Non-heuristic reduction of the graph in graph-cut optimization

François Malgouyres[1] **and Nicolas Lermé**[2]

[1]Institut de Mathématiques de Toulouse, CNRS UMR 5219, Université de Toulouse, France
[2]LAGA, CNRS UMR 7539 and LIPN CNRS UMR 7030, Université de Paris 13, France

E-mail:
[1]Francois.Malgouyres@math.univ-toulouse.fr [2]nicolas.lerme@lipn.univ-paris13.fr

Abstract. During the last ten years, graph cuts had a growing impact in shape optimization. In particular, they are commonly used in applications of shape optimization such as image processing, computer vision and computer graphics. Their success is due to their ability to efficiently solve (apparently) difficult shape optimization problems which typically involve the perimeter of the shape. Nevertheless, solving problems with a large number of variables remains computationally expensive and requires a high memory usage since underlying graphs sometimes involve billion of nodes and even more edges. Several strategies have been proposed in the literature to improve graph-cuts in this regards. In this paper, we give a formal statement which expresses that a simple and local test performed on every node before its construction permits to avoid the construction of useless nodes for the graphs typically encountered in image processing and vision. A useless node is such that the value of the maximum flow in the graph does not change when removing the node from the graph. Such a test therefore permits to limit the construction of the graph to a band of useful nodes surrounding the final cut.

1. Introduction

Shape optimization is a field which has many practical applications. Among shape optimization methods, graph-cut methods have recently received a growing interest because of their ability to minimize pairwise Markov Random Fields of the form

$$E(u) = \beta \sum_{p\in\mathcal{P}} E_p(u_p) + \sum_{(p,q)\in(\mathcal{P}\times\mathcal{P})} E_{p,q}(u_p, u_q), \qquad \beta \in \mathbb{R}^+, \tag{1}$$

among $u \in \{0,1\}^{\mathcal{P}}$, for a set of pixels $\mathcal{P} \subset \mathbb{Z}^d$ and for a positive integer d. Indeed, when the terms $E_{p,q}(.)$ are submodular, there is a way to build a graph (see [1, 2]) such that if, for any $S \subset \mathcal{P}$, we denote by $u^S \in \{0,1\}^{\mathcal{P}}$ the binary partition of $\mathcal{P}$ defined for all $p \in \mathcal{P}$ by

$$u^S_p = \begin{cases} 1 & \text{, if} \quad p \in S, \\ 0 & \text{, if} \quad p \notin S, \end{cases} \tag{2}$$

we have

$$\mathrm{val}_{\mathcal{G}}(S) = E(u^S) + K, \tag{3}$$

for some (irrelevant in the current minimization context) constant $K \in \mathbb{R}$ and where $\mathrm{val}_{\mathcal{G}}(S)$ is the value of the cut defined by $S \subset \mathcal{P}$ (see next section for a precise definition).

Moreover, since (2) makes a one to one correspondence between cuts and elements of $\{0,1\}^{\mathcal{P}}$, we trivially obtain from (3) that to any minimum cut (min-cut) in $\mathcal{G}$ corresponds a minimizer of (1) and vice versa. Moreover, a min-cut can efficiently be computed with a maximum flow (max-flow) algorithm such as the one described in [3, 4]. The max flow is indeed the dual of the min-cut problem.

One drawback of graph-cuts is that the graph contains as many nodes as the number of pixels and that its number of edges behaves like

$$O\left(\sum_{p\in\mathcal{P}} \#\sigma(p)\right),$$

where $\#\sigma(p)$ is the number of neighbors of a pixel $p \in \mathcal{P}$ (see Section 2 for a detailed description of the graph). Such a memory requirement is a strong limitation when minimizing (1) for $\#\mathcal{P}$ large and/or for d large ($\#\sigma(p)$ typically grows like a radius raised to the power d...)

As a consequence, many strategies have been proposed to avoid the construction of the whole graph. We can find in the literature few methods that actually provide a global solution of the minimization problem [5, 6, 7]. However, most of he proposed methods are heuristics who may fail to accurately preserve thin structures [8, 9, 10, 11, 12]. The strategy developed in the current paper aims at building a thin band surrounding the final boundary between the object and the background. It shares this property with the other band-based methods (see [13, 10, 7, 9, 8]).

In this paper, we propose not to build any node satisfying a local test. This test is similar to the ones in [14, 13]. The nodes that fail to comply with the test are built in the reduced graph. The difference with the results stated in [14, 13] is that the test has been slightly modified in order to obtain a formal statement guaranteeing that the value of the max-flow in the reduced graph is equal to the value of the max-flow in the initial graph. As a consequence, any max-flow in the reduced graph is also a max-flow in the initial graph and the cut defined by this flow is a min-cut in the initial graph. In other words, despite the graph reduction, the max-flow algorithm still permits to build a minimizer of E.

The proof of the result presented in this paper as well as experiments are provided in [15] and will also be submitted for publication in a long version of the current paper.

2. Definition of the graph

The graph constructed in [1] has the form described below. This constitute the main motivation for considering this graph structure.

We consider two terminal nodes s and t and the set of nodes

$$\mathcal{V} = \mathcal{P} \cup \{s, t\}.$$

We consider a set of directed edges $\mathcal{E} \subset (\mathcal{V} \times \mathcal{V})$ such that $(\mathcal{V}, \mathcal{E})$ is a simple directed graph. We also assume that for every $p \in \mathcal{P}$,

$$(p, s) \notin \mathcal{E} \text{ and } (t, p) \notin \mathcal{E}.$$

We denote the neighbors of any nodes $p \in \mathcal{V}$ by

$$\sigma_{\mathcal{E}}(p) = \{q \in \mathcal{V}, (p, q) \in \mathcal{E} \text{ or } (q, p) \in \mathcal{E}\}.$$

We define the capacities as a mapping $c : (\mathcal{V} \times \mathcal{V}) \to \mathbb{R}^+$ and denote the capacity of any edge $(p, q) \in (\mathcal{V} \times \mathcal{V})$ by

$$c_{p,q} \geq 0.$$

Notice that usually, the capacities are only defined over $\mathcal{E}$. We extend their definition to $(\mathcal{V}\times\mathcal{V})$ in order to simplify notations. This extension is performed by setting

$$c_{p,q} = 0, \text{ whenever } (p,q) \notin \mathcal{E}. \tag{4}$$

We assume, without loss of generality (see [2, 1]), that capacities are such that for every $p \in \mathcal{P}$

$$c_{s,p} \neq 0 \quad \Rightarrow \quad c_{p,t} = 0.$$

We therefore sum up the capacities of the edges linked to the terminal nodes and set for all $p \in \mathcal{P}$

$$c_p = c_{s,p} - c_{p,t}.$$

For any $S \subset \mathcal{P}$, we denote by $\mathrm{val}_{\mathcal{G}}(S)$ the value of the s-t cut

$$(S \cup \{s\}, (\mathcal{P} \setminus S) \cup \{t\}),$$

in $\mathcal{G}$.

We also define flows in the graph as any mapping $f : (\mathcal{V}\times\mathcal{V}) \to \mathbb{R}^+$ satisfying both the capacity constraints

$$0 \leq f_{p,q} \leq c_{p,q} \quad \text{, for all } (p,q) \in (\mathcal{V}\times\mathcal{V}), \tag{5}$$

and the flow conservation

$$\sum_{q\in\sigma_{\mathcal{E}}(p)} f_{q,p} = \sum_{q\in\sigma_{\mathcal{E}}(p)} f_{p,q}, \qquad \text{for all } p \in \mathcal{P}. \tag{6}$$

As usual, the value of the flow f in $\mathcal{G}$ is defined by

$$\mathrm{val}_{\mathcal{G}}(f) = \sum_{p\in\sigma_{\mathcal{E}}(s)} f_{s,p}. \tag{7}$$

Notice that we use the same notation for the value of a flow and the value of a s-t cut in $\mathcal{G}$. This abuse of notation will never be ambiguous, once in context.

We call max-flow any solution f^* of the linear program

$$\begin{cases} \max_f \quad \mathrm{val}_{\mathcal{G}}(f), \\ \text{subject to (5) and (6).} \end{cases}$$

In the sequel, we also consider a subset $B \subset \mathbb{Z}^d$ and assume that B and $\mathcal{G}$ are such that

$$\forall p \in \mathcal{P}, \quad (\sigma_{\mathcal{E}}(p) \cap \mathcal{P}) \ \subset \ B_p, \tag{8}$$

where

$$B_p = \{p + q, q \in B\}. \tag{9}$$

In practice, we typically think of B as a ball centered at the origin. In such a case, (8) means that neighbors in the graph $\mathcal{G}$ are close to each other in $\mathbb{Z}^d$.

3. Main result of the paper

Theorem 1 *Let $\mathcal{G}$ be a graph as defined in Section 2, let B satisfy* (8) *and let us assume that $p \in \mathcal{P}$ satisfies*

$$\begin{cases} \textit{either} & \forall q \in B_p, \quad c_q \geq \sum_{q' \notin B_p} c_{q,q'}, \\ \textit{or} & \forall q \in B_p, \quad c_q \leq -\sum_{q' \notin B_p} c_{q',q}. \end{cases} \tag{10}$$

Then, there exists a max-flow f in $\mathcal{G}$ such that

$$\forall q \in \sigma_{\mathcal{E}}(p), \qquad f_{p,q} = f_{q,p} = 0.$$

As a consequence, removing the node p from the graph $\mathcal{G}$ does not modify its max-flow value.

Algorithmically, the above theorem guarantees that we can test every node, during the graph construction, before it is added to the graph. If the node satisfies (10), it is not useful to the max-flow evaluation. It can therefore be removed from the graph $\mathcal{G}$ without modifying its max-flow value.

Let us describe the meaning of the test corresponding to the upper line of (10). For every node q in B_p

- either q is in the interior of B_p in which case, for all $q' \notin B_p$, $c_{q,q'} = 0$ and the test simply becomes $c_q \geq 0$.
- or q is on the border of B_p and then the test takes the form $c_q \geq \sum_{q' \notin B_p} c_{q,q'} \geq 0$ and is generally more difficult to satisfy.

Notice also that a similar theorem with the test

$$\begin{cases} \text{either} & \forall q \in B_p, \quad c_q \geq \sum_{q' \in \sigma_{\mathcal{E}}(q)} c_{q,q'}, \\ \text{or} & \forall q \in B_p, \quad c_q \leq -\sum_{q' \in \sigma_{\mathcal{E}}(q)} c_{q',q}, \end{cases}$$

is easy to obtain. The two results should not be confused.

Let us now describe how to build the minimizer of (1), once the minimum cut

$$\Big(S \cup \{s\} \ , \ \big((\mathcal{P} \setminus \{p\}) \setminus S\big) \cup \{t\}\Big),$$

where $S \subset (\mathcal{P} \setminus \{p\})$, has been computed in the graph from which the node p satisfying (10) has been removed. The minimizer $u^* \in \{0,1\}^{\mathcal{P}}$ of (1) is defined by

$$u^*_r = \begin{cases} 1 & \text{, if } r = p \text{ and } \forall q \in B_p, \quad c_q \geq \sum_{q' \notin B_p} c_{q,q'}, \\ 0 & \text{, if } r = p \text{ and } \forall q \in B_p, \quad c_q \leq -\sum_{q' \notin B_p} c_{q',q}, \\ 1 & \text{, if } r \neq p \text{ and } r \in S, \\ 0 & \text{, if } r \neq p \text{ and } r \notin S. \end{cases}$$

The proof of the above theorem as well as some experiments illustrating the situations in which it can successfully be applied are provided in [16].

References

[1] Kolmogorov V and Zabih R 2004 *IEEE Transactions on Pattern Analysis and Machine Intelligence* **26** 147–159

[2] Picard J and Ratliff H 1975 *Networks* **5** 357–370

[3] Boykov Y and Kolmogorov V 2004 *IEEE Transactions on Pattern Analysis and Machine Intelligence* **26** 1124–1137

[4] Hochbaum D S 2008 *Operation research* **56** 992–1009

[5] Delong A and Boykov Y 2008 *CVPR* pp 1–8

[6] Strandmark P and Kahl F 2010 *CVPR* pp 2085–2092
[7] Lempitsky V and Boykov Y 2007 *CVPR* pp 1–8
[8] Lombaert H, Sun Y, Grady L and Xu C 2005 *ICCV* vol 1 pp 259–265
[9] Sinop A and Grady L 2006 *MICCAI* vol 2 pp 896–903
[10] Kohli P, Lempitsky V and Rother C 2010 *DAGM* pp 242–251
[11] Li Y, Sun J, Tang C and Shum H 2004 *ACM Transactions on Graphics* **23** 303–308
[12] Stawiaski J, Decencière E and Bidault F 2007 *ISMM* pp 349–360
[13] Lermé N and Malgouyres F 2011 A reduction method for graph cut optimization Tech. Rep. hal-00606921 CCSD
[14] Lermé N, Malgouyres F and Létocart L 2010 *ICIP* pp 3045–3048
[15] Lermé N 2011 *Réduction de graphes et application la segmentation de tumeurs pulmonaires* Ph.D. thesis Université Paris 13
[16] Malgouyres F and lermé N 2012 A non-heuristic reduction method for graph cut optimization Tech. Rep. hal-00692464 CCSD preprint