Active Learning and Proofreading for Delineation of Curvilinear Structures (supplementary material)

***

***

1 Fast Reconstruction of Curvilinear Structures

To delineate networks of curvilinear structures, we rely on the algorithm of [3], which involves solving the following problem:

Min-Weight Tree Containing r (MinTree)

Given: A graph $\mathcal{G} = (V, \mathcal{E})$, a root vertex $r \in V$, weights on edges $w : \mathcal{E} \to \mathbb{R}$. Weights may be negative.
Find: A tree $\mathcal{R} \subseteq \mathcal{G}$ containing the vertex r, minimizing the sum of weights of picked edges $\sum_{e \in \mathcal{R}} w(e)$.

In our approach, MinTree is used when we expect the ground-truth image to be a tree. If such an assumption is not realistic (loopy networks, such as blood vessels), then we are instead interested in the following problem MinSubgraph:

Min-Weight Connected Subgraph Containing r (MinSubgraph)

Given: A graph $\mathcal{G} = (V, \mathcal{E})$, a root vertex $r \in V$, weights on edges $w : \mathcal{E} \to \mathbb{R}$. Weights may be negative.
Find: A connected subgraph $\mathcal{R} \subseteq \mathcal{G}$ which contains the vertex r (and is not necessarily a tree), minimizing the sum of weights of picked edges $\sum_{e \in \mathcal{R}} w(e)$.

Both problems are significantly harder than the Minimum Spanning Tree problem, because $\mathcal{R}$ does not need to connect the entire graph and also the weights may be negative. In fact, both problems are NP-complete; we demonstrate this later in Proposition 2. In both [3] and our approach they are solved

using a Mixed Integer Programming (**MIP**) formulation, which is given as input to the Gurobi solver.[1]

However, the previously considered formulation (see the model Arbor-IP in [3] and also the model M-DG in [1]) has $|V||\mathcal{E}|$ variables and as many constraints. This makes solving it costly for small graphs and impossible for larger ones. Our contribution is a new, linear-size **MIP** model for this problem.

The organization of this section is the following: in Section 1.1 we introduce our formulation and argue about its correctness, in Section 1.2 we measure the major running time improvements it brings about, in Section 1.3 we show how using a **MIP** formulation (rather than a simple Minimum Spanning Tree based method) boosts the performance of Active Learning, and finally in Section 1.4 we prove the NP-hardness of the considered problems.

Let us mention in passing that Blum and Calvo [1] also propose a "matheuristic" approach to solving MinTree – although with no optimality guarantees.

1.1 Our Formulation

First, we describe how to obtain a **MIP** for MinTree. We replace each undirected edge with two directed edges, so as to work with a directed graph. Our objective is to find a directed tree whose each edge is directed away from the root r (a so-called r-arborescence).

We associate a binary variable $x_{uv} \in \{0, 1\}$ with each directed edge $(u, v) \in \mathcal{E}$, denoting the presence of the edge in the solution $\mathcal{R}$. The first two linear constraints to consider are:

- any vertex v has at most one incoming edge (r has none) (see equations (1–2) below),
- an edge (u, v) can be in the solution only if u has an incoming edge in the solution (or $u = r$) (3).

These conditions almost require the solution to be an r-arborescence, but not quite; namely, there can still appear directed cycles (possibly with some adjoined trees). One way to deal with this issue is to enforce that every non-isolated vertex is connected to the root; this can be done using network flows. The constraints in the previous formulation require that, for every v with an incoming edge, there should exist a flow $\{f_e^v\}_{e\in\mathcal{E}}$ of value 1 from r to v. However, this leads to a large program ($|V||\mathcal{E}|$ variables).

Our way around this is to instead require the existence of a single flow $\{f_e\}_{e\in\mathcal{E}}$ from the source vertex r to some set of sinks. The main constraints are that:

- for every vertex $v \neq r$, if v has an incoming edge (i.e., v is not an isolated vertex in the solution, but is spanned by $\mathcal{R}$), then the inflow into v is at least 1 more than the outflow (otherwise it is greater or equal to the outflow) (4),
- f is supported only on the support of x (that is, the flow f only uses edges which are used by the solution $\mathcal{R}$) (5).

[1] [3] use quadratic weights, i.e., weights on pairs of adjacent edges, rather than a linear weight function; this makes the computational burden even heavier.

Since x has no edges into the root, neither does f. Thus f is indeed a flow (within the x-subgraph) from the source r to the sink set being the set of all active vertices.

We write down our **MIP** formulation below. We use the following notation: $x(F) = \sum_{e \in F} x(e)$ for a subset $F \subseteq \mathcal{E}$, $\delta^+(v)$ is the set of (directed) edges outgoing from vertex v, and $\delta^-(v)$ is the set of (directed) edges incoming into vertex v. Thus e.g. $f(\delta^+(v))$ is the total f-flow outgoing from vertex v.

$$
\begin{aligned}
\text{minimize} \quad & \sum_{(u,v) \in \mathcal{E}} w(u,v) x_{uv} & & \\
\text{subject to} \quad & x_{uv} \in \{0,1\} & \forall (u,v) \in \mathcal{E} & \\
& x(\delta^-(v)) \leq 1 & \forall v \in V \setminus \{r\} & (1) \\
& x(\delta^-(r)) = 0 & & (2) \\
& x_{uv} \leq x(\delta^-(u)) & \forall (u,v) \in \mathcal{E}, u \neq r & (3) \\
& f(\delta^-(v)) - f(\delta^+(v)) \geq x(\delta^-(v)) & \forall v \in V \setminus \{r\} & (4) \\
& f_{uv} \geq 0 & \forall (u,v) \in \mathcal{E} & \\
& f_{uv} \leq (|V| - 1) \cdot x_{uv} & \forall (u,v) \in \mathcal{E}. & (5)
\end{aligned}
$$

The following proposition explains the correctness of our formulation.

Proposition 1. *For any $\mathcal{R} \subseteq \mathcal{E}$, the corresponding vector $x \in \{0,1\}^{\mathcal{E}}$ is feasible for the **MIP** formulation[2] iff $\mathcal{R}$ is a tree containing the root r.*

Proof. ($\Longrightarrow$) By (1), edges (u,v) with $x_{uv} = 1$ form a (directed) subgraph where every vertex has indegree at most 1. It is not hard to see that each connected component of such a graph is either a tree or a cycle (possibly with adjoined trees); the cycle case is impossible if the component contains r (by (2)). We show that actually there is no connected component except the one containing r. Towards a contradiction suppose that $S \subseteq V \setminus \{r\}$ is such a component; we will show that the flow conservation constraints (4) must be violated. Denote by $\delta^+(S) = \{(u,v) \in \mathcal{E} : u \in S, v \notin S\}$ the outgoing edges of S, and by $\delta^-(S)$ the incoming edges. We have $x(\delta^+(S)) = x(\delta^-(S)) = 0$ and thus, by (5), $f(\delta^+(S)) = f(\delta^-(S)) = 0$. However, by summing up (4) over $v \in S$ we get $f(\delta^-(S)) - f(\delta^+(S)) \geq \sum_{v \in S} x(\delta^-(v))$; the left side is 0 but the right side is positive, a contradiction.[3]

[2] More precisely, there exists $f \in \mathbb{R}_+^{\mathcal{E}}$ such that (x,f) is feasible for the **MIP** formulation, where x is obtained from $\mathcal{R}$ by directing all edges to point away from r.

[3] The observant reader will notice that the constraint (3) is redundant. However, we keep it for clarity of exposition and because it makes solving the program faster in practice.

($\Longleftarrow$) It is easy to see that constraints (1–3) are satisfied by x. To obtain the flow, we begin with $f = 0$. Then, for each vertex v with $x(\delta^-(v)) = 1$, we route 1 unit of flow from r to v inside $\mathcal{R}$ (that is, we only use edges e with $x_e = 1$) and add that flow to f. (This is possible since $\mathcal{R}$ is connected.) This way we will satisfy (4). Since the number of such vertices is at most $|V| - 1$, any edge will hold at most $|V| - 1$ units of flow, thus satisfying (5).

So far we have discussed MinTree. To get a formulation for MinSubgraph, one only needs to omit the constraint (1) and adjust the constraint (5) to become $f_{uv} \leq |\mathcal{E}| \cdot x_{uv}$. Then x is obtained from $\mathcal{R}$ by choosing any spanning tree of $\mathcal{R}$ and orienting tree edges to point away from r and non-tree edges arbitrarily. In the proof of Proposition 1 we route $x(\delta^-(v)) \geq 1$ units of flow (rather than 1 unit) for each v (now any edge holds at most $|\mathcal{E}|$ units of flow). These are the only changes.

1.2 Running Time Improvements

The runtimes of our formulation compared to the one presented in [3] are shown in Table 1. The optimization was executed on a 2x Intel E5-2680 v2 system (20 cores).

	Axons1	*Axons2*	*Axons3*	*Axons4*	*Axons5*	*Axons6*
# edges	164	223	224	265	932	2638
MIP [3]	0.91	1.04	1.19	1.45	78.3	393.7
MIP ours	0.03	0.10	0.04	0.23	0.10	5.23
speedup	26.1x	10.1x	27.3x	6.3x	743.5	75.2x

	BFNeuron1	*BFNeuron2*	*OPF1*	*OPF2*	*BFNeuron3*	*BFNeuron4*
# edges	120	338	363	380	645	2826
MIP [3]	0.48	2.25	1.53	1.65	2.13	308.23
MIP ours	0.02	0.12	0.05	0.08	0.26	2.30
speedup	18.2x	17.7x	29.4x	19.9x	8.1x	134.0x

Table 1: Per-reconstruction runtimes (in seconds) of the **MIP** formulation of [3] and ours for the proofreading task.

Our formulation can be solved under 6 seconds for all real-world graph examples we have tried; the maximum for the formulation of [3] is over 6 minutes.

We also compared the runtimes on randomly generated graphs of various sizes – see Table 2. The speed-ups remain similar. In Table 3 we collect runtimes of our method on larger randomly generated graphs. If we assumed (more or less arbitrarily) 2 seconds to be the threshold of what is practical in an interactive setting (given that this optimization needs to be run multiple times), then we can see that the method of [3] can deal with graphs of size at most 300, whereas our method copes with graphs having around 2000 edges.

# edges	99	132	220	330	440	660	924	1320	1540
MIP [3]	0.16	0.30	1.13	3.39	8.35	29.35	73.16	112.59	149.01
MIP ours	0.03	0.04	0.06	0.12	0.15	0.29	0.36	0.67	0.42
speedup	6.1x	7.5x	17.9x	29.4x	53.9x	102.8x	201.8x	167.8x	348.1x

Table 2: Per-reconstruction runtimes (in seconds) of the **MIP** formulation of [3] and ours on random graphs.

# edges	1760	2420	3520	4400	5720	9900
MIP ours	1.60	2.71	6.59	9.57	15.52	81.55

Table 3: Per-reconstruction runtimes (in seconds) of our **MIP** formulation on random graphs.

One further practical method for speeding up the solver is to initialize it with a nonzero feasible solution. In cases where we needed to explore a large number of reconstructions resulting from altering just one weight at a time (which was the setting of our paper), we initialized the new solution to the current optimal solution. Note that this scenario makes performance considerations especially relevant, as $|\mathcal{E}|$ reconstructions need to be made; even though they can be run in parallel, a high running time of a single **MIP** solution would make the approach impractical.

1.3 Active Learning Accuracy

In Fig. 1 we can see that using **MIP** formulations indeed helps improve the AL results, compared to a more basic method Minimum Spanning Tree with Pruning [2] (**MSTP**), as it produces more accurate reconstructions and thus we can more reliably detect mistakes. This is visible especially in case of *Blood Vessels*, which in reality can form loops. Those can be reconstructed using MinSubgraph **MIP**, but not with **MSTP**.

1.4 Hardness

In this section we argue that our problems are extremely unlikely to be solvable in polynomial time. This makes solving **MIP** formulations using state-of-the-art solvers one of the most natural and efficient methods available.

Proposition 2. *The problems MinTree and MinSubgraph are NP-complete.*

Proof. Clearly both are in NP. We will show an NP-hardness reduction from the Steiner tree problem in graphs (STP), which is a well-known NP-hard problem. An instance of STP consists of a graph $G = (V, E)$ with weights on edges $w : E \to \mathbb{R}_+$ and a set of terminal vertices $T \subseteq V$. The objective is to find a

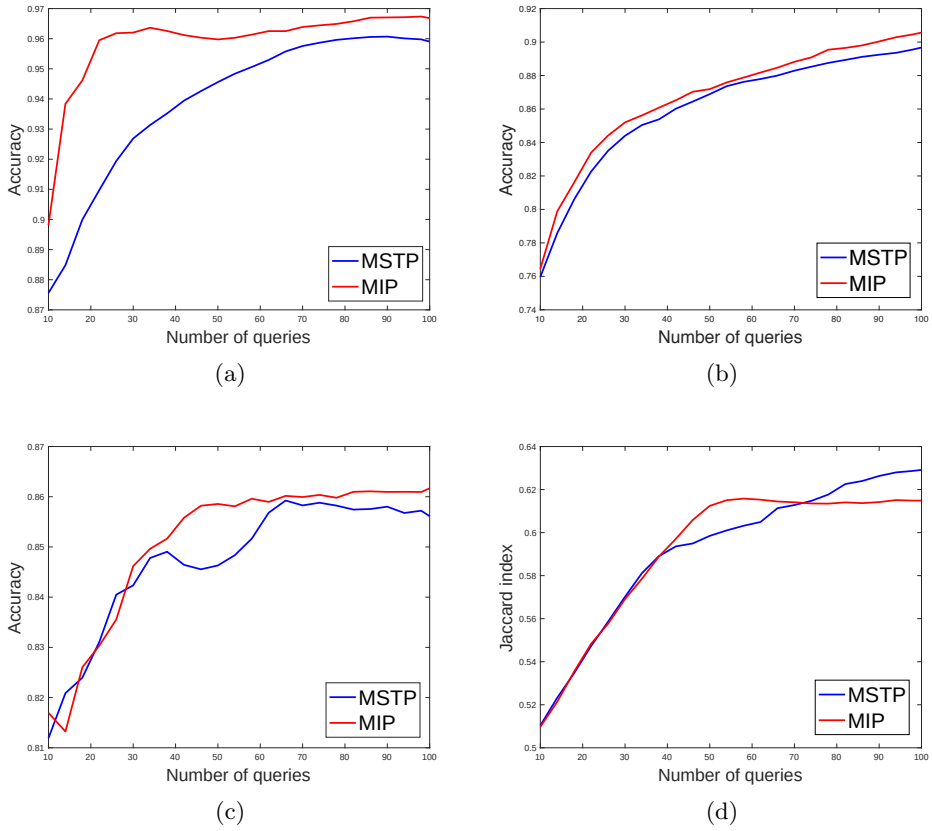
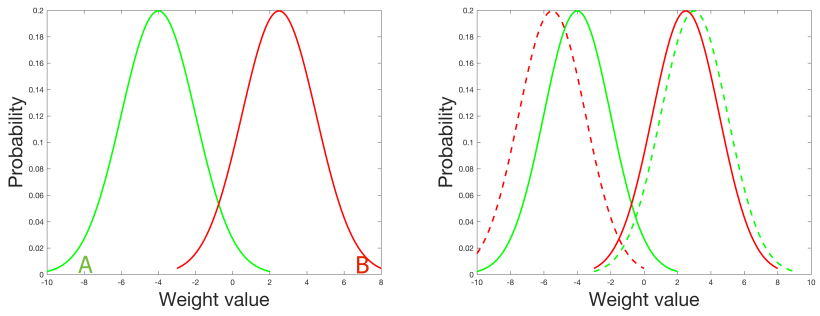


Fig. 1: Comparison of our AL strategy when using **MSTP** and **MIP**. (a) *Blood Vessels*. (b) *Axons*. (c) *Brightfield Neurons*. (d) *Olfactory Projection Fibers*. In all cases using **MIP** facilitates Active Learning compared to **MSTP**, with the exception of the right-hand side of (d): it is a comparatively easy case and the **MIP** delineation stops changing faster than the **MSTP** reconstruction, after which point error-based queries are no longer informative.

minimum-weight tree in G which connects the set T. To obtain an instance of MinTree (or MinSubgraph) from STP, we do the following for each $t \in T$: adjoin a new vertex t' to t using a new edge (t, t') of weight $-M$, where M is a very large weight (say $M = 1 + \sum_{e \in E} |w(e)|$). Then set the root r to be any of these new vertices.

To see that an optimal solution of the MinTree instance corresponds to an optimal solution of the STP instance, note that the former must necessarily contain all the new edges (as we set their weight to be so low that it makes sense to take them even if it requires us to also take many positive-weight edges). Since the MinTree solution must be connected, it will therefore connect all the terminal vertices; removing the new edges from the MinTree solution gives an optimal STP solution. (The same reduction also works for MinSubgraph, since the weights of all original edges are positive and thus the optimal solution for MinSubgraph is the same as the optimal solution for MinTree.)

2 Changing Weights



(a) (b)

Fig. 2: (a) Two Gaussian distributions corresponding to positive (green) and negative (red) classes of edges. (b) The effect of weight transformation; the original distributions are drawn with solid lines, while the corresponding distributions after the transformation are drawn with dashed lines. The described transformation causes the "swapping" of distributions corresponding to the two classes.

As described in Section 2.2, we alter the weight of each edge to measure its influence on the delineation and to detect possible mistakes. In practice, the weights of positive-class edges tend to follow a Gaussian distribution with a negative mean and variance such that few of them have positive values, as shown in Fig. 2(a). Similarly, negative edges follow a Gaussian distribution with positive mean and few of them being negative. As a result, weights of most of the mistaken edges have small absolute values.

In order for our delineation-change metric of Section 2.2 to be informative, we must ensure that attention-worthy edges (probable mistakes) have high values of Δc_i. One possible transformation would be to simply flip the sign of the weight (implying assigning it to the opposite class). However, many of the mistakes with $|w_i| \approx 0$ could be omitted due to smaller values of Δc_i compared to edges with weights of higher absolute value, which are much less likely to be mistakes. Consequently, we also have to increase the absolute value of most likely mistakes. The above requirements can be satisfied with the following transformation:

$$w_i' = \begin{cases} A + w_i & \text{if } w_i > 0, \\ B + w_i & \text{if } w_i < 0. \end{cases} \tag{6}$$

This transformation is equivalent to swapping the distributions corresponding to positive and negative edges, as shown in Fig 2(b).

References

1. Blum, C., Calvo, B.: A matheuristic for the minimum weight rooted arborescence problem. Journal of Heuristics 21(4), 479–499 (2015)
2. Gonzalez, G., Fleuret, F., Fua, P.: Automated Delineation of Dendritic Networks in Noisy Image Stacks. In: ECCV. pp. 214–227 (October 2008)
3. Turetken, E., Benmansour, F., Andres, B., Glowacki, P., Pfister, H., Fua, P.: Reconstructing Curvilinear Networks Using Path Classifiers and Integer Programming. PAMI (2016)