\mathbf{y}$ less than or equal to f, as in (5)), then we will also have succeeded in bringing the costs $\mathbf{c}^T\mathbf{x}^*$ and $\mathbf{c}^T\mathbf{x}$ close to each other as well (for example, the ratio $r_0 = \mathbf{c}^T\mathbf{x}/\mathbf{c}^T\mathbf{x}^*$ will also be less than f), thus proving that $\mathbf{x}$ is indeed an f-approximation to $\mathbf{x}^*$. Put otherwise, what the above principle does is to make use of the fact that the primal LP gives a lower bound to the primal integer program (IP) and, thus, the dual of the LP, which is (by weak duality) always a lower bound to the primal LP, will also give a lower bound to the primal IP as well (that is, $\text{dual-LP} \le \text{primal-LP} \le \text{primal-IP}$). This also implies that the quality of solution $\mathbf{x}$ will depend on how tight the LP relaxation is with respect to the IP.

The abovementioned principle lies at the heart of any primal-dual technique. In fact, the various primal-dual methods mostly differ in the way in which they manage to estimate a pair $(\mathbf{x}, \mathbf{y})$ satisfying the fundamental inequality (5). One very common way for that (but not the only one) is by relaxing the so-called primal complementary slackness conditions [18].

Theorem 3.1 (Relaxed Complementary Slackness). *If the pair* $(\mathbf{x}, \mathbf{y})$ *of integral-primal and dual feasible solutions satisfies the so-called relaxed primal complementary slackness conditions*

$$\forall\, x_j > 0 \Rightarrow \sum_{i=1}^{m} a_{ij} y_i \ge c_j / f_j,$$

then $(\mathbf{x}, \mathbf{y})$ *also satisfies the Primal-Dual Principle with* $f = \max_j f_j$ *and, therefore,* $\mathbf{x}$ *is an* f*-approximation to the optimal integral solution.*

To prove this, one must simply combine the relaxed complementary slackness conditions with the fact that solutions $\mathbf{x}$ and $\mathbf{y}$ satisfy the feasibility conditions of the primal and dual program, respectively (the fundamental inequality (5) then follows trivially). Thus, based on the abovementioned theorem, the following iterative schema is usually applied during a primal-dual f-approximation algorithm:

Primal-Dual Schema. *Keep generating pairs of integral-primal and dual solutions* $\{(\mathbf{x}^k, \mathbf{y}^k)\}_{k=1}^{t}$ *until the elements* $\mathbf{x}^t$ *and* $\mathbf{y}^t$ *of the last pair are both feasible and satisfy the relaxed primal complementary slackness conditions.*

This schema is illustrated graphically in Fig. 2b. At each iteration, based on just the current dual feasible solution $\mathbf{y}^k$, we perturb the current primal feasible solution $\mathbf{x}^k$ so that its primal cost $\mathbf{c}^T\mathbf{x}^k$ comes closer to the dual cost $\mathbf{b}^T\mathbf{y}^k$. This is also applied in reverse (that is, $\mathbf{y}^k$ is perturbed as well) and a new primal-dual pair, say $(\mathbf{x}^{k+1}, \mathbf{y}^{k+1})$, is thus generated. This

```
1: $\mathbf{x} \leftarrow$INIT_PRIMALS( );  $\mathbf{y} \leftarrow$INIT_DUALS( );  $\mathbf{x}_{\text{old}} \leftarrow \mathbf{x}$
2: for each label $c$ in $\mathcal{L}$ do
3:     $\mathbf{y} \leftarrow$ PREEDIT_DUALS$(c, \mathbf{x}, \mathbf{y})$;
4:     $[\mathbf{x}', \mathbf{y}'] \leftarrow$UPDATE_DUALS_PRIMALS$(c, \mathbf{x}, \mathbf{y})$;
5:     $\mathbf{y}' \leftarrow$POSTEDIT_DUALS$(c, \mathbf{x}', \mathbf{y}')$;
6:     $\mathbf{x} \leftarrow \mathbf{x}', \ \mathbf{y} \leftarrow \mathbf{y}'$;
7: end for
8: if $\mathbf{x} \neq \mathbf{x}_{\text{old}}$ then $\mathbf{x}_{\text{old}} \leftarrow \mathbf{x}$, goto 2;
9: if algorithm $\neq$ PD1 then $\mathbf{y}^{\text{fit}} \leftarrow$DUAL_FIT$(\mathbf{y})$;
```

Fig. 3. The primal dual schema, as applied by algorithms PD1, PD2, and PD3.

is repeated until the costs of the final primal-dual pair are close enough. The remarkable thing with this procedure is that the two processes (that is, the primal and the dual) make local improvements to each other and yet they manage to achieve an approximately global objective at the end. Also, it is worth mentioning that *one can thus devise different approximation algorithms merely by specifying a different set of complementary conditions (that is, different f_j) each time*, which is exactly what we will do for the case of ML and thus derive three different types of algorithms: PD1, PD2, and PD3.

3.1 Applying the Primal-Dual Schema to ML

For the case of ML, our primal linear program will be IP (1), after first relaxing its $\{0,1\}$ constraints to $x_p(\cdot) \geq 0$, $x_{pq}(\cdot,\cdot) \geq 0$. The dual of that LP will then be

$$\max \ \mathbf{z}^T \cdot \mathbf{1}, \tag{6}$$

$$\text{s.t. } \mathbf{z} \leq \min_{a \in \mathcal{L}} \mathbf{h}_a \quad (\min_{a \in \mathcal{L}} \mathbf{h}_a \text{ takes the elementwise minimum between vectors } \mathbf{h}_a), \tag{7}$$

$$y_{pq}(a) + y_{qp}(b) \leq w_{pq} d(a,b) \quad \forall a,b \in L, \ \forall (p,q) \in E. \tag{8}$$

In this case, dual variables consist of 1) a vector $\mathbf{z} = \{z_p\}_{p \in \mathcal{V}}$ with one component per vertex of $\mathcal{G}$, 2) an auxiliary vector $\mathbf{h}_a = \{h_p(a)\}_{p \in \mathcal{V}}$ per label a (each $\mathbf{h}_a$ has one component per vertex of $\mathcal{G}$), and 3) a vector $\mathbf{y}$ containing all variables $y_{pq}(\cdot)$ and $y_{qp}(\cdot)$, called the "*balance variables*" hereafter. Also, any two variables $y_{pq}(a)$ and $y_{qp}(a)$ will be referred to as "*conjugate balance variables*."

The variables $h_p(\cdot)$ are named the "*height variables*" and are just auxiliary variables that implicitly depend on the balance variables as follows:

$$h_p(\cdot) \equiv \boldsymbol{c}_p(\cdot) + \sum\nolimits_{q:q\sim p} y_{pq}(\cdot). \tag{9}$$

The reason for giving this name to the $h_p(\cdot)$ variables, as well as for introducing these redundant variables in the first place, will become clear in the sections that follow. Also, note that, due to (6) and (7), the z_p variables should always be set as follows:

$$\mathbf{z} = \min_{a \in \mathcal{L}} \mathbf{h}_a \tag{10}$$

and, so, we no longer have to worry about (7) or how to estimate the z_p variables. Furthermore, for defining a dual solution, only the balance variables $y_{pq}(\cdot)$ must be specified since the height variables $h_p(\cdot)$ can then be computed by (9).

Since we will be considering only feasible $\{0,1\}$-primal solutions, instead of the variables $x_p(\cdot)$ and $x_{pq}(\cdot,\cdot)$, a primal solution $\mathbf{x}$ will hereafter simply refer to a set of labels $\{x_p\}_{p \in \mathcal{V}}$, where x_p denotes the label assigned to vertex p. Then, $x_p(a) = 1$ is equivalent to $x_p = a$, whereas $x_{pq}(a,b) = 1$ means that $x_p = a$ and $x_q = b$ and, so, under this notation, it is not difficult to see that the complementary condition related to a nonzero $x_p(a)$ variable reduces to

$$z_p \geq c_p(x_p)/f_1 + \sum\nolimits_{q:q\sim p} y_{pq}(x_p), \tag{11}$$

whereas the complementary condition related to a nonzero $x_{pq}(a,b)$ variable reduces to

$$x_p \neq x_q \Rightarrow y_{pq}(x_p) + y_{qp}(x_q) \geq w_{pq} d(x_p, x_q)/f_2, \tag{12}$$

$$x_p = x_q = a \Rightarrow y_{pq}(a) + y_{qp}(a) = 0. \tag{13}$$

Our objective will therefore be to find feasible solutions $\mathbf{x}, \mathbf{y}$ satisfying (11), (12), and (13) for specific values of f_1 and f_2. Condition (13) simply says that conjugate balance variables are opposite to each other. For this reason, we set, by definition,

$$y_{qp}(\cdot) \equiv -y_{pq}(\cdot) \quad \forall \ (p,q) \in E \tag{14}$$

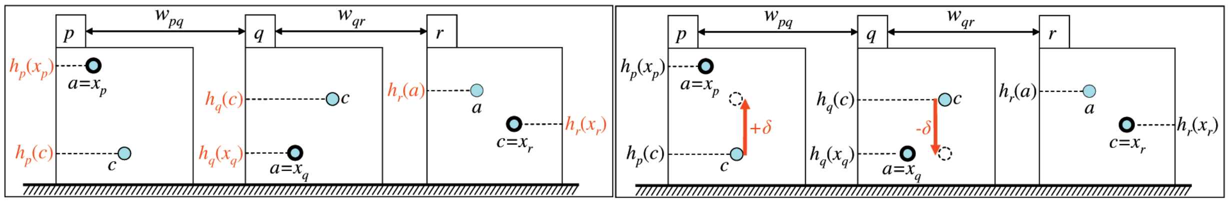
and, so, we do not have to worry about (13) hereafter.

Most of our primal-dual algorithms will achieve an approximation factor of $f_{\text{app}} = 2\frac{d_{\max}}{d_{\min}}$ (that is, $\max\{f_1, f_2\} = f_{\text{app}}$), where $d_{\min} \equiv \min_{a \neq b} d(a,b)$ and $d_{\max} \equiv \max_{a \neq b} d(a,b)$. Their basic structure can be seen in Fig. 3. The initial primal-dual solutions are generated inside INIT_PRIMALS and INIT_DUALS. During an inner iteration (lines 3-6 in Fig. 3), a label c is selected and a new primal-dual pair of solutions $(\mathbf{x}', \mathbf{y}')$ is generated by updating the current pair $(\mathbf{x}, \mathbf{y})$. During this iteration, among all balance variables of $\mathbf{y}$ (that is, $y_{pq}(.)$), only the balance variables of the c labels (that is, $y_{pq}(c)$) are modified. We call this a c-iteration of the algorithm. $|\mathcal{L}|$ such iterations (one c-iteration for each label c in the set $\mathcal{L}$) make up an outer iteration (lines 2-7 in Fig. 3) and the algorithm terminates if no vertex changes its label during the current outer iteration.

During an inner iteration, the main update of the primal and dual variables takes place inside UPDATE_DUALS_PRIMALS, whereas PREEDIT_DUALS and POSTEDIT_DUALS modify the dual variables before and after the main update. The DUAL_FIT routine, which is used only in algorithms PD2 and PD3, serves only the purpose of applying a scaling operation to the last dual solution.

4 The PD1 Algorithm

An intuitive view of the dual variables which will prove useful for designing our approximation algorithms is the following: For each vertex p, we consider a separate copy of all labels in $\mathcal{L}$. It is then assumed that all these labels represent balls, which float at certain heights relative to a reference plane. The role of the height variables is then to



(a) (b)

Fig. 4. Visualization of the dual variables for a graph $\mathcal{G}$ with vertices p, q, r and labels $\mathcal{L} = \{a, c\}$. (a) A copy of labels $\{a, c\}$ exists for each vertex and all of these labels are represented as balls floating above a reference plane. The role of the height variables is to specify the balls' height. (b) Furthermore, the balls are not static but may move in pairs by updating conjugate balance variables. For example, here, ball c at p is pulled up by $+\delta$ (due to an increase of $y_{pq}(c)$ by $+\delta$) and, so, ball c at q moves down by $-\delta$ (due to a decrease of $y_{qp}(c)$ by $-\delta$). Active labels are drawn with a thicker circle.

determine the balls' height (see Fig. 4a). For example, the height of label a at vertex p is given by the dual variable $h_p(a)$. Expressions like "label a at p is below/above label b" imply $h_p(a) \lessgtr h_p(b)$. Furthermore, the balls are not static but may move in pairs through updating pairs of conjugate balance variables. For example, in Fig. 4b, label c at p is raised by $+\delta$ (due to adding $+\delta$ to $y_{pq}(c)$) and, so, label c at q has to move down by $-\delta$ (due to subtracting $-\delta$ from $y_{qp}(c)$) so that conjugate variables remain opposite to each other. Therefore, the role of balance variables is to raise or lower labels. In particular, due to (9), the height of label a at p may change only if at least one of the balance variables $\{y_{pq}(a)\}_{q:q\sim p}$ changes as well. The value of balance variable $y_{pq}(a)$ thus represents the partial raise of label a at p due to edge pq, whereas the total raise of a at p equals the sum of all partial raises due to edges in $\mathcal{G}$ incident to p. Note that each $y_{pq}(\cdot)$ represents a net raise (just called raise hereafter) and not a relative raise. For example, in Fig. 4b, label a at p has a relative raise $+\delta$, but its (net) raise is $y_{pq}(c) + \delta$, where $y_{pq}(c)$ is the previous value of the balance variable.

Before proceeding to PD1, let us define some terminologies. Let $(\mathbf{x}, \mathbf{y})$ be a pair of integral-primal dual solutions. We call the label that $\mathbf{x}$ assigns to p (that is, x_p) the *active label at* p. The sum of heights of all active labels is called the "*Approximate Primal Function*" (APF), that is, $\text{APF}^{\mathbf{x},\mathbf{y}} = \sum_p h_p(x_p)$. This function's name comes from the fact that, if $\mathbf{x}$, $\mathbf{y}$ satisfies the relaxed slackness conditions, then it is easy to prove that APF approximates the primal objective function. Also, any balance variable of an active label at p (that is, any variable in $\{y_{pq}(x_p)\}_{q:q\sim p}$) will be called an *active balance variable at vertex* p. The "load" between neighbors p and q (denoted by load_{pq}) is then defined as $\text{load}_{pq} = y_{pq}(x_p) + y_{qp}(x_q)$ (that is, as the sum of two active balance variables at p, q) and represents the partial raises of active labels at p and q due to edge pq. If relaxed slackness conditions (12) hold, then, due to (12) and (8), it is easy to see that $w_{pq}d(x_p, x_q)/f_2 \le \text{load}_{pq} \le w_{pq}d(x_p, x_q)$ and, so, the load of p, q can be also thought of as a *virtual separation cost* that approximates the actual separation cost $w_{pq}d(x_p, x_q)$ of p, q (this will prove useful later for our PD3 algorithms).

Our first algorithm, called PD1, assumes that $d(\cdot,\cdot)$ is merely a nonmetric distance and then tries to find feasible $\mathbf{x}$, $\mathbf{y}$ satisfying complementary conditions (11) and (12) with $f_1 = 1$ and $f_2 = f_{\text{app}}$ (recall that $f_{\text{app}} \equiv 2\frac{d_{\max}}{d_{\min}}$). If $f_1 = 1$, then (11) becomes $z_p \ge h_p(x_p)$ and, so, since $z_p = \min_a h_p(a)$ (due to (10)), complementary condition (11) finally reduces to

$$h_p(x_p) = \min_a h_p(a). \tag{15}$$

Also, if $f_2 = f_{\text{app}}$, then complementary condition (12) reduces to

$$x_p \ne x_q \Rightarrow \text{load}_{pq} \ge w_{pq}d(x_p, x_q)/f_{\text{app}}. \tag{16}$$

Furthermore, to ensure feasibility of $\mathbf{y}$, PD1 enforces (for any label a) that

$$y_{pq}(a) \le w_{pq}d_{\min}/2. \tag{17}$$

To see that (17) ensures feasibility, it suffices to observe that $y_{pq}(a) + y_{qp}(b) \le 2w_{pq}d_{\min}/2 = w_{pq}d_{\min} \le w_{pq}d(a, b)$ and, so, the dual constraints (8) hold true.

Therefore, the goal of PD1 is to find $\mathbf{x}$, $\mathbf{y}$ satisfying (15), (16), and (17) To this end, it ensures that (16) and (17) always hold true (which is easy) and then iteratively drives $\mathbf{x}$, $\mathbf{y}$ toward satisfying (15) as well by alternating between updates of primal and dual variables. For this, it also maintains the invariant that *active balance variables are nonnegative*, that is,

$$\forall p \in \mathcal{V}, \quad y_{pq}(x_p) \ge 0. \tag{18}$$

To then see how the update of primal and dual variables should proceed, one simply needs to reinterpret (15), (16), and (17) based on the dual variables' aforementioned interpretation. For example,

- (15) simply says that, at each vertex, the active label should have the lowest height,
- (16) requires that any two active labels should be raised proportionally to their separation costs, and
- (17) says that there is an upper bound on how much we can raise a label.

Based on these, and assuming that (16) and (17) already hold true at the current (outer) iteration, the update of the primal and dual variables for the next (outer) iteration proceeds as follows:

Dual variables update. Given the current active labels (that is, the current primal), any nonactive label (which is below the corresponding active label) is raised (by increasing the appropriate balance variables) until it either reaches the active label or attains the maximum raise allowed by (17). Note that (16) still holds true since no active label has moved.

Primal variables update. Given the new heights (that is, the new dual), there might still be vertices violating (15), that is, their active labels are not at the lowest height. For each such vertex p, we select a nonactive label which is below x_p but has already reached the maximum raise allowed by (17). That

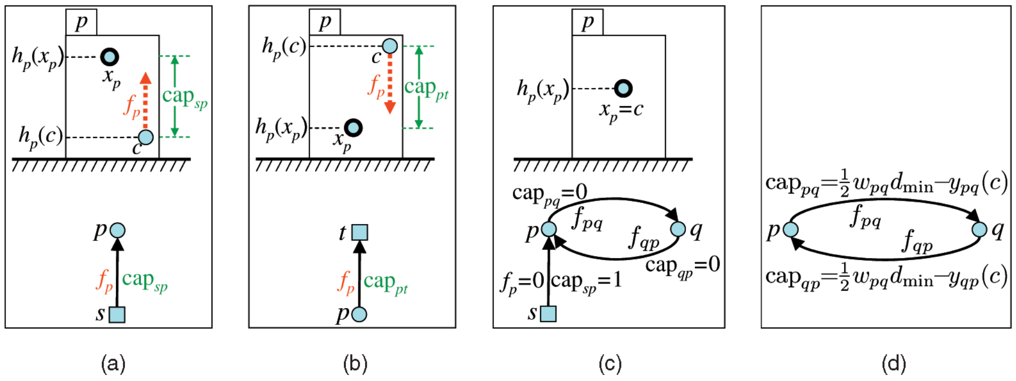


Fig. 5. Four simple rules for constructing graph $\mathcal{G}^c$. (a) If c at p is below x_p, we connect node p to source s and flow f_p through sp represents the total relative raise of c (and also $\text{cap}_{sp} = h_p(x_p) - h_p(c)$). (b) If c is above x_p, we connect node p to sink t and flow f_p at pt equals the total relative decrease in the height of c (and $\text{cap}_{pt} = h_p(c) - h_p(x_p)$). (c) If c is the active label at p, it need not move (that is, flow f_p through sp should be 0) and we thus set $\text{cap}_{pq} = \text{cap}_{qp} = 0$ (and, by convention, $\text{cap}_{sp} = 1$). (d) Capacities of interior edges are set so that (17) always holds true.

label, say c, is then made the new active label of p, that is, we set $x_p = c$. One can then show that (16) will still hold for the new active label. To see that, it suffices to observe that, because c has reached its maximum raise at p, then, for any neighbor q, it will hold that $y_{pq}(c) = \frac{w_{pq}d_{\min}}{2}$, which is $\geq \frac{w_{pq}d_{\min}}{2}\frac{d(x_p,x_q)}{d_{\max}} = \frac{w_{pq}d(x_p,x_q)}{f_{\text{app}}}$. This, in conjunction with the nonnegativity of any variable $y_{qp}(x_q)$ (due to (18)), thus proves that (16) will still hold. This way, since we keep assigning lower active labels to vertices, one may easily show that (15) will finally hold true in the end, that is, after a finite number of outer iterations.

During an outer iteration, the update of the dual variables takes place in groups, one group per inner iteration. In particular, during an inner c-iteration, only the heights of the c-labels are rearranged so that as many of these labels as possible are raised above the corresponding active labels. To this end, solution $\mathbf{y}$ is changed into solution $\mathbf{y}'$ by changing only variables $y_{pq}(c)$ (that is, the balance variables of all c-labels) into $y'_{pq}(c)$. This way, the new heights $h'_p(c)$ are produced. We must be careful, though, during this update of the c-heights. For example, in Fig. 4a, we would like the c-label at p to move at least as high as $x_p = a$ (the c-label at q is already above $x_q = a$, whereas the c-label at r does not need to move at all as it is already the active label of r). However, if we raise label c at p until it reaches x_p, say by increasing $y_{pq}(c)$, then label c at q will go below x_q due to the decrease of the conjugate variable $y_{qp}(c)$, thus breaking (15) for q.

It turns out that the optimal update of the c-heights can be simulated by pushing the maximum amount of flow through a directed graph $\mathcal{G}^c = (\mathcal{V}^c, \mathcal{E}^c, \mathcal{C}^c)$. Capacities $\mathcal{C}^c$ of this graph depend on $\mathbf{x}$, $\mathbf{y}$, whereas its nodes $\mathcal{V}^c$ consist of all nodes of graph $\mathcal{G}$ (the *internal* nodes) plus two *external* nodes, the source s and the sink t. Furthermore, all nodes of $\mathcal{G}^c$ are connected by two types of edges, *interior* and *exterior* edges, which are constructed using the following simple rules (see also Fig. 5):

Interior edges. For each edge $(p,q) \in \mathcal{G}$, we insert two directed interior edges pq and qp in graph $\mathcal{G}^c$. Flows f_{pq} (through pq), f_{qp} (through qp) will represent the increase and decrease of balance variable $y_{pq}(c)$, respectively. The net change of $y_{pq}(c)$ will therefore be $f_{pq} - f_{qp}$, that is,

$$y'_{pq}(c) = y_{pq}(c) + f_{pq} - f_{qp}. \tag{19}$$

Similarly, the net change of $y_{qp}(c)$ will be $f_{qp} - f_{pq}$ and, so, $y'_{pq}(c) = -y'_{qp}(c)$, that is, conjugate balance variables remain opposite to each other, as they should.

Due to (19), it is obvious that the capacity cap_{pq} of edge pq determines the maximum allowable value of $y'_{pq}(c)$ (attained at $f_{pq} = \text{cap}_{pq}$, $f_{qp} = 0$) and a similar conclusion holds for cap_{qp} and $y'_{qp}(c)$. However, for example, $y'_{pq}(c)$ represents the new partial raise of label c at p due to edge pq. Therefore, if the c-labels at p and q are not active (that is, $x_p \neq c$ and $x_q \neq c$) and may thus move, then cap_{pq} and cap_{qp} are set so that these c-labels cannot raise too much and violate (17), that is, they are set so that $y'_{pq}(c)$ and $y'_{qp}(c)$ cannot exceed $\frac{1}{2}w_{pq}d_{\min}$, thus ensuring that (17) holds true for new dual solution $\mathbf{y}'$ as well (see also Fig. 5d):

$$\text{cap}_{pq} + y_{pq}(c) = \frac{1}{2}w_{pq}d_{\min} = \text{cap}_{qp} + y_{qp}(c). \tag{20}$$

On the other hand, if c is already the active label of p (or q), then label c at p (or q) need not move (from their current positions) and, so, $y_{pq}(c)$ and $y_{qp}(c)$ should equal $y'_{pq}(c)$ and $y'_{qp}(c)$, that is (see also Fig. 5c)

$$x_p = c \ \text{ or } \ x_q = c \Rightarrow \text{cap}_{pq} = \text{cap}_{qp} = 0. \tag{21}$$

Exterior edges. Each internal node p connects to either the source node s or the sink node t (but not to both of them) through an exterior edge. We have three possible cases to consider:

Case 1 (*c is "below" x_p, that is,* $h_p(c) < h_p(x_p)$). We would then like to raise label c as much as needed so that it reaches label x_p (for example, see Fig. 5a). To this end, we connect source node s to node p through a directed edge sp. The flow f_p through that edge will then represent the total relative raise of label c, that is,[2]

$$h'_p(c) = h_p(c) + f_p. \tag{22}$$

2. To verify (22), it suffices to combine (19) with the flow conservation at node p, which reduces to $f_p = \sum_{q:q\sim p}(f_{pq} - f_{qp})$. It then holds that $h_p(c) + f_p \overset{(9)}{=} (\mathbf{c}_p(c) + \sum_{q:q\sim p} y_{pq}(c)) + f_p = (\mathbf{c}_p(c) + \sum_{q:q\sim p} y_{pq}(c)) + \sum_{q:q\sim p}(f_{pq} - f_{qp}) \overset{(19)}{=} (\mathbf{c}_p(c) + \sum_{q:q\sim p} y_{pq}(c)) + \sum_{q:q\sim p}(y'_{pq}(c) - y_{pq}(c)) = \mathbf{c}_p(c) + \sum_{q:q\sim p} y'_{pq}(c) \overset{(9)}{=} h'_p(c)$.

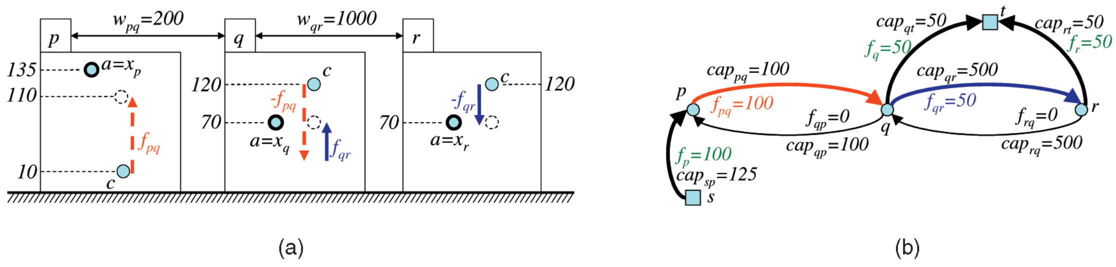


Fig. 6. (a) Arrows show how c-labels will move due to the update of balance variables. Dashed circles indicate new positions of c-labels. (b) Associated graph $\mathcal{G}^c$ and resulting flows responsible for the update of balance variables in the left figure (current balance variables were assumed to be 0). Based on the reassign rule, only vertex p will have to change its active label into c since only edge sp of $\mathcal{G}^c$ is unsaturated, whereas any path to q or r is not. This is indeed the right choice since, as can be seen (after the update), only p will have label c below its previous active label a. Also, as expected, flows f_p, f_q, and f_r at exterior edges are equal to the total relative movement of the c-labels at p, q, and r, respectively. (The Potts distance has been used in this example, that is, $a \neq b \Rightarrow d(a,b) = 1$.)

Therefore, based on (22), the capacity cap_{sp} of edge sp will represent the maximum allowable relative raise in the height of c. Since we need to raise c only as high as the current active label of p, but not higher than that, we therefore set $\text{cap}_{sp} = h_p(x_p) - h_p(c)$ (see Fig. 5a).

Case 2 (*c is not "below" x_p, that is, $h_p(c) \geq h_p(x_p)$ and not the active label of p, that is, $c \neq x_p$*). We can then afford a decrease in the height of c at p as long as c remains "above" x_p. To this end, we connect p to the sink node t through directed edge pt (for example, see Fig. 5b). This time, the flow f_p through edge pt will equal the total relative decrease in the height of c, that is,

$$h'_p(c) = h_p(c) - f_p \tag{23}$$

and, so, cap_{pt} will represent the maximum value of such a decrease. Therefore, based on the fact that c has to remain above x_p, we set $\text{cap}_{pt} = h_p(c) - h_p(x_p)$ (see Fig. 5b).

Case 3 (*c is the active label of p, that is, $c = x_p$*). We then want to keep the height of c fixed at the current iteration. As in Case 1, we again connect the source node s to node p through directed edge sp (see Fig. 5c). This time, however, the flow for any interior edge pq or qp incident to p will be zero (due to (21)). Therefore, $f_p = 0$ as well (due to flow conservation at p) and, so, $h'_p(c) = h_p(c)$ (see (22)), as was intended. By convention, we set $\text{cap}_{sp} = 1$.

4.1 Main Routines for Updating Primal and Dual Variables during a c-Iteration

We are now ready to summarize the main actions executed during an inner c-iteration of PD1:

PREEDIT_DUALS. This routine's role is to edit current solution $\mathbf{y}$ before the construction of the graph $\mathcal{G}^c$. In the case of the PD1 algorithm, no such editing is needed.

UPDATE_DUALS_PRIMALS. The primal-dual pair $(\mathbf{x}', \mathbf{y}')$ is generated here. For generating $\mathbf{y}'$, the graph $\mathcal{G}^c$ is constructed and a maximum flow algorithm is applied to it. The resulting flows are used in updating only the $y_{pq}(c)$ variables as explained in the previous section (see (19)), that is,

$$y'_{pq}(c) = y_{pq}(c) + f_{pq} - f_{qp}. \tag{24}$$

Therefore, due to (22) and (23), only the c-heights will change as follows:

$$h'_p(c) = h_p(c) + \begin{cases} f_p & \text{if } p \text{ is connected to node } s \\ -f_p & \text{if } p \text{ is connected to node } t. \end{cases} \tag{25}$$

Based on the new heights, we now need to update $\mathbf{x}$ into $\mathbf{x}'$, that is, assign new active labels. As only the c-heights have changed (that is, only c-labels may have gone above or below an active label), this amounts to deciding whether a vertex keeps its current active label or is assigned the label c. This can again be achieved by considering only the flows in $\mathcal{G}^c$ and applying the following rule:

REASSIGN RULE. *Label c will be the new label of p (that is, $x'_p = c$) $\Leftrightarrow \exists$ unsaturated*[3] *path between the source node s and node p. In all other cases, p keeps its current label, that is, $x'_p = x_p$.*

Intuitively, on one hand, this rule ensures that, if $c \neq x_p$, then (after the heights' update) the "lowest" of c and x_p is assigned to p (that is, it ensures Property 1 below so that (15) finally holds true). To see that, assume, for example, that path sp in Fig. 5a is unsaturated, that is, $f_p < \text{cap}_{sp}$. But then, since f_p equals the total relative raise of c (that is, $f_p = h'_p(c) - h_p(c)$) and $\text{cap}_{sp} = h_p(x_p) - h_p(c)$, it follows that $h'_p(c) < h_p(x_p) \overset{x_p \neq c}{=} h'_p(x_p)$. That is, label c should indeed be assigned to p as it is "lower" than the previous active label x_p (see also label c at p in Fig. 6 for another such example).

On the other hand, this rule also ensures that, if c is the new label assigned to p, then c has raised high enough so that (16) still holds. This is ensured by Property 2 below (together with $y'_{qp}(x'_q) \geq 0$ from (18) and the definition of cap_{pq} in (20)). The reason Property 2 holds is because, for example, in the previous example, if sp is unsaturated, then forward arc pq, as well as backward arc qp, must both be saturated (or else an unsaturated path from s to t can be shown to exist, which is impossible by max-flow min-cut). But then, $f_{pq} = \text{cap}_{pq}$ and $f_{qp} = 0$ and, so, Property 2 arises due to (24).

Due to the "reassign rule," these three properties can thus be proven [10] for the new solutions $\mathbf{x}'$ and $\mathbf{y}'$:[4]

1. $h'_p(x'_p) = \min\{h'_p(x_p), h'_p(c)\}$,
2. $x'_p = c \neq x'_q \Rightarrow y'_{pq}(x'_p) = \text{cap}_{pq} + y_{pq}(c)$, and
3. $\mathbf{x} \neq \mathbf{x}' \Rightarrow \text{APF}^{\mathbf{x}',\mathbf{y}'} < \text{APF}^{\mathbf{x},\mathbf{y}}$.

The last one (that is, Property 3) proves the algorithm terminates (assuming integer capacities) and the intuition for being true is due to the reassign rule, which ensures that

3. A path is unsaturated if "flow $<$ capacity" for all forward arcs and "flow > 0" for all backward arcs.

4. The reassign rule and, thus, Properties 1, 2, and 3 apply not only to PD1, but also to our other primal-dual algorithms.

```
INIT_PRIMALS: initialize $\mathbf{x}$ by a random label assignment

INIT_DUALS
• $\mathbf{y} = \mathbf{0}$
• for all $(p,q) \in \mathcal{E}$ with $x_p \neq x_q$ do   {impose (17)}
      $y_{pq}(x_p) = -y_{qp}(x_p) = w_{pq} d_{\min}/2 = y_{qp}(x_q) = -y_{pq}(x_q)$
• $\mathbf{z} = \min_a \mathbf{h}_a$   {impose (10)}

PREEDIT_DUALS: $\mathbf{y} \leftarrow \mathbf{y}$  {no change needed for $\mathbf{y}$}

UPDATE_DUALS_PRIMALS  {generate $\mathbf{x}', \mathbf{y}'$}
• $\mathbf{x}' = \mathbf{x},\ \mathbf{y}' = \mathbf{y}$
• Apply max-flow to $\mathcal{G}^c$ and compute flows $f_p, f_{pq}$
• $y'_{pq}(c) = y_{pq}(c) + f_{pq} - f_{qp} \quad \forall p,q : p \sim q$
• $\forall p \in \mathcal{V},\ x'_p = c \Leftrightarrow \exists$ unsaturated path $s \rightsquigarrow p$ in $\mathcal{G}^c$

POSTEDIT_DUALS  {edit $\mathbf{y}'$}
• for all $(p,q) \in \mathcal{E}$ with $x'_p = x'_q = c$ do   {impose (18)}
      if $y'_{pq}(c) < 0$ or $y'_{qp}(c) < 0$ then $y'_{pq}(c) = y'_{qp}(c) = 0$
• $\mathbf{z}' = \min_a \mathbf{h}'_a$   {impose (10)}
```

(a)

```
INIT_PRIMALS: initialize $\mathbf{x}$ at random

INIT_DUALS
• $\mathbf{y} = 0$
• for all $(p,q) \in \mathcal{E}$ with $x_p \neq x_q$ do   {impose (26)}
      $y_{pq}(x_p) = -y_{qp}(x_p) = \mu w_{pq} d(x_p, x_q)/2$
      $y_{qp}(x_q) = -y_{pq}(x_q) = \mu w_{pq} d(x_p, x_q)/2$
• $\mathbf{z} = \min_a \mathbf{h}_a$   {impose (10)}

PREEDIT_DUALS  {edit $\mathbf{y}$}
• for each $(p,q) \in \mathcal{E}$ with $x_p \neq c, x_q \neq c$ do
      $y_{qp}(c) = -y_{qp}(c) = \mu w_{pq} d(x_p, c) - y_{pq}(x_p)$

POSTEDIT_DUALS  {edit $\mathbf{y}'$}
• edit $\mathbf{y}'$ so that its active balance variables are $\geq 0$
• $\mathbf{z}' = \min_a \mathbf{h}'_a$   {impose (10)}

DUAL_FIT:  $\mathbf{y}^{\text{fit}} = \frac{\mathbf{y}}{\mu f_{\text{app}}}$
```

(b)

Fig. 7. (a) Pseudocode of PD1. (b) Pseudocode of PD2_μ. The routine UPDATE_DUALS_PRIMALS is common to both algorithms (and is thus shown only for PD1). Also, regarding the construction of $\mathcal{G}^c$, the only difference between the two algorithms is that a subset of the edges of $\mathcal{G}^c$ are assigned different capacities (see (29) and (30)).

a new active label always has lower height than the previous active label, that is, $h'_p(x'_p) \leq h_p(x_p)$.

POSTEDIT_DUALS. This routine's role is to restore (18) for the next iteration. It thus changes $\mathbf{y}'$ so that its active balance variables are all ≥ 0, whereas neither the APF nor any "load" is altered during this change. For PD1, one can show that only if $x'_p = x'_q$ (and never if $x'_p \neq x'_q$) may (18) then not hold, in which case, POSTEDIT_DUALS simply sets $y'_{pq}(x'_p) = y'_{qp}(x'_q) = 0$.

Based on the analysis above (see also the pseudocode in Fig. 7), the next theorem can thus be proven [10], asserting that PD1 always leads to an f_{app}-approximate solution:

Theorem 4.1. *The final primal-dual solutions generated by* PD1 *satisfy (15), (16), and (17) and, thus, they satisfy the relaxed complementary slackness conditions with $f_1 = 1$ and $f_2 = f_{\text{app}}$.*

5 The PD2 Algorithm

Algorithm PD2 (unlike PD1) applies only if $d(\cdot,\cdot)$ is a metric. In fact, PD2 represents a family of algorithms parameterized by a variable $\mu \in [\frac{1}{f_{\text{app}}} 1]$. PD2_μ will achieve slackness conditions (11) and (12) with $f_1 = \mu f_{\text{app}}$ and $f_2 = f_{\text{app}}$. The reason for $\mu \geq \frac{1}{f_{\text{app}}}$ is because $f_1 < 1$ can never hold.

A main difference between algorithms PD1 and PD2_μ is that PD1 always generates a feasible dual solution at any of its inner iterations, whereas PD2_μ may allow any such dual solution to become infeasible. However, PD2_μ ensures that the (probably infeasible) final dual solution is "*not too far away from feasibility*." Practically, this means that, if that solution is divided by a suitable factor, it will become feasible again. This method (that is, turning an infeasible dual solution into a feasible one by scaling) is also known as "*dual fitting*" [18] in the LP literature.

More specifically, PD2_μ generates a series of intermediate pairs, all of them satisfying complementary condition (12) as an equality with $f_2 = \frac{1}{\mu}$, that is,

$$x_p \neq x_q \Rightarrow \text{load}_{pq} = \mu w_{pq} d(x_p, x_q). \qquad (26)$$

In addition, using a similar strategy with PD1, PD2_μ drives the last intermediate pair toward satisfying complementary condition 11 with $f_1 = 1$ again, that is,

$$h_p(x_p) = \min_a h_p(a), \qquad (27)$$

whereas, also like PD1, it tries to maintain nonnegativity of active balance variables, that is, (18).

However, unlike PD1, the dual solution of the last intermediate pair may be infeasible since, in place of (8), it can be shown to satisfy only the following conditions:

$$y_{pq}(a) + y_{qp}(b) \leq 2\mu w_{pq} d_{max} \quad \forall a, b \in L,\ \forall (p,q) \in E. \qquad (28)$$

Nevertheless, these conditions ensure that the last dual solution, say $\mathbf{y}$, is not "too far away from feasibility." This means that, by replacing $\mathbf{y}$ with $\mathbf{y}^{\text{fit}} = \frac{\mathbf{y}}{\mu f_{\text{app}}}$, we can then show that

$$\begin{aligned} y^{\text{fit}}_{pq}(a) + y^{\text{fit}}_{qp}(b) &= \frac{y_{pq}(a) + y_{qp}(b)}{\mu f_{\text{app}}} \overset{(28)}{\leq} \frac{2\mu w_{pq} d_{\max}}{\mu f_{\text{app}}} \\ &= \frac{2\mu w_{pq} d_{\max}}{\mu 2 d_{\max}/d_{\min}} = w_{pq} d_{\min} \leq w_{pq} d_{ab}, \end{aligned}$$

which means that $\mathbf{y}^{\text{fit}}$ satisfies (8) and is thus feasible. Furthermore, the primal-dual pair $(\mathbf{x}, \mathbf{y}^{\text{fit}})$ ($\mathbf{x}$ is the last primal solution) satisfies complementary conditions (11) and (12) with $f_1 = \mu f_{\text{app}}$ and $f_2 = f_{\text{app}}$, thus leading to an f_{app}-approximate solution as well. Indeed, it holds that

$$z_p^{\text{fit}} \equiv \frac{z_p}{\mu f_{\text{app}}} \overset{(10)}{=} \frac{\min_a h_p(a)}{\mu f_{\text{app}}} \overset{(27)}{=} \frac{h_p(x_p)}{\mu f_{\text{app}}} = \frac{\boldsymbol{c}_p(x_p) + \sum_{q:q\sim p} y_{pq}(x_p)}{\mu f_{\text{app}}}$$
$$= \frac{\boldsymbol{c}_p(x_p)}{\mu f_{\text{app}}} + \sum_{q:q\sim p} y_{pq}^{\text{fit}}(x_p) \ ,$$
$$y_{pq}^{\text{fit}}(x_p) + y_{qp}^{\text{fit}}(x_q) = \frac{y_{pq}(x_p) + y_{qp}(x_q)}{\mu f_{\text{app}}}$$
$$= \frac{\text{load}_{pq}}{\mu f_{\text{app}}} \overset{(26)}{=} \frac{\mu w_{pq} d(x_p, x_q)}{\mu f_{\text{app}}} = \frac{w_{pq} d(x_p, x_q)}{f_{\text{app}}}.$$

The generation of $\mathbf{y}^{\text{fit}}$ (given $\mathbf{y}$) is exactly what the DUAL_FIT routine does.

5.1 Main Routines for Updating Primal and Dual Variables during a c-Iteration

PD2_μ routines (see Fig. 7) are mostly similar to those of PD1. The main difference (which is also the only difference regarding the construction of $\mathcal{G}^c$) is the definition of capacity for all interior edges pq and qp whose endpoints have labels $\neq c$ at the start of the current c-iteration, that is, $x_p \equiv a \neq c$ and $x_q \equiv b \neq c$. In place of (20), we then define

$$\text{cap}_{pq} = \mu w_{pq}\big(d(a,c) + d(c,b) - d(a,b)\big), \tag{29}$$

$$\text{cap}_{qp} = 0. \tag{30}$$

Furthermore, in this case, PREEDIT_DUALS edits $\mathbf{y}$ so that $y_{pq}(a) + y_{qp}(c) = \mu w_{pq} d(a,c)$.

The above difference is because, in PD1, the "reassign rule" needed to ensure that load_{pq} satisfied (16), whereas now, load_{pq} must fulfill (26), even if new labels are assigned by $\mathbf{x}'$ (that is, $\mathbf{x}' \neq \mathbf{x}$), which is exactly the rationale behind (29) and (30), as well as PREEDIT_DUALS. To see that, assume, for example, that $\mathbf{x}'$ assigns a new label c to q (that is, $x'_q = c \neq x_q$) but not to p (that is, $x'_p = x_p \neq c$), then

$$y'_{pq}(x'_p) \overset{x'_p \neq c}{=} y_{pq}(x'_p) \overset{x'_p = x_p}{=} y_{pq}(x_p) \overset{a \equiv x_p}{=} y_{pq}(a) \quad \text{and}$$
$$y'_{qp}(x'_q) \overset{\text{property B}}{=} \text{cap}_{qp} + y_{qp}(c) \overset{(30)}{=} y_{qp}(c).$$

Combining these with the definition of PREEDIT_DUALS immediately proves that (26) remains true in this case (with other cases being handled similarly as well). Finally, as in PD1, the role of POSTEDIT_DUALS is again to restore (18), that is, nonnegativity of active balance variables. Also, note that (29) explains why $d(\cdot,\cdot)$ must be a metric (or else it would hold that $\text{cap}_{pq} < 0$).

5.2 Equivalence of Algorithms $\text{PD2}_{\mu=1}$ and α-Expansion

It can thus be shown that PD2_μ indeed generates an f_{app}-approximate solution. Furthermore, it holds that all PD2_μ algorithms with $\mu < 1$ are nongreedy algorithms, which means that neither the primal nor the dual objective function necessarily decreases or increases per iteration. Instead, it is APF that constantly decreases (see Property 3 in Section 4.1), but, since APF is always kept close to the primal function, the decrease in APF is finally reflected to the values of the primal function as well. In fact, a notable thing happens if $\mu = 1$. In that case, due to (26), the load of any p, q equals exactly their separation cost (that is, $\text{load}_{pq} = w_{pq} d(x_p, x_q)$) and it can then be shown that APF coincides with the primal function, that is, $\text{APF} = \text{PRIMAL}$, whereas, in any other case, $\text{APF} \leq \text{PRIMAL}$. Furthermore, it turns out that, during a c-iteration, $\text{PD2}_{\mu=1}$ chooses an $\mathbf{x}'$ that minimizes APF with respect to any other c-expansion, say $\bar{\mathbf{x}}$, of the current solution $\mathbf{x}$ (to see that, recall that APF is the sum of active labels' heights and $\text{PD2}_{\mu=1}$ always tries choosing the "lowest" label among x_p and c, see Property 1). All these can be formally summarized in the following lemma [10]:

Lemma 5.1. *Let* $(\mathbf{x}', \mathbf{y}') \equiv$ *next primal-dual pair due to c-iteration and $\bar{\mathbf{x}} \equiv c$-expansion of current primal. Then,*

$$\text{PRIMAL}^{\mathbf{x}'} = \text{APF}^{\mathbf{x}',\mathbf{y}'} \leq \text{APF}^{\bar{\mathbf{x}},\mathbf{y}'} \leq \text{PRIMAL}^{\bar{\mathbf{x}}},$$
$$(\text{PRIMAL}^{\mathbf{x}} \equiv \text{ primal cost of } \mathbf{x}).$$

But this, due to $\text{PRIMAL}^{\mathbf{x}'} \leq \text{PRIMAL}^{\bar{\mathbf{x}}}$, actually proves that the c-expansion algorithm in [1] (which was interpreted only as a greedy local search technique up to now) is equivalent to $\text{PD2}_{\mu=1}$!

Theorem 5.2 [10]. *The label assignment $\mathbf{x}'$ selected during a c-iteration of $\text{PD2}_{\mu=1}$ has a smaller primal cost than any other label assignment $\bar{\mathbf{x}}$, which is a c-expansion of current solution $\mathbf{x}$.*

6 PD3: Extending PD2 to the Nonmetric Case

By modifying PD2_μ, three different variations (PD3_a, PD3_b, and PD3_c) may result, which are applicable even if $d(\cdot,\cdot)$ is a nonmetric distance function. For simplicity, we will consider only the $\mu = 1$ case, that is, only the variations of $\text{PD2}_{\mu=1}$. We also recall a fact that will prove to be useful for explaining the rationale behind the algorithms' definition: The load between any p, q represents a virtual separation cost, which should be equal to the actual separation cost of p and q if the current primal-dual solutions are optimal (OPTIMALITY CRITERION).

The main difficulty of extending $\text{PD2}_{\mu=1}$ to the nonmetric case relates to all edges pq with capacity defined by (29) during a c-iteration, that is, all interior edges pq whose endpoints p and q are currently assigned labels $\neq c$ (that is, $x_p \equiv a \neq c$ and $x_q \equiv b \neq c$), whereas, in addition, the following inequality holds: $d(a,b) > d(a,c) + d(c,b)$. Hereafter, we will call any such pair (p,q) a "*conflicting pair*" and the corresponding labels (a,b,c) a "*conflicting label-triplet*." Depending on the way we deal with such a "conflicting pair," three different variations of $\text{PD2}_{\mu=1}$ may arise.

PD3_a algorithm. We choose to set $\text{cap}_{pq} = 0$ in place of (29). In this case, it can be shown that if $\mathbf{x}'$ assigns the pair of labels c and b to the objects p and q, respectively, then the resulting load of p and q will be $w_{pq}(d(a,b) - d(a,c))$, that is, it will be greater than the actual separation cost $w_{pq} d(c,b)$ of p and q because $d(a,b) > d(a,c) + d(c,b)$ as (a,b,c) is a "conflicting label-triplet." Equivalently, this says that the virtual separation cost of p, q overestimates their actual separation cost, contrary to the OPTIMALITY CRITERION mentioned above (in all other cases, one can prove that there is no such overestimation). Therefore, in this case, POSTEDIT_DUALS modifies the dual variables so that the equality between the load and the actual separation cost is restored and, thus, the violation of the OPTIMALITY CRITERION is canceled by the start of the next iteration. No other differences between $\text{PD2}_{\mu=1}$ and PD3_a exist.

One may also view this cost overestimation as an equivalent overestimation of the corresponding distance between labels. In the above case, for example, we saw that, if labels c and b are assigned to p and q by $\mathbf{x}'$, then, instead of the actual separation cost $w_{pq}d(c,b)$, the resulting overestimated cost would have been $w_{pq}\bar{d}(c,b)$, with $\bar{d}(c,b) = d(a,b) - d(a,c)$. This is equivalent to saying that the algorithm has assigned the virtual distance $\bar{d}(c,b) > d(c,b)$ to labels c and b instead of their actual distance $d(c,b)$. On the other hand, if (a,b) or (a,c) are assigned to p and q by $\mathbf{x}'$, then no cost overestimation takes place and, so, the virtual distances for these labels coincide with their actual distances, that is, $\bar{d}(a,b) = d(a,b), \bar{d}(a,c) = d(a,c)$. *Since $\bar{d}(a,c) + \bar{d}(c,b) = \bar{d}(a,b)$, one could then argue that, by replacing d with $\bar{d}$, what $PD3_a$ actually did was to overestimate the distance between labels c, b in order to restore the triangle inequality for the current "conflicting label-triplet"* (a,b,c). Put otherwise, it is as if a "dynamic approximation" of the nonmetric d by a varying metric $\bar{d}$ is taking place, with this metric $\bar{d}$ being constantly modified. Note also that, for restoring the triangle inequality, we could have instead designed our algorithm so that it overestimates the distance between labels a and c in place of that between c and b. Not only that, but we could have also defined an application-dependent function, say RESOLVE, which would decide (based on the current "conflicting pair") which one of the two distances (that is, $d(a,c)$ or $d(c,b)$) should be overestimated each time.

Based on these observations, it can be shown [10] that the primal-dual solutions generated by both PD3_a and $\text{PD2}_{\mu=1}$ satisfy exactly the same conditions (26), (27), and (28) and, so, PD3_a is always guaranteed to lead to an f_{app}-approximate solution as well. *Therefore, PD3_a directly generalizes $\text{PD2}_{\mu=1}$ (that is, the α-expansion) to the case of a nonmetric distance function $d(\cdot,\cdot)$.* We should note here that, recently, Rother et al. [19] have also described an extension of the α-expansion technique, which can be applied to the nonmetric case and seems related to our PD3 method.

PD3_b algorithm. We choose to set $\text{cap}_{pq} = +\infty$ and no further differences between PD3_b and $\text{PD2}_{\mu=1}$ exist. This has the following important effect: *The solution $\mathbf{x}'$ produced at the current iteration can never assign the pair of labels c, b to the objects p and q, respectively (due to this fact, we will call labels c and b the "excluded labels"*[5]). To prove this, it suffices to recall the "reassign rule" and also observe that the directed edge pq can never become saturated by increasing its flow (since $\text{cap}_{pq} = +\infty$). Therefore, if label c is assigned to p by $\mathbf{x}'$ (which, by the "reassign rule," means that there is an unsaturated path $s \rightsquigarrow p$), then label b can never be assigned to q since, in that case, the path $s \rightsquigarrow p \rightarrow q$ would also be unsaturated (since $\text{cap}_{pq} = +\infty$) and, by the "reassign rule" again, q would have to be assigned label c as well. Put otherwise, it is as if an infinite overestimation of the distance $d(c,b)$ between labels c and b takes place by the algorithm and, so, those labels are implicitly prevented from being assigned to the "conflicting pair." The price for that is that no guarantees about the algorithm's optimality can be provided. The reason is that the balance variables may now increase without bound (since $\text{cap}_{pq} = +\infty$) and, so, we cannot make sure that the generated dual solutions satisfy a "not too far away from feasibility" condition like (28). This, in turn, implies that no dual-fitting technique can be applied in this case. However, PD3_b has a nice interpretation in the primal domain due to the following theorem:

Theorem 6.1 [10]. *The solution $\mathbf{x}'$, selected by PD3_b during a c-iteration has the minimum primal cost among all solutions that result after a c-expansion of current solution $\mathbf{x}$, except for those that assign "excluded labels" to "conflicting pairs."*

This theorem designates the price we pay for $d(\cdot,\cdot)$ not being a metric: In the metric case, we can choose the best assignment among all c-expansion moves (see Theorem 5.2), whereas, in the nonmetric case, we are only able to choose the best one among a certain subset of these c-expansion moves. Despite this fact, the considered subset contains an exponential number of c-expansion moves, which makes the algorithm a perfect candidate as a local minimizer.

PD3_c algorithm. PD3_c first adjusts (if needed) the dual solution $\mathbf{y}$ so that, for any two neighbors p and q, it holds that $\text{load}_{pq} \le w_{pq}(d(a,c) + d(c,b))$. After this initial adjustment, which is always easy to achieve, PD3_c proceeds exactly as $\text{PD2}_{\mu=1}$, except for the fact that the term $d(a,b)$ in (29) is replaced with the distance $\bar{d}(a,b)$, which is defined as $\bar{d}(a,b) = \frac{\text{load}_{pq}}{w_{pq}}$. Obviously, $\bar{d}(a,b) \le d(a,c) + d(c,b)$ and, so, cap_{pq} in (29) is valid, that is, $\text{cap}_{pq} \ge 0$. PD3_c and $\text{PD2}_{\mu=1}$ have no other differences.

It is now interesting to examine what happens if (p,q) is a "conflicting pair" with current labels a and b (that is, $x_p \equiv a \ne c$ and $x_q \equiv b \ne c$). In that case, it also holds that $d(a,c) + d(c,b) < d(a,b)$ and, so,

$$\bar{d}(a,b) = \frac{\text{load}_{pq}}{w_{pq}} \le \frac{w_{pq}\big(d(a,c) + d(c,b)\big)}{w_{pq}} < \frac{w_{pq}d(a,b)}{w_{pq}} = d(a,b).$$

Furthermore, it is easy to show that, if none of p and q is assigned a new label by $\mathbf{x}'$ (that is, they both retain their current labels a and b), then the resulting load will be equal to $w_{pq}\bar{d}(a,b)$, that is, it will underestimate the actual separation cost $w_{pq}d(a,b)$ since $\bar{d}(a,b) < d(a,b)$, as was shown above (in all other cases, the load will coincide with the actual separation cost).

Based on these observations, one can then see that the PD3_c algorithm works in a complementary way to the PD3_a algorithm: In order to restore the triangle inequality for the "conflicting label-triplet" (a,b,c), instead of overestimating the distance between either labels (c,b) or (a,c) (like PD3_a did), it chooses to underestimate the distance between labels (a,b). Again, one may view this as a "dynamic approximation" of the nonmetric d by a constantly varying metric $\bar{d}$; this time, however, we set $\bar{d}(a,b) = \frac{\text{load}_{pq}}{w_{pq}} < d(a,b)$, $\bar{d}(a,c) = d(a,c)$, and $\bar{d}(c,b) = d(c,b)$.

It can be shown that the intermediate primal-dual solutions generated by algorithms PD3_c and $\text{PD2}_{\mu=1}$ satisfy exactly the same conditions, except for (26). In place of that condition, the intermediate solutions of PD3_c satisfy

$$\text{load}_{pq} \ge w_{pq}\hat{d}(x_p, x_q), \tag{31}$$

where $\hat{d}(a,b) = \min_{c \in \mathcal{L}}(d(a,c) + d(c,b))$. By then applying the same (as in $\text{PD2}_{\mu=1}$) dual fitting factor to the last dual

5. Note that, as in PD3_a, we can modify PD3_b so that a function RESOLVE chooses which labels (that is, (a,c) or (c,b)) are "excluded" each time. Moreover, RESOLVE could perhaps be defined based on a priori knowledge about each specific problem.

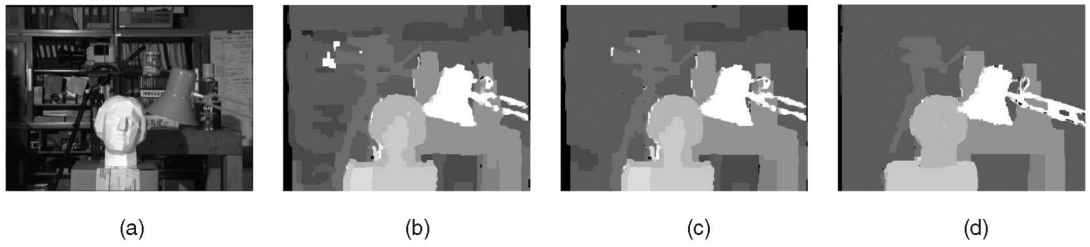


Fig. 8. (a) Tsukuba image. (b) Disparity estimated by the PD1 and (c) $\mathrm{PD2}_{\mu=1}$ algorithm. The Potts distance (a metric) was used and, so, $\mathrm{PD3}_a$, $\mathrm{PD3}_b$, and $\mathrm{PD3}_c$ produced the same result with the $\mathrm{PD2}_{\mu=1}$. No tuning of parameters took place in (b) and (c). (d) Our result when using the same parameters as in [21] (following the notation in [21], we have used $s = 50$, $T = 4$, and $P = 2$).

solution of $\mathrm{PD3}_c$, one can easily prove that $\mathrm{PD3}_c$ leads to an f'_{app}-approximate solution where

$$f'_{\mathrm{app}} = f_{\mathrm{app}} \cdot c_0 \quad \text{with} \quad c_0 = \max_{a \neq b} \frac{d(a,b)}{\hat{d}(a,b)}. \tag{32}$$

Finally, we should note that, if d_{ab} is a metric, then $\mathrm{PD3}_a$, $\mathrm{PD3}_b$, and $\mathrm{PD3}_c$ all coincide with $\mathrm{PD2}_{\mu=1}$.

7 Experimental Results

We first describe certain properties of the proposed algorithms that prove to be very useful in practice (Section 7.1). We then proceed to demonstrate our algorithms' effectiveness in MRF optimization. To this end, we apply them to a variety of low-level vision tasks, such as stereo matching (Sections 7.1 and 7.2), image restoration (Section 7.3), and image completion (Section 7.3), as well as optical flow estimation (Section 7.4). Finally, to further analyze their performance, results on synthetic problems are shown in Section 7.5. We note that, in each experiment, identical settings (that is, parameters and initial solution) have been used for all algorithms and, in addition, initialization was chosen randomly.

7.1 Per-Instance Suboptimality Bounds

An important advantage of any primal-dual algorithm is that, after its execution, it can always tell (for free) how well it performed with respect to any given instance of ML. In particular, as implied by the Primal-Dual Principle of Section 3, given any pair $(\mathbf{x}, \mathbf{y})$ of integral-primal dual feasible solutions, then the ratio $r = \mathbf{c}^T\mathbf{x}/\mathbf{b}^T\mathbf{y}$ of their costs automatically provides a new suboptimality bound in the sense that $\mathbf{x}$ is then guaranteed to be an r-approximation to the optimal integral solution. This leads to the following consequence: *By considering all primal-dual solutions $\{\mathbf{x}^k, \mathbf{y}^k\}_{k=1}^t$ generated during the primal-dual schema, the quantity $\min_k r_k$ (where $r_k \equiv \mathbf{c}^T\mathbf{x}^k/\mathbf{b}^T\mathbf{y}^k$) defines a new per-instance suboptimality bound.*

In practice, this per-instance bound turns out to be much tighter (that is, much closer to 1) than the worst-case bound predicted in theory and, so, this allows one to have a much clearer view about the goodness of the generated solution. This has been verified experimentally by applying our algorithms to the stereo matching problem. In this case, labels correspond to image pixel disparities and they can be chosen from a set $\mathcal{L} = \{0, 1, \ldots, K\}$ of discretized disparities, where K denotes the maximum allowable disparity. The vertices of the graph $\mathcal{G}$ are the image pixels and the edges of $\mathcal{G}$ connect each pixel to its four immediate neighbors in the image. During our tests, the label cost for assigning disparity a to the image pixel p has been set equal to

$$\boldsymbol{c}_p(a) = |I_{\mathrm{right}}(p-a) - I_{\mathrm{left}}(p)|, \tag{33}$$

where I_{left} and I_{right} represent the intensities of the left and right images, respectively.

We have applied our algorithms to the well-known Tsukuba stereo data set [20], setting the maximum disparity value equal to $K = 14$ based on the provided ground truth data. Samples from the results produced when using our algorithms are shown in Fig. 8. We should note that no special tuning of parameters took place and all edge weights w_{pq} have been set equal to each other, instead of properly adjusting their values based on image intensity edges (which would improve the results considerably for this specific example; for example, see Fig. 8d for one such result produced with our method). The reason for this, as well as for using the very simple label cost presented in (33), is because our main goal was not to produce the best possible disparity estimation, but to test the tightness of the suboptimality bounds that are provided by our algorithms, that is, to test the effectiveness of these algorithms in minimizing the objective function.

To this end, three different distances $d(\cdot,\cdot)$ have been used during our experiments. These are the Potts distance d_1 (a metric), the truncated linear distance d_2 (also a metric), and the truncated quadratic distance d_3 (a nonmetric), which are defined as follows (where λ denotes some constant):

$$d_1(a,b) = 1 \ \forall a \neq b, \quad d_2(a,b) = \min(\lambda, |a-b|),$$
$$d_3(a,b) = \min(\lambda, |a-b|^2).$$

Each experiment consisted of selecting an approximation algorithm and a distance function and then using them for computing disparities for each one of the Tsukuba stereo pairs. The average values (over all Tsukuba stereo pairs) of the obtained suboptimality bounds are displayed in Table 1. The columns $f_{\mathrm{app}}^{\mathrm{PD1}}$, $f_{\mathrm{app}}^{\mathrm{PD2}_{\mu=1}}$, $f_{\mathrm{app}}^{\mathrm{PD3}_a}$, $f_{\mathrm{app}}^{\mathrm{PD3}_b}$, and $f_{\mathrm{app}}^{\mathrm{PD3}_c}$ of that table list these averages for the algorithms PD1, $\mathrm{PD2}_{\mu=1}$, $\mathrm{PD3}_a$, $\mathrm{PD3}_b$, and $\mathrm{PD3}_c$, respectively. In addition, the last column lists the value of the corresponding approximation factor f_{app}, which, as already proven, makes up a worst-case suboptimality bound for most of the above algorithms. By observing Table 1, one can conclude that the per-instance suboptimality bounds are often much tighter (that is, much closer to 1) than the worst-case bounds predicted in theory. In our stereo experiments, this was true for all combinations

TABLE 1
Average Suboptimality Bounds (Columns 2-6) Obtained over All Tsukuba Stereo Pairs

Distance	$f_{\mathrm{app}}^{\mathrm{PD1}}$	$f_{\mathrm{app}}^{\mathrm{PD2}_{\mu=1}}$	$f_{\mathrm{app}}^{\mathrm{PD3}_a}$	$f_{\mathrm{app}}^{\mathrm{PD3}_b}$	$f_{\mathrm{app}}^{\mathrm{PD3}_c}$	f_{app}
Potts	1.0104	1.0058	1.0058	1.0058	1.0058	2
Trunc. Linear $^{\lambda=5}$	1.0226	1.0104	1.0104	1.0104	1.0104	10
Trunc. quad. $^{\lambda=5}$	1.0280	-	1.0143	1.0158	1.0183	10

As expected, these bounds are much closer to 1 than the theoretical suboptimality bounds f_{app} listed in the last column and, thus, a nearly optimal solution is obtained in all cases. Note that $\mathrm{PD2}_{\mu=1}$ can be applied only if distance $d(\cdot,\cdot)$ is a metric and, in that case, $\mathrm{PD2}_{\mu=1}$, $\mathrm{PD3}_a$, $\mathrm{PD3}_b$, and $\mathrm{PD3}_c$ (as well as their bounds) coincide.

TABLE 2
The Average Suboptimality Bounds (Columns 2, 4, 6, 8, and 10) Obtained When Applying Our Stereo Matching Algorithms to One Scan Line at a Time (Instead of the Whole Image)

Distance	$f_{\mathrm{app}}^{\mathrm{PD1}}$	$f_{\mathrm{true}}^{\mathrm{PD1}}$	$f_{\mathrm{app}}^{\mathrm{PD2}_{\mu=1}}$	$f_{\mathrm{true}}^{\mathrm{PD2}_{\mu=1}}$	$f_{\mathrm{app}}^{\mathrm{PD3}_a}$	$f_{\mathrm{true}}^{\mathrm{PD3}_a}$	$f_{\mathrm{app}}^{\mathrm{PD3}_b}$	$f_{\mathrm{true}}^{\mathrm{PD3}_b}$	$f_{\mathrm{app}}^{\mathrm{PD3}_c}$	$f_{\mathrm{true}}^{\mathrm{PD3}_c}$
Potts	1.0098	1.0036	1.0066	1.0004	1.0066	1.0004	1.0066	1.0004	1.0066	1.0004
Trunc. Linear	1.0202	1.0107	1.0115	1.0021	1.0115	1.0021	1.0115	1.0021	1.0115	1.0021
Trunc. quad.	1.0255	1.0130	-	-	1.0135	1.0011	1.0144	1.0020	1.0160	1.0036

In this case, we are also able to compute the true average suboptimality (Columns 3, 5, 7, 9, and 11) of the generated solutions using dynamic programming. As can be seen, by inspecting the table, the suboptimality bounds approximate the true suboptimality relatively well, which means that they can be safely used as a measure for judging the goodness of the generated solution in this case.

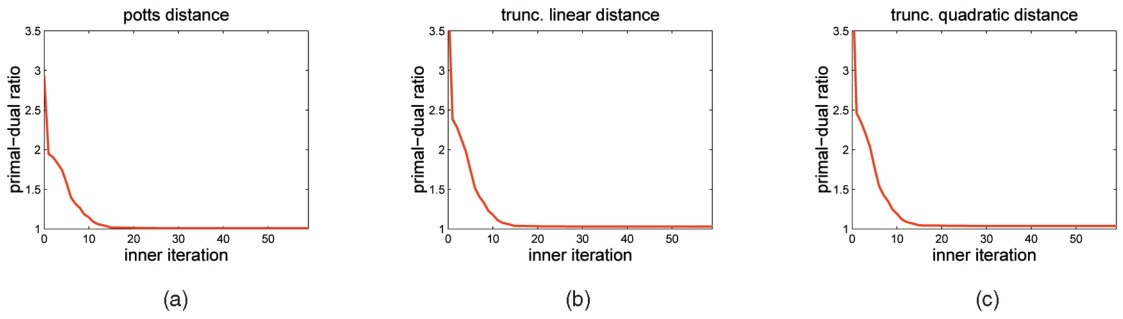
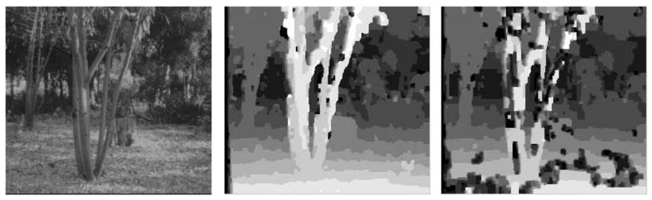


Fig. 9. These three plots show how the primal-dual ratios vary during the first four outer iterations (or, equivalently, the first $60 = 4 \cdot 15$ inner iterations) using the Tsukuba sequence as input. (a) The Potts function, (b) the truncated linear function, and (c) the truncated quadratic function have been used, respectively, as label distance $d(\cdot,\cdot)$. Notice how rapidly the ratios drop in all cases (that is, they get very close to 1 just after a few inner iterations).

of algorithms and distances and, so, in this particular case, *the presented algorithms were able to extract a nearly optimal solution even when a nonmetric distance was used*.

Besides the tightness of the per-instance suboptimality bounds, another important issue is their accuracy, that is, how well these bounds predict the true suboptimality of the generated solutions. To investigate this issue, we modified our experiments in the following way: We applied our stereo matching algorithms to one image scan line at a time (instead of the whole image). In this case, the graph $\mathcal{G}$ reduces to a chain and the true optimum can be easily computed using dynamic programming. This, in turn, implies that we are able to compute the true suboptimality of a solution. By using this fact, we have thus constructed Table 2. Its columns $f_{\mathrm{true}}^{\mathrm{PD1}}$, $f_{\mathrm{true}}^{\mathrm{PD2}_{\mu=1}}$, $f_{\mathrm{true}}^{\mathrm{PD3}_a}$, $f_{\mathrm{true}}^{\mathrm{PD3}_b}$, and $f_{\mathrm{true}}^{\mathrm{PD3}_c}$ contain the true average suboptimality of the solutions of PD1, $\mathrm{PD2}_{\mu=1}$, $\mathrm{PD3}_a$, $\mathrm{PD3}_b$, and $\mathrm{PD3}_c$, respectively, where the average is taken over all image scan lines. By examining that table, one may easily conclude that (for this particular experiment) the true suboptimality of an algorithm's solution was close to the corresponding estimated suboptimality bound, which means that these bounds were relatively accurate and, therefore, reliable for judging the solution's goodness. Furthermore, in this way, we can decide if a badly generated solution is the result of a bad optimization procedure or a bad modeling of the problem at hand. At this point, however, we should also note that one must be careful in extrapolating results on scan lines to that on grids. One potential issue is that, in the former case, the integrality gap is 1, whereas, in the latter, it may be greater than 1, which may contribute to the inaccuracy of the suboptimality bound for graphs with loops.

For the Tsukuba sequence, on the average, four outer iterations (or, equivalently, $60 = 4 \cdot 15$ inner iterations) are needed for the algorithms to terminate. The corresponding running time is 46 s (measured on a 2.4 GHz CPU). The plots in Fig. 9 show how the primal-dual ratios vary during the execution of our algorithms (for the Tsukuba data set). For the first two plots, a metric distance between labels has



(a) (b) (c)

Fig. 10. (a) One image from the SRI tree image sequence. (b) Computed disparities when using $\mathrm{PD3}_a$ and the distance d_4 with $(\kappa, \lambda) = (2, 10)$. (c) Disparities computed by the α-β-swap algorithm using the same distance.

been used, whereas, for the last one, a nonmetric distance has been chosen. It is worth noticing how rapidly the primal-dual ratios drop in all cases. They come very close to 1 just after a few inner iterations, which means that the algorithms converge really fast while computing an almost optimal solution at the same time. Based on this observation, one may also use the values of these ratios to control the algorithms' convergence (for example, if the ratios are close to 1 and do not vary too much per iteration, one may decide that convergence has been reached). This way, one may further reduce the running times.

7.2 Stereo Matching

Besides the Tsukuba data set, we have also applied our algorithms to image pairs from the Stanford Research Institute (SRI) tree image sequence (Fig. 10a). The selected pairs had a maximum disparity of 11 pixels. Given our algorithms' ability to handle both metric and nonmetric distances equally well, the following nonmetric distance has been used in this case: $d_4(a,b) = |a-b|$ if $|a-b| <= \kappa$; otherwise, $d_4(a,b) = \lambda$. We always assume $\kappa < \lambda$. In this specific example, we have used $(\kappa, \lambda) = (2, 10)$. The rationale behind this distance is that it assigns a low penalty to small (that is, $\leq \kappa$) changes in disparity (thus allowing surfaces with smoothly varying disparity, like the slanted ground in the SRI image) but assigns a high penalty λ to large disparity gaps. Despite the fact that d_4 is not a metric, our algorithms did not face any problem in efficiently minimizing the corresponding objective function and thus localizing the trees, as well as the slanted ground in the SRI image. The resulting disparity is shown in Fig. 10b. The average running time to convergence has been 33 s. We have also applied the α-β-swap algorithm [1] to the SRI data set, using exactly the same settings. Although this graph-cut-based algorithm is applicable even in the case of a nonmetric label distance, its disadvantage is that it may get trapped to a bad local minimum, that is, it cannot make any guarantees about the optimality of the solutions it generates. This is indeed the case here since, despite the fact that exactly the same objective function has been minimized by both algorithms, the final energy produced by α-β-swap was 8.3 percent higher than the energy estimated by our method. The corresponding disparity is shown in Fig. 10c.

As a further example, we illustrate how one could favor disparities that are not violating the uniqueness constraint, just by the use of an appropriate nonmetric distance $d(\cdot,\cdot)$. This can possibly lead to a better handling of occlusions as well in some cases. To this end, an extra label for occlusions, say $\hat{o}$, is introduced first whose label cost is equal to $\boldsymbol{c}_{\hat{o}}$ for all pixels, that is, $\boldsymbol{c}_p(\hat{o}) = \boldsymbol{c}_{\hat{o}}$. Assuming (without loss of generality) that image scan lines coincide with the epipolar lines, we then introduce additional horizontal edges in the graph $\mathcal{G}$: We connect any pixel (x,y) in the left image to the K pixels to its right $(x+1,y),\ldots,(x+K,y)$, where K is the maximum disparity (see Fig. 11a). For measuring the separation cost between the labels of (x,y), $(x+k,y)$, we will use the distance function hdist^k. We will therefore use K different distance functions in total for all of the horizontal edges. On the other hand, no additional vertical edges are introduced and, so, any pixel will be connected only to its immediate vertical neighbors, as before, with vdist^1 denoting the common distance function for all these edges.

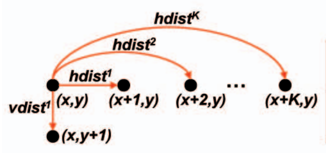
Distances hdist^1 and vdist^1 (which are related to edges connecting pixels adjacent in the image) will be used for enforcing the smoothness of the disparity field, as before. For example, both can be set equal to the Potts metric: $\mathrm{hdist}^1 = \mathrm{vdist}^1 = d_1$. The rest of hdist^k will be used just for assigning an extra penalty M to all pairs of labels violating the uniqueness constraint. For all other pairs of distinct labels, hdist^k then simply assigns a very small distance ε (with $\varepsilon \ll M$):

$$b = a + k \Rightarrow \mathrm{hdist}^k(a,b) = M,$$
$$b \neq a + k \;\&\; b \neq a \Rightarrow \mathrm{hdist}^k(a,b) = \varepsilon,$$
$$b = a \Rightarrow \mathrm{hdist}^k(a,b) = 0.$$

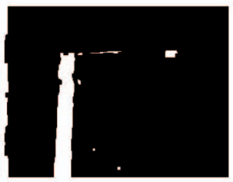
Fig. 11 contains the result of applying this distance (with $\boldsymbol{c}_{\hat{o}} = 23, M = 10, \varepsilon = 0.01$) to the well-known *map* stereo pair (which appears in Fig. 7 in [20]). Error statistics are shown in Fig. 11b.

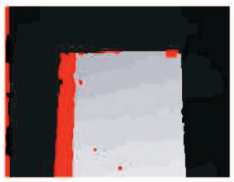
7.3 Image Restoration and Image Completion

In image restoration, we are given as input a corrupted (by noise) image and the objective is to extract the original (uncorrupted) image. In this case, the labels represent intensities (or colors), while the label cost for assigning



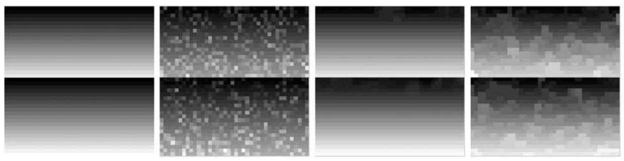
Percentage of pixels with absolute error > 1 in non-occluded regions	1.14%
Percentage of missed occluded pixels in occluded regions	12.31%





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

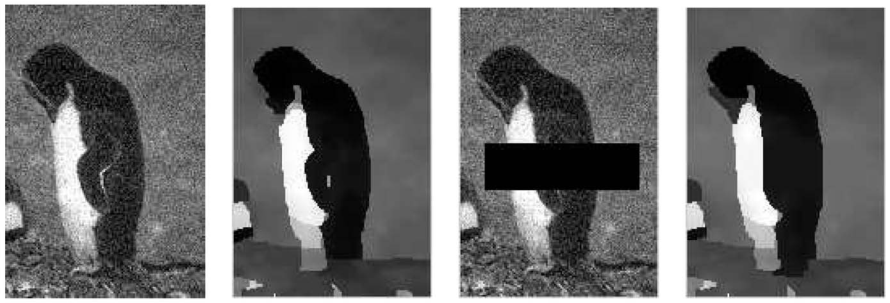
Fig. 11. White pixels in (c) indicate occlusions. (a) Additional edges in $\mathcal{G}$. (b) Error statistics for the "*map*" pair. (c) Detected occlusions for the "*map*" pair. (d) Estimated disparity for nonoccluded pixels of the "*map*" pair.



Metric $d(\cdot,\cdot)$	Non-metric $d(\cdot,\cdot)$
41.4% pixels with error	8.2% pixels with error
1.12 avg. intensity error	0.21 avg. intensity error

(a) (b) (c) (d) (e) (f)

Fig. 12. (a) Original uncorrupted image. (b) Noisy input image. (c) Restored image using nonmetric distance d_4 with $(\kappa, \lambda) = (2, 30)$. (d) Restored image using truncated linear metric d_2 with $\lambda = 30$. (e) Error statistics.



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

Fig. 13. Examples of image restoration and image completion. (a) Noisy input. (b) Restored. (c) Noisy input with mask. (d) Restored and completed.

intensity a to pixel p can be set equal to $\boldsymbol{c}_p(a) = |I(p) - a|$, where I represents the array of intensities of the input image. The graph $\mathcal{G}$, which will be used when solving the ML problem, coincides again with the image grid.

The example of Fig. 12 illustrates the importance of using nonmetric distances $d(\cdot,\cdot)$ on the task of image restoration as well. The original image (Fig. 12a) consists of two identical patterns placed vertically. Each pattern's intensity is kept constant along the horizontal direction and increases linearly with step 2 from top to bottom. The input image is then formed by corrupting the original image with white noise (Fig. 12b). Although our algorithms managed to restore the original image with only a few errors by the use of the nonmetric distance d_4 (Fig. 12c), this was not the case when the truncated linear metric d_2 (or the Potts metric) has been used, despite tweaking the λ parameter. The best obtained result with such a metric (after tweaking λ) is shown in Fig. 12d. The error statistics for this restoration example are shown in Fig. 12e.

Another nonmetric distance, which is very commonly used in image restoration problems, is the truncated quadratic distance $d_3(a,b) = \min(|a-b|^2, \lambda)$. That distance with $\lambda = 200$ was used in the restoration of the contaminated (with Gaussian noise) image of Fig. 13a. In this case, the following function (which is more robust against outliers) has been used for the label costs: $\boldsymbol{c}_p(a) = \lambda_0 \min(|I(p) - a|^2, \lambda_1)$, with $\lambda_0 = 0.05$ and $\lambda_1 = 10^4$. Notice that our algorithm managed not only to remove the noise completely (see Fig. 13b), but also to maintain the boundaries of the objects at the same time.

The same distance (that is, the truncated quadratic) can also be used for the task of image completion. Besides containing Gaussian noise, the image in Fig. 13c also has a part that has been masked. The label costs of masked pixels have been set to zero, whereas, for the rest of the pixels, the costs have been set as before. As can be seen in Fig. 13d, our algorithm managed not only to remove the noise again, but also to fill the missing part in a plausible way.

7.4 Optical Flow Estimation

Global methods [22], [23] estimate optical flow u_x, u_y by minimizing a function of the form

$$E(u_x, u_y) = \int_I \rho_D(I_x u_x + I_y u_y + I_t) + \lambda \cdot \rho_S\left(\sqrt{|\nabla u_x|^2 + |\nabla u_y|^2}\right) dx dy,$$

where I_x, I_y, and I_t denote spatial and temporal image derivatives, whereas ρ_D, ρ_S denote penalty functions. By discretizing $E(u_x, u_y)$, we can easily incorporate all such methods into our framework: The first term (which expresses the *optic flow constraint equation*) and the second term (which is a regularizer) will then correspond to the label costs and separation costs, respectively. Furthermore, due to our weak assumptions on $d(\cdot,\cdot)$, our framework allows us to set ρ_S equal to any of the so-called *robust penalty functions* [22] (for example, the Lorentzian $\rho_S(x) = \log(1 + \frac{1}{2}(x/\sigma)^2)$), which are known to better cope with outliers or flow discontinuities. Due to this fact, our framework can also incorporate the state-of-the-art combined local-global method (CLG) [23], which just replaces I_x, I_y, and I_t (in the first term of the function above) with a structure tensor. This is important since our algorithms can always compute a solution near the global minimum and, so, by using them as initializers to CLG (or to any other global method), we can help such methods to avoid a local minimum.

Besides using the *optic flow constraint equation* in our label costs, our framework also allows the use of other label costs. For example, we can set $\boldsymbol{c}_p(a) = |I_1(p + a) - I_0(a)|$, where I_0 and I_1 are the current and next images. In this case, due to the 2D nature of optical flow, it is important that, not only the magnitudes, but especially the directions of the optical flow vectors are estimated correctly as well. To this end, the following nonmetric distance between labels can be used: $d(a,b) = \mathrm{dist}(a,b) + \tau \cdot \mathrm{angledist}(a,b)$. Here, $\mathrm{dist}(a,b)$ denotes

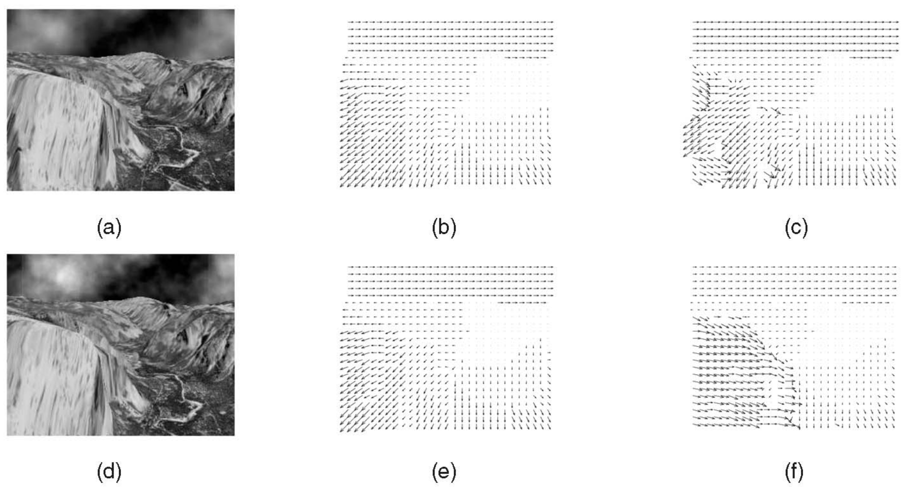


Fig. 14. Estimated flow between frames 4, 5 (first row) and 11, 12 (second row) of the *yosemite* sequence. Although more outer iterations were used by α-β-swap, its optical flow had 19.2 and 56.7 percent higher energy than our optical flow. (a) Fourth frame of the *yosemite* sequence with clouds. (b) Our flow: 6.97 degrees average angular error, four iterations. (c) α-β-swap flow: 14.73 degrees average angular error, 11 iterations, 19.2 percent higher energy than (b). (d) Eleventh frame of the *yosemite* sequence with clouds. (e) Our flow: 6.91 degrees average angular error, four iterations. (f) α-β-swap flow: 39.29 degrees average angular error, 23 iterations, 56.7 percent higher energy than (e).

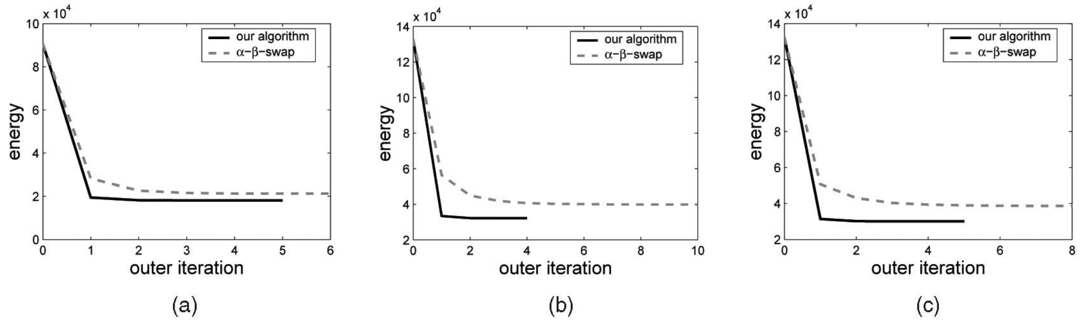


Fig. 15. α-β-swap produces an energy that is higher by (a) 17 percent, (b) 23 percent, and (c) 28 percent with respect to our algorithm's energy. Notice that, as the number of labels increases, the gap in performance increases as well. (a) $\mathcal{G}$ is a tree, $K = 60$ labels. (b) $\mathcal{G}$ is a grid, $K = 60$ labels. (c) $\mathcal{G}$ is a grid, $K = 180$ labels.

a truncated euclidean distance between the optical flow vectors a and b, that is, $\mathrm{dist}(a, b) = \min(\|a - b\|, \lambda)$, whereas the second term is used for giving even more weight to the correct estimation of the vectors' direction. In particular, it penalizes (in a robust way) abrupt changes in the direction of the vectors a and b and is defined as follows: $\mathrm{angledist}(a, b)$ equals 1 if the angle (in degrees) between a and b is greater than 45 degrees, whereas, in all other cases, it equals 0. We have applied both our algorithm and the α-β-swap algorithm to the well-known *yosemite* image sequence, using as label distance the abovementioned distance, with parameters $\lambda = 5$ and $\tau = 5$. The results, as well as the error statistics, are shown in Fig. 14. Due to the bigger number of labels, the runtimes of our algorithm for this example were approximately six minutes on the average. *We note that, although both algorithms are trying to minimize exactly the same objective function, the resulting solutions of α-β-swap have much higher energy.* It seems that, contrary to our method, α-β-swap needs to be properly initialized or else is not powerful enough to escape from bad local minima in this case.

7.5 Synthetic Problems

To further examine the ability of our algorithms to optimize the energy of an MRF, we also tested them on a set of synthetic problems. In these problems, the vertices of a 30×30 grid were chosen as the nodes of the graph $\mathcal{G}$, whereas the total number of labels was set equal to K. The label costs for all nodes were generated randomly by drawing samples from a uniform distribution in the $[\varrho_0\ \varrho_1]$ interval, whereas, for the pairwise potentials, a random nonmetric distance has been used which was constructed as follows: Equal labels were assigned zero distance, whereas the distance for different labels was generated randomly in the $[\varrho_0\ \varrho_1]$ interval again.

Three experiments have been conducted: In the first one (Fig. 15a), a random spanning tree of the 30×30 grid was used as the graph $\mathcal{G}$ and the number of labels was $K = 60$, whereas, in the second (Fig. 15b) and third (Fig. 15c) experiments, the graph $\mathcal{G}$ inherited the structure of the underlying grid and the number of labels was $K = 60$ and $K = 180$, respectively. For each experiment, 100 random problems were constructed (all with $\varrho_0 = 1$ and $\varrho_1 = 100$), and the resulting average energies per outer iteration for both our algorithm and the α-β-swap algorithm are shown in the plots of Fig. 15. Notice that, compared to α-β-swap, our algorithm manages to produce a solution of lower energy in all cases. At the same time, it needs fewer iterations to converge. This behavior is a typical one and has been observed in real problems as well. Notice also that, as the

Label costs

$c(\cdot)$	p	q	r
a	0	T	T
b	T	0	T
c	2	2	0

Label distance

$d(\cdot,\cdot)$	a	b	c
a	0	T/2	T
b	T/2	0	T/2
c	T	T/2	0

Labeling A (Local minimum)

p	q	r
a	b	c

Labeling B (Global minimum)

p	q	r
c	c	c

Fig. 16. A synthetic example, where the graph $\mathcal{G}$ has three vertices $\{p, q, r\}$ and two edges $\{pq, qr\}$ while the labels $\mathcal{L}$ are $\{a, b, c\}$. Label costs $c_p(\cdot)$ and the distance $d(\cdot, \cdot)$ (not a metric) are shown. The α-β-swap algorithm can get stuck in labeling A whose cost is T, that is, arbitrarily larger than the true minimum cost, which is 4 (labeling B). On the contrary, $\mathrm{PD3}_a$, $\mathrm{PD3}_b$, and $\mathrm{PD3}_c$ can always locate the optimal labeling B. Example taken from [1].

number of labels or the graph complexity increases, the gap in performance between the two algorithms increases as well.

The efficiency of our algorithms in the case where $d(\cdot, \cdot)$ is not a metric can also be illustrated by the synthetic example of Fig. 16. Although $\mathrm{PD3}_a$, $\mathrm{PD3}_b$, and $\mathrm{PD3}_c$ are always able to locate the exact global minimum for this example, the α-β-swap algorithm may get stuck at a local minimum that can be arbitrarily far from the true minimum.

8 Conclusions

A new theoretical framework has been proposed for both understanding and developing algorithms that can approximately optimize MRFs with both metric and nonmetric energy functions. This set of MRFs can model a very important class of problems in computer vision. The above framework includes the state-of-the-art α-expansion algorithm merely as a special case (for metric energy functions). Moreover, it provides algorithms, which have guaranteed optimality properties even for the case of nonmetric potentials. In fact, in all cases, our primal-dual algorithms are capable of providing per-instance suboptimality bounds, which, in practice, prove to be very tight (that is, very close to 1), which means that the resulting solutions are nearly optimal. The theoretical setting of the proposed framework rests on duality theory of LP, which is entirely different than the setting of the original graph-cut work. This way, an alternative and more general view of the very successful graph-cut algorithms for approximately optimizing MRFs is provided, which is an important advance. We strongly believe that this more general view of graph-cut techniques may give rise to new related research, which could lead to even more powerful MRF optimization algorithms in the future. Moreover, a novel optimization technique, the primal-dual schema, has been introduced to the field of computer vision, and the resulting algorithms have proven to give excellent experimental results on a variety of low-level vision tasks, such as stereo matching, image restoration, image completion, and optical flow estimation.

For metric MRFs, PD2 with $\mu = 1$ has, in general, given the best results experimentally (although, in some cases, the results were only slightly better compared, for example, to PD1 (see Tables 1 and 2)), which is consistent with the good performance of α-expansion in many problems up to now. However, we believe that PD2 with $\mu < 1$, as well as PD1, can also be very useful as initializers since they are less greedy and can more easily avoid local minima. For nonmetric MRFs, PD3 algorithms exhibit similar performance and gave the best results in practice. Furthermore, they reduce to $\mathrm{PD2}_{\mu=1}$ in the case of a metric potential function. Therefore, based also on the fact that $\mathrm{PD3}_a$ and $\mathrm{PD3}_c$ can, in both cases, provide worst case guarantees, we recommend the use of either one of these two algorithms in the general case. Also, as already mentioned, all of our algorithms apply without change even if each edge pq has its own distance d_{pq}. The distance giving the worst approximation factor then dominates and, so, the new suboptimality bound becomes

$$f_{\mathrm{app}} = \max_{pq}\left[2\frac{\max_{a\neq b} d_{pq}(a,b)}{\min_{a\neq b} d_{pq}(a,b)}\right].$$

Finally, we should note that, for certain special cases of the ML problem, our algorithms' theoretical approximation factors coincide with the so-called *integrality gap* of the linear program in (1), which is essentially the best possible approximation factor a primal-dual algorithm may achieve [18]. For example, such is the case with the Generalized Potts model, whose integrality gap is known to be 2 [2], that is, equal to f_{app}. This explains, in yet another way, why graph-cut techniques are so good in optimizing problems related to the Potts energy. In conclusion, a new powerful optimization tool has been added to the arsenal of computer vision capable of tackling a very wide class of problems.

References

[1] Y. Boykov, O. Veksler, and R. Zabih, "Fast Approximate Energy Minimization via Graph Cuts," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 23, no. 11, Nov. 2001.

[2] J. Kleinberg and E. Tardos, "Approximation Algorithms for Classification Problems with Pairwise Relationships: Metric Labeling and Markov Random Fields," *J. ACM,* vol. 49, pp. 616-630, 2002.

[3] O. Veksler, "Efficient Graph-Based Energy Minimization Methods in Computer Vision," PhD dissertation , Cornell Univ., 1999.

[4] S. Roy and I. Cox, "A Maximum-Flow Formulation of the n-Camera Stereo Correspondence Problem," *Proc. Sixth IEEE Int'l Conf. Computer Vision,* 1998.

[5] A. Gupta and E. Tardos, "Constant Factor Approximation Algorithms for a Class of Classification Problems," *Proc. 32nd Ann. ACM Symp. Theory of Computing,* pp. 652-658, 2000.

[6] H. Ishikawa and D. Geiger, "Segmentation by Grouping Junctions," *Proc. Conf. Computer Vision and Pattern Recognition,* 1998.

[7] C. Chekuri, S. Khanna, J. Naor, and L. Zosin, "Approximation Algorithms for the Metric Labeling Problem via a New Linear Programming Formulation," *Proc. 12th Ann. ACM-SIAM Symp. Discrete Algorithms,* pp. 109-118, 2001.

[8] A. Archer, J. Fakcharoenphol, C. Harrelson, R. Krauthgamer, K. Talvar, and E. Tardos, "Approximate Classification via Earthmover Metrics," *Proc. 15th Ann. ACM-SIAM Symp. Discrete Algorithms,* 2004.

[9] N. Komodakis and G. Tziritas, "A New Framework for Approximate Labeling via Graph-Cuts," *Proc. 10th IEEE Int'l Conf. Computer Vision,* 2005.

[10] N. Komodakis and G. Tziritas, "Approximate Labeling via the Primal-Dual Schema," Technical Report CSD-TR-05-01, Computer Science Dept., http://www.csd.uoc.gr/~komod/publications/docs/CSD_TR_2005_01.pdf, Feb. 2005.

[11] M. Wainwright, T. Jaakkola, and A. Willsky, "Map Estimation via Agreement on (Hyper)Trees: Messagepassing and Linear Programming Approaches," *Proc. 40th Ann. Allerton Conf. Comm., Control, and Computing,* 2002.

[12] C.H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, 1982.

[13] V. Kolmogorov and R. Zabih, "Multi-Camera Scene Reconstruction via Graph Cuts," *Proc. Seventh European Conf. Computer Vision,* pp. 82-96, 2002.

[14] R. Zabih and V. Kolmogorov, "Spatially Coherent Clustering Using Graph Cuts," *Proc. Conf. Computer Vision and Pattern Recognition,* pp. 437-444, 2004.

[15] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1988.

[16] V. Kolmogorov, "Convergent Tree-Reweighted Message Passing for Energy Minimization," *Proc. 10th Int'l Workshop Artificial Intelligence and Statistics,* 2005.

[17] T. Meltzer, C. Yanover, and Y. Weiss, "Globally Optimal Solutions for Energy Minimization in Stereo Vision Using Reweighted Belief Propagation," *Proc. 10th IEEE Int'l Conf. Computer Vision,* 2005.

[18] V. Vazirani, *Approximation Algorithms.* Springer, 2001.

[19] C. Rother, S. Kumar, V. Kolmogorov, and A. Blake, "Digital Tapestry," *Proc. Conf. Computer Vision and Pattern Recognition,* 2005.

[20] D. Scharstein and R. Szeliski, "A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms," *Int'l J. Computer Vision,* vol. 47, nos. 1-3, pp. 7-42, Apr.-June 2002.

[21] M.F. Tappen and W.T. Freeman, "Comparison of Graph Cuts with Belief Propagation for Stereo, Using Identical mrf Parameters," *Proc. Ninth IEEE Int'l Conf. Computer Vision,* pp. 900-907, 2003.

[22] M.J. Black and P. Anandan, "The Robust Estimation of Multiple Motions: Parametric and Piecewise-Smooth Flow Fields," *Computer Vision and Image Understanding,* vol. 63, no. 1, pp. 75-104, 1996.

[23] A. Bruhn, J. Weickert, and C. Schnörr, "Lucas/Kanade Meets Horn/Schunck: Combining Local and Global Optic Flow Methods," *Int'l J. Computer Vision,* vol. 61, no. 3, pp. 211-231, 2005.

Nikos Komodakis received the PhD degree in computer science (with highest honors) from the University of Crete in 2006. He is currently a postdoctoral associate in the Computer Science Department at the University of Crete. His research interests include MRF optimization techniques (with applications to computer vision), image segmentation, and stereo matching, as well as topics that are at the confluence of computer vision and computer graphics, such as image-based modeling and rendering, image completion, and texture synthesis.

Georgios Tziritas received the Diploma in electrical engineering (1977) from the Technical University of Athens and the Diplôme d'Etudes Approfondies (DEA, 1978), the Diplôme de Docteur Ingénieur (1981), and the Diplôme de Docteur d'Etat (1985) from the Institut National Polytechnique de Grenoble. Beginning in 1982, he was a researcher of the Centre National de la Recherche Scientifique, with the Centre d'Etudes des Phénomènes Aléatoires (CEPHAG) until August 1985, with the Institut National de Recherche en Informatique et Automatique (INRIA) until January 1987, and with the Laboratoire des Signaux et Systèmes (LSS). Beginning in September 1992, he was an associate professor and, since January 2003, a full professor, at the Department of Computer Science, University of Crete, teaching digital image processing, digital video processing, pattern recognition, and data and signal compression, among others. His research interests include multimedia signal processing, image processing and analysis, computer vision, motion analysis, and image-based augmented reality. He is a coauthor (with C. Labit) of the book *Motion Analysis for Image Sequence Coding* (Elsevier, 1994) and of more than 90 journals and conference papers on signal and image processing, image and video analysis, computer vision, and pattern recognition. He is a senior member of the IEEE.