New Algorithms for Convex Cost Tension Problem with Application to Computer Vision[☆]

Vladimir Kolmogorov

Adastral Park Campus, University College London, Adastral Park, Martlesham Heath, IP5 3RE, UK

Akiyoshi Shioura

Graduate School of Information Sciences, Tohoku University, Sendai 980-8579, Japan

Abstract

Motivated by various applications to computer vision, we consider the convex cost tension problem, which is the dual of the convex cost flow problem. In this paper, we first propose a primal algorithm for computing an optimal solution of the problem. Our primal algorithm iteratively updates primal variables by solving associated minimum cut problems. We show that the time complexity of the primal algorithm is $\mathrm{O}(K \cdot T(n,m))$, where K is the range of primal variables and $T(n,m)$ is the time needed to compute a minimum cut in a graph with n nodes and m edges. We then develop an improved version of the primal algorithm, called the primal-dual algorithm, by making good use of dual variables in addition to primal variables. Although its time complexity is the same as that of the primal algorithm, we can expect a better performance in practice. We finally consider an application to a computer vision problem called the panoramic image stitching.

Key words: minimum cost tension, minimum cost flow, discrete convex function, submodular function

1. Introduction

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a simple undirected graph. We assume that if $(u,v) \in \mathcal{E}$ then $(v,u) \notin \mathcal{E}$. We consider the following optimization problem called the convex cost tension problem:

$$\text{(CTP): Minimize} \quad E(\boldsymbol{x}) = \sum_{u \in \mathcal{V}} D_u(x_u) + \sum_{(u,v) \in \mathcal{E}} V_{uv}(x_v - x_u) \quad \text{subject to} \quad \boldsymbol{x} \in \mathbb{Z}^{\mathcal{V}},$$

[☆]Preliminary version appeared as two separate technical reports [24, 34].

Email addresses: vnk@adastral.ucl.ac.uk (Vladimir Kolmogorov), shioura@dais.is.tohoku.ac.jp (Akiyoshi Shioura)

where $D_u : \mathbb{Z} \to \mathbb{R} \cup \{+\infty\}$ ($u \in \mathcal{V}$) and $V_{uv} : \mathbb{Z} \to \mathbb{R} \cup \{+\infty\}$ ($(u, v) \in \mathcal{E}$) are convex functions such that $\operatorname{dom} D_u = \{\alpha \in \mathbb{Z} \mid D_u(\alpha) < +\infty\}$ and $\operatorname{dom} V_{uv} = \{\alpha \in \mathbb{Z} \mid V_{uv}(\alpha) < +\infty\}$ are finite intervals. This problem is known as the dual of the convex cost flow problem and extensively discussed in the literature (see, e.g., [1, 20, 31, 33]). Recently, various applications of the problem (CTP) have been studied in the area of image processing and computer vision (see, e.g, [8, 9, 11, 19, 21, 26, 36, 37, 38, 39]). In this paper, we propose new algorithms for the problem (CTP) and prove a tight bound for the number of iterations required by the algorithms.

Previous Algorithms. Let $n = |\mathcal{V}|$, $m = |\mathcal{E}|$, and K be a positive integer such that

$$|\alpha - \beta| \leq K \qquad (\forall \alpha, \beta \in \operatorname{dom} D_u,\ \forall u \in \mathcal{V}). \tag{1.1}$$

Ishikawa [21] and Ahuja et al. [2] reduce the problem (CTP) to a minimum s-t cut problem in a graph with $O(nK)$ nodes and $O(mK^2)$ edges. In the important special case where the functions $V_{uv}(\cdot)$ are given by piecewise linear functions with a constant number of breakpoints, the number of edges is reduced to $O(mK)$. A disadvantage of this approach is that it needs a large amount of memory (either $O(mK^2)$ or $O(mK)$).

Algorithms in [2, 21] can be seen as primal algorithms since they directly solve the problem (CTP). An alternative is to solve the dual problem instead. Several dual algorithms are proposed by Karzanov and McCormick [22] and Ahuja et al. [1]. The worst-case complexity of the latter algorithm is $O(nm \log(n^2/m) \log(nK))$, which is the best known for (CTP).

It is known that the problem (CTP) can be reduced to a linear cost flow problem on a graph with $O(rm)$ edges, where r ($\leq K$) is the maximum number of breakpoints of piecewise-linear convex functions $D_u(\cdot)$ and $V_{uv}(\cdot)$. Therefore, it is possible to use any existing method for the linear cost flow problem. One of them, the primal-dual algorithm of Ford and Fulkerson [13, 14], is related to the technique that we develop in this paper. In particular, the two algorithms are equivalent in a special case when functions $D_u(\cdot)$ are linear and functions $V_{uv}(\cdot)$ have one breakpoint ($r = 1$). However, if $r > 1$ then the techniques are different; our algorithm works with a graph with $O(m)$ rather than $O(rm)$ edges.

Our Contributions. In this paper we propose two algorithms for the problem (CTP): primal and primal-dual. Our primal algorithm finds an optimal solution of (CTP) by at most $O(K)$ computation of a minimum cut of a graph with n nodes and m edges. The algorithm is similar to the steepest descent algorithm of Murota for the minimization of L$^\natural$-convex functions [29, 30, 31]. The minimization of L$^\natural$-convex functions is a more general problem than (CTP) (see Section 2.1 for the definition of L$^\natural$-convexity). The algorithm is also similar to that of Bioucas-Dias and Valadão [4] which is originally applied to the following problem without functions $D_u(\cdot)$:

$$(\mathrm{CTP}_0)\text{: Minimize} \quad E_0(\boldsymbol{x}) = \sum_{(u,v)\in\mathcal{E}} V_{uv}(x_v - x_u) \quad \text{subject to} \quad \boldsymbol{x} \in \mathbb{Z}^{\mathcal{V}}.$$

It should be noted that the problem (CTP) can be easily reduced to the problem (CTP_0), while (CTP_0) is apparently a special case of (CTP) (see Section 1.1).

Our major contribution is to provide a tight bound on the number of iterations, while bounds in [4, 31] are much weaker. Our proof is based on the analysis of the L_∞ distance between the current feasible solution and an optimal solution, and it is shown that the distance decreases monotonically in each iteration. The proof is applicable not only to the problem (CTP), but also to the minimization of $L^\natural$-convex functions. Indeed, our analysis is based on the theory of $L^\natural$-convex functions. Hence, our result also implies a better bound on the number of iterations for Murota's steepest descent algorithm. In particular, our result shows that Murota's algorithm as well as our primal algorithm yields the best known technique for minimizing $L^\natural$-convex functions.

One drawback of the primal algorithm is that it solves different min-cut/max-flow problems independently, although these problems are strongly related. Thus, a natural idea for speeding up computations is to use maximum flow obtained in one iteration as an initialization for the next iteration. This is a motivation of our primal-dual method, which is an improved version of the primal algorithm by making good use of dual variables in addition to primal variables. This method can be viewed as a generalization of the primal-dual method of Ford and Fulkerson [13, 14] for linear costs to convex costs. Our primal-dual algorithm is also closely related to (but different from) the out-of-kilter algorithm and the successive shortest path algorithm for the convex cost flow problem (see, e.g., [3, 27, 33]) since both of the algorithms as well as their analysis use similar primal-dual techniques. Our contribution with regard to the primal-dual algorithm is to analyze the behavior and the running time of the algorithm from the viewpoint of the convex cost tension problem, not of the convex cost flow problem.

The time complexity of both algorithms is $O(K \cdot T(n, m))$, where $T(n, m)$ is the running time of a single max-flow/min-cut computation on a graph with n nodes and m edges. This is worse than the complexity of the algorithm in [1]. Our techniques, however, have a practical advantage: they rely only on a max-flow/min-cut algorithm, which is more readily available. For example, it is possible to use a max-flow/min-cut algorithm that is specifically tuned to computer vision problems [7]. Experimental results of our algorithms are shown in Section 4

Although the algorithms described above are pseudo-polynomial, it is possible to apply proximity scaling technique to get an algorithm polynomial in $\log K$ rather than K (see, e.g., [30, Sec. 10.3.2]). In particular, combining proximity scaling technique with our algorithms yields the time complexity $O(n \log K \cdot T(n, m))$, as shown in Section 2.3.

Other Related Work. Hochbaum [19] gives a very efficient algorithm for a special case of (CTP). Namely, if functions V_{uv} are given as $V_{uv}(x_v - x_u) = \lambda_{uv}|x_v - x_u|$, then the technique in [19] has almost the same time complexity as that of a single max-flow computation on a graph with n nodes and m edges. Similar ideas appear in [9, 11, 36, 37, 38]. The method is applicable to the problem of image restoration using total variation minimization [9, 11, 19, 36].

If functions $D_u(\cdot)$ are arbitrary and $V_{uv}(\cdot)$ are convex, then the problem can be solved exactly in time $T(nK, mK^2)$ or $T(nK, mK)$, depending on the structure of the functions V_{uv} [2, 21]. If both $D_u(\cdot)$ and $V_{uv}(\cdot)$ are arbitrary then the problem becomes NP-hard. Boykov et al. [8], Kleinberg and Tardos [23], and Komodakis and Tziritas [25] give constant factor approximation algorithms in the case when the functions $V_{uv}(\cdot)$ are metrics. Veksler [35] uses the same procedures as our primal algorithm, as a heuristic tool for minimizing a function with nonconvex terms of the form $V_{uv}(x_v - x_u) = \lambda_{uv} \min\{|x_v -$

$x_u|, 1\}$. For the problem (CTP_0) with nonconvex $V_{uv}(\cdot)$, Bioucas-Dias and Valadão [4] use their primal algorithm to obtain a good feasible solution.

Application to Computer Vision Problems. The problem (CTP) arises in many applications in computer vision such as panoramic image stitching [26, 39], image restoration [8], minimization of total variation [11], and phase unwrapping in SAR images [4]. In such applications, the node set $\mathcal{V}$ of the undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ usually corresponds to the set of pixels in a given image, and variable x_u represents the label of the pixel $u \in \mathcal{V}$ which must belong to a finite set of integers $\{0, 1, \ldots, K-1\}$. For motion or stereo, labels are disparities, while for image restoration they represent intensities. Functions $D_u(\cdot)$ encode unary data penalty functions, and $V_{uv}(\cdot)$ are pairwise interaction potentials. The objective function of (CTP) is often derived in the context of Markov random fields [16]; a minimum of E corresponds to a maximum a-posteriori labeling $\boldsymbol{x}$.

In this paper, we consider the panoramic image stitching problem which inspired our work. Given different portions of the same scene with some overlap, the goal of panoramic image stitching is to generate an output image which is similar to the original images and does not have a visible seam. The approach of [26, 39] is to compute the image whose gradients match the gradients of the two input images, which can be done by solving an instance of (CTP). We apply our algorithms to some instances of (CTP) arising from actual panoramic image stitching problems, and test the empirical performance of our algorithms.

Outline. In Section 2, we describe a primal algorithm and prove a bound on the number of iterations. In Section 3, we review the dual problem and present a primal-dual algorithm. Finally, an application to the panoramic image stitching is discussed in Section 4. In Appendix we discuss the relationship between our primal algorithm and Murota's steepest descent algorithm.

1.1. Equivalence between (CTP) *and* (CTP_0)

We discuss the equivalence between the two problems (CTP) and (CTP_0). While (CTP_0) is a special case of (CTP), it is known that (CTP) can be reduced to (CTP_0), as shown below. Hence, (CTP) and (CTP_0) are essentially equivalent to each other, and any algorithm for the one problem can be adapted to the other.

Let $E : \mathbb{Z}^{\mathcal{V}} \to \mathbb{R} \cup \{+\infty\}$ be the objective function of (CTP). Define a function $\tilde{E} : \mathbb{Z}^{\tilde{\mathcal{V}}} \to \mathbb{R} \cup \{+\infty\}$ by

$$\tilde{E}(x_0, \boldsymbol{x}) = E(\boldsymbol{x} - x_0\mathbf{1}) \quad (x_0 \in \mathbb{Z}, \boldsymbol{x} \in \mathbb{Z}^{\mathcal{V}}),$$

where 0 denotes a new vertex not in $\mathcal{V}$, $\tilde{\mathcal{V}} = \{0\} \cup \mathcal{V}$, and $\mathbf{1} \in \mathbb{Z}^{\mathcal{V}}$ is the vector with all components equal to one. Then, we have

$$\begin{aligned} \tilde{E}(x_0, \boldsymbol{x}) &= E(\boldsymbol{x} - x_0\mathbf{1}) \\ &= \sum_{u \in \mathcal{V}} D_u(x_u - x_0) + \sum_{(u,v) \in \mathcal{E}} V_{uv}(x_v - x_u) \quad (x_0 \in \mathbb{Z}, \boldsymbol{x} \in \mathbb{Z}^{\mathcal{V}}). \end{aligned}$$

We put $\tilde{\mathcal{E}} = \{(0, u) \mid u \in \mathcal{V}\} \cup \mathcal{E}$ and define a function $V_{0u} : \mathbb{Z} \to \mathbb{R} \cup \{+\infty\}$ $(u \in \mathcal{V})$ by

$$V_{0u}(\alpha) = D_u(\alpha) \qquad (\alpha \in \mathbb{Z}).$$

Input: initial feasible solution $\boldsymbol{x} := \boldsymbol{x}^\circ \in \operatorname{dom} E$.
Step 1: Set SuccessUp := false, SuccessDown := false.
Step 2: Do UP or DOWN in any order until SuccessUp = SuccessDown = true:
UP (do only if SuccessUp is false):
- Compute $X^+ \in \arg\min\{E(\boldsymbol{x} + \chi_X) \mid X \subseteq \mathcal{V}\}$.
- If $E(\boldsymbol{x} + \chi_{X^+}) = E(\boldsymbol{x})$, set SuccessUp := true; otherwise set $\boldsymbol{x} := \boldsymbol{x} + \chi_{X^+}$.

DOWN (do only if SuccessDown is false):
- Compute $X^- \in \arg\min\{E(\boldsymbol{x} - \chi_X) \mid X \subseteq \mathcal{V}\}$.
- If $E(\boldsymbol{x} + \chi_{X^-}) = E(\boldsymbol{x})$, set SuccessDown := true; otherwise set $\boldsymbol{x} := \boldsymbol{x} - \chi_{X^-}$.

Step 3: Output $\boldsymbol{x}$ and stop.

Figure 1: Primal algorithm

Then, each V_{0u} is a convex function and it holds that

$$\tilde{E}(x_0, \boldsymbol{x}) = \sum_{(u,v)\in\tilde{\mathcal{E}}} V_{uv}(x_v - x_u) \qquad (x_0 \in \mathbb{Z}, \boldsymbol{x} \in \mathbb{Z}^{\mathcal{V}}).$$

Hence, we obtain an objective function of (CTP_0). This shows that (CTP) can be reduced to (CTP_0).

2. Primal Algorithm

As we have mentioned in Introduction, our primal algorithm is very similar to those in [4] and in [31]. It iteratively invokes the following subroutine: given a current feasible solution $\boldsymbol{x}$, compute the minimum of the function $\hat{E}(\boldsymbol{b}) = E(\boldsymbol{x} + \sigma\boldsymbol{b})$, where $\sigma = \pm 1$ is fixed and $\boldsymbol{b}$ is a 0-1 vector. Function $\hat{E}(\boldsymbol{b})$ can be written as the sum of functions in binary variables:

$$\hat{E}(\boldsymbol{b}) = \sum_{u\in\mathcal{V}} \hat{D}_u(b_u) + \sum_{(u,v)\in\mathcal{E}} \hat{V}_{uv}(b_u, b_v),$$

where

$$\hat{D}_u(b_u) = D_u(x_u + \sigma b_u), \qquad \hat{V}_{uv}(b_u, b_v) = V_{uv}((x_v + \sigma b_v) - (x_u + \sigma b_u)).$$

Note that function $\hat{E}(\boldsymbol{b})$ can be minimized in polynomial time by computing a minimum cut of an appropriately constructed graph (see [6], for example).

Our primal algorithm is presented in Fig. 1, where for any subset X of $\mathcal{V}$, we denote by $\chi_X \in \{0,1\}^{\mathcal{V}}$ the characteristic vector of X, i.e.,

$$(\chi_X)_u = \begin{cases} 1 & (u \in X), \\ 0 & (u \in \mathcal{V} \setminus X). \end{cases}$$

Its difference from the algorithm of Bioucas-Dias and Valadão [4] is very minor: the latter is applicable only to (CTP_0) and uses only procedure UP. Murota's algorithm can be seen as a specialized implementation of our primal algorithm; while our algorithm has a flexibility in the choice of the procedures UP and DOWN, Murota's algorithm computes

both of X^+ and X^- and chooses a better one by comparing the function values of $E(\boldsymbol{x}+\chi_{X^+})$ and $E(\boldsymbol{x}-\chi_{X^-})$; see Appendix for more discussion on Murota's algorithm.

In the following, we analyze the number of iterations of our primal algorithm. This analysis is a major contribution of our paper with regard to the primal algorithm. It leads to a tight bound on the number of iterations improving the bounds in [4] and [31]. Bioucas-Dias and Valadão [4] show that if a feasible solution $\boldsymbol{x}$ is not an optimal solution, then the objective function value $E(\boldsymbol{x})$ is decreased in the next iteration. This gives a non-polynomial bound on the number of iterations, assuming that E is an integer-valued function. Murota [31] proves that his algorithm terminates in $\mathrm{O}(nK)$ iterations.

We will show that our primal algorithm terminates in $\mathrm{O}(K)$ iterations.

Theorem 2.1. *Our primal algorithm finds an optimal solution of the problem* (CTP) *in* $2K+2$ *iterations.*

Proof. This is an immediate corollary of a more general result (Theorem 2.8) to be shown in Section 2.2. □

Remark 2.2. The tight bound $\mathrm{O}(K)$ is originally shown in the technical report version [24] of this paper. The same bound is also shown in Darbon's PhD thesis [10], which is published after [24]; indeed, [10] cites [24]. Recently, Bioucas-Dias and Valadão [5] (the journal version of [4]) show that their algorithm for (CTP_0) terminates in $\mathrm{O}(K)$ iterations, where their proof is a simplified version of those in [10, 24]. □

Clearly, in some cases this bound is tight. For example, consider the following problem:

$$\text{Minimize} \quad D_1(x_1)+D_2(x_2) \quad \text{subject to} \quad (x_1,x_2)\in\mathbb{Z}^2,$$

where k is a positive integer and $D_1, D_2 : \mathbb{Z}\to\mathbb{R}\cup\{+\infty\}$ are functions defined by

$$D_1(\alpha)=\begin{cases}\alpha & (0\le\alpha\le k),\\ +\infty & (\text{otherwise}),\end{cases} \qquad D_2(\alpha)=\begin{cases}-\alpha & (0\le\alpha\le k),\\ +\infty & (\text{otherwise}).\end{cases}$$

This problem is a special case of (CTP) with $K=k$ and $(0,k)$ is a unique optimal solution. If the primal algorithm starts with $\boldsymbol{x}^\circ=(k,0)$, then it requires $2k+2=2K+2$ iterations.

Our proof for the bound relies on the theory of discrete convex functions called $L^\natural$-*convex functions*. The next section gives some background on $\mathrm{L}^\natural$-convex functions.

2.1. $L^\natural$-convex Functions

The concept of $\mathrm{L}^\natural$-convexity is introduced by Fujishige and Murota [15] as a variant of L-convexity by Murota [28]. In this section we review some fundamental results on $\mathrm{L}^\natural$-convex functions.

A function $E:\mathbb{Z}^{\mathcal{V}}\to\mathbb{R}\cup\{+\infty\}$ with nonempty $\operatorname{dom}E$ is called L-convex if it satisfies the following properties:

(LF1) $E(\boldsymbol{x})+E(\boldsymbol{y})\ge E(\boldsymbol{x}\wedge\boldsymbol{y})+E(\boldsymbol{x}\vee\boldsymbol{y}) \quad (\forall\boldsymbol{x},\boldsymbol{y}\in\operatorname{dom}E)$,
(LF2) $\exists r\in\mathbb{R}$ such that $E(\boldsymbol{x}+\lambda\mathbf{1})=E(\boldsymbol{x})+\lambda r \quad (\forall\boldsymbol{x}\in\operatorname{dom}E,\ \forall\lambda\in\mathbb{Z})$,

where $\operatorname{dom} E = \{\boldsymbol{x} \in \mathbb{Z}^V \mid E(\boldsymbol{x}) < +\infty\}$ and the vectors $\boldsymbol{x} \wedge \boldsymbol{y}, \boldsymbol{x} \vee \boldsymbol{y} \in \mathbb{Z}^{\mathcal{V}}$ are defined by

$$(\boldsymbol{x} \wedge \boldsymbol{y})_u = \min\{x_u, y_u\}, \quad (\boldsymbol{x} \vee \boldsymbol{y})_u = \max\{x_u, y_u\} \quad (u \in \mathcal{V}).$$

Throughout the paper, we assume that the value r in the property (LF2) is zero. Note, without this condition an L-convex function E does not have a minimum.

A function $E : \mathbb{Z}^{\mathcal{V}} \to \mathbb{R} \cup \{+\infty\}$ with $\operatorname{dom} E \neq \emptyset$ is called $\mathrm{L}^\natural$-convex if the function $\tilde{E} : \mathbb{Z}^{\tilde{\mathcal{V}}} \to \mathbb{R} \cup \{+\infty\}$ defined by

$$\tilde{E}(x_0, \boldsymbol{x}) = E(\boldsymbol{x} - x_0\mathbf{1}) \quad (x_0 \in \mathbb{Z}, \boldsymbol{x} \in \mathbb{Z}^{\mathcal{V}}) \tag{2.1}$$

is L-convex, where 0 denotes a new element not in $\mathcal{V}$ and $\tilde{\mathcal{V}} = \{0\} \cup \mathcal{V}$. $\mathrm{L}^\natural$-convex functions are conceptually equivalent to L-convex functions, while the class of $\mathrm{L}^\natural$-convex functions contains that of L-convex functions as a proper subclass. $\mathrm{L}^\natural$-convexity is equivalent to the combination of submodularity and integral convexity [12] (see [30] for details).

The next property shows that the problem (CTP) (resp., (CTP_0)) is a special case of the minimization of an $\mathrm{L}^\natural$-convex function (resp., L-convex function).

Proposition 2.3 (cf. [30, Sec. 7.3]).
(i) *The objective function $E : \mathbb{Z}^{\mathcal{V}} \to \mathbb{R} \cup \{+\infty\}$ of* (CTP) *is $L^\natural$-convex.*
(ii) *The objective function $E_0 : \mathbb{Z}^{\mathcal{V}} \to \mathbb{R} \cup \{+\infty\}$ of* (CTP_0) *is L-convex with $r = 0$ in* (LF2).

$\mathrm{L}^\natural$-convexity of a function can be characterized by the following properties. We denote by $\mathbb{Z}_+$ the set of nonnegative integers. For a vector $\boldsymbol{x} \in \mathbb{Z}^{\mathcal{V}}$ we define

$$\arg\max\{x_u \mid u \in \mathcal{V}\} = \{u \in \mathcal{V} \mid x_u \geq x_v \ (\forall v \in \mathcal{V})\}.$$

Theorem 2.4 ([30, Th. 7.1, 7.7]). *Let $E : \mathbb{Z}^{\mathcal{V}} \to \mathbb{R} \cup \{+\infty\}$ be a function with* $\operatorname{dom} E \neq \emptyset$.
(i) *E is $L^\natural$-convex if and only if for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}^{\mathcal{V}}$ with $\{u \in \mathcal{V} \mid x_u > y_u\} \neq \emptyset$, we have*

$$E(\boldsymbol{x}) + E(\boldsymbol{y}) \geq E(\boldsymbol{x} - \chi_W) + E(\boldsymbol{y} + \chi_W),$$

where $W = \arg\max\{x_u - y_u \mid u \in \mathcal{V}\}$.
(ii) *E is $L^\natural$-convex if and only if for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}^{\mathcal{V}}$ and $\lambda \in \mathbb{Z}_+$, we have*

$$E(\boldsymbol{x}) + E(\boldsymbol{y}) \geq E((\boldsymbol{x} - \lambda\mathbf{1}) \vee \boldsymbol{y}) + E(\boldsymbol{x} \wedge (\boldsymbol{y} + \lambda\mathbf{1})).$$

In particular, an $L^\natural$-convex function E satisfies the submodular inequality $E(\boldsymbol{x}) + E(\boldsymbol{y}) \geq E(\boldsymbol{x} \wedge \boldsymbol{y}) + E(\boldsymbol{x} \vee \boldsymbol{y})$ $(\forall \boldsymbol{x}, \boldsymbol{y} \in \operatorname{dom} E)$.

We denote by $\arg\min E$ the set of minimizers of a function $E : \mathbb{Z}^{\mathcal{V}} \to \mathbb{R} \cup \{+\infty\}$, i.e.,

$$\arg\min E = \{\boldsymbol{x} \in \operatorname{dom} E \mid E(\boldsymbol{x}) \leq E(\boldsymbol{y}) \ (\forall \boldsymbol{y} \in \mathbb{Z}^{\mathcal{V}})\}.$$

Minimizers of an $\mathrm{L}^\natural$-convex function can be characterized by local optimality.

Theorem 2.5 ([30, Th. 7.14]). *Let $E : \mathbb{Z}^{\mathcal{V}} \to \mathbb{R} \cup \{+\infty\}$ be an $L^\natural$-convex function and $\boldsymbol{x} \in \operatorname{dom} E$. Then, $\boldsymbol{x} \in \arg\min E$ if and only if $E(\boldsymbol{x}) \leq E(\boldsymbol{x} + \chi_X)$ for all $X \subseteq \mathcal{V}$ and $E(\boldsymbol{x}) \leq E(\boldsymbol{x} - \chi_X)$ for all $X \subseteq \mathcal{V}$.*

2.2. Analysis of Primal Algorithm

As shown in Proposition 2.3 (i), the problem (CTP) is a special case of the minimization of an $\mathrm{L}^\natural$-convex function. In the rest of this section, we mainly consider the minimization of an $\mathrm{L}^\natural$-convex function $E : \mathbb{Z}^{\mathcal{V}} \to \mathbb{R} \cup \{+\infty\}$, and show that the primal algorithm finds an optimal solution in $\mathrm{O}(K_\infty)$ iterations, where

$$K_\infty = \max\{||\boldsymbol{x} - \boldsymbol{y}||_\infty \mid \boldsymbol{x}, \boldsymbol{y} \in \operatorname{dom} E\}.$$

For any vector $\boldsymbol{x} \in \operatorname{dom} E$, the vector $\boldsymbol{x}^+$ denotes the unique minimal vector in $\arg\min\{E(\boldsymbol{z}) \mid \boldsymbol{z} \geq \boldsymbol{x}\}$ and $\boldsymbol{x}^-$ denotes the unique maximal vector in $\arg\min\{E(\boldsymbol{z}) \mid \boldsymbol{z} \leq \boldsymbol{x}\}$. The existence of such $\boldsymbol{x}^+$ and $\boldsymbol{x}^-$ follows from the submodularity of function E (see Theorem 2.4 (ii)). To analyze the number of iterations, we define values $\rho^+(\boldsymbol{x})$ and $\rho^-(\boldsymbol{x})$ for a vector $\boldsymbol{x} \in \operatorname{dom} E$ as

$$\rho^+(\boldsymbol{x}) = ||\boldsymbol{x}^+ - \boldsymbol{x}||_\infty, \qquad \rho^-(\boldsymbol{x}) = ||\boldsymbol{x}^- - \boldsymbol{x}||_\infty.$$

The following optimality condition follows immediately from Theorem 2.5.

Lemma 2.6. *For $\boldsymbol{x} \in \operatorname{dom} E$, if $\rho^+(\boldsymbol{x}) = \rho^-(\boldsymbol{x}) = 0$ then $\boldsymbol{x} \in \arg\min E$.*

Note that $\rho^+(\boldsymbol{x}) = 0$ (resp. $\rho^-(\boldsymbol{x}) = 0$) alone implies $\boldsymbol{x} \in \arg\min\{E(\boldsymbol{z}) \mid \boldsymbol{z} \geq \boldsymbol{x}\}$ (resp. $\boldsymbol{x} \in \arg\min\{E(\boldsymbol{z}) \mid \boldsymbol{z} \leq \boldsymbol{x}\}$), but does not imply $\boldsymbol{x} \in \arg\min E$ in general.

Each iteration of the primal algorithm increases neither of $\rho^+(\boldsymbol{x})$ nor $\rho^-(\boldsymbol{x})$ and decreases strictly at least one of $\rho^+(\boldsymbol{x})$ and $\rho^-(\boldsymbol{x})$. The proof is given at the end of this section.

Lemma 2.7. *In each iteration of the primal algorithm, we have the following:*
(i) *If $\rho^+(\boldsymbol{x}) > 0$, then $\rho^+(\boldsymbol{x} + \chi_{X^+}) = \rho^+(\boldsymbol{x}) - 1$ and $\rho^-(\boldsymbol{x} + \chi_{X^+}) \leq \rho^-(\boldsymbol{x})$.*
(ii) *If $\rho^-(\boldsymbol{x}) > 0$, then $\rho^-(\boldsymbol{x} - \chi_{X^-}) = \rho^-(\boldsymbol{x}) - 1$ and $\rho^+(\boldsymbol{x} - \chi_{X^-}) \leq \rho^+(\boldsymbol{x})$.*

Theorem 2.8.
(i) *The output $\boldsymbol{x}$ of the primal algorithm satisfies $\boldsymbol{x} \in \arg\min E$.*
(ii) *The number of iterations of the primal algorithm is bounded by $\rho^+(\boldsymbol{x}^\circ) + \rho^-(\boldsymbol{x}^\circ) + 2$, which is further bounded by $2K_\infty + 2$.*

Proof. The claim (ii) is immediate from Lemma 2.7 and the fact that $\rho^+(\boldsymbol{x}) \leq K_\infty$ and $\rho^-(\boldsymbol{x}) \leq K_\infty$ for any $\boldsymbol{x} \in \operatorname{dom} E$. We then prove (i). We see from Lemma 2.7 that $\rho^+(\boldsymbol{x}) = \rho^-(\boldsymbol{x}) = 0$ holds at the end of the algorithm. Therefore, the claim (i) follows from Lemma 2.6. □

In each iteration of the primal algorithm, we need to compute $\min\{E(\boldsymbol{x} + \chi_X) \mid X \subseteq \mathcal{V}\}$ or $\min\{E(\boldsymbol{x} - \chi_X) \mid X \subseteq \mathcal{V}\}$, which can be reduced to the submodular set function minimization (cf. Theorem 2.4 (ii)). Hence, the running time of the primal algorithm is given as follows, where $T_{\mathrm{sfm}}(n)$ denotes the time complexity for solving the minimization of a submodular set function $f : 2^{\mathcal{V}} \to \mathbb{R}$ with $|\mathcal{V}| = n$. Currently, we have $T_{\mathrm{sfm}}(n) = O(n^6)$ by Orlin's algorithm [32].

Corollary 2.9. *The primal algorithm finds a minimizer of an $L^\natural$-convex function $E : \mathbb{Z}^{\mathcal{V}} \to \mathbb{R} \cup \{+\infty\}$ in $\mathrm{O}(K_\infty \cdot T_{\mathrm{sfm}}(n))$ time.*

We also note that if the function E is given as the objective function of (CTP), then the integer K given by (1.1) satisfies $K \geq K_\infty$. Hence, Theorem 2.1 follows immediately from Theorem 2.8.

We now prove Lemma 2.7, where the following property is useful.

Lemma 2.10. *Let $\boldsymbol{x}, \boldsymbol{y} \in \operatorname{dom} E$.*
(i) *Suppose that $\boldsymbol{x} \leq \boldsymbol{y}$ and $E(\boldsymbol{y}) = \min\{E(\boldsymbol{z}) \mid \boldsymbol{x} \leq \boldsymbol{z} \leq \boldsymbol{y}\}$. Then, $\rho^+(\boldsymbol{y}) \leq \rho^+(\boldsymbol{x})$ and $\rho^-(\boldsymbol{y}) \leq \rho^-(\boldsymbol{x})$. In particular, we have $\boldsymbol{y}^+ = \boldsymbol{x}^+ \vee \boldsymbol{y}$.*
(ii) *Suppose that $\boldsymbol{x} \geq \boldsymbol{y}$ and $E(\boldsymbol{y}) = \min\{E(\boldsymbol{z}) \mid \boldsymbol{x} \geq \boldsymbol{z} \geq \boldsymbol{y}\}$. Then, $\rho^+(\boldsymbol{y}) \leq \rho^+(\boldsymbol{x})$ and $\rho^-(\boldsymbol{y}) \leq \rho^-(\boldsymbol{x})$. In particular, we have $\boldsymbol{y}^- = \boldsymbol{x}^- \wedge \boldsymbol{y}$.*

Proof. We prove (i) only. We note that $\boldsymbol{y}^+ = \boldsymbol{x}^+ \vee \boldsymbol{y}$ implies $\rho^+(\boldsymbol{y}) \leq \rho^+(\boldsymbol{x})$ since

$$\rho^+(\boldsymbol{y}) = ||\boldsymbol{y}^+ - \boldsymbol{y}||_\infty = ||(\boldsymbol{x}^+ \vee \boldsymbol{y}) - \boldsymbol{y}||_\infty \leq ||\boldsymbol{x}^+ - \boldsymbol{x}||_\infty = \rho^+(\boldsymbol{x}).$$

[Proof of "$\boldsymbol{y}^+ = \boldsymbol{x}^+ \vee \boldsymbol{y}$"] By Theorem 2.4 (ii), we have

$$E(\boldsymbol{x}^+) + E(\boldsymbol{y}) \geq E(\boldsymbol{x}^+ \vee \boldsymbol{y}) + E(\boldsymbol{x}^+ \wedge \boldsymbol{y}). \tag{2.2}$$

Since $\boldsymbol{x} \leq \boldsymbol{x}^+ \wedge \boldsymbol{y} \leq \boldsymbol{y}$, we have $E(\boldsymbol{y}) \leq E(\boldsymbol{x}^+ \wedge \boldsymbol{y})$, which, together with (2.2), implies

$$E(\boldsymbol{x}^+) \geq E(\boldsymbol{x}^+ \vee \boldsymbol{y}). \tag{2.3}$$

Since $\boldsymbol{y}^+ \geq \boldsymbol{y} \geq \boldsymbol{x}$ and $\boldsymbol{x}^+ \in \arg\min\{E(\boldsymbol{z}) \mid \boldsymbol{z} \geq \boldsymbol{x}\}$, we have

$$E(\boldsymbol{y}^+) \geq E(\boldsymbol{x}^+). \tag{2.4}$$

Similarly, since $\boldsymbol{x}^+ \vee \boldsymbol{y} \geq \boldsymbol{y}$ and $\boldsymbol{y}^+ \in \arg\min\{E(\boldsymbol{z}) \mid \boldsymbol{z} \geq \boldsymbol{y}\}$, we have

$$E(\boldsymbol{x}^+ \vee \boldsymbol{y}) \geq E(\boldsymbol{y}^+). \tag{2.5}$$

It follows from (2.3), (2.4), and (2.5) that $E(\boldsymbol{x}^+) = E(\boldsymbol{y}^+) = E(\boldsymbol{x}^+ \vee \boldsymbol{y})$, which implies

$$\boldsymbol{y}^+ \in \arg\min\{E(\boldsymbol{z}) \mid \boldsymbol{z} \geq \boldsymbol{x}\}, \qquad \boldsymbol{x}^+ \vee \boldsymbol{y} \in \arg\min\{E(\boldsymbol{z}) \mid \boldsymbol{z} \geq \boldsymbol{y}\}.$$

It follows from the choices of $\boldsymbol{x}^+$ and $\boldsymbol{y}^+$ that $\boldsymbol{x}^+ \leq \boldsymbol{y}^+$ and $\boldsymbol{y}^+ \leq \boldsymbol{x}^+ \vee \boldsymbol{y}$. These inequalities and $\boldsymbol{y} \leq \boldsymbol{y}^+$ imply $\boldsymbol{y}^+ = \boldsymbol{x}^+ \vee \boldsymbol{y}$.

[Proof of "$\rho^-(\boldsymbol{y}) \leq \rho^-(\boldsymbol{x})$"] We first show that $\boldsymbol{x}^- \leq \boldsymbol{y}^-$. By Theorem 2.4 (ii), we have

$$E(\boldsymbol{x}^-) + E(\boldsymbol{y}^-) \geq E(\boldsymbol{x}^- \vee \boldsymbol{y}^-) + E(\boldsymbol{x}^- \wedge \boldsymbol{y}^-). \tag{2.6}$$

Since $\boldsymbol{x}^- \in \arg\min\{E(\boldsymbol{z}) \mid \boldsymbol{z} \leq \boldsymbol{x}\}$ and $\boldsymbol{x}^- \wedge \boldsymbol{y}^- \leq \boldsymbol{x}^- \leq \boldsymbol{x}$, we have $E(\boldsymbol{x}^-) \leq E(\boldsymbol{x}^- \wedge \boldsymbol{y}^-)$, which, together with (2.6), implies $E(\boldsymbol{y}^-) \geq E(\boldsymbol{x}^- \vee \boldsymbol{y}^-)$. Since $\boldsymbol{y}^- \leq \boldsymbol{x}^- \vee \boldsymbol{y}^- \leq \boldsymbol{y}$ and $\boldsymbol{y}^-$ is the maximal vector in $\arg\min\{E(\boldsymbol{z}) \mid \boldsymbol{z} \leq \boldsymbol{y}\}$, we have $\boldsymbol{y}^- = \boldsymbol{x}^- \vee \boldsymbol{y}^-$, i.e., $\boldsymbol{x}^- \leq \boldsymbol{y}^-$.

We may assume that $\rho^-(\boldsymbol{y}) > 0$ since otherwise the inequality holds immediately. Put $\lambda = \rho^-(\boldsymbol{y})$ and $W = \{u \in \mathcal{V} \mid y_u - (y^-)_u = \lambda\}$. By Theorem 2.4 (i), we have

$$E(\boldsymbol{y}) + E(\boldsymbol{y}^-) \geq E(\boldsymbol{y} - \chi_W) + E(\boldsymbol{y}^- + \chi_W).$$

We also have $E(\boldsymbol{y}^-) < E(\boldsymbol{y}^- + \chi_W)$ by the definition of $\boldsymbol{y}^-$ and the inequality $\boldsymbol{y}^- + \chi_W \leq \boldsymbol{y}$. Hence, it holds that $E(\boldsymbol{y} - \chi_W) < E(\boldsymbol{y})$. Since $E(\boldsymbol{y}) = \min\{E(\boldsymbol{z}) \mid \boldsymbol{x} \leq \boldsymbol{z} \leq \boldsymbol{y}\}$, if $\boldsymbol{y} - \chi_W \geq \boldsymbol{x}$ then we have $E(\boldsymbol{y} - \chi_W) \geq E(\boldsymbol{y})$, a contradiction. Hence, there exists some $u \in \mathcal{V}$ such that $x_u = y_u$ and $u \in W$. This implies that $x_u = y_u = (y^-)_u + \lambda \geq (x^-)_u + \lambda$. Therefore, $\rho^-(\boldsymbol{x}) \geq x_u - (x^-)_u \geq \lambda = \rho^-(\boldsymbol{y})$. □

Input: initial feasible solution $\boldsymbol{x} := \boldsymbol{x}^\circ \in \operatorname{dom} E$.
Step 1: Set $\alpha = 2^{\lceil \log_2(K_\infty/2n) \rceil}$.
Step 2: Find an integer vector $\boldsymbol{y}$ that minimizes $E(\boldsymbol{x} + \alpha\boldsymbol{y})$ and set $\boldsymbol{x} := \boldsymbol{x} + \alpha\boldsymbol{y}$.
Step 3: If $\alpha = 1$, then stop ($\boldsymbol{x}$ is a minimizer of E).
Step 4: Set $\alpha := \alpha/2$ and go to Step 1.

Figure 2: Murota's scaling algorithm

Proof of Lemma 2.7. We prove (i) only; the claim (ii) can be shown in the same way.

Put $\boldsymbol{y} = \boldsymbol{x} + \chi_{X^+}$. Then, we have $E(\boldsymbol{y}) = \min\{E(\boldsymbol{z}) \mid \boldsymbol{x} \le \boldsymbol{z} \le \boldsymbol{y}\}$, which implies $\rho^+(\boldsymbol{y}) \le \rho^+(\boldsymbol{x})$ and $\rho^-(\boldsymbol{y}) \le \rho^-(\boldsymbol{x})$ by Lemma 2.10. To prove $\rho^+(\boldsymbol{y}) = \rho^+(\boldsymbol{x}) - 1$, it suffices to show that

$$S \subseteq X^+, \quad \text{where } S = \arg\max\{(x^+)_u - x_u \mid u \in \mathcal{V}\}.$$

If $S \subseteq X^+$, then Lemma 2.10 implies the desired equation as follows:

$$\rho^+(\boldsymbol{y}) = ||\boldsymbol{y}^+ - \boldsymbol{y}||_\infty = ||(\boldsymbol{x}^+ \vee \boldsymbol{y}) - \boldsymbol{y}||_\infty = ||\boldsymbol{x}^+ - \boldsymbol{x}||_\infty - 1 = \rho^+(\boldsymbol{x}) - 1.$$

Assume, to the contrary, that $S \setminus X^+ \neq \emptyset$. Put

$$S' = \arg\max\{(x^+)_u - x_u - (\chi_{X^+})_u \mid u \in \mathcal{V}\} = S \setminus X^+.$$

Theorem 2.4 (i) implies

$$E(\boldsymbol{x}^+) + E(\boldsymbol{x} + \chi_{X^+}) \ge E(\boldsymbol{x}^+ - \chi_{S'}) + E(\boldsymbol{x} + \chi_{X^+} + \chi_{S'}).$$

Since $\chi_{X^+} + \chi_{S'} = \chi_{X^+ \cup S}$, we have $E(\boldsymbol{x} + \chi_{X^+} + \chi_{S'}) = E(\boldsymbol{x} + \chi_{X^+ \cup S}) \ge E(\boldsymbol{x} + \chi_{X^+})$, where the inequality is by the choice of X^+. Hence, we have $E(\boldsymbol{x}^+) \ge E(\boldsymbol{x}^+ - \chi_{S'})$, a contradiction to the fact that $\boldsymbol{x}^+$ is the minimal vector in $\arg\min\{E(\boldsymbol{z}) \mid \boldsymbol{z} \ge \boldsymbol{x}\}$ since $\boldsymbol{x}^+ - \chi_{S'} \ge \boldsymbol{x}$. □

2.3. Application of Scaling Technique

Scaling is one of common techniques in obtaining a polynomial-time algorithm from a pseudo-polynomial-time algorithm. As shown in the previous section, our primal algorithm is a pseudo-polynomial-time algorithm, i.e., the number of iterations depends on K, not on $\log K$. In this section, we consider an application of scaling technique to our primal algorithm.

A scaling framework for L$^\natural$-convex function minimization is already proposed by Murota [31] (see also [30, Sec. 10.3.2]), which is shown in Fig. 2. We can use either of our primal algorithm and Murota's steepest descent algorithm to find a vector $\boldsymbol{y}$ in Step 2 since the function $E_\alpha : \mathbb{Z}^{\mathcal{V}} \to \mathbb{R} \cup \{+\infty\}$ defined by

$$E_\alpha(\boldsymbol{y}) = E(\boldsymbol{x} + \alpha\boldsymbol{y}) \qquad (\boldsymbol{y} \in \mathbb{Z}^{\mathcal{V}})$$

is also an L$^\natural$-convex function. By a proximity theorem for L$^\natural$-convex functions [30, Theorem 7.18], there exists a minimizer $\boldsymbol{y}$ of $E(\boldsymbol{x} + \alpha\boldsymbol{y})$ such that $-n \le y_u \le n$ ($u \in \mathcal{V}$). Hence, Corollary 2.9 implies that Step 2 can be done in $O(n \cdot T_{\mathrm{sfm}}(n))$ time. Since the number of scaling phases is $O(\log(K_\infty/2n))$, we have the following:

Theorem 2.11. *The scaling algorithm combined with our primal algorithm or Murota's steepest descent algorithm finds a minimizer of an $L^\natural$-convex function $E : \mathbb{Z}^{\mathcal{V}} \to \mathbb{R} \cup \{+\infty\}$ in $\mathrm{O}(n \log(K_\infty/2n) \cdot T_{\mathrm{sfm}}(n))$ time.*

The bound shown in Theorem 2.11 improves the previous bound $\mathrm{O}(n^2 \log(K_\infty/2n) \cdot T_{\mathrm{sfm}}(n))$ in [31] by a factor of n.

3. Primal-Dual Algorithm

In this section, we explain our primal-dual algorithm, which is an improved version of the primal algorithm by making good use of dual variables. For this purpose, we first review the convex cost flow problem, which is the dual of the problem (CTP), in Section 3.1. Based on this, we then present our primal-dual algorithm in Section 3.2. The primal-dual algorithm also uses procedures UP and DOWN; however, during these procedures the algorithm updates not only primal variables $\boldsymbol{x}$ but also dual variables called flow. We show the validity of the algorithm and analyze the time complexity in Section 3.3.

3.1. Convex Cost Flow Problem

It is well known that the convex cost flow problem can be obtained as the dual of the problem (CTP) in the following way (see, e.g., [1], [20, Ch. IV], [33, Sec. 8]). Let $(\mathcal{V}, \mathcal{A})$ be a directed graph corresponding to the undirected graph $(\mathcal{V}, \mathcal{E})$, where $\mathcal{A} = \{(u, v) \mid (u, v) \in \mathcal{E}\} \cup \{(v, u) \mid (u, v) \in \mathcal{E}\}$. A flow is a vector $f \in \mathbb{R}^{\mathcal{V} \cup \mathcal{A}}$ satisfying

$$\begin{array}{lcll} f_{uv} & = & -f_{vu} & ((u,v) \in \mathcal{E}) \quad \text{(antisymmetry)}, \\ f_u & = & \sum_{(u,v) \in \mathcal{E}} f_{uv} & (u \in \mathcal{V}) \quad \text{(flow conservation)}. \end{array}$$

Given a flow f, we define a function $E^f(\boldsymbol{x}) : \mathbb{Z}^{\mathcal{V}} \to \mathbb{R} \cup \{+\infty\}$ as follows:

$$E^f(\boldsymbol{x}) = \sum_{u \in \mathcal{V}} D_u[-f_u](x_u) + \sum_{(u,v) \in \mathcal{E}} V_{uv}[-f_{uv}](x_v - x_u), \tag{3.1}$$

where for $\beta \in \mathbb{R}$ the functions $D_u[\beta]$ and $V_{uv}[\beta]$ are respectively defined by

$$D_u[\beta](\alpha) = D_u(\alpha) + \beta\alpha, \qquad V_{uv}[\beta](\alpha) = V_{uv}(\alpha) + \beta\alpha \qquad (\alpha \in \mathbb{Z}).$$

It is not difficult to check that for any flow f we have $E^f(\boldsymbol{x}) = E(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathbb{Z}^{\mathcal{V}}$. Furthermore, the functions $D_u[\beta]$ and $V_{uv}[\beta]$ are convex.

For a flow f, let us define a function $H : \mathbb{R}^{\mathcal{V} \cup \mathcal{A}} \to \mathbb{R} \cup \{-\infty\}$ as

$$H(f) = \sum_{u \in \mathcal{V}} \min_{\alpha \in \mathbb{Z}} D_u[-f_u](\alpha) + \sum_{(u,v) \in \mathcal{E}} \min_{\alpha \in \mathbb{Z}} V_{uv}[-f_{uv}](\alpha).$$

We note that $\min_{\alpha \in \mathbb{Z}} D_u[-f_u](\alpha)$ (resp., $\min_{\alpha \in \mathbb{Z}} V_{uv}[-f_{uv}](\alpha)$) is a concave function in variable f_u (resp., in variable f_{uv}).

We now consider the following convex cost flow problem, which is the dual of (CTP):

$$\text{(CFP):} \quad \text{Maximize} \quad H(f) \quad \text{subject to} \quad f \in \mathbb{R}^{\mathcal{V} \cup \mathcal{A}},\ f \text{ is a flow.}$$

Input: initial feasible solution $\boldsymbol{x} := \boldsymbol{x}^\circ \in \operatorname{dom} E$.

1. INITIALIZE-FLOW (updates f)
2. Set SuccessUp := false, SuccessDown := false.
3. Do UP or DOWN in any order until SuccessUp = SuccessDown = true:
   UP (do only if SuccessUp is false):
   - MAXFLOW-UP (updates $\boldsymbol{x}$ and f)
   - If $\mathcal{V}^+(\boldsymbol{x}, f) = \emptyset$, set SuccessUp := true
   - Optional: DIJKSTRA-UP (updates $\boldsymbol{x}$)

   DOWN (do only if SuccessDown is false):
   - MAXFLOW-DOWN (updates $\boldsymbol{x}$ and f)
   - If $\mathcal{V}^-(\boldsymbol{x}, f) = \emptyset$, set SuccessDown := true
   - Optional: DIJKSTRA-DOWN (updates $\boldsymbol{x}$)
4. Optional: DIJKSTRA-DOWN; set $\boldsymbol{x}^{\min} := \boldsymbol{x}$.
5. Optional: DIJKSTRA-UP; set $\boldsymbol{x}^{\max} := \boldsymbol{x}$.

Figure 3: Our primal-dual algorithm. Upon termination $\boldsymbol{x}$ is a minimizer of E, $\boldsymbol{x}^{\min}$ is a unique minimal minimizer, $\boldsymbol{x}^{\max}$ is a unique maximal minimizer, and f is an optimal flow.

Clearly, $H(f)$ is a lower bound of the function value $E^f(\boldsymbol{x}) = E(\boldsymbol{x})$, i.e., $H(f) \le E(\boldsymbol{x})$ holds for any flow f and any feasible solution $\boldsymbol{x} \in \operatorname{dom} E$. It turns out that strong duality holds as well.

Theorem 3.1 (cf. [1, 20, 33]).
(a) [strong duality] $H(f^*) = E(\boldsymbol{x}^*)$ *holds for any optimal solution* $f^* \in \mathbb{R}^{\mathcal{V}\cup\mathcal{A}}$ *of* (CFP) *and any optimal solution* $\boldsymbol{x}^* \in \operatorname{dom} E$ *of* (CTP).
(b) [optimality condition] *A flow* $f \in \mathbb{R}^{\mathcal{V}\cup\mathcal{A}}$ *and a feasible solution* $\boldsymbol{x} \in \operatorname{dom} E$ *are optimal solutions of the problems* (CFP) *and* (CTP)*, respectively, if and only if the following conditions hold:*

$$D_u[-f_u](x_u) = \min_{\alpha\in\mathbb{Z}} D_u[-f_u](\alpha) \qquad (\forall u \in \mathcal{V}), \tag{3.2a}$$

$$V_{uv}[-f_{uv}](x_v - x_u) = \min_{\alpha\in\mathbb{Z}} V_{uv}[-f_{uv}](\alpha) \qquad (\forall (u,v) \in \mathcal{E}). \tag{3.2b}$$

We will prove that our algorithm finds a pair $(\boldsymbol{x}, f)$ satisfying the optimality condition (3.2). Since the functions $D_u[-f_u](\cdot)$ and $V_{uv}[-f_{uv}](\cdot)$ are convex, these conditions are equivalent to

$$\Delta^- D_u[-f_u](x_u) \le 0 \le \Delta^+ D_u[-f_u](x_u),$$
$$\Delta^- V_{uv}[-f_{uv}](x_v - x_u) \le 0 \le \Delta^+ V_{uv}[-f_{uv}](x_v - x_u),$$

where we use the following notation for a function $g : \mathbb{Z} \to \mathbb{R} \cup \{+\infty\}$:

$$\Delta^- g(\alpha) = g(\alpha) - g(\alpha - 1), \quad \Delta^+ g(\alpha) = g(\alpha + 1) - g(\alpha) \qquad (\alpha \in \mathbb{Z}).$$

3.2. Algorithm

We are now ready to present our primal-dual algorithm. It maintains a feasible solution $\boldsymbol{x} \in \operatorname{dom} E$ and a flow $f \in \mathbb{R}^{\mathcal{V}\cup\mathcal{A}}$ satisfying the condition (3.2b), and iteratively

updates $\boldsymbol{x}$ and f so that the condition (3.2a) is satisfied. It is convenient to use the following notation for sets of nodes violating the condition (3.2a):

$$\mathcal{V}^+(\boldsymbol{x}, f) = \{u \in \mathcal{V} \mid \Delta^+ D_u[-f_u](x_u) < 0\}, \quad \mathcal{V}^-(\boldsymbol{x}, f) = \{u \in \mathcal{V} \mid \Delta^- D_u[-f_u](x_u) > 0\}.$$

Note that since the function $D_u[-f_u](\cdot)$ is convex, we have $\mathcal{V}^+(\boldsymbol{x}, f) \cap \mathcal{V}^-(\boldsymbol{x}, f) = \emptyset$. Furthermore, the condition (3.2a) holds if and only if $\mathcal{V}^+(\boldsymbol{x}, f) = \mathcal{V}^-(\boldsymbol{x}, f) = \emptyset$.

The outline of the algorithm is shown in Fig. 3. We now give details of each procedure.

INITIALIZE-FLOW. Its goal is to set flow f so that the condition (3.2b) is satisfied for every edge $(u, v) \in \mathcal{E}$. Since $\Delta^{\pm} V_{uv}[-f_{uv}](x_v - x_u) = \Delta^{\pm} V_{uv}(x_v - x_u) - f_{uv}$, a necessary and sufficient condition for flow f_{uv} is

$$\Delta^- V_{uv}(x_v - x_u) \leq f_{uv} \leq \Delta^+ V_{uv}(x_v - x_u).$$

After setting f_{uv}, values f_u are computed from the flow conservation constraint.

MAXFLOW-UP. This operation is similar to procedure UP of the primal algorithm, except that it modifies not only feasible solution $\boldsymbol{x}$ but also flow f. We will show later that it tries to move towards satisfying (3.2a). In particular, sets $\mathcal{V}^+(\boldsymbol{x}, f)$ and $\mathcal{V}^-(\boldsymbol{x}, f)$ will not grow. The procedure can be summarized as follows.

First, we construct a directed graph $\hat{\mathcal{G}} = (\hat{\mathcal{V}}, \hat{\mathcal{A}})$ such that

$$\begin{aligned} \hat{\mathcal{V}} &= \mathcal{V} \cup \{s, t\}, \\ \hat{\mathcal{A}} &= \mathcal{A} \cup \{(s, u), (u, s) \mid u \in \mathcal{V} \setminus \mathcal{V}^+(\boldsymbol{x}, f)\} \cup \{(u, t), (t, u) \mid u \in \mathcal{V}^+(\boldsymbol{x}, f)\}, \end{aligned}$$

where s and t are called the source and the sink, respectively. We also consider a capacity $\hat{c}_{uv}$ for $(u, v) \in \hat{\mathcal{A}}$ defined by

$$\begin{array}{lll} \hat{c}_{uv} = \Delta^+ V_{uv}[-f_{uv}](x_v - x_u), & \hat{c}_{vu} = -\Delta^- V_{uv}[-f_{uv}](x_v - x_u) & \text{for } (u, v) \in \mathcal{E}, \\ \hat{c}_{su} = \Delta^+ D_u[-f_u](x_u), & \hat{c}_{us} = 0 & \text{for } u \in \mathcal{V} \setminus \mathcal{V}^+(\boldsymbol{x}, f), \\ \hat{c}_{ut} = -\Delta^+ D_u[-f_u](x_u), & \hat{c}_{tu} = 0 & \text{for } u \in \mathcal{V}^+(\boldsymbol{x}, f). \end{array}$$

We note that all of the capacities are nonnegative by the condition (3.2b) and the definition of $\mathcal{V}^+(\boldsymbol{x}, f)$. We then solve the following maximum flow problem:

$$\begin{array}{lll} \text{Maximize} & \displaystyle\sum_{(s,u) \in \hat{\mathcal{A}}} \hat{f}_{su} & \\ \text{subject to} & \hat{f}_{uv} \leq \hat{c}_{uv} & (\forall (u, v) \in \hat{\mathcal{A}}), \\ & \hat{f}_{uv} = -\hat{f}_{vu} & (\forall (u, v) \in \hat{\mathcal{A}}), \\ & \displaystyle\sum_{u:(u,v) \in \hat{\mathcal{A}}} \hat{f}_{uv} = 0 & (\forall v \in \hat{\mathcal{V}} \setminus \{s, t\}), \\ & \hat{f} \in \mathbb{R}^{\hat{\mathcal{A}}}. & \end{array}$$

Finally, we update the flow f by using $\hat{f}$ as follows:

$$\begin{array}{rcll} f_{uv} & := & f_{uv} + \hat{f}_{uv} & \text{for } (u, v) \in \mathcal{A}, \\ f_u & := & f_u + \hat{f}_{su} & \text{for } (s, u) \in \hat{\mathcal{A}}, \\ f_u & := & f_u - \hat{f}_{ut} & \text{for } (u, t) \in \hat{\mathcal{A}}. \end{array}$$

It is easy to see that the output flow f satisfies antisymmetry and flow conservation constraints, since the same holds for flow $\hat{f}$.

A new feasible solution $\boldsymbol{y}$ is computed from a minimum s-t cut of the graph $\hat{\mathcal{G}}$. An s-t cut (S,T) of the graph $\hat{\mathcal{G}}$ is a pair of subsets of $\hat{\mathcal{V}}$ such that $\{S,T\}$ is a partition of $\hat{\mathcal{V}}$ and $s \in S$, $t \in T$. We denote by $\mathrm{cap}(S,T)$ the capacity of an s-t cut (S,T), i.e.,

$$\mathrm{cap}(S,T) = \sum\{\hat{c}_{uv} \mid (u,v) \in \hat{\mathcal{A}},\ u \in S,\ v \in T\}.$$

A minimum s-t cut is an s-t cut (S,T) minimizing the capacity $\mathrm{cap}(S,T)$. If we obtain a minimum s-t cut (S,T), then we set $\boldsymbol{y} = \boldsymbol{x} + \chi_{X^+}$, where $X^+ = T \setminus \{t\}$. As we will show later, the output feasible solution $\boldsymbol{y}$ is the same as that of procedure UP in the primal algorithm, i.e., $X^+ \in \arg\min\{E(\boldsymbol{x} + \chi_X) \mid X \subseteq \mathcal{V}\}$.

MAXFLOW-DOWN. This operation is the same as MAXFLOW-UP, except for the definition of the arc set $\hat{\mathcal{A}}$ and the update of $\boldsymbol{x}$. The arc set $\hat{\mathcal{A}}$ is given by

$$\hat{\mathcal{A}} = \mathcal{A} \cup \{(s,u),(u,s) \mid u \in \mathcal{V}^-(\boldsymbol{x},f)\} \cup \{(u,t),(t,u) \mid u \in \mathcal{V} \setminus \mathcal{V}^-(\boldsymbol{x},f)\},$$

Capacities $\hat{c}_{uv}$ are defined by

$$\begin{array}{lll}
\hat{c}_{uv} = \Delta^+ V_{uv}[-f_{uv}](x_v - x_u), & \hat{c}_{vu} = -\Delta^- V_{uv}[-f_{uv}](x_v - x_u) & \text{for } (u,v) \in \mathcal{E}, \\
\hat{c}_{su} = \Delta^- D_u[-f_u](x_u), & \hat{c}_{us} = 0 & \text{for } u \in \mathcal{V}^-(\boldsymbol{x},f), \\
\hat{c}_{ut} = -\Delta^- D_u[-f_u](x_u), & \hat{c}_{tu} = 0 & \text{for } u \in \mathcal{V} \setminus \mathcal{V}^-(\boldsymbol{x},f).
\end{array}$$

A new feasible solution $\boldsymbol{y}$ is computed from a minimum s-t cut (S,T) of the graph $\hat{\mathcal{G}}$ by $\boldsymbol{y} = \boldsymbol{x} - \chi_{X^-}$, where $X^- = S \setminus \{s\}$.

DIJKSTRA-UP. This operation is optional. It does not affect the worst-case complexity of the algorithm, but may improve empirical performance. In this procedure we fix flow and compute a maximal feasible solution $\boldsymbol{y} \geq \boldsymbol{x}$ such that functions $D_u[-f_u](\cdot)$ are non-increasing on $[x_u, y_u]$ and the condition (3.2b) holds. If we denote $d_u = y_u - x_u \geq 0$, then these constraints are equivalent to

$$\begin{array}{rcl}
d_u & \leq & d_u^{\max} = \max\{d \in \mathbb{Z}_+ \mid D_u[-f_u](x_u + d) \leq \cdots \leq D_u[-f_u](x_u + 1) \leq D_u[-f_u](x_u)\}, \\
d_v - d_u & \leq & d_{uv}^{\max} = \max\{d \in \mathbb{Z}_+ \mid V_{uv}[-f_{uv}](x_v - x_u + d) = V_{uv}[-f_{uv}](x_v - x_u)\}, \\
d_u - d_v & \leq & d_{vu}^{\max} = \max\{d \in \mathbb{Z}_+ \mid V_{uv}[-f_{uv}](x_v - x_u - d) = V_{uv}[-f_{uv}](x_v - x_u)\}.
\end{array}$$

It is well known (see, e.g., [3]) that finding a maximal vector $\boldsymbol{d}$ satisfying these constraints can be reduced to a single-source shortest path problem, and therefore such a vector $\boldsymbol{d}$ can be computed efficiently by using Dijkstra's algorithm.

DIJKSTRA-DOWN. This operation is similar to the previous one; we compute a minimal feasible solution $\boldsymbol{y} \leq \boldsymbol{x}$ such that functions $D_u[-f_u](\cdot)$ are non-decreasing on $[y_u, x_u]$ and the condition (3.2b) holds. If we denote $d_u = x_u - y_u \geq 0$, then these constraints are equivalent to

$$\begin{array}{rcl}
d_u & \leq & d_u^{\max} = \max\{d \in \mathbb{Z}_+ \mid D_u[-f_u](x_u - d) \leq \cdots \leq D_u[-f_u](x_u - 1) \leq D_u[-f_u](x_u)\}, \\
d_v - d_u & \leq & d_{uv}^{\max} = \max\{d \in \mathbb{Z}_+ \mid V_{uv}[-f_{uv}](x_v - x_u - d) = V_{uv}[-f_{uv}](x_v - x_u)\}, \\
d_u - d_v & \leq & d_{vu}^{\max} = \max\{d \in \mathbb{Z}_+ \mid V_{uv}[-f_{uv}](x_v - x_u + d) = V_{uv}[-f_{uv}](x_v - x_u)\}.
\end{array}$$

3.3. Analysis of Primal-Dual Algorithm

First we analyze the behavior of the algorithm without procedures DIJKSTRA-UP and DIJKSTRA-DOWN. In the theorem below we assume that the input pair $(\boldsymbol{x}, f')$ satisfies the condition (3.2b). The proof is given at the end of this section.

Theorem 3.2.
1. *Let* $(\boldsymbol{y}, f)$ *be the output of* MAXFLOW-UP *applied to* $(\boldsymbol{x}, f')$. *Then,*
(a) *The condition* (3.2b) *holds for* $(\boldsymbol{y}, f)$.
(b) *Any minimum s-t cut* (S, T) *of* $\hat{\mathcal{G}}$ *satisfies* $T \setminus \{t\} \in \arg\min\{E(\boldsymbol{x} + \chi_X) \mid X \subseteq \mathcal{V}\}$.
(c) *There hold* $\mathcal{V}^+(\boldsymbol{y}, f) \subseteq \mathcal{V}^+(\boldsymbol{x}, f')$ *and* $\mathcal{V}^-(\boldsymbol{y}, f) \subseteq \mathcal{V}^-(\boldsymbol{x}, f')$.
(d) *If* $\rho^+(\boldsymbol{x}) = 0$, *then* $\mathcal{V}^+(\boldsymbol{y}, f) = \emptyset$.
2. *Let* $(\boldsymbol{y}, f)$ *be the output of* MAXFLOW-DOWN *applied to* $(\boldsymbol{x}, f')$. *Then,*
(a) *The condition* (3.2b) *holds for* $(\boldsymbol{y}, f)$.
(b) *Any minimum s-t cut* (S, T) *of* $\hat{\mathcal{G}}$ *satisfies* $S \setminus \{s\} \in \arg\min\{E(\boldsymbol{x} - \chi_X) \mid X \subseteq \mathcal{V}\}$.
(c) *There hold* $\mathcal{V}^+(\boldsymbol{y}, f) \subseteq \mathcal{V}^+(\boldsymbol{x}, f')$ *and* $\mathcal{V}^-(\boldsymbol{y}, f) \subseteq \mathcal{V}^-(\boldsymbol{x}, f')$.
(d) *If* $\rho^-(\boldsymbol{x}) = 0$, *then* $\mathcal{V}^-(\boldsymbol{y}, f) = \emptyset$.

Combining Lemma 2.7 and Theorem 3.2, we can show that the algorithm terminates in at most $2K + 2$ iterations and yields an optimal primal-dual pair $(\boldsymbol{x}, f)$ upon termination. Indeed, 1 (a) and 2 (a) of Theorem 3.2 imply that the condition (3.2b) always holds. After at most K iterations of procedure UP, the quantity $\rho^+(\boldsymbol{x})$ becomes zero, and therefore after at most $K + 1$ iterations the set $\mathcal{V}^+(\boldsymbol{x}, f)$ becomes empty. At this point flag SuccessUp is set to true, and set $\mathcal{V}^+(\boldsymbol{x}, f)$ will remain empty. Similar argumentation holds for procedure DOWN. When the algorithm terminates, both of the sets $\mathcal{V}^+(\boldsymbol{x}, f)$ and $\mathcal{V}^-(\boldsymbol{x}, f)$ are empty, so the optimality condition (3.2) holds for the pair $(\boldsymbol{x}, f)$.

This analysis remains valid even with procedures DIJKSTRA-UP or DIJKSTRA-DOWN, as shown below.

Theorem 3.3. *Suppose that* $(\boldsymbol{x}, f)$ *satisfies the condition* (3.2b). *Let* $\boldsymbol{y}$ *be the output of* DIJKSTRA-UP *or* DIJKSTRA-DOWN *applied to* $(\boldsymbol{x}, f)$. *Then,*
(a) *The condition* (3.2b) *holds for* $(\boldsymbol{y}, f)$.
(b) *There hold* $\mathcal{V}^+(\boldsymbol{y}, f) \subseteq \mathcal{V}^+(\boldsymbol{x}, f)$ *and* $\mathcal{V}^-(\boldsymbol{y}, f) \subseteq \mathcal{V}^-(\boldsymbol{x}, f)$.
(c) *There hold* $\rho^+(\boldsymbol{y}) \leq \rho^+(\boldsymbol{x})$ *and* $\rho^-(\boldsymbol{y}) \leq \rho^-(\boldsymbol{x})$.

Proof. We consider only procedure DIJKSTRA-UP; the proof for procedure DIJKSTRA-DOWN is completely analogous. The statements (a) and (b) follow directly from the definition of $\boldsymbol{y}$. From (a) and the non-increasing property of the functions $D_u[-f_u](\cdot)$ on $[x_u, y_u]$, it follows that $E(\boldsymbol{y}) = E^f(\boldsymbol{y}) = \min\{E^f(\boldsymbol{z}) \mid \boldsymbol{x} \leq \boldsymbol{z} \leq \boldsymbol{y}\} = \min\{E(\boldsymbol{z}) \mid \boldsymbol{x} \leq \boldsymbol{z} \leq \boldsymbol{y}\}$, which implies (c) by Lemma 2.10. □

It can be seen that if procedure DIJKSTRA-UP is applied to an optimal pair $(\boldsymbol{x}, f)$ then the output $\boldsymbol{y}$ is the maximal optimal solution. Indeed, according to Theorem 3.1 a feasible solution $\boldsymbol{y}$ is optimal if and only if it satisfies

$$D_u[-f_u](y_u) = D_u[-f_u](x_u) \qquad (\forall\, u \in \mathcal{V}),$$
$$V_{uv}[-f_{uv}](y_v - y_u) = V_{uv}[-f_{uv}](x_v - x_u) \qquad (\forall\, (u, v) \in \mathcal{E}).$$

For feasible solutions $\boldsymbol{y} \geq \boldsymbol{x}$ this is equivalent to saying that functions $D_u[-f_u](\cdot)$ are non-increasing on $[x_u, y_u]$ and the condition (3.2b) holds. By construction, DIJKSTRA-UP finds the maximal feasible solution satisfying these conditions. Similarly, we can show

that applying DIJKSTRA-DOWN to an optimal pair $(\boldsymbol{x}, f)$ yields the minimal optimal solution.

The following theorem allows to simplify slightly the algorithm's implementation. The proof is given at the end of this section.

Theorem 3.4.
1. *Let* $(\boldsymbol{y}, f)$ *be the output of* MAXFLOW-UP *applied to* $(\boldsymbol{x}, f')$. *Then, applying* DIJKSTRA-UP *to* $(\boldsymbol{x}, f)$ *and to* $(\boldsymbol{y}, f)$ *would yield the same feasible solution* $\boldsymbol{z}$.
2. *Let* $(\boldsymbol{y}, f)$ *be the output of* MAXFLOW-DOWN *applied to* $(\boldsymbol{x}, f')$. *Then, applying* DIJKSTRA-DOWN *to* $(\boldsymbol{x}, f)$ *and to* $(\boldsymbol{y}, f)$ *would yield the same feasible solution* $\boldsymbol{z}$.

Thus, if DIJKSTRA-UP is applied immediately after MAXFLOW-UP then it is not necessary to update variables $\boldsymbol{x}$ in MAXFLOW-UP (and similarly for DOWN); that is, MAXFLOW-UP updates only dual variables f and then DIJKSTRA-UP updates only primal variables $\boldsymbol{x}$ in this implementation.

We now turn to the proofs of Theorems 3.2 and 3.4. We omit proofs of part 2 of Theorems 3.2 and 3.4 since they are very similar to those of part 1. For simplicity, we will assume without loss of generality that the flow f' is equal to zero; the general case can be shown in the same way by replacing the functions $D_u(\cdot)$ and $V_{uv}(\cdot)$ with $D_u[-f'_u](\cdot)$ and $V_{uv}[-f'_{uv}](\cdot)$, respectively.

Proof of Theorem 3.2, part 1(a). From the capacity constraints we get $f_{uv} \le \Delta^+ V_{uv}(x_v - x_u)$, $-f_{uv} = f_{vu} \le -\Delta^- V_{uv}(x_v - x_u)$. Therefore, we have

$$\begin{aligned} \Delta^+ V_{uv}[-f_{uv}](x_v - x_u) &= \Delta^+ V_{uv}(x_v - x_u) - f_{uv} \ge 0, \\ \Delta^- V_{uv}[-f_{uv}](x_v - x_u) &= \Delta^- V_{uv}(x_v - x_u) - f_{uv} \le 0, \end{aligned}$$

which implies that $V_{uv}[-f_{uv}](x_v - x_u) = \min_{\alpha \in \mathbb{Z}} V_{uv}[-f_{uv}](\alpha)$. Thus, if $y_v - y_u = x_v - x_u$, then the condition (3.2b) holds for edge (u, v). Let us consider the case $y_v - y_u = x_v - x_u + 1$. This can only happen when $u \in S$ and $v \in T$, which means that edge (u, v) must be saturated. Therefore, we have $f_{uv} = \hat{f}_{uv} = \hat{c}_{uv} = \Delta^+ V_{uv}(x_v - x_u)$, implying $\Delta^+ V_{uv}[-f_{uv}](x_v - x_u) = 0$. Hence, we have

$$V_{uv}[-f_{uv}](y_v - y_u) = V_{uv}[-f_{uv}](x_v - x_u + 1) = V_{uv}[-f_{uv}](x_v - x_u) = \min_{\alpha \in \mathbb{Z}} V_{uv}[-f_{uv}](\alpha).$$

The case $y_v - y_u = x_v - x_u - 1$ can be shown similarly. □

Proof of Theorem 3.2, part 1(b). Let (S, T) be an s-t cut of the graph $\hat{\mathcal{G}}$, and put $\boldsymbol{y} = \boldsymbol{x} + \chi_{T \setminus \{t\}}$. Then, we have

$$\begin{aligned} \mathrm{cap}(S,T) &= \sum\{\hat{c}_{su} \mid u \in (T \setminus \{t\}) \setminus \mathcal{V}^+(\boldsymbol{x}, 0)\} + \sum\{\hat{c}_{ut} \mid u \in (S \setminus \{s\}) \cap \mathcal{V}^+(\boldsymbol{x}, 0)\} \\ &\quad + \sum\{\hat{c}_{uv} \mid (u,v) \in \mathcal{E},\ u \in S,\ v \in T\} + \sum\{\hat{c}_{vu} \mid (u,v) \in \mathcal{E},\ u \in T,\ v \in S\} \\ &= \sum\{\Delta^+ D_u(x_u) \mid u \in T \setminus \{t\}\} - \sum\{\Delta^+ D_u(x_u) \mid u \in \mathcal{V}^+(\boldsymbol{x}, 0)\} \\ &\quad + \sum\{\Delta^+ V_{uv}(x_v - x_u) \mid (u,v) \in \mathcal{E},\ u \in S,\ v \in T\} \\ &\quad - \sum\{\Delta^- V_{uv}(x_v - x_u) \mid (u,v) \in \mathcal{E},\ u \in T,\ v \in S\} \\ &= E(\boldsymbol{y}) - E(\boldsymbol{x}) - \sum\{\Delta^+ D_u(x_u) \mid u \in \mathcal{V}^+(\boldsymbol{x}, 0)\}. \end{aligned}$$

This equation shows that (S,T) is a minimum s-t cut if and only if $T\backslash\{t\} \in \arg\min\{E(\boldsymbol{x}+\chi_X) \mid X \subseteq \mathcal{V}\}$. □

Proof of Theorem 3.2, part 1(c). We consider two possible cases.

[Case 1: $u \in \mathcal{V}^+(\boldsymbol{x},0)$] We need to show that $u \notin \mathcal{V}^-(\boldsymbol{y},f)$. By the definition of capacity, we have $-f_u = \hat{f}_{ut} \leq \hat{c}_{ut} = -\Delta^+ D_u(x_u)$, i.e., $\Delta^+ D_u[-f_u](x_u) \leq 0$ holds. Hence, we have

$$\Delta^- D_u[-f_u](y_u) \leq \Delta^- D_u[-f_u](x_u+1) = \Delta^+ D_u[-f_u](x_u) \leq 0,$$

where the first inequality follows from $y_u \leq x_u + 1$ and convexity of $D_u[-f_u](\cdot)$. This implies $u \notin \mathcal{V}^-(\boldsymbol{y},f)$.

[Case 2: $u \notin \mathcal{V}^+(\boldsymbol{x},0)$] We first show that $u \notin \mathcal{V}^+(\boldsymbol{y},f)$ holds. By the definition of capacity, we have $f_u = \hat{f}_{su} \leq \Delta^+ D_u(x_u)$, i.e., $\Delta^+ D_u[-f_u](x_u) \geq 0$ holds. This implies

$$\Delta^+ D_u[-f_u](y_u) \geq \Delta^+ D_u[-f_u](x_u) \geq 0,$$

where the first inequality follows from $y_u \geq x_u$ and convexity of $D_u[-f_u](\cdot)$. Hence, $u \notin \mathcal{V}^+(\boldsymbol{y},f)$.

Now suppose that $u \notin \mathcal{V}^-(\boldsymbol{x},0)$, i.e., $\Delta^- D_u(x_u) \leq 0$. We need to show that $u \notin \mathcal{V}^-(\boldsymbol{y},f)$. If $y_u = x_u$, then this follows from

$$\Delta^- D_u[-f_u](y_u) = \Delta^- D_u(x_u) - f_u \leq 0.$$

If $y_u = x_u + 1$, then $u \in T$, implying that edge (s,u) must be saturated, i.e., $f_u = \hat{f}_{su} = \Delta^+ D_u(x_u)$. Therefore, we have

$$\Delta^- D_u[-f_u](y_u) = \Delta^+ D_u[-f_u](x_u) = \Delta^+ D_u(x_u) - f_u = 0.$$

This implies $u \notin \mathcal{V}^-(\boldsymbol{y},f)$. □

Proof of Theorem 3.2, part 1(d). We show that $u \notin \mathcal{V}^+(\boldsymbol{y},f)$ for all $u \in \mathcal{V}$. For nodes $u \notin \mathcal{V}^+(\boldsymbol{x},0)$ this follows from part (c). Let us consider a node $u \in \mathcal{V}^+(\boldsymbol{x},0)$. The condition $\rho^+(\boldsymbol{x}) = 0$ means that $\emptyset \in \arg\min\{E(\boldsymbol{x}+\chi_X) \mid X \subseteq \mathcal{V}\}$. Therefore, according to part 1(b), cut $(\mathcal{V}\cup\{s\},\{t\})$ is a minimum s-t cut of the graph $\hat{\mathcal{G}}$. Thus, edge (u,t) must be saturated, i.e., $f_u = -\hat{f}_{ut} = -\hat{c}_{ut} = \Delta^+ D_u(x_u)$. This implies that

$$\Delta^+ D_u[-f_u](y_u) \geq \Delta^+ D_u[-f_u](x_u) = \Delta^+ D_u(x_u) - f_u = 0,$$

implying $u \notin \mathcal{V}^+(\boldsymbol{y},f)$, as desired. □

Proof of Theorem 3.4, part 1. Let us show that (i) $D_u[-f_u](y_u) \leq D_u[-f_u](x_u)$ for all nodes u, and (ii) $V_{uv}[-f_{uv}](y_v - y_u) = V_{uv}[-f_{uv}](x_v - x_u)$ for all edges (u,v). The theorem will then follow from the description of DIJKSTRA-UP.

If $y_u = x_u$ for node u then the fact (i) is trivial. Suppose that $y_u = x_u + 1$; we need to show that $\Delta^- D_u[-f_u](y_u) = \Delta^+ D_u[-f_u](x_u) \leq 0$. If $u \in \mathcal{V}^+(\boldsymbol{x},0)$ then this holds since $u \notin \mathcal{V}^-(\boldsymbol{y},f)$ by Theorem 3.2, part 1(c). If $u \notin \mathcal{V}^+(\boldsymbol{x},0)$, then $f_u = \hat{f}_{su} = \hat{c}_{su} = \Delta^+ D_u(x_u)$ since $u \in T$ and edge (s,u) is saturated. Therefore, $\Delta^+ D_u[-f_u](x_u) = \Delta^+ D_u(x_u) - f_u = 0$.

Finally, the fact (ii) was shown earlier (see the proof of Theorem 3.2, part 1(a)). □

4. Application to Panoramic Image Stitching

We discuss an application of our algorithms to panoramic image stitching.

Given two input images I^1 and I^2 defined on overlapping domains $\mathcal{V}^1$ and $\mathcal{V}^2$, the goal of the panoramic image stitching is to compute an output image without a visible seam. Levin et al. [26, 39] proposed several techniques for this problem. One of them, GIST1 algorithm under l_1 norm, is shown to outperform many other stitching methods. It involves minimizing the following function for each color channel:

$$E(\boldsymbol{x}) = \sum_{(u,v)\in\mathcal{E}^1} w^1_{uv}|(x_v - x_u) - (I^1_v - I^1_u)| + \sum_{(u,v)\in\mathcal{E}^2} w^2_{uv}|(x_v - x_u) - (I^2_v - I^2_u)|,$$

where $(u, v) \in \mathcal{E}^i$ if and only if $u, v \in \mathcal{V}^i$ are neighboring pixels for $i = 1, 2$. In other words, we want the gradient of image $\boldsymbol{x}$ to match gradients of images I^1 and I^2. Weights w^1_{uv} and w^2_{uv} are determined as follows. For edges $(u, v) \in \mathcal{E}^1$, if $u, v \in \mathcal{V}^1 \cap \mathcal{V}^2$ we set $w^1_{uv} = 1$; otherwise set $w^1_{uv} = 2$. Similarly, for edges $(u, v) \in \mathcal{E}^2$, if $u, v \in \mathcal{V}^1 \cap \mathcal{V}^2$ we set $w^2_{uv} = 1$; otherwise set $w^2_{uv} = 2$.

It is easy to see that an optimal solution for the minimization of the function E is determined only up to an additive constant. Similar to [26, 39], we computed this constant so that median intensity of I^1 in $\mathcal{V}^1$ matches that of the output image. This does not uniquely determines the solution, however, since there may be multiple optimal solutions $\boldsymbol{x}$ satisfying this requirement. Levin et al. do not discuss how to choose between them.

We propose the following technique. We put constraints $x_u \in [0, K-1]$ on the variables, where K is sufficiently large (e.g., 512). We then compute the minimal optimal solution $\boldsymbol{x}^{\min}$, the maximal optimal solution $\boldsymbol{x}^{\max}$, and their average $\boldsymbol{x}^{\text{av}} = \lfloor(\boldsymbol{x}^{\min} + \boldsymbol{x}^{\max})/2\rfloor$ which is also an optimal solution due to $\text{L}^\natural$-convexity of the objective function (cf. [30, Th. 7.7]). Furthermore, these optimal solutions have the minimum possible range defined as $\max_u\{x_u\} - \min_u\{x_u\} + 1$. In our experiments it was very close to 256. Having a small range may be advantageous since intensities must be mapped to interval [0,255]; if the range is too large then some regions may become too dark or saturated.

Fig. 4 shows panoramas corresponding to feasible solutions $\boldsymbol{x}^{\min}$, $\boldsymbol{x}^{\text{av}}$, and $\boldsymbol{x}^{\max}$. It can be seen that the solution $\boldsymbol{x}^{\text{av}}$ looks significantly better than the other two. The overlap area is too dark in $\boldsymbol{x}^{\min}$ and too bright in $\boldsymbol{x}^{\max}$.

Algorithms tested. We tested the speed of several algorithms on the panoramic image stitching application. We compared the speed of three different algorithms. The first two are the primal-dual method without/with DIJKSTRA-UP and DIJKSTRA-DOWN. We note that the primal-dual method without DIJKSTRA-UP and DIJKSTRA-DOWN can be seen as a particular implementation of the primal method (see Sections 3.2 and 3.3). Procedure DOWN is applied only after SuccessUp becomes true. We used the max flow algorithm of Boykov and Kolmogorov [7] available at http://www.cs.cornell.edu/People/vnk/software.html (version 3.0).

The third technique that we tried is as follows. We converted the original problem to the linear minimum cost flow problem, where we did not enforce constraints $x_u \in [0, K-1]$. We then applied an algorithm of Goldberg [17] available at http://www.avglab.com/andrew/soft.html (version 4.0). It has one free parameter, namely scaling factor; we set it to 32 (results for other factors were faster by at most one percent). The problem (CTP) can be converted

Figure 4: Results of panoramic stitching. First two columns: input images (courtesy of A. Zomet). Rectangles show the area of overlap. Last three columns: results corresponding to $\boldsymbol{x}^{\min}$, $\boldsymbol{x}^{\text{av}}$, and $\boldsymbol{x}^{\max}$, respectively (note that images are cropped). The additive constant is chosen as described in the text.

to the linear cost flow problem in many different ways. We used a transformation with the following property: if the initial feasible solution satisfied the optimality condition, then so did the resulting linear minimum cost flow problem.[1] In all codes we used 32-bit integers.

We note that we did not test the cost scaling algorithm of Ahuja et al. [1]. Their algorithm essentially solves the dual problem (CFP) instead of the primal problem (CTP) by using the cost scaling technique similar to [17], and at the termination an optimal solution (CTP) is obtained as a byproduct. Since the algorithm of Ahuja et al. [1] works with the original graph, it could potentially be faster than converting the problem to a linear minimum cost flow problem and then applying the algorithm in [17]. In our application, however, graph sizes would differ only slightly, and we argue that direct implementation of the technique in [1] is unlikely to beat the implementation in [17].

[1]More precisely, we do the following. We denote by $\boldsymbol{x}^\circ$ and by f° the initial primal and dual solutions, respectively. (The initial flow f° may be non-zero during the second stage of the two-stage procedure, described later.) Suppose that term $V_{uv}[-f^\circ_{uv}](x_v - x_u)$ is represented by breakpoints $b_1 < b_2 < \ldots < b_k$ and slopes $s_0 < s_1 < \ldots < s_k$. For each breakpoint $i = 1, 2, \ldots, k$ we add an arc from node u to v with cost $-(x^\circ_v - x^\circ_u) + b_i$ and the reverse arc from v to u with cost $(x^\circ_v - x^\circ_u) - b_i$. Arc capacities are computed as follows: (i) if $s_{i-1} \leq 0$ and $s_i \geq 0$ then $c_{uv} = s_i$, $c_{uv} = -s_{i-1}$; (ii) if $s_{i-1} > 0$ then $c_{uv} = s_i - s_{i-1}$, $c_{vu} = 0$; (iii) if $s_i < 0$ then $c_{uv} = 0$, $c_{vu} = s_i - s_{i-1}$. Finally, if $s_0 < 0$ or $s_k > 0$ then we add δ to the excess of node v and subtract δ from the excess of node u, where $\delta = s_0$ if $s_0 > 0$ and $\delta = s_k$ if $s_k < 0$. Unary terms are handled similar to pairwise terms. In fact, we can use the description above, if we convert unary terms to pairwise terms as described in Section 1.1.

This reduction corresponds to converting the convex cost flow problem (CFP) to a linear minimum cost flow problem. Note, in the first version of the paper [24] we used a similar procedure, only we first applied the minimal change to flow f° to ensure that condition (3.2b) holds for all edges. Due to this step the running times reported in [24] were significantly slower. Also, in [24] we used the *reversed* graph.

Indeed, the latter is highly optimized and includes many heuristics which significantly improve the empirical performance.

We neither test the algorithms in [2, 21, 26], although they are applicable to the panoramic image stitching application. The algorithms from [2, 21] use a huge graph, so it seems natural that it would be significantly slower. The paper of [26] uses some iterative optimization technique which converges in the limit (and thus not a polynomial-time algorithm).

Initialization and two-stage procedure. Algorithms were initialized with the following feasible solution $\boldsymbol{x}^\circ$: $x^\circ_u = I^1_u$ in region $\mathcal{V}^1 \setminus \mathcal{V}^2$, $x^\circ_u = I^2_u$ in $\mathcal{V}^2 \setminus \mathcal{V}^1$, and $x^\circ_u = \lfloor (I^1_u + I^2_u)/2 \rfloor$ in $\mathcal{V}^1 \cap \mathcal{V}^2$. Besides applying an algorithm directly to the original problem, we also tested the following two-stage procedure. First we solve the problem for a subgraph induced by subset $\mathcal{V}'$ obtained by the erosion of the set $\mathcal{V}^1 \cap \mathcal{V}^2$ by one pixel. In other words, we fix nodes in $\mathcal{V} \setminus \mathcal{V}'$ by adding terms $C|x_u - x^\circ_u|$ to the objective function for nodes $u \in \mathcal{V} \setminus \mathcal{V}'$, where C is a sufficiently large constant. (In implementation nodes which are not connected to nodes in $\mathcal{V}'$ can be safely omitted.) Then we apply the algorithm to the whole problem using the optimal solution and the flow obtained in the first stage as an initialization.

Experiments. We used three datasets D0, D1, and D2 shown in Fig. 4. Their dimensions are 449×193 for D0 and 577×257 for D1 and D2. The percentages of overlap area are 4.9%, 10.0% and 6.9%, respectively. We also used scaled-down datasets D0-s, D1-s and D2-s (both X and Y dimensions are reduced by 2 times). Note that results for scaled-down images visually look worse.

The table below shows running times in seconds (we measure the total time for 3 color channels). The tests were performed on a machine with Intel Celeron 1.4GHz processor in Microsoft Windows XP environment, using Microsoft Visual Studio 7.0 C++ compiler.

	D0-s	D1-s	D2-s	D0	D1	D2
primal-dual, no Dijkstra, 1 stage	12.8	25.7	29.3	61.6	148	160
primal-dual, no Dijkstra, 2 stages	3.55	15.0	15.9	22.8	101	115
primal-dual with Dijkstra, 1 stage	2.47	6.15	7.97	14.9	26.0	56.0
primal-dual with Dijkstra, 2 stages	0.44	0.71	0.75	1.93	3.19	3.17
linear minimum cost flow, 1 stage	1.00	2.49	2.21	10.9	19.3	21.5
linear minimum cost flow, 2 stages	0.94	0.34	0.80	3.93	1.83	1.56

While a naive implementation of the primal-dual algorithm is quite slow, the implementation with Dijkstra computations and two-stage procedure is much faster and competitive with the linear minimum cost flow approach. Our preliminary experiments are not enough to get a robust conclusion and choose between the two. The result indicate, however, that the two-stage procedure is a promising heuristic for the panoramic image stitching application.

Acknowledgements

The authors thank Kazuo Murota for valuable comments on the manuscript. The first author is supported by EPSRC. The second author is partially supported by Grant-in-Aid of the Ministry of Education, Culture, Sports, Science and Technology of Japan.

Input: initial feasible solution $\boldsymbol{x} := \boldsymbol{x}^\circ \in \operatorname{dom} E$.
Step 1: Compute $X^+ \in \arg\min\{E(\boldsymbol{x} + \chi_X) \mid X \subseteq \mathcal{V}\}$.
Step 2: Compute $X^- \in \arg\min\{E(\boldsymbol{x} - \chi_X) \mid X \subseteq \mathcal{V}\}$.
Step 3: If $E(\boldsymbol{x}) = \min\{E(\boldsymbol{x} + \chi_{X^+}), E(\boldsymbol{x} - \chi_{X^-})\}$, then output x and stop.
Step 4: If $E(\boldsymbol{x} + \chi_{X^+}) \leq E(\boldsymbol{x} - \chi_{X^-})$, then set $\boldsymbol{x} := \boldsymbol{x} + \chi_{X^+}$; otherwise set $\boldsymbol{x} := \boldsymbol{x} - \chi_{X^-}$.
Step 5: Go to Step 1.

Figure 5: Murota's steepest descent algorithm for the minimization of an $\mathrm{L}^\natural$-convex function. The algorithm described here is slightly different from the original one in the choice of X^+ and X^-; in the original algorithm X^+ is the unique minimal minimizer and X^- is the unique maximal minimizer.

A. Appendix: Analysis of Murota's Steepest Descent Algorithm

Our primal algorithm is very similar to the steepest descent algorithm of Murota [30, 31] for minimizing an $\mathrm{L}^\natural$-convex function (see Fig. 5). In this section, we discuss the relationship between our primal algorithm and Murota's algorithm. We will assume throughout this section that $E : \mathbb{Z}^{\mathcal{V}} \to \mathbb{R} \cup \{+\infty\}$ is an $\mathrm{L}^\natural$-convex function, and consider the minimization of the function E.

Since Murota's steepest descent algorithm for $\mathrm{L}^\natural$-convex function can be seen as a specialized implementation of our primal algorithm, Theorem 2.8 implies that Murota's algorithm terminates in $O(K_\infty)$ iterations, which is much better than the previous bound $O(K_1)$ shown in [31], where

$$K_1 = \max\{||\boldsymbol{x} - \boldsymbol{y}||_1 \mid \boldsymbol{x}, \boldsymbol{y} \in \operatorname{dom} E\}.$$

In Appendix, we show the following:

- Our primal algorithm requires the same or larger number of iterations.
- Our primal algorithm requires the same or fewer total number of calls to the minimization procedure in UP and DOWN.

Note that one iteration of Murota's algorithm makes two calls to the procedure for minimizing a submodular function, so it is roughly twice as expensive as one iteration of the primal algorithm.

A.1. Analysis of Steepest Descent Algorithm for L-convex Functions

In [31], Murota firstly proposes a steepest descent algorithm for L-convex functions, which is then adapted to $\mathrm{L}^\natural$-convex functions through the relation (2.1). For the simplicity of the proof, we firstly analyze the number of iterations required by the algorithm for L-convex functions, and then restate the result in terms of $\mathrm{L}^\natural$-convex functions.

Murota's steepest descent algorithm for L-convex functions is described in Fig. 6, where $\tilde{\mathcal{V}} = \{0\} \cup \mathcal{V}$ and $\tilde{E} : \mathbb{Z}^{\tilde{\mathcal{V}}} \to \mathbb{R} \cup \{+\infty\}$ is an L-convex function with $\operatorname{dom} \tilde{E} \neq \emptyset$. We note that Murota's steepest descent algorithm coincides with the one by Bioucas-Dias and Valadão [4] when it is applied to the problem (CTP_0), which is a special case of L-convex function minimization (see Proposition 2.3 (ii)).

Input: initial feasible solution $\tilde{\boldsymbol{x}} := \tilde{\boldsymbol{x}}^\circ \in \operatorname{dom} \tilde{E}$.
Step 1: Compute $\tilde{X}^+ \in \arg\min\{\tilde{E}(\tilde{\boldsymbol{x}} + \chi_{\tilde{X}}) \mid \tilde{X} \subseteq \tilde{\mathcal{V}}\}$.
Step 2: If $\tilde{E}(\tilde{\boldsymbol{x}}) = \tilde{E}(\tilde{\boldsymbol{x}} + \chi_{\tilde{X}^+})$, then output $\tilde{\boldsymbol{x}}$ and stop.
Step 3: Set $\tilde{\boldsymbol{x}} := \tilde{\boldsymbol{x}} + \chi_{\tilde{X}^+}$.
Step 4: Go to Step 1.

Figure 6: Murota's steepest descent algorithm for the minimization of an L-convex function. The algorithm described here is slightly different from the original one in the choice of $\tilde{X}^+$; in the original algorithm $\tilde{X}^+$ is the unique minimal minimizer.

Given a vector $\tilde{\boldsymbol{x}} \in \mathbb{Z}^{\tilde{\mathcal{V}}}$, we denote by $\tilde{\boldsymbol{x}}^+$ the unique minimal vector in the set $\arg\min\{\tilde{E}(\tilde{\boldsymbol{z}}) \mid \tilde{\boldsymbol{z}} \geq \tilde{\boldsymbol{x}}\}$ and define $\tilde{\mu}(\tilde{\boldsymbol{x}}) = ||\tilde{\boldsymbol{x}}^+ - \tilde{\boldsymbol{x}}||_\infty$. It should be mentioned that the definition of $\tilde{\mu}(\tilde{\boldsymbol{x}})$ does not change even if $\tilde{\boldsymbol{x}}^+$ is replaced by the unique maximal vector in $\arg\min\{\tilde{E}(\tilde{\boldsymbol{z}}) \mid \tilde{\boldsymbol{z}} \leq \tilde{\boldsymbol{x}}\}$.

The property (LF2) of L-convex functions implies that

$$(\tilde{\boldsymbol{x}}^\circ)^+ \in \arg\min\{\tilde{E}(\tilde{\boldsymbol{z}}) \mid \tilde{\boldsymbol{z}} \geq \tilde{\boldsymbol{x}}^\circ\} \subseteq \arg\min \tilde{E},$$

i.e., the vector $(\tilde{\boldsymbol{x}}^\circ)^+$ is a minimizer of the function $\tilde{E}$. On the other hand, it is easy to see that Murota's algorithm is the same as our primal algorithm except that procedure DOWN is missing. Therefore, the discussion in Section 2.2 shows that Murota's algorithm outputs the vector $(\tilde{\boldsymbol{x}}^\circ)^+$ in $\tilde{\mu}(\tilde{\boldsymbol{x}}^\circ) + 1$ iterations.

Theorem A.1. *The number of iterations of Murota's steepest descent algorithm for L-convex function $\tilde{E}$ is equal to $\tilde{\mu}(\tilde{\boldsymbol{x}}^\circ) + 1$.*

A.2. Analysis of Steepest Descent Algorithm for $L^\natural$-convex Functions

We now analyze the number of iterations required by the steepest descent algorithm for $L^\natural$-convex functions.

The behavior of the steepest descent algorithm for an $L^\natural$-convex function E with the initial vector $\boldsymbol{x}^\circ$ is essentially the same as that of the steepest descent algorithm for the L-convex function $\tilde{E}$ defined by (2.1) with the initial vector $\tilde{\boldsymbol{y}}^\circ = (0, \boldsymbol{x}^\circ) \in \mathbb{Z} \times \mathbb{Z}^{\mathcal{V}}$. The correspondence between the two steepest descent algorithms is as follows (see [31]):

$L^\natural$-convex E		L-convex $\tilde{E}$
$\boldsymbol{x} \to \boldsymbol{x} + \chi_X$	$\Longleftrightarrow$	$\tilde{\boldsymbol{y}} \to \tilde{\boldsymbol{y}} + (0, \chi_X)$
$\boldsymbol{x} \to \boldsymbol{x} - \chi_X$	$\Longleftrightarrow$	$\tilde{\boldsymbol{y}} \to \tilde{\boldsymbol{y}} + (1, \chi_{\mathcal{V} \setminus X})$

where $\tilde{\boldsymbol{y}} = (x_0, \boldsymbol{x} + x_0\mathbf{1})$ and x_0 is a nonnegative integer representing the number of iterations with "$\boldsymbol{x} \to \boldsymbol{x} - \chi_X$" so far.

For a vector $\boldsymbol{x} \in \mathbb{Z}^{\mathcal{V}}$ we define $\mu(\boldsymbol{x}) = \tilde{\mu}(0, \boldsymbol{x})$. As an immediate corollary of Theorem A.1 we obtain the following bound on the number of iterations.

Theorem A.2. *The number of iterations of Murota's steepest descent algorithm for $L^\natural$-convex function E is equal to $\mu(\boldsymbol{x}^\circ) + 1$.*

We will show that our primal algorithm requires the same or larger number of iterations than Murota's algorithm.

Theorem A.3. *The number of iterations of our primal algorithm for $L^\natural$-convex function E is at least $\mu(\boldsymbol{x}^\circ)+1$.*

Proof. Let $\boldsymbol{x}$ be the output of our primal algorithm applied to $\boldsymbol{x}^\circ$. Denote

$$\begin{aligned} d^+ &= \max\big[0, \max\{x_u - x_u^\circ \mid u \in \mathcal{V},\ x_u > x_u^\circ\}\big], \\ d^- &= \max\big[0, \max\{x_u^\circ - x_u \mid u \in \mathcal{V},\ x_u < x_u^\circ\}\big]. \end{aligned}$$

Clearly, the primal algorithm calls procedure UP (resp., DOWN) at least $d^+ + 1$ (resp., $d^- + 1$) times. We will show next that $d^+ + d^- \geq \mu(\boldsymbol{x}^\circ)$, which will imply the theorem.

Consider vector $\tilde{\boldsymbol{y}} = (d^-, \boldsymbol{x} + d^-\mathbf{1})$. Since $\tilde{E}(\tilde{\boldsymbol{y}}) = \tilde{E}(0, \boldsymbol{x})$ and $(0, \boldsymbol{x})$ is a minimizer of $\tilde{E}$, vector $\tilde{\boldsymbol{y}}$ is also a minimizer. Furthermore, $\tilde{\boldsymbol{y}} \geq (0, \boldsymbol{x}^\circ)$. Thus, $||\tilde{\boldsymbol{y}} - (0, \boldsymbol{x}^\circ)||_\infty \geq \tilde{\mu}(0, \boldsymbol{x}^\circ)$. It remains to notice that $||\tilde{\boldsymbol{y}} - (0, \boldsymbol{x}^\circ)||_\infty = d^+ + d^-$. □

We then show that our primal algorithm requires the same or fewer total number of calls to the minimization procedure in UP and DOWN than Murota's algorithm.

Theorem A.4. *For any feasible solution $\boldsymbol{x} \in \operatorname{dom} E$ there hold $\rho^+(\boldsymbol{x}) \leq \mu(\boldsymbol{x})$ and $\rho^-(\boldsymbol{x}) \leq \mu(\boldsymbol{x})$.*

Proof. We prove only the first inequality. Let $\tilde{\boldsymbol{x}}^*$ be the minimal vector in $\arg\min\{\tilde{E}(\tilde{\boldsymbol{y}}) \mid \tilde{\boldsymbol{y}} \geq (0, \boldsymbol{x})\}$. Then, $\mu(\boldsymbol{x}) = ||\tilde{\boldsymbol{x}}^* - (0, \boldsymbol{x})||_\infty$. We will show next that $(0, \boldsymbol{x}^+) \leq \tilde{\boldsymbol{x}}^*$ (recall the definition of $\boldsymbol{x}^+$ in Section 2.2). This will imply the desired inequality since $\rho^+(\boldsymbol{x}) = ||(0, \boldsymbol{x}^+) - (0, \boldsymbol{x})||_\infty$.

We define vectors $\tilde{\boldsymbol{y}} = (y_0, \boldsymbol{y}) \in \mathbb{Z} \times \mathbb{Z}^{\mathcal{V}}$ and $\tilde{\boldsymbol{z}} = (z_0, \boldsymbol{z}) \in \mathbb{Z} \times \mathbb{Z}^{\mathcal{V}}$ by $\tilde{\boldsymbol{y}} = \tilde{\boldsymbol{x}}^* \wedge (0, \boldsymbol{x}^+)$ and $\tilde{\boldsymbol{z}} = \tilde{\boldsymbol{x}}^* \vee (0, \boldsymbol{x}^+)$, respectively. Clearly, $y_0 = 0$. We have

$$E(\boldsymbol{y}) = \tilde{E}(\tilde{\boldsymbol{y}}) \leq \tilde{E}(0, \boldsymbol{x}^+) + [\tilde{E}(\tilde{\boldsymbol{x}}^*) - \tilde{E}(\tilde{\boldsymbol{z}})] \leq \tilde{E}(0, \boldsymbol{x}^+) = E(\boldsymbol{x}^+),$$

where the first inequality follows from submodularity of $\tilde{E}$, and the second inequality follows from the optimality of $\tilde{\boldsymbol{x}}^*$ and the fact that $\tilde{\boldsymbol{z}} \geq (0, \boldsymbol{x})$. Since $\boldsymbol{y} \geq \boldsymbol{x}$ and $E(\boldsymbol{y}) \leq E(\boldsymbol{x}^+)$, we have $\boldsymbol{x}^+ \leq \boldsymbol{y}$. Thus, $(0, \boldsymbol{x}^+) \leq \tilde{\boldsymbol{y}} \leq \tilde{\boldsymbol{x}}^*$, as claimed. □

We note that our algorithm makes at most $\rho^+(\boldsymbol{x}^\circ) + \rho^+(\boldsymbol{x}^\circ) + 2$ calls to the procedure for minimizing a submodular function, while Murota's algorithm makes $2\mu(\boldsymbol{x}^\circ) + 2$ such calls. Thus, the theorem implies that our algorithm makes the same of fewer number of calls.

It should be mentioned that Murota's algorithm can be implemented so that it calls the procedure for minimizing a submodular function only once in each iteration. Instead of computing both of X^+ and X^- and choosing a better one, we just need to compute $\tilde{X}^+ \in \arg\min\{\tilde{E}(\tilde{\boldsymbol{x}} + \chi_{\tilde{X}}) \mid \tilde{X} \subseteq \tilde{\mathcal{V}}\}$, as in Murota's algorithm for L-convex function, and then compute X^+ or X^- by using $\tilde{X}^+$.

References

[1] R. K. Ahuja, D. S. Hochbaum, J. B. Orlin, Solving the convex cost integer dual network flow problem, Management Sci. 49 (2003) 950–964.

[2] R. K. Ahuja, D. S. Hochbaum, J. B. Orlin, A cut based algorithm for the convex dual of the minimum cost network flow problem, Algorithmica 39 (2004) 189–208.

[3] R. K. Ahuja, T. L. Magnanti, J. B. Orlin, Network Flows: Theory, Algorithms, and Applications, Prentice Hall, Englewood, NJ, 1993.
[4] J. Bioucas-Dias, G. Valadão, Phase unwrapping via graph cuts, in: Proc. of Pattern Recognition and Image Analysis, 2nd Iberian Conf. (IbPRIA), LNCS 3533, Springer, Berlin, 2005, pp. 360–367.
[5] J. Bioucas-Dias, G. Valadão. Phase unwrapping via graph cuts, IEEE Trans. Image Process. 16 (2007) 698–709.
[6] E. Boros, P. L. Hammer, Pseudo-boolean optimization, Discrete Appl. Math. 123 (2002) 155–225.
[7] Y. Boykov, V. Kolmogorov, An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision, IEEE Trans. Pattern Anal. Mach. Intell. 26 (2004) 1124–1137.
[8] Y. Boykov, O. Veksler, R. Zabih, Fast approximate energy minimization via graph cuts, IEEE Trans. Pattern Anal. Mach. Intell. 23 (2001) 1222–1239.
[9] A. Chambolle, Total variation minimization and a class of binary MRF models, in: 5th International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition, LNCS 3757, Springer, Berlin, 2005, pp. 136–152.
[10] J. Darbon, Composants logiciels et algorithmes de minimisation exacte d'énergies dédiées au traitement des images, PhD thesis, Ecole Nationale Supérieure des Télécommunications, October 2005. (In French)
[11] J. Darbon, M. Sigelle, A fast and exact algorithm for total variation minimization, in: Proc. of Pattern Recognition and Image Analysis, 2nd Iberian Conf. (IbPRIA), LNCS 3533, Springer, Berlin, 2005, pp. 351–359.
[12] P. Favati, F. Tardella, Convexity in nonlinear integer programming, Ricerca Operativa 53 (1990) 3–44.
[13] L. R. Ford, D. R. Fulkerson, A primal-dual algorithm for the capacitated Hitchcock problem, Naval Res. Logist. Quart. 4 (1957) 47–54.
[14] L. R. Ford, D. R. Fulkerson, Flows in Networks, Princeton Univ. Press, Princeton, NJ, 1962.
[15] A. Fujishige, K. Murota, Notes on L-/M-convex functions and the separation theorems, *Math. Programming* 88 (2000) 129–146.
[16] S. Geman, D. Geman, Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images, IEEE Trans. Pattern Anal. Mach. Intell. 6 (1984) 721–741.
[17] A. V. Goldberg, An efficient implementation of a scaling minimum-cost flow algorithm, J. Algorithms 22 (1997) 1–29.
[18] A. V. Goldberg, R. E. Tarjan, A new approach to the maximum flow problem, J. ACM 35 (1988) 921–940.
[19] D. S. Hochbaum, An efficient algorithm for image segmentation, Markov random fields and related problems, J. ACM 48 (2001) 686–701.
[20] M. Iri, Network Flow, Transportation and Scheduling—Theory and Algorithms, Academic Press, New York, NY, 1969.
[21] H. Ishikawa, Exact optimization for Markov random fields with convex priors, IEEE Trans. Pattern Anal. Mach. Intell. 25 (2003) 1333–1336.
[22] A. B. Karzanov, S. T. McCormick, Polynomial methods for separable convex optimization in unimodular linear spaces with applications, SIAM J. Comput. 4 (1997) 1245–1275.
[23] J. Kleinberg, E. Tardos, Approximation algorithms for classification problems with pairwise relationships: metric labeling and Markov random fields, in: Proc. of 40th Annual Symp. on Foundations of Computer Science (FOCS), IEEE Computer Society, Washington DC, 1999, pp. 14–23.
[24] V. Kolmogorov, Primal-dual algorithm for convex Markov random fields, Technical Report MSR-TR-2005-117, Microsoft Research, 2005.
[25] N. Komodakis, G. Tziritas, A new framework for approximate labeling via graph cuts, in: Proc. of 10th IEEE Int. Conf. on Computer Vision (ICCV), IEEE Computer Society, Washington DC, 2005, pp. 1018–1025.
[26] A. Levin, A. Zomet, S. Peleg, Y. Weiss, Seamless image stitching in the gradient domain, in: Proc. of 8th European Conf. on Computer Vision, LNCS 3024, Springer, Berlin, 2004, pp. 377–389.
[27] M. Minoux, A polynomial algorithm for minimum quadratic cost flow problems, European J. Oper. Res. 18 (1984) 377–387.
[28] K. Murota, Discrete convex analysis, *Math. Programming* 83 (1998) 313–371.
[29] K. Murota, Algorithms in discrete convex analysis, IEICE Trans. Inform. Syst., E83-D (2000) 344–352.
[30] K. Murota, Discrete Convex Analysis, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2003.
[31] K. Murota, On steepest descent algorithms for discrete convex functions, SIAM J. Opt. 14 (2003)

699–707.
[32] J. B. Orlin, A faster strongly polynomial time algorithm for submodular function minimization, Math. Programming 118 (2009) 237–251.
[33] R. T. Rockafellar, Network Flows and Monotropic Optimization, Wiley, New York, NY, 1984.
[34] A. Shioura, Note on $L^\natural$-convex function minimization algorithms: comparison of Murota's and Kolmogorov's algorithms, Technical Report METR 2006-03, University of Tokyo, 2006.
[35] O. Veksler, Efficient graph-based energy minimization methods in computer vision, PhD thesis, Cornell University, 1999.
[36] B. Zalesky, Network flow optimization for restoration of images, J. Appl. Math. 2 (2002) 199–218.
[37] B. Zalesky, Fast algorithms of Bayesian segmentation of images, preprint, arXiv:math/0206184v2 (2002).
[38] B. Zalesky, Efficient determination of Gibbs estimators with submodular energy functions, preprint, arXiv:math/0304041v1 (2003).
[39] A. Zomet, A. Levin, S. Peleg, Y. Weiss, Seamless image stitching by minimizing false edges, IEEE Trans. Image Process. 15 (2006) 969–977.