Approximate Labeling via Graph Cuts Based on Linear Programming

Nikos Komodakis, Georgios Tziritas

Computer Science Department, University of Crete

E-mails: {komod, tziritas}@csd.uoc.gr

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 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, e.g. even for MRFs with non-metric potentials. In addition, they are capable of providing per-instance suboptimality bounds in all occasions, including discrete Markov Random Fields with an arbitrary potential function. These bounds prove to be very tight in practice (i.e. 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.

I. 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, image restoration, to mention only a few of them. Therefore, an issue of paramount

importance, that 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 Problem (or ML for short), that has been introduced by Kleinberg and Tardos [2] recently, 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, 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 2 kinds.

On one hand, for each $p \in V$, there is a *label cost* $\mathfrak{c}_p(a) \geq 0$ 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, q that are related (i.e. connected by an edge in the graph $\mathcal{G}$), there is a so-called *separation cost* for assigning labels $a = f_p, b = f_q$ 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, q, while $d(a, b)$ is a distance function between labels, measuring how similar two labels are. The intuition behind this definition of the separation cost is that objects which are strongly related to each other should be assigned similar labels. This helps in preserving the spatial coherence of the final labeling. To simplify 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 Metric Labeling, the distance $d(a, b)$ was assumed to be a metric, i.e. $d(a, b) = 0 \Leftrightarrow a = b, \ d(a, b) = d(b, a) \geq 0, \ d(a, b) \leq d(a, c) + d(c, b)$, but here we will relax this assumption. Based on these definitions the total cost of a labeling f equals:

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

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

The Metric Labeling problem is directly connected to the theory of Markov Random Fields (MRFs). In fact, optimizing the cost in the Metric Labeling 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 Metric Labeling problem is (in general) NP-hard and therefore one can only hope for methods that provide approximate solutions. To this end, two are the main classes of methods that have

been proposed so far: those based on combinatorial optimization [1], [3], [4], [5], [6], as well as those based on linear programming [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 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 that they formulate Metric Labeling as an equivalent integer program with a very large number of variables. E.g. one such formulation, introduced in [7], is the following:

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

$$\text{s.t.} \sum_a x_p(a) = 1 \qquad \forall \ p \in V \tag{2}$$

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

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

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

The $\{0,1\}$-variable $x_p(a)$ indicates that vertex p is assigned label a, while the $\{0,1\}$-variable $x_{pq}(a,b)$ indicates that vertices p,q are assigned labels a,b respectively. The variables $x_{pq}(a,b)$, $x_{qp}(b,a)$ 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), (q,p) belongs to $\mathcal{E}$ for any neighbors p,q. The notation "$p \sim q$" will hereafter denote that p,q are neighbors, i.e. "either only $(p,q) \in \mathcal{E}$ or only $(q,p) \in \mathcal{E}$". The first constraints (2) simply express the fact that each vertex must receive exactly one label, while constraints (3), (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 $x_q(b) = 1$ holds 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 Metric Labeling problem (and thus of a very wide class of MRFs frequently encountered in computer vision). It makes use of the primal-dual schema of linear programming in order to derive efficient (i.e. 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:
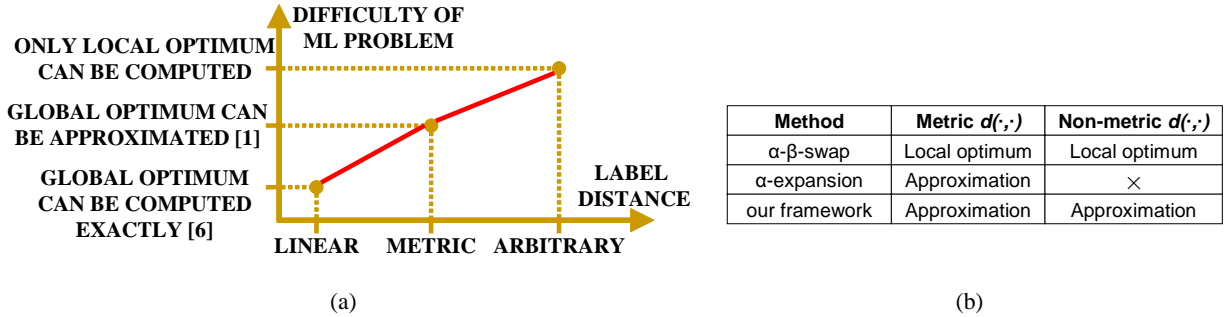
Fig. 1: **(a)** The difficulty of the ML problem depends critically on the type of chosen label distance $d(\cdot,\cdot)$ **(b)** A comparison of our framework with respect to state-of-the-art optimization methods that are based on graph-cuts.

1) It turns out that the difficulty of the Metric Labeling problem depends critically on the type of the chosen distance $d(\cdot,\cdot)$ between labels (see Figure 1(a)). 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, i.e. 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 non-metric distance function that satisfies $d(a,b) = 0 \Leftrightarrow a = b, \ d(a,b) = d(b,a) \geq 0$, 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 etc.). 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 for the quality of the generated solutions.

2) Furthermore, the quality of these solutions also depends critically 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 non-metric distance, it can still guarantee that the generated solution will always be within a known factor of the global optimum, i.e. a worst-case suboptimality bound can be provided in this case (see Figure 1(b)). This is in contrast to local MRF optimization methods, such as the ICM algorithm or the Highest Confidence First method, for which no such

---

[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 "non-metric" 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.

theoretical guarantees (i.e. no such analysis) can be provided. We should also note that, although any algorithm (e.g. the α-expansion) can be converted to handle non-metric distances without a loss in the worst case bounds (e.g. by replacing non-metric terms with Potts terms, see [1]), this completely misses any structure of the non-metric distance function. On the contrary, our method can handle both metric as well as non-metric 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, i.e. they are nearly optimal. This can be verified thanks to our algorithms' ability of also providing 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 (i.e. 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 (e.g., 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 be also 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 proved to be equivalent to the α-expansion graph cut technique (i.e. a method which 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 in terms of principles drawn from duality theory of linear programming, thus shedding further light on their essence (e.g. a connection between α-expansion and the belief propagation algorithm TRBP [11], which also tries to solve exactly the same dual LP relaxation, can thus be established). This is an important advance which, 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 has already been used for tackling many LP problems [12], e.g. for providing an alternate way to derive Dijkstra's algorithm, Ford-Fulkerson's algorithm, the Hungarian method, etc.), is now also introduced to the field of computer vision, which can prove to be a great benefit too.

The rest of the paper is organized as follows. We review related work in section II. In section

III the primal-dual schema is presented, which will guide the design of all of our approximation algorithms. These algorithms are described in sections IV - VI. More specifically, to handle the various cases of the distance function, we will progressively present 3 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 IV. In section V, we derive $PD2_\mu$ (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 $\mu = 1$, i.e. it is equivalent to algorithm $PD2_{\mu=1}$. In section VI, we present algorithms PD3, which make up the third family of our primal-dual methods. These algorithms manage to extend, as well as generalize the α-expansion method (i.e. algorithm $PD2_{\mu=1}$) to the case of non-metric MRFs. In addition, despite this generalization, these algorithms manage to maintain the theoretical approximation guarantees of the $PD2_{\mu=1}$ algorithm. Experimental results are shown in section VII, while we conclude in section VIII. 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].

II. 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, multi-scale MRFs, relaxation labeling, graduated nonconvexity and mean field annealing, to mention just a few of them. However, all of the above-mentioned 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 (i.e. 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. A linear programming 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 (i.e. one with lower energy) by finding the minimum cut in a suitable graph. This state-of-the-art method has proved 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].

Related to α-expansion is also the α-β-swap algorithm [1]. Although this is a more general method, as it applies to non-metric 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 which are capable of extracting the exact global optimum [4], [6]. These, however, require the potential function to be convex, as well as the labels to be one-dimensional, a fact which restricts their applicability.

Finally, we should also mention that there also exist those optimization algorithms that are based on belief propagation [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].

III. THE PRIMAL-DUAL SCHEMA

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

(a) The primal-dual principle

(b) The primal-dual schema
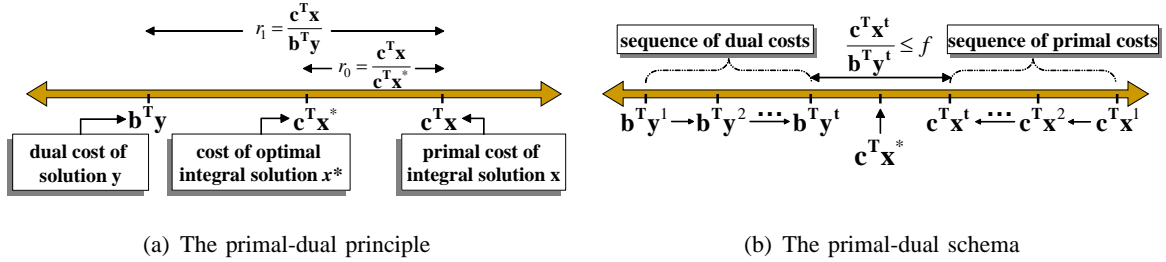
Fig. 2: (a) By weak duality, the optimal cost $\mathbf{c}^T\mathbf{x}^*$ will lie between the costs $\mathbf{b}^T\mathbf{y}$ and $\mathbf{c}^T\mathbf{x}$ of any pair $(\mathbf{x}, \mathbf{y})$ of integral-primal and dual feasible solutions. Therefore, if $\mathbf{b}^T\mathbf{y}$ and $\mathbf{c}^T\mathbf{x}$ are close enough (e.g. their ratio r_1 is $\leq f$), so are $\mathbf{c}^T\mathbf{x}^*$ and $\mathbf{c}^T\mathbf{x}$ (e.g. their ratio r_0 is $\leq f$ 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 $\mathbf{b}^T\mathbf{y}^t$, $\mathbf{c}^T\mathbf{x}^t$ are close enough (e.g. their ratio is $\leq f$). We can then apply the primal-dual principle (as in Fig. (a)) and thus conclude that $\mathbf{x}^t$ is an f-approximation to $\mathbf{x}^*$.

$$\text{PRIMAL:} \quad \min \mathbf{c}^T\mathbf{x} \qquad\qquad \text{DUAL:} \quad \max \mathbf{b}^T\mathbf{y}$$
$$\text{s.t.} \quad \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0} \qquad\qquad \text{s.t.} \quad \mathbf{A}^T\mathbf{y} \leq \mathbf{c}$$

Here $\mathbf{A} = [a_{ij}]$ represents an $m \times n$ rectangular matrix, while $\mathbf{b}, \mathbf{c}$ are column vectors of size m, 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 use of the following principle:

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

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

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

The reason that this principle holds true is rather simple and is illustrated graphically in Figure 2(a): 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 $\mathbf{c}^T\mathbf{x}$, i.e. $\mathbf{b}^T\mathbf{y} \leq \mathbf{c}^T\mathbf{x}^* \leq \mathbf{c}^T\mathbf{x}$. If we therefore manage to bring the two quantities $\mathbf{b}^T\mathbf{y}$ and $\mathbf{c}^T\mathbf{x}$ close to each other (e.g. by making their ratio $r_1 = \mathbf{c}^T\mathbf{x}/\mathbf{b}^T\mathbf{y}$ less 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 (e.g. 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 IP (integer program) 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 (i.e. dual-LP$\leq$primal-LP$\leq$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 above principle lies at the heart of any primal-dual technique. In fact, the various primal-dual methods mostly differ in the way that 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 (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:*

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

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 $\mathbf{x}$, $\mathbf{y}$ satisfy the feasibility conditions of the primal and dual program respectively (fundamental inequality (5) then follows trivially). Thus, based on the above theorem, the following iterative schema is usually applied during a primal-dual f-approximation algorithm:

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

This schema is illustrated graphically in Figure 2(b). At each iteration, based just on 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 (i.e. $\mathbf{y}^k$ is perturbed as well) and a new primal-dual pair, say $(\mathbf{x}^{k+1}, \mathbf{y}^{k+1})$, is thus generated. This 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 (i.e. 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 (i.e. different f_j) each time*, which is exactly what we will do for the case of Metric Labeling and thus derive 3 different types of algorithms PD1, PD2 and PD3.

A. Applying the primal-dual schema to Metric Labeling

For the case of Metric Labeling, our primal linear program will be integer program (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:

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

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

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

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 $\mathcal{G}$), **3)** as well as a vector $\mathbf{y}$ containing all variables $y_{pq}(\cdot), y_{qp}(\cdot)$, called the *"balance variables"* hereafter. Also, any 2 variables $y_{pq}(a)$, $y_{qp}(a)$ will be referred to as *"conjugate balance variables"*.

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

$$h_p(\cdot) \equiv \mathfrak{c}_p(\cdot) + \sum_{q:q \sim p} y_{pq}(\cdot) \tag{9}$$

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 are following. 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 do not have to worry about constraints (7) or how to estimate the z_p variables any more. 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 $x_p = a$, while $x_{pq}(a, b) = 1$ means $x_p = a, x_q = b$ and so, under this notation, it is not difficult to see that the complementary condition related to a non-zero $x_p(a)$ variable reduces to:

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

while the complementary condition related to a non-zero $x_{pq}(a, b)$ variable reduces to:

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

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

1: $\mathbf{x} \leftarrow$ INIT_PRIMALS( ); $\mathbf{y} \leftarrow$ INIT_DUALS( ); LabelChange$\leftarrow 0$	6: **if** $\mathbf{x}' \neq \mathbf{x}$ **then** LabelChange$\leftarrow 1$
2: **for** each label c in $\mathcal{L}$ **do**	7: $\mathbf{x} \leftarrow \mathbf{x}'$, $\mathbf{y} \leftarrow \mathbf{y}'$;
3: $\mathbf{y} \leftarrow$ PREEDIT_DUALS$(c, \mathbf{x}, \mathbf{y})$;	8: **end for**
4: $[\mathbf{x}', \mathbf{y}'] \leftarrow$ UPDATE_DUALS_PRIMALS$(c, \mathbf{x}, \mathbf{y})$;	9: **if** LabelChange$= 1$ **then** goto 2;
5: $\mathbf{y}' \leftarrow$ POSTEDIT_DUALS$(c, \mathbf{x}', \mathbf{y}')$;	10: **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.

Our objective will therefore be to find feasible solutions $\mathbf{x}, \mathbf{y}$ satisfying the above conditions (11), (12) and (13) for specific values of f_1 and f_2. Conditions (13) simply say 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 conditions (13) hereafter.

Most of our primal-dual algorithms will achieve an approximation factor of $f_{\text{app}} = 2\frac{d_{\max}}{d_{\min}}$ (i.e. $\max\{f_1, f_2\} = f_{\text{app}}$), 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 Figure 3. The initial primal-dual solutions are generated inside INIT_PRIMALS and INIT_DUALS. During an inner iteration (lines 4-8 in Figure 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 $\mathbf{y}$ (i.e. $y_{pq}(.)$), only the balance variables of the c labels (i.e. $y_{pq}(c)$) 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 2-9 in Figure 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, while 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.

IV. THE PD1 ALGORITHM

An intuitive view of the dual variables, that 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 determine the balls' height (see Figure 4(a)). E.g. 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, balls are not static, but may move in pairs through updating pairs of conjugate balance variables. E.g., in Figure

(a) 1st type of dual variables: height variables   (b) 2nd type of dual variables: balance variables
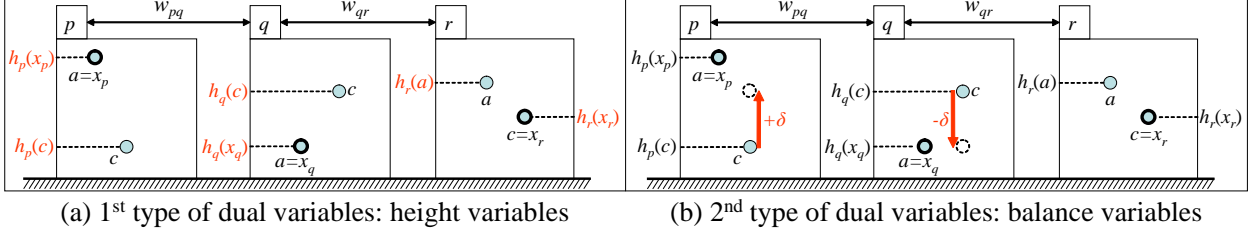
Fig. 4: Visualization of the dual variables for a graph $\mathcal{G}$ with vertices p, q, r and labels $\mathcal{L} = \{a, c\}$. **(a)** A copy of labels $\{a, c\}$ exists for each vertex and all these labels are represented as balls floating above a reference plane. The role of the height variables is to specify the balls' height. **(b)** Furthermore, balls are not static, but may move in pairs by updating conjugate balance variables. E.g., here, ball c at p is pulled up by $+\delta$ (due to increase of $y_{pq}(c)$ by $+\delta$) and so ball c at q moves down by $-\delta$ (due to decrease of $y_{qp}(c)$ by $-\delta$). Active labels are drawn with a thicker circle.

4(b), 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, while 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 net raise (called just raise hereafter) and not relative raise. E.g., in Fig. 4(b), label a at p has 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 terminology. Let $\mathbf{x}, \mathbf{y}$ be a pair of integral-primal, dual solutions. We call the label that $\mathbf{x}$ assigns to p (i.e. x_p) the *active label at p*. The sum of heights of all active labels is called the *"Approximate Primal Function"* (or APF for short), i.e. $\text{APF}^{\mathbf{x}, \mathbf{y}} = \sum_p h_p(x_p)$. This function's name comes from the fact that, if $\mathbf{x}, \mathbf{y}$ satisfy 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 (i.e. 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, q (denoted by load_{pq}) is then defined as $\text{load}_{pq} = y_{pq}(x_p) + y_{qp}(x_q)$ (i.e. as the sum of 2 active balance variables at p, q) and represents the partial raises of active labels at p, q due to edge pq. If relaxed slackness conditions (12) hold, then, due to (12) and (8), it is easy to see that $w_{pq} d(x_p, x_q) / f_2 \leq \text{load}_{pq} \leq w_{pq} d(x_p, x_q)$ and so the load of p, q can be also thought of as a *virtual separation cost* which 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 non-metric distance and

then tries to find feasible $\mathbf{x}, \mathbf{y}$ satisfying complementary conditions (11), (12) with $f_1 = 1$ and $f_2 = f_{\mathrm{app}}$ (recall that $f_{\mathrm{app}} \equiv 2\frac{d_{\max}}{d_{\min}}$). If $f_1 = 1$, then condition (11) becomes $z_p \geq h_p(x_p)$ 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 $f_2 = f_{\mathrm{app}}$, then complementary condition (12) reduces to:

$$x_p \neq x_q \Rightarrow \mathrm{load}_{pq} \geq w_{pq}d(x_p, x_q)/f_{\mathrm{app}} \tag{16}$$

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

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

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

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

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

To see then how the update of primal and dual variables should proceed, one simply needs to reinterpret conditions (15)-(17) based on the dual variables' aforementioned interpretation. E.g.:

- (15) simply says that, at each vertex, the active label should have the lowest height,
- (16) requires that any 2 active labels should be raised proportionally to their separation costs,
- and, finally, (17) says that there is an upper bound on how much we can raise a label.

Based on these, and assuming that (16), (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 (i.e. the current primal), any non-active 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 conditions (16) still hold true, since no active labels have moved.

PRIMAL VARIABLES UPDATE: Given the new heights (i.e. the new dual), there might still be vertices violating (15), i.e. their active labels are not at the lowest height. For each such vertex p, we select a non-active label, which is below x_p, but has already reached the maximum raise allowed by (17). That label, say c, is then made the new active label of p, i.e. we set $x_p = c$. One can then show that conditions (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 $\geq \frac{w_{pq}d_{\min}}{2} \frac{d(x_p,x_q)}{d_{\max}} = \frac{w_{pq}d(x_p,x_q)}{f_{\mathrm{app}}}$. This, in conjunction with the nonnegativity of any variable $y_{qp}(x_q)$ (due to (18)), thus proves that condition (16) will still hold. This way, since we keep assigning lower active labels to vertices, one may easily show that conditions (15) will finally hold true in the end, i.e. 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 $y_{pq}(c)$ (i.e. 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. E.g., in Figure 4(a), 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 $x_q{=}a$, while 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 condition (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 $\mathbf{x}, \mathbf{y}$, while its nodes $\mathcal{V}^c$ consist of all nodes of graph $\mathcal{G}$ (the *internal* nodes) plus 2 *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 2 directed interior edges pq and qp in graph $\mathcal{G}^c$. Flows f_{pq} (through pq), f_{qp} (through qp) will represent respectively the increase, decrease of balance variable $y_{pq}(c)$. The net change of $y_{pq}(c)$ will therefore be $f_{pq} - f_{qp}$, i.e.:

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

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

Due to (19), it is obvious that capacity cap_{pq} of edge pq determines the maximum allowed value of $y'_{pq}(c)$ (attained at $f_{pq}{=}\mathrm{cap}_{pq}, f_{qp}{=}0$), while a similar conclusion holds for cap_{qp} and $y'_{qp}(c)$. But, e.g. $y'_{pq}(c)$ represents the new partial raise of label c at p due to edge pq. Therefore, if the c-labels at p, q aren't active (i.e. $x_p{\neq}c, x_q{\neq}c$) and may thus move, then cap_{pq}, cap_{qp} are set so that these c-labels cannot raise too much and violate (17), i.e. they are set so that $y'_{pq}(c), y'_{qp}(c)$
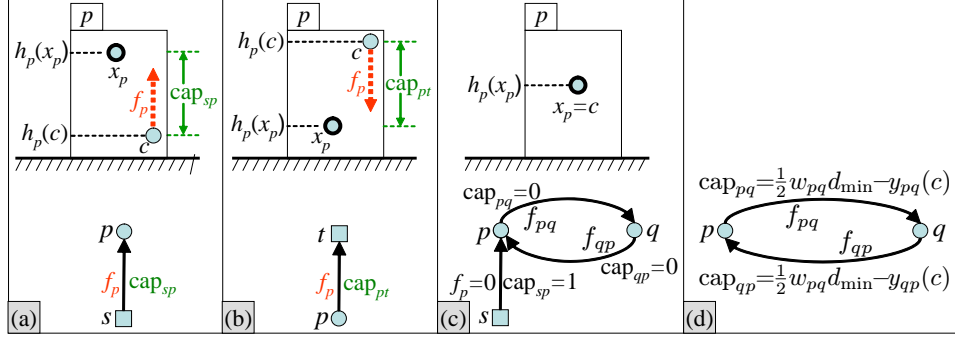
Fig. 5: Four simple rules for constructing graph $\mathcal{G}^c$: **(a)** if c at p is below x_p, we connect node p to source s and flow f_p through sp represents the total relative raise of c (and also $\text{cap}_{sp} = h_p(x_p) - h_p(c)$). **(b)** if c is above x_p, we connect node p to sink t and 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 (i.e. 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 constraints (17) always hold true.

can't 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. 5(d)):

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

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 $y_{pq}(c)$, $y_{qp}(c)$ should equal $y'_{pq}(c)$, $y'_{qp}(c)$, i.e. (see also Fig. 5(c)):

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

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 3 possible cases to consider:

– *CASE 1 $\big(c$ is "below" x_p, i.e. $h_p(c) < h_p(x_p)\big)$:* we would then like to raise label c as much as needed so that it reaches label x_p (e.g. see Fig. 5(a)). 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, i.e.:[2]
$$h'_p(c) = h_p(c) + f_p \tag{22}$$

Therefore, based on (22), capacity cap_{sp} of edge sp will represent the maximum allowed 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. 5(a)).

– *CASE 2 $\big(c$ is not "below" x_p, i.e. $h_p(c) \geq h_p(x_p)$, and not the active label of p, i.e. $c \neq x_p\big)$:* 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 (e.g. see Fig. 5(b)). This time the

---

[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: $h_p(c) + f_p \overset{(9)}{=} \big(\mathfrak{c}_p(c) + \sum_{q:q\sim p} y_{pq}(c)\big) + f_p = \big(\mathfrak{c}_p(c) + \sum_{q:q\sim p} y_{pq}(c)\big) + \sum_{q:q\sim p}(f_{pq} - f_{qp}) \overset{(19)}{=} \big(\mathfrak{c}_p(c) + \sum_{q:q\sim p} y_{pq}(c)\big) + \sum_{q:q\sim p}(y'_{pq}(c) - y_{pq}(c)) = \mathfrak{c}_p(c) + \sum_{q:q\sim p} y'_{pq}(c) \overset{(9)}{=} h'_p(c)$

flow f_p through edge pt will equal the total relative decrease in the height of c i.e.:

$$h'_p(c) = h_p(c) - f_p \qquad (23)$$

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. 5(b)).

   $-$ *CASE 3* $\big(c$ *is the active label of p, i.e. $c = x_p\big)$:* 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. 5(c)). 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 it was intended. By convention we set $\text{cap}_{sp} = 1$.

A. *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)), i.e.:

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

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

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

Based on the new heights, we now need to update $\mathbf{x}$ into $\mathbf{x}'$, i.e. assign new active labels. As only the c-heights have changed (i.e. 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 (i.e. $x'_p = c$) $\Leftrightarrow \exists$ unsaturated[3] path between the source node s and node p. In all other cases, p keeps its current label i.e. $x'_p = x_p$.*

Intuitively, on one hand, this rule ensures that if $c \neq x_p$, then (after the heights' update) the "lowest" of c, x_p is assigned to p (i.e. it ensures property A below, so that conditions (15) finally

---

[3]A path is unsaturated if "flow $<$ capacity" for all forward arcs and "flow > 0" for all backward arcs
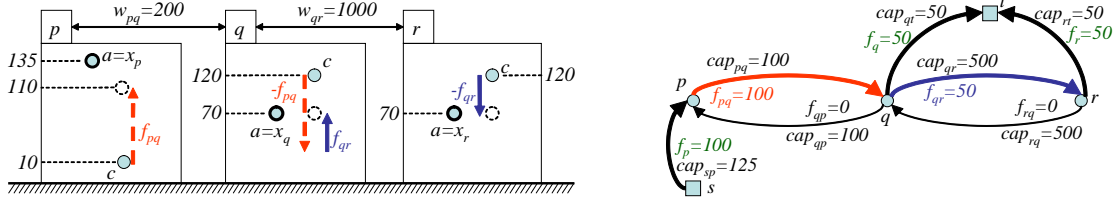
Fig. 6: (Left) Red/blue arrows show how c-labels will move due to update of balance variables. Dashed circles indicate new positions of c-labels. **(Right)** Associated graph $\mathcal{G}^c$ and resulting flows responsible for the update of balance variables in the left figure (current balance variables 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, f_r at exterior edges equal the total relative movement of the c-labels at p, q, r respectively. (The Potts distance has been used in this example, i.e. $a \neq b \Rightarrow d(a,b) = 1$).

hold true). To see that, assume e.g. path sp in Fig. 5(a) is unsaturated, i.e. $f_p < \mathrm{cap}_{sp}$. But then, since f_p equals the total relative raise of c (i.e. $f_p = h'_p(c) - h_p(c)$), and $\mathrm{cap}_{sp} = h_p(x_p) - h_p(c)$, it follows that $h'_p(c) < h_p(x_p) \overset{x_p \neq c}{=} h'_p(x_p)$. I.e. label c should indeed be assigned to p, as it is "lower" than 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 B below (together with $y'_{qp}(x'_q) \geq 0$ from (18) and the definition of cap_{pq} in (20)). The reason property B holds is because, e.g. 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 $f_{pq} = \mathrm{cap}_{pq}, f_{qp} = 0$ and so property B arises due to (24).

Due to "reassign rule", these 3 properties can thus be proved [10] for the new solutions $\mathbf{x}', \mathbf{y}'$:[4]
(A) $h'_p(x'_p) = \min\{h'_p(x_p), h'_p(c)\}$, **(B)** $x'_p = c \neq x'_q \Rightarrow y'_{pq}(x'_p) = \mathrm{cap}_{pq} + y_{pq}(c)$, **(C)** $\mathbf{x} \neq \mathbf{x}' \Rightarrow \mathrm{APF}^{\mathbf{x}',\mathbf{y}'} < \mathrm{APF}^{\mathbf{x},\mathbf{y}}$. The last one (i.e. property C) proves the algorithm terminates (assuming integer capacities), and the intuition for being true is due to the reassign rule, which ensures that a new active label has always lower height than the previous active label, i.e. $h'_p(x'_p) \leq h_p(x_p)$.

POSTEDIT_DUALS: This routine's role is to restore invariant (18) for the next iteration. It thus changes $\mathbf{y}'$ so that its active balance variables are all ≥ 0, while 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 then (18) not hold, in which case POSTEDIT_DUALS simply sets $y'_{pq}(x'_p) = y'_{qp}(x'_q) = 0$.

---

[4]The reassign rule, and thus properties (A), (B) and (C), apply not only to PD1, but to our other primal-dual algorithms as well.

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)*}

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 ≥ 0
- $\mathbf{z}' = \min_a \mathbf{h}'_a$ {*impose (10)*}

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

Fig. 7: Left: Pseudocode of PD1 **Right:** Pseudocode of PD2$_\mu$. 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 2 algorithms is that a subset of the edges of $\mathcal{G}^c$ are assigned different capacities (see (29), (30)).

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

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

V. 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$_\mu$ will achieve slackness conditions (11), (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$_\mu$ is that, PD1 always generates a feasible dual solution at any of its inner iterations, whereas PD2$_\mu$ may allow any such dual solution to become infeasible. However, PD2$_\mu$ ensures that the (probably infeasible) final dual solution is *"not too far away from feasibility"*. This practically means that if that solution is divided by a suitable factor, it will become feasible again. This method (i.e. turning an infeasible dual solution into a feasible one by scaling) is also known as *"dual-fitting"* [18] in the linear programming literature.

More specifically, PD2$_\mu$ generates a series of intermediate pairs, all of them satisfying complementary condition (12) as an equality with $f_2 = \frac{1}{\mu}$, i.e.:

$$x_p \neq x_q \Rightarrow \mathrm{load}_{pq} = \mu w_{pq} d(x_p, x_q) \tag{26}$$

In addition, using a similar strategy with PD1, $\mathrm{PD2}_\mu$ drives the last intermediate pair towards satisfying complementary condition (11) with $f_1 = 1$ again, i.e.:

$$h_p(x_p) = \min_a h_p(a), \tag{27}$$

while, also like PD1, it tries to maintain nonnegativity of active balance variables, i.e. (18).

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

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

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 $\mathbf{y}^{\mathrm{fit}} = \frac{\mathbf{y}}{\mu f_{\mathrm{app}}}$, we can then show that:

$$y_{pq}^{\mathrm{fit}}(a) + y_{qp}^{\mathrm{fit}}(b) = \frac{y_{pq}(a) + y_{qp}(b)}{\mu f_{\mathrm{app}}} \overset{(28)}{\leq} \frac{2\mu w_{pq} d_{\max}}{\mu f_{\mathrm{app}}} = \frac{2\mu w_{pq} d_{\max}}{\mu 2 d_{\max}/d_{\min}} = w_{pq} d_{\min} \leq w_{pq} d_{ab} \ ,$$

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

$$z_p^{\mathrm{fit}} \equiv \frac{z_p}{\mu f_{\mathrm{app}}} \overset{(10)}{=} \frac{\min_a h_p(a)}{\mu f_{\mathrm{app}}} \overset{(27)}{=} \frac{h_p(x_p)}{\mu f_{\mathrm{app}}} = \frac{\mathbf{c}_p(x_p) + \sum_{q:q\sim p} y_{pq}(x_p)}{\mu f_{\mathrm{app}}} = \frac{\mathbf{c}_p(x_p)}{\mu f_{\mathrm{app}}} + \sum_{q:q\sim p} y_{pq}^{\mathrm{fit}}(x_p) \ ,$$

$$y_{pq}^{\mathrm{fit}}(x_p) + y_{qp}^{\mathrm{fit}}(x_q) = \frac{y_{pq}(x_p) + y_{qp}(x_q)}{\mu f_{\mathrm{app}}} = \frac{\mathrm{load}_{pq}}{\mu f_{\mathrm{app}}} \overset{(26)}{=} \frac{\mu w_{pq} d(x_p, x_q)}{\mu f_{\mathrm{app}}} = \frac{w_{pq} d(x_p, x_q)}{f_{\mathrm{app}}}$$

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

A. Main routines for updating primal and dual variables during a c-iteration

$\mathrm{PD2}_\mu$ 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, qp whose endpoints have labels $\neq c$ at the start of current c-iteration, i.e. $x_p \equiv a \neq c$ and $x_q \equiv b \neq c$. In place of (20), we then define:

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

$$\mathrm{cap}_{qp} = 0 \tag{30}$$

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 $\mathbf{x}'$ (i.e. $\mathbf{x}' \neq$

x), which is exactly the rationale behind definitions (29), (30), as well as PREEDIT_DUALS. To see that, assume e.g. $\mathbf{x}'$ assigns a new label c to q (i.e. $x'_q = c \neq x_q$), but not to p (i.e. $x'_p = x_p \neq c$), then:

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

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), i.e. nonnegativity of active balance variables. Also, note that (29) explains why $d(\cdot, \cdot)$ must be a metric (or else it would hold $\text{cap}_{pq} < 0$).

B. Equivalence of algorithms PD2$_{\mu=1}$ and α-expansion

It can thus be shown that PD2$_\mu$ indeed generates an f_{app}-approximate solution. Furthermore, it holds that all PD2$_\mu$ algorithms with $\mu < 1$ are non-greedy algorithms, meaning that neither the primal (nor the dual) objective function necessarily decreases (increases) per iteration. Instead, it is APF which constantly decreases (see property C in section IV-A), 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 (i.e. $\text{load}_{pq} = w_{pq}d(x_p, x_q)$) and it can then be shown that APF coincides with the primal function, i.e. APF = PRIMAL, whereas in any other case APF $\leq$ PRIMAL. Furthermore, it turns out that, during a c-iteration, PD2$_{\mu=1}$ chooses an $\mathbf{x}'$ that minimizes APF with respect to any other c-expansion, say $\bar{\mathbf{x}}$, of current solution $\mathbf{x}$ (to see that, recall that APF is the sum of active labels' heights and PD2$_{\mu=1}$ always tries choosing the "lowest" label among x_p and c, see property A). All these can be formally summarized in the next lemma [10]:

Lemma V.1. Let $(\mathbf{x}', \mathbf{y}') \equiv$ next primal-dual pair due to c-iteration, $\bar{\mathbf{x}} \equiv c$-expansion of current primal. Then: PRIMAL$^{\mathbf{x}'}$ = APF$^{\mathbf{x}',\mathbf{y}'} \leq$ APF$^{\bar{\mathbf{x}},\mathbf{y}'} \leq$ PRIMAL$^{\bar{\mathbf{x}}}$, (PRIMAL$^{\mathbf{x}} \equiv$ primal cost of $\mathbf{x}$)

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

Theorem V.2 ([10]). *The label assignment $\mathbf{x}'$ selected during a c-iteration of PD2$_{\mu=1}$ has smaller primal cost than any other label assignment $\bar{\mathbf{x}}$ which is a c-expansion of current solution $\mathbf{x}$.*

VI. PD3: EXTENDING PD2 TO THE NON-METRIC CASE

By modifying PD2$_\mu$, three different variations (PD3$_a$, PD3$_b$, PD3$_c$) may result, that are applicable even if $d(\cdot, \cdot)$ is a non-metric distance function. For simplicity, we will consider

only the $\mu = 1$ case, i.e. only variations of PD2$_{\mu=1}$. We also recall a fact that will prove to be useful for explaining the rationale behind the algorithms' definition: (OPTIMALITY CRITERION) *the load between any p, q represents a virtual separation cost, which should be equal to the actual separation cost of p, q, if the current primal-dual solutions are optimal.*

The main difficulty of extending PD2$_{\mu=1}$ to the non-metric case relates to all edges pq with capacity defined by (29) during a c-iteration, i.e. all interior edges pq whose endpoints p, q are currently assigned labels $\neq c$ (i.e. $x_p \equiv a \neq c, \ x_q \equiv b \neq c$) while, 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 PD2$_{\mu=1}$ may arise.

PD3$_a$ algorithm: We choose to set $\mathrm{cap}_{pq} = 0$ in place of (29). In this case, it can be shown that if $\mathbf{x}'$ assigns the pair of labels c, b to the objects p, q respectively, then the resulting load of p, q will be $w_{pq}\big(d(a, b) - d(a, c)\big)$, i.e. it will be greater than the actual separation cost $w_{pq}d(c, b)$ of p, 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 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 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, b are assigned to p, 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, b instead of their actual distance $d(c, b)$. On the other hand, if (a, b) or (a, c) are assigned to p, q by $\mathbf{x}'$, then no cost overestimation takes place and so the virtual distances for these labels coincide with their actual distances, i.e. $\bar{d}(a, b) = d(a, b), \bar{d}(a, c) = d(a, c)$. *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 non-metric 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, c in place of that between c, 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 (i.e. $d(a, c)$ 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)-(28) and so PD3_a is always guaranteed to lead to an f_{app}-approximate solution as well. *Therefore,* PD3_a *directly generalizes* $\text{PD2}_{\mu=1}$ *(i.e. the α-expansion) to the case of a non-metric 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 non-metric 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, q respectively (due to this fact we will call labels c, 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, 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:

---

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

Theorem VI.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 V.2), whereas in the non-metric 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.

Algorithm PD3$_c$: PD3$_c$ first adjusts (if needed) the dual solution $\mathbf{y}$ so that, for any 2 neighbors p, q, it holds: $\text{load}_{pq} \leq w_{pq}\big(d(a,c) + d(c,b)\big)$. After this initial adjustment, which is always easy to achieve, PD3$_c$ proceeds exactly as 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 $\bar{d}(a,b) \leq d(a,c) + d(c,b)$ and so cap_{pq} in (29) is valid, i.e. $\text{cap}_{pq} \geq 0$. PD3$_c$, 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, b (i.e. $x_p \equiv a \neq c$, $x_q \equiv b \neq c$). In that case it also holds that $d(a,c) + d(c,b) < d(a,b)$ and so:

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

Furthermore, it is easy to show that if none of p, q is assigned a new label by $\mathbf{x}'$ (i.e. they both retain their current labels a, b), then the resulting load will be equal to $w_{pq}\bar{d}(a,b)$, i.e. 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 non-metric 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 PD2$_{\mu=1}$ satisfy exactly the same conditions, except for condition (26). In place of that condition, the intermediate solutions of PD3$_c$ satisfy:

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

where $\hat{d}(a,b) = \min_{c \in \mathcal{L}}\big(d(a,c) + d(c,b)\big)$. By applying then the same (as in PD2$_{\mu=1}$) dual fitting factor to the last dual solution of PD3$_c$, one can easily prove that PD3$_c$ leads to an f'_{app}-approximate solution where:

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

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

VII. Experimental results

We first describe certain properties of the proposed algorithms that prove to be very useful in practice (Section VII-A). We then proceed and 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 VII-A, VII-B), image restoration (Section VII-C), image completion (Section VII-C), as well as optical flow estimation (Section VII-D). Finally, to further analyze their performance, results on synthetic problems are shown in Section VII-E. We note that, in each experiment, identical settings (i.e. parameters and initial solution) have been used for all algorithms and, in addition, initialization was chosen randomly.

A. 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 Metric Labeling. In particular, as implied by the Primal-Dual Principle of section III, 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 (i.e. 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 allowed disparity. The vertices of the graph $\mathcal{G}$ are the image pixels
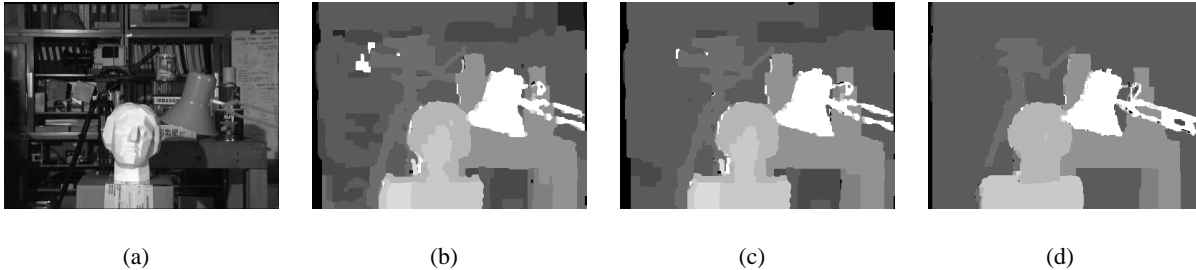
Fig. 8: (a) Tsukuba image **(b)** Disparity estimated by PD1 **(c)** and PD2$_{\mu=1}$ algorithm. The Potts distance (a metric) was used and so PD3$_a$, PD3$_b$, PD3$_c$ produced the same result with PD2$_{\mu=1}$. No tuning of parameters took place in (b) and (c). **(d)** Our result when using same parameters as in [21] (following the notation of [21], we have used $s = 50$, $T = 4$, $P = 2$).

and the edges of $\mathcal{G}$ connect each pixel to its 4 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:

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

where $I_{\text{left}}, I_{\text{right}}$ 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. A sample 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, e.g. see Fig.8(d) 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, i.e. to test the effectiveness of these algorithms in minimizing the objective function.

To this end, 3 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 non-metric), defined as follows (where λ denotes some constant):

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

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 I. The columns $f_{\text{app}}^{\text{PD1}}$, $f_{\text{app}}^{\text{PD2}_{\mu=1}}$, $f_{\text{app}}^{\text{PD3}_a}$, $f_{\text{app}}^{\text{PD3}_b}$, $f_{\text{app}}^{\text{PD3}_c}$ of that table list these averages for the

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

TABLE I: Average suboptimality bounds (colums 2-6) obtained over all Tsukuba stereo pairs. 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 PD2$_{\mu=1}$ can be applied only if distance $d(\cdot, \cdot)$ is a metric and in that case PD2$_{\mu=1}$, PD3$_a$, PD3$_b$ and PD3$_c$ (as well as their bounds) coincide.

algorithms PD1, PD2$_{\mu=1}$, PD3$_a$, PD3$_b$ and PD3$_c$ respectively. In addition, the last column lists the value of the corresponding approximation factor f_{app}, which, as already proved, makes up a worst-case suboptimality bound for most of the above algorithms. By observing table I, one can conclude that the per-instance suboptimality bounds are often much tighter (i.e. much closer to 1) than the worst-case bounds predicted in theory. In our stereo experiments, this was true for all combinations of algorithms and distances, and so in this particular case *the presented algorithms were able to extract a nearly optimal solution even when a non-metric distance was used.*

Besides the tightness of the per instance suboptimality bounds, another important issue is their accuracy, i.e. 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 scanline 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 II. Its columns $f_{\mathrm{true}}^{\mathrm{PD1}}$, $f_{\mathrm{true}}^{\mathrm{PD2}_{\mu=1}}$, $f_{\mathrm{true}}^{\mathrm{PD3}_a}$, $f_{\mathrm{true}}^{\mathrm{PD3}_b}$, $f_{\mathrm{true}}^{\mathrm{PD3}_c}$ contain the true average suboptimality of the solutions of PD1, PD2$_{\mu=1}$, PD3$_a$, PD3$_b$ and PD3$_c$ respectively, where the average is taken over all image scanlines. 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, meaning that these bounds were relatively accurate and therefore reliable for judging the solution's goodness. Furthermore, in this way, we can decide if a bad 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 scanlines to that on grids. One potential issue is that in the former case the integrality gap is 1, while in the latter it may be greater than 1, which may contribute to the inaccuracy of the suboptimality bound for graphs with loops.
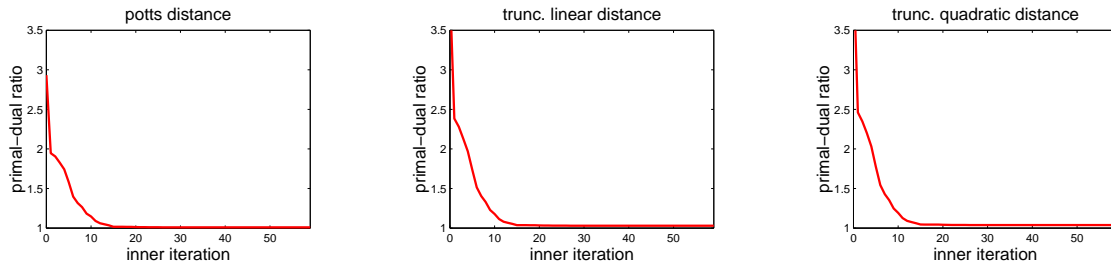
Fig. 9: These 3 plots show how the primal-dual ratios vary during the first 4 outer iterations (or equivalently the first $60 = 4 \cdot 15$ inner iterations) using the Tsukuba sequence as input. **(Left)** The Potts function, **(Middle)** the trunc. linear function and **(Right)** the trunc. quad. function have been used respectively as label distance $d(\cdot, \cdot)$. Notice how rapidly the ratios drop in all cases (i.e. they get very close to 1 just after a few inner iterations).

For the Tsukuba sequence, on average 4 outer iterations (or equivalently $60 = 4 \cdot 15$ inner iterations) are needed for the algorithms to terminate. The corresponding running time is 46 secs (measured on a 2.4GHz CPU). The plots in Figure 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 been used, whereas for the last one a non-metric 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, meaning 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 (e.g. 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.

B. Stereo matching

Besides the Tsukuba dataset, we have also applied our algorithms to image pairs from the SRI tree image sequence (Fig. 10(a)). The selected pairs had a maximum disparity of 11 pixels. Given

Distance	$f_{\text{app}}^{\text{PD1}}$	$f_{\text{true}}^{\text{PD1}}$	$f_{\text{app}}^{\text{PD2}_{\mu=1}}$	$f_{\text{true}}^{\text{PD2}_{\mu=1}}$	$f_{\text{app}}^{\text{PD3}_a}$	$f_{\text{true}}^{\text{PD3}_a}$	$f_{\text{app}}^{\text{PD3}_b}$	$f_{\text{true}}^{\text{PD3}_b}$	$f_{\text{app}}^{\text{PD3}_c}$	$f_{\text{true}}^{\text{PD3}_c}$
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

TABLE II: The average suboptimality bounds (columns 2-4-6-8-10), obtained when applying our stereo matching algorithms to one scanline at a time (instead of the whole image). In this case, we are also able to compute the true average suboptimality (columns 3-5-7-9-11) of the generated solutions, using dynamic programming. As can be seen, by inspecting the table, the suboptimality bounds approximate the true suboptimality relatively well, meaning that they can be safely used as a measure for judging the goodness of the generated solution in this case.

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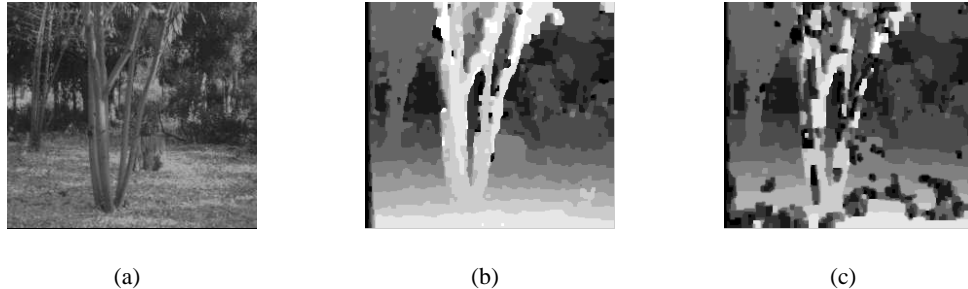
Fig. 10: (a) One image from the SRI tree image sequence. **(b)** Computed disparities when using PD3$_a$ and the distance d_4 with $(\kappa, \lambda) = (2, 10)$. **(c)** Disparities computed by the α-β-swap algorithm using the same distance.

our algorithms' ability to handle both metric and non-metric distances equally well, the following non-metric 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 (i.e. $\leq \kappa$) 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 Figure 10(b). The average running time to convergence has been 33 secs. We have also applied the α-β-swap algorithm [1] to the SRI dataset, using exactly the same settings. Although this graph-cut based algorithm is applicable even in the case of a non-metric label distance, its disadvantage is that it may get trapped to a bad local minimum, i.e. 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% higher than the energy estimated by our method. The corresponding disparity is shown in Figure 10(c).

As a further example, we illustrate how one could favor disparities that are not violating the uniqueness constraint, just by use of an appropriate non-metric 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 $\mathfrak{c}_{\hat{o}}$ for all pixels, i.e. $\mathfrak{c}_p(\hat{o}) = \mathfrak{c}_{\hat{o}}$. Assuming (without loss of generality) that image scanlines 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. 11(a)). 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 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, 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. E.g. both can be set equal to the Potts metric: $\text{hdist}^1 = \text{vdist}^1 = d_1$. 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 $\varepsilon \ll M$):
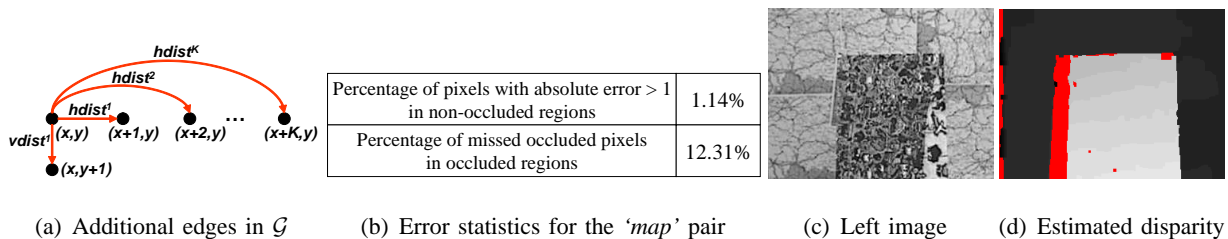
$$b = a + k \Rightarrow \text{hdist}^k(a, b) = M, \quad b \neq a + k \ \& \ b \neq a \Rightarrow \text{hdist}^k(a, b) = \varepsilon, \quad b = a \Rightarrow \text{hdist}^k(a, b) = 0$$

A result of applying this distance (with $\mathfrak{c}_{\hat{o}} = 23, M = 10, \varepsilon = 0.01$) to the *map* stereo pair, appears in Fig. 11(d). Error statistics are displayed in Fig. 11(b).
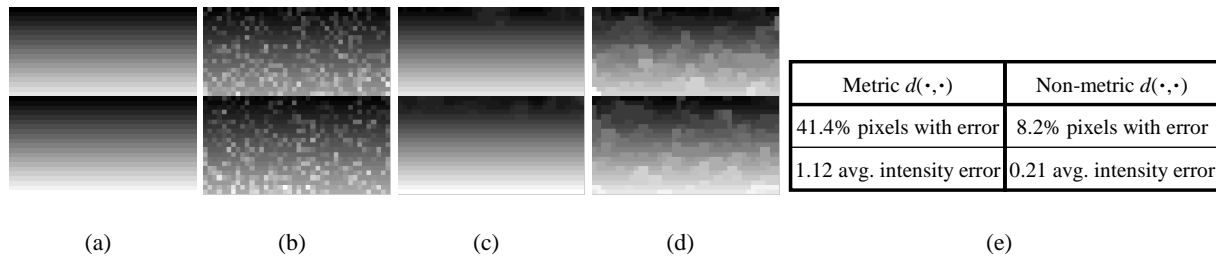
C. Image restoration and image completion

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

The example of Fig. 12 illustrates the importance of using non-metric distances $d(\cdot, \cdot)$ on the task of image restoration as well. The original image (Fig. 12(a)) consists of 2 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. 12(b)). Although our algorithms managed to restore

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

(a) Additional edges in $\mathcal{G}$ (b) Error statistics for the *'map'* pair   (c) Left image   (d) Estimated disparity

Fig. 11: Red pixels in (d) indicate occlusions

| 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)

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

the original image with only a few errors, by use of the non-metric distance d_4 (Fig. 12(c)), this wasn't 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. 12(d). The error statistics for this restoration example are shown in table 12(e).

Another non-metric 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. 13(a). In this case, the following function (which is more robust against outliers) has been used for the label costs: $\mathfrak{c}_p(a) = \lambda_0 \min(|I(p) - a|^2, \lambda_1)$, with $\lambda_0 = 0.05, \lambda_1 = 10^4$. Notice that our algorithm managed not only to remove the noise completely (see Fig. 13(b)), but also to maintain the boundaries of the objects at the same time.

The same distance (i.e. the truncated quadratic) can be also used for the task of image completion. Besides containing Gaussian noise, the image in Fig. 13(c) also has a part which has been masked. The labels costs of masked pixels have been set to zero, while for the rest of the pixels the costs have been set as before. As can be seen, from Fig. 13(d), our algorithm

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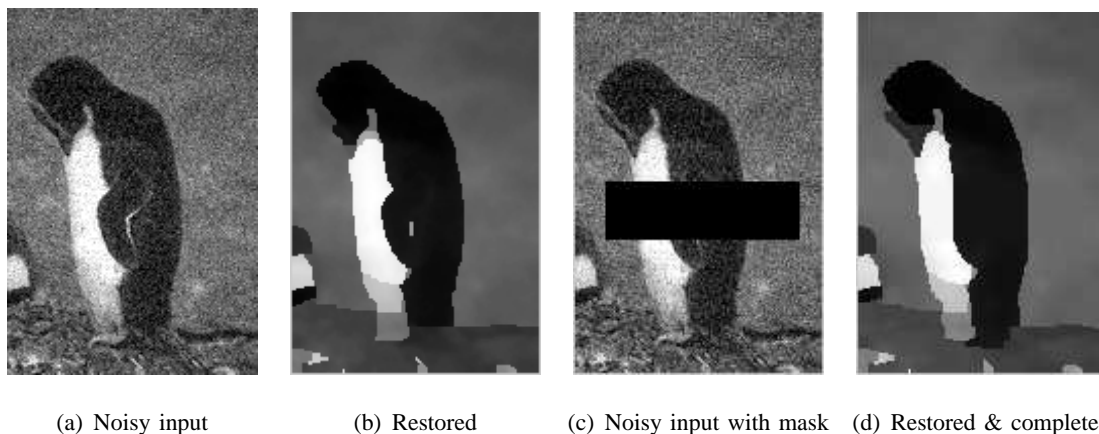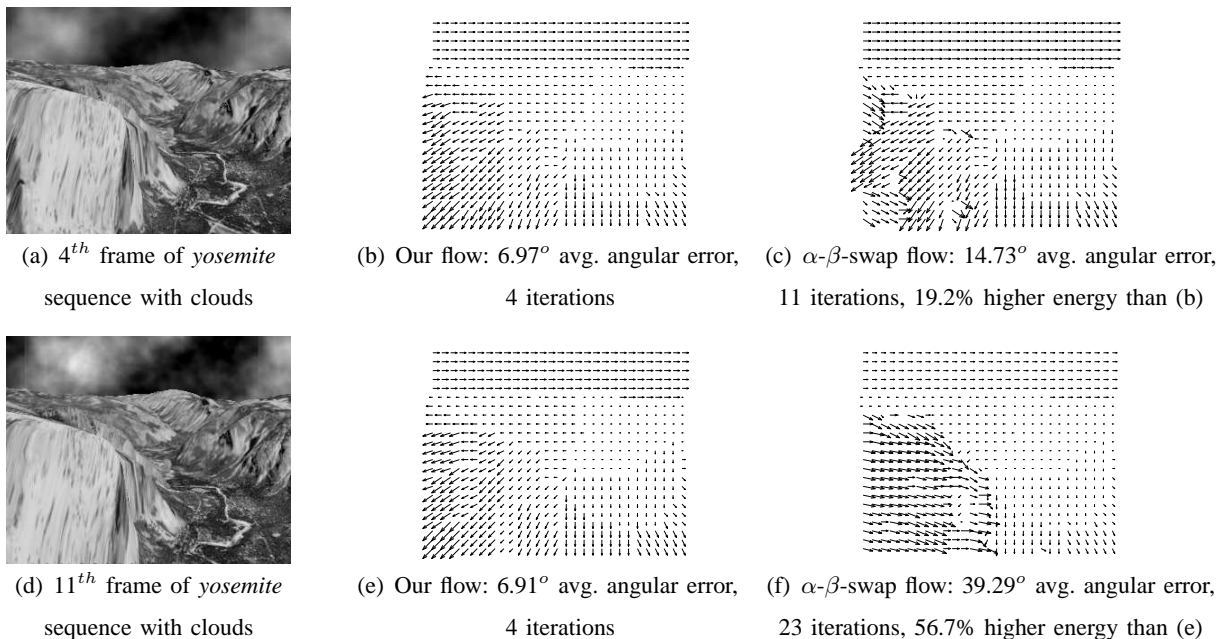
Fig. 13: Examples of image restoration and image completion

managed not only to remove the noise again, but also to fill the missing part in a plausible way.

D. Optical flow estimation

Global methods [22], [23] estimate optical flow u_x, u_y by minimizing a functional of the form: $E(u_x, u_y) = \int_I \rho_D(I_x u_x + I_y u_y + I_t) + \lambda \cdot \rho_S(\sqrt{|\nabla u_x|^2 + |\nabla u_y|^2}) dx dy$, where I_x, I_y, I_t denote spatial and temporal image derivatives, while 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 1st term (which expresses the *optic flow constraint equation*) and the 2nd 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] (e.g. 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, I_t (in the 1st term of the above functional) 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. E.g. we can set $\mathfrak{c}_p(a) = |I_1(p + a) - I_0(a)|$, where I_0, I_1 are the current and next image. In this case, due to the two-dimensional 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 non-metric distance between labels can be used: $d(a, b) = \text{dist}(a, b) + \tau \cdot \text{angledist}(a, b)$. Here, $\text{dist}(a, b)$ denotes a truncated euclidean distance between the optical flow vectors a, b, i.e. $\text{dist}(a, b) = \min(||a - b||, \lambda)$, while the 2nd 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, b and is defined as follows: $\text{angledist}(a, b)$ equals 1 if the angle (in degrees) between a and b is greater than 45°, while in all other cases 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 above distance with parameters $\lambda = 5, \tau = 5$. The results, as well as error statistics, are shown in Figure 14. Due to the bigger number of labels, the run times of our algorithm for this example were approximately 6 minutes on average. *We note that, although both algorithms are trying*

(a) 4th frame of *yosemite* sequence with clouds

(b) Our flow: 6.97^o avg. angular error, 4 iterations

(c) α-β-swap flow: 14.73^o avg. angular error, 11 iterations, 19.2% higher energy than (b)

(d) 11th frame of *yosemite* sequence with clouds

(e) Our flow: 6.91^o avg. angular error, 4 iterations

(f) α-β-swap flow: 39.29^o avg. angular error, 23 iterations, 56.7% higher energy than (e)

Fig. 14: Estimated flow between frames 4, 5 (1st row) and 11, 12 (2nd row) of *yosemite* sequence. Although more outer iterations were used by α-β-swap, its optical flow had 19.2% and 56.7% higher energy than our optical flow.
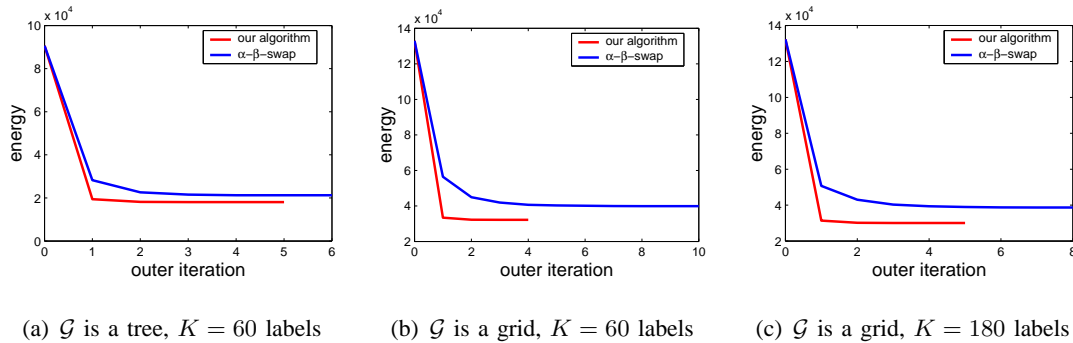
to minimize exactly the same objective function, the resulting solutions of α-β-swap have much higher energy. It seems that, contrary to our method, α-β-swap needs to be properly initialized, or else is not powerful enough to escape from bad local minima in this case.

E. 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 $\mathcal{G}$, while 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, while, for the pairwise potentials, a random non-metric distance has been used, that 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 1st one (Fig. 15(a)), a random spanning tree of the 30×30 grid was used as the graph $\mathcal{G}$ and the number of labels was $K = 60$, while, in the 2nd (Fig. 15(b)) and 3rd (Fig. 15(c)) experiment, the graph $\mathcal{G}$ had 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 $\varrho_0 = 1, \varrho_1 = 100$) and the

(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

Fig. 15: α-β-swap produces an energy which is higher by **(a)** 17%, **(b)** 23% and **(c)** 28% with respect to our algorithm's energy. Notice that as the number of labels increases the gap in performance increases as well.

resulting average energies per outer iteration, for both our algorithm and the α-β-swap algorithm, are shown in the plots of Figure 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 less iterations to converge. This behavior is a typical one and has been observed in real problems as well. Notice also that, as the number of labels or the graph complexity increases, the gap in performance between the 2 algorithms increases as well.

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

Fig. 16: A synthetic example, where the graph $\mathcal{G}$ has 3 vertices $\{p, q, r\}$ and 2 edges $\{pq, qr\}$, while the labels $\mathcal{L}$ are $\{a, b, c\}$. Label costs $\mathfrak{c}_p(\cdot)$ 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, i.e. arbitrarily larger than the true minimum cost, which is 4 (labeling B). On the contrary, PD3$_a$, PD3$_b$ and PD3$_c$ can always locate the optimal labeling B. Example taken from [1].

VIII. CONCLUSIONS

A new theoretical framework has been proposed, for both understanding and developing algorithms that can approximately optimize MRFs with both metric and non-metric 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 non-metric 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 (i.e. very close to 1), meaning that the resulting solutions are nearly optimal. The theoretical setting of the proposed framework rests on duality theory of linear programming, 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 proved 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, e.g., to PD1 (see tables I and II)), 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 non-metric MRFs, PD3 algorithms exhibit similar performance and gave the best results in practice. Furthermore, they reduce to PD2$_{\mu=1}$ in the case of a metric potential function. Therefore, based also on the fact that PD3$_a$ and 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]. E.g., such is the case with the Generalized Potts model, whose integrality gap is known to be 2 [2], i.e. 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," *PAMI*, vol. 23, no. 11, 2001.

[2] J. Kleinberg and E. Tardos, "Approximation algorithms for classification problems with pairwise relaionships: metric labeling and markov random fields," *Journal of the ACM*, vol. 49, pp. 616–630, 2002.

[3] O. Veksler, "Efficient graph-based energy minimization methods in computer vision," Ph.D. dissertation, Cornell, 1999.

[4] S. Roy and I. Cox, "A maximum-flow formulation of the n-camera stereo correspondence problem," in *ICCV*, 1998.

[5] A. Gupta and E. Tardos, "Constant factor approximation algorithms for a class of classification problems," in *Proceedings of the 32nd Annual ACM Symposium on the theory of Computing*, 2000, pp. 652–658.

[6] H. Ishikawa and D. Geiger, "Segmentation by grouping junctions," in *CVPR*, 1998.

[7] C. Chekuri, S. Khanna, J. Naor, and L. Zosin, "Approximation algorithms for the metric labeling problem via a new linear programming formulation," in *12th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2001, pp. 109–118.

[8] A. Archer, J. Fakcharoenphol, C. Harrelson, R. Krauthgamer, K. Talvar, and E. Tardos, "Approximate classification via earthmover metrics," in *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2004.

[9] N. Komodakis and G. Tziritas", "A new framework for approximate labeling via graph-cuts," in *ICCV*, 2005.

[10] N. Komodakis and G. Tziritas, "Approximate labeling via the primal-dual schema," Computer Science Department, Tech. Rep. CSD-TR-05-01, February 2005 (http://www.csd.uoc.gr/~komod/publications/docs/CSD_TR_2005_01.pdf).

[11] M. Wainwright, T. Jaakkola, and A. Willsky, "Map estimation via agreement on (hyper)trees: messagepassing and linear programming approaches," in *Allerton Conference on Communication, 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." in *ECCV*, 2002, pp. 82–96.

[14] R. Zabih and V. Kolmogorov, "Spatially coherent clustering using graph cuts." in *CVPR*, 2004, pp. 437–444.

[15] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*.   San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988.

[16] V. Kolmogorov, "Convergent tree-reweighted message passing for energy minimization," in *AISTATS*, 2005.

[17] T. Meltzer, C. Yanover, and Y. Weiss, "Globally optimal solutions for energy minimization in stereo vision using reweighted belief propagation." in *ICCV*, 2005.

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

[19] C. Rother, S. Kumar, V. Kolmogorov, and A. Blake, "Digital tapestry," in *CVPR*, 2005.

[20] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision*, vol. 47, no. 1/2/3, pp. 7–42, April-June 2002.

[21] M. F. Tappen and W. T. Freeman, "Comparison of graph cuts with belief propagation for stereo, using identical mrf parameters." in *ICCV*, 2003, pp. 900–907.

[22] M. J. Black and P. Anandan, "The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields," *CVIU*, 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," *IJCV*, vol. 61, no. 3, pp. 211–231, 2005.