UMich, Summer 18

ATSP Paper

Ansh Nagda, Harsh Parekh

1 Laminarly Weighted Instances

Definition 1.1. Given a graph, let $\boldsymbol{\delta(U, V)}$ be the set of all edges from U to V. Let $\boldsymbol{\delta^+(U)}$ and $\boldsymbol{\delta^-(V)}$ be the set of outgoing and incoming edges from U respectively. Let $\boldsymbol{\delta(U)} := \delta^+(U) \cup \delta^-(U)$.

Definition 1.2. Given an edge-weighted ATSP instance (G,w), consider the following LPs:

LP(G,w):

$$\begin{aligned} \min \quad & \sum_{e \in E} w_e \cdot x_e & \\ s.t. \quad & x(\delta^+(v)) = x(\delta^-(v)) & v \in V \\ & x(\delta(S)) \geq 2 & \emptyset \neq S \subseteq V \\ & x \geq 0 & \end{aligned}$$

DUAL(G,w):

$$\begin{aligned} \max \quad & \sum_{S \subseteq V} 2 \cdot y_S & \\ s.t. \quad & \sum_{S:(u,v) \in \delta(S)} y_S + \alpha_u - \alpha_v \leq w(u,v) & (u,v) \in E \\ & y \geq 0 & \end{aligned}$$

Definition 1.3. (G, L, x, y) is called a **laminarly weighted instance** if L forms a laminar family of subsets of V and the following hold:

1. x is a solution to $LP(G, 0)$ and $x > 0$
2. y is a solution to $DUAL(G, 0)$, and $y_S > 0 \iff S \in L$
3. The primal constraint corresponding to each $S \in L$ is tight

Sets in L are called laminar sets.

If w is the weight function induced by y (i.e the weight of an edge equals the total y-value of the laminar sets it crosses), and if we set α to 0, then complementary slackness tells us that x and (y, α) are optimal solutions to LP(G,w) and DUAL(G,w) respectively.

Claim. *Given a graph G, we can find an optimal dual solution (α, y) such that the support of y forms a laminar family.*

This can be shown using an "uncrossing" argument - given a dual solution, if the support of y is not laminar, then there must exist two sets in its support, say X and Y, such that $X \setminus Y$ and $Y \setminus X$ are nonempty. WLOG, assume that $y_X \leq y_Y$. Then y_X amount of value can be redistributed from X and Y to $X \setminus Y$ and $Y \setminus X$ without disturbing feasibility or optimality. Note that X will no longer be in the support of y, so X and Y are now "uncrossed". Applying this procedure repeatedly (and choosing X and Y carefully), we can obtain a solution with laminar support in polynomial time.

Theorem 1.1. *An α-approximation algorithm for ATSP on laminarly weighted instances implies an α-approximation algorithm for ATSP on general instances.*

Proof. We are given $I = (G, w)$ as input to general ATSP. Let x be a primal optimal solution to LP(G,w) and (y, α) be the dual optimal solution to DUAL(G,w) guaranteed by the above claim. Let E' be the support of x, and L be the support of y. Then complimentary slackness implies that $I' = ((V, E'), L, x, y)$ is a laminarly weighted instance.

Note that (by complimentary slackness) the induced weight on I' is $w'(u, v) = w(u, v) + \alpha_v - \alpha_u$. Although these weights are different from our original weights, the weight of every circulation in I' is the same as its weight in I (this can be seen using a cycle decomposition of x). In particular, the weight of a tour remains the same, and the optimal LP-value remains the same. This leads to the final algorithm - simply run the α-approximation algorithm for laminarly weighted instances on I' and return the resulting tour, say T. The analysis is easy:

$$w(T) = w'(T) \leq \alpha \cdot \text{value}(I') = \alpha \cdot \text{value}(I)$$

□

We have now reduced ATSP to laminarly weighted instances.

2 Paths in Laminar Sets

Consider any $S \in L$. Let $S_1, \ldots, S_k$ be the strongly connected components of S in topologically sorted order. We know that

1. $x(\delta^-(S_1)) \geq 1$
2. $x(\delta^-(S)) = 1$
3. Every edge into S_1 is also an edge into S

It follows that $x(\delta^-(S_1)) = 1$. Using induction along with a similar argument, one can also see that for all i, $x(\delta^-(S_i)) = x(\delta^+(S_i)) = 1$, and the only edges from one SCC to another are edges from S_i to S_{i+1}. As a consequence, there always exists a path from $u \in S_{in}$ to $v \in S_{out}$ within S. This structure allows us to "shortcut" paths that enter or exit S multiple times - we can replace the entire portion of the path between the first time it enters S and the last time it exits S with a path completely inside S.

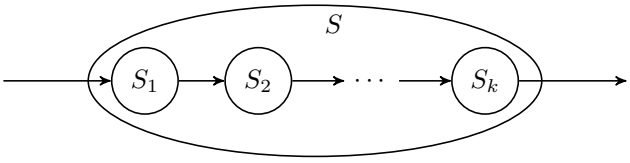

Figure 1: Paths traverse through a tight set S by entering at S_1, visiting each connected component in S exactly once before leaving from S_l.

Definition 2.1. Let $\textbf{value}_{\boldsymbol{I}}(\boldsymbol{S})$ be the LP value of I stored within S, and let $\textbf{value}(\boldsymbol{I})$ be the total LP value of I. Formally,

$$\textbf{value}_{\boldsymbol{I}}(\boldsymbol{S}) := \sum_{R \in L : R \subsetneq S} 2 \cdot y_R \qquad \textbf{value}(\boldsymbol{I}) := \text{value}_I(V)$$

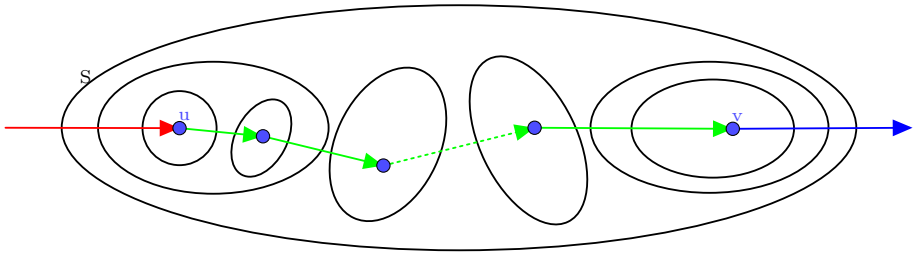

Figure 2: The Red edge is paid for by $\sum_{R \in L: u \in R \subsetneq S} y_R$, the shortest $u - v$ path is shown in green and, the blue edge is paid for by $\sum_{R \in L: v \in R \subsetneq S} y_R$; summing to $D_S(u, v)$.

Definition 2.2. Let $\boldsymbol{d_S(u, v)}$ be the weight of the shortest path from u to v inside S. Also, let

$$\boldsymbol{D_S(u, v)} := \sum_{R \in L: u \in R \subsetneq S} y_R + d_s(u, v) + \sum_{R \in L: v \in R \subsetneq S} y_R \tag{1}$$

Furthermore, let $D_{\max}(S) = \max_{u \in S_{in}, v \in S_{out}} D_S(u, v)$.

Intuitively, $D_s(u, v)$ is the total cost of entering u from outside S, a shortest path from u to v, and exiting from v to outside S. See Figure 2.

Lemma 2.1. *$S \subseteq V$ be any set such that $L \cup \{S\}$ is a laminar family. Then*

(1) $d_S(u, v) \leq \text{value}_I(S)$ if there exists a $u - v$ path inside S.

(2) $D_S(u, v) \leq \text{value}_I(S)$ if $u \in S_{in}$ or $v \in S_{out}$

Proof. The main idea for the proof for (1) is that a path that enters some laminar set multiple times can be shortcut to just enter that laminar set once. After repeatedly shortcutting until each laminar set is entered at most once, the cost of the path that each set R is responsible for is at most $2 \cdot y_R$ (one for entering R, and one for exiting R), and (1) follows from the definition of $\text{value}_I(S)$.

The idea for (2) is similar. One can show (using a case-by-case analysis) that for all $R \in L$ with $R \subsetneq S$, after shortcutting the path, at most two of the following are true:

1. $u \in R$
2. $v \in R$
3. The path enters R
4. The path exits R

In other words, every laminar set $R \subsetneq S$ appears at most twice in the right hand side of Equation 1. (2) follows from the definition of $\text{value}_I(S)$. □

This gives us useful bounds on costs our algorithm will be incurring.

3 Operations on Laminar Sets

Contracting

Definition 3.1. The instance $\boldsymbol{\mathcal{I}/S} = (G', \mathcal{L}', x', y')$ obtained by contracting $S \in \mathcal{L}$ is called the contracted instance. It is obtained as follows:

- G' is obtained by contracting the set $S \in \mathcal{L}$ to a single vertex s, keeping parallel edges but removing self-loops.

- For each edge $e' \in E(G')$, $x'(e') = x(e)$ where e is the pre-image of e' in G.
- $\mathcal{L}' = \{(R \setminus S) \cup \{s\} : R \in \mathcal{L}, S \subseteq R\} \cup \{R : R \in \mathcal{L}, S \cap R = \emptyset\}$
  i.e. contract all occurrences of set S into a single vertex s.
- $y'_s = y_S + \frac{1}{2}D_{\max}(S)$. For all other sets R', $y'_{R'} = y_R$ where, R is the pre-image of R'.

Fact 3.1. $\text{value}(\mathcal{I}/S) = \text{value}(\mathcal{I}) - (\text{value}_{\mathcal{I}}(S) - D_{\max}(S)) \leqslant \text{value}(\mathcal{I})$.

Proof. $\text{value}(\mathcal{I}/S) - \text{value}(\mathcal{I}) = D_{\max}(S) - \text{value}_{\mathcal{I}}(S)$ and from Lemma 2.1 □

Now that we have defined contracted instance, we would like to convert any tour T in $\mathcal{I}/S$ to a subtour in $\mathcal{I}$. We call this procedure Lift.

Definition 3.2. For a tour T in $\mathcal{I}/S$, we define its lift to be the subtour in $\mathcal{I}$ formed by replacing every consecutive pair $(u_{in}, s), (s, v_{out})$ of incoming and outgoing edges by their pre-images $(u_{in}, v_{in}), (u_{out}, v_{out})$ and the shortest $v_{in} - u_{out}$ path in S.

Lemma 3.1. *Let T be a tour in $\mathcal{I}/S$. Then the lift F of T satisfies $w_{\mathcal{I}}(F) \leqslant w_{\mathcal{I}/S}(T)$.*

Proof. Let T cross the set $R \in \mathcal{L}$, α_R times. Let $\left(v_{in}^{(1)}, u_{out}^{(1)}\right), \left(v_{in}^{(2)}, u_{out}^{(2)}\right), \ldots, \left(v_{in}^{(\alpha_S)}, u_{out}^{(\alpha_S)}\right)$ be the pairs of consecutive incoming and outgoing vertices for S.

$$\begin{aligned} w_{\mathcal{I}}(F) &= \sum_{R \in \mathcal{L}: R \not\subseteq S} \alpha_R y_R + \sum_{i=1}^{\alpha_S} \left(2y_S + D_S\left(v_{in}^{(i)}, u_{out}^{(i)}\right)\right) \\ &\leqslant \sum_{R \in \mathcal{L}: R \not\subseteq S} \alpha_R y_R + \sum_{i=1}^{\alpha_S} 2\left(y_S + \frac{1}{2}D_{\max}(S)\right) = w_{\mathcal{I}/S}(T) \end{aligned}$$

□

Inducing

Definition 3.3. The instance $\boldsymbol{\mathcal{I}[S]} = (G', \mathcal{L}', x', y')$ is called the induced instance by inducing on a tight set S. It is obtained as follows:

- $G' = G/\bar{S}$ where, $\bar{S} = V \setminus S$. Let $\bar{s}$ be the vertex corresponding to set $\bar{S}$.
- For each edge $e' \in E(G')$, $x'(e') = x(e)$ where e is the pre-image of e' in G.
- $\mathcal{L}' = \{R \in \mathcal{L} : R \subsetneq S\} \cup \{\{\bar{s}\}\}$.
- $y'_{\bar{s}} = \text{value}(S)/2$ and for every other set $R \in \mathcal{L}'$ its the same as y_R.

Fact 3.2. $\text{value}(I[S]) = 2\text{value}(S)$

The sub-tour obtained by lifting a tour in contracted instance is guaranteed to be a single component but its not guaranteed to cover all vertices, for this we use a tour in the induced graph to patch in the vertices that are not visited.

Let $S_1, \ldots, S_l$ be the SCC in $S \in \mathcal{L}$, indexed in topological order.

Definition 3.4. For a tour T in $\mathcal{I}[S]$, we define its lift to be the following:
For any strongly connected component $S_i \subseteq S$ we obtain a tour F_i of S_i by taking all the edges in T that are contained within S_i and for every time the T exits S_i from u_{out} and enters the next time to a vertex v_{in}, we add the shortest $u_{out} - v_{in}$ path in S_i. $F = \cup_{i \in [l]} F_i$ is the lift of T.

Definition 3.5. We say that a set $S \in \mathcal{L}$ is contractible with respect to an Eulerian set of egdes $F \subseteq E$ if the lift of any tour of $\mathcal{I}/S$ plus the set F is a tour in $\mathcal{I}$.

We note that F, the lift of a tour in $\mathcal{I}[S]$ makes S contractible as lift of any tour in $\mathcal{I}/S$ will visit each $S_i \in S$ at least once.

Lemma 3.2. *Let T be a tour in $\mathcal{I}[S]$. Then the lift F of T satisfies $w_{\mathcal{I}}(F) \leqslant w_{\mathcal{I}[S]}(T)$.*

Proof. We can show that any connected component S_i agrees with the laminar family $\mathcal{L}'$. With this, we can bound the cost of the shortest $u_{out} - v_{in}$ path in S_i by $\text{value}_{\mathcal{I}[S]}(S_i)$.Also, let k be the number of times T visits $\bar{s}$. Therefore weight of F is

$$\begin{aligned} w_{\mathcal{I}}(F) &= \sum_{i\in[l]} w_{\mathcal{I}} F_i \\ &\leqslant \sum_{i\in[l]} [k\text{value}(S) + w_{\mathcal{I}}(T \cap E(S_i))] \\ &\leqslant 2k\frac{\text{value}(S)}{2} + \sum_{i\in[l]} w_{\mathcal{I}}(T \cap E(S_i)) = w_{\mathcal{I}[S]}(T). \end{aligned}$$

□

4 Reduction to Irreducible Instances

Definition 4.1. Fix some δ. A set $S \in L$ is called **reducible** if $\max_{u\in S_{in}, v\in S_{out}} D_S(u,v) \leq \delta \cdot \text{value}_I(S)$, and **irreducible** otherwise. An instance I is irreducible if every laminar set is irreducible.

By this definition, we see that if we contract a reducible set S, the LP-value goes down by at least $\text{value}_I(S) - \max D_S(u,v) \geq (1-\delta)\cdot \text{value}_I(S)$.

Theorem 4.1. *A ρ-approximation algorithm for ATSP on irreducible instances implies a $\frac{2\rho}{1-\delta}$-approximation algorithm for ATSP on laminarly weighted instances.*

Algorithm 1: LAMINARLY WEIGHTED → IRREDUCIBLE

1. If I is irreducible, simply use the algorithm for irreducible instances
2. $S \leftarrow$ any minimal reducible set of L
3. $T \leftarrow$ tour in $I \setminus S$ found using recursion
4. $T_S \leftarrow$ tour in $I[S]$ found using algorithm for irreducible instances
5. $T' \leftarrow \text{LIFT}(T), T'_S \leftarrow \text{LIFT}(T_S)$
6. **return** $T' \cup T'_S$

Proof. Note that step 4 of the algorithm is valid because $I[S]$ is irreducible - all the laminar sets inside S were already irreducible, and all singleton sets are trivially irreducible, so the new laminar set is also irreducible.
Also note that the base case (i.e I is already irreducible) is trivial.

For the sake of induction, assume that recursing gives us a tour T of I/S such that $w_{I/S}(T) \leq \rho \cdot \text{value}(I/S)$. We know that $w_{I[S]}(T_S) \leq \rho \cdot \text{value}(I[S]) = 2\rho \cdot \text{value}_S(I)$. Now,

$$ALG = w_I(T') + w_I(T'_S) \leq w_{I[S]}(T) + w_{I/S}(T_S) \leq \frac{2\cdot\rho}{1-\delta}\text{value}(I/S) + \rho\cdot\text{value}(I[S])$$

$$\leq \frac{2\cdot\rho}{1-\delta}(\text{value}(I) - (1-\delta)\cdot\text{value}_I(S)) + 2\rho\cdot\text{value}_I(S) = \frac{2\cdot\rho}{1-\delta}\cdot\text{value}(I)$$

□

5 Reduction to Vertebrate Pairs

Definitions

Definition 5.1. We say that a subtour B, is a backbone of the instance $\mathcal{I}$ if for every non-singleton set $S \in \mathcal{L}$. B crosses S i.e. $\delta(S) \cap B \neq \emptyset$.
The pair $(\mathcal{I}, B)$ is called the vertebrate pair.

Definition 5.2. For an instance $\mathcal{I}$ we say that a subtour B a quasi-backbone if

$$2 \sum_{S \in \mathcal{L}^*} y_S \leqslant (1 - \delta)\,\text{value}(\mathcal{I}) \tag{2}$$

where $\mathcal{L}^*$ is the set of all $S \in \mathcal{L}$ that are not crossed by B.

Here we note that a quasi-backbone may not be a backbone and vice-versa especially if majority of value($\mathcal{I}$) is concentrated on singleton laminar sets in $\mathcal{L}$.[1]

Finding a Quasi-Backbone

Lemma 5.1. *Given an irreducible instance $\mathcal{I}$, we can construct a quasi-backbone B such that it crosses all maximal non-singleton sets in $\mathcal{L}$ and $w(B) \leqslant (\alpha_{NW} + 3)\text{value}(\mathcal{I})$.*

Proof. Let $\mathcal{L}_{\max}$ be the set of all maximal sets in $\mathcal{L}$ and $\mathcal{I}'$ be the instance obtained by contracting each set in $\mathcal{L}_{\max}$. The instance $\mathcal{I}'$ is node weighted and to find a subtour B' in $\mathcal{I}$ we find a tour T in $\mathcal{I}'$ using α_{NW} approx algorithm for Node-Weighted ATSP and lift it. From Lemma 3.1 and Fact 3.1 we know that $w_{\mathcal{I}}(B') \leqslant w_{\mathcal{I}'}(T) \leqslant \alpha_{NW}\text{value}(\mathcal{I}') \leqslant \alpha_{NW}\text{value}(\mathcal{I})$ and that B' crosses all maximal non-singleton laminar sets but, we still need to ensure that it crosses at least δvalue($\mathcal{I}$) of sets. For this we modify B' to get our quasi-backbone B as follows:

- Let u^S, v^S be the first entry and exit vertex for S in B' respectively and, $u^S_{\max}, v^S_{\max}$ be the vertices corresponding to $D_{\max}(S)$.
- Replace the $u^S - v^S$ path by the path P composed of the shortest paths[2] from u^S to $u^S_{\max}$ to $v^S_{\max}$ to v^S, from Lemma 2.1 $w(P) \leqslant 3\text{value}_{\mathcal{I}}(S)$

As S is irreducible, with this modification we automatically satisfy our last requirement while only increasing the weight by at most 3value($\mathcal{I}$). □

Obtaining a Vertebrate Pair

Now we would like to find a way to convert a quasi-backbone into a backbone and then use it, along with an approximation algorithm for ATSP on vertebrate pair to approximate ATSP on irreducible instances.
Theorem 5.1. *Given an irreducible instance $\mathcal{I}$, Algorithm 2 returns a Tour of $\mathcal{I}$ of cost no more than ρvalue($\mathcal{I}$) where, $\rho = \frac{\kappa + \eta(\alpha_{NW}+3)}{1-2(1-\delta)}$.*

Proof.

$$w(F) \leqslant w(T') \leqslant \kappa\text{value}(\mathcal{I}) + \eta(\alpha_{NW} + 3)\text{value}(\mathcal{I})$$
$$w(F) \leqslant (\kappa + \eta(\alpha_{NW} + 3))\,\text{value}(\mathcal{I})$$

[1] While the backbones we construct will also be quasi-backbones but our future reductions do not use this fact.
[2] These paths exists as $u^S, u^S_{\max} \in S_1$ and $v^S, v^S_{\max} \in S_l$.

Algorithm 2: Irreducible ATSP (A_{irr})

Input: Irreducible instance $\mathcal{I} = (G, \mathcal{L}, x, y)$
Algorithm A for ATSP on vertebrate pair that returns a tour of cost at most $\kappa \text{value}(\mathcal{I}') + \eta w(B)$.

1 Use Lemma 5.1 to obtain a quasi-backbone B.
2 $L^*_{\max} \leftarrow$ all non-singleton maximal sets in $\mathcal{L}^*$.
3 **for** *each* $S \in \mathcal{L}^*_{\max}$ **do**
4 | Find $T_S \leftarrow A_{irr}(\mathcal{I}[S])$.
5 | Use T_S to find F_S for which S is contractible.
6 **end**
7 $\mathcal{I}' =$ Instance obtained by contracting every $S \in \mathcal{L}^*_{\max}$.
8 $T' \leftarrow A(\mathcal{I}', B)$ and $T \leftarrow$ lift of T'.
9 **return** $T \cup \left(\cup_{S \in \mathcal{L}^*_{\max}} F_S\right)$

$$w(F_S) \leqslant w_{\mathcal{I}[S]}(T_S) \leqslant \rho \text{value}(\mathcal{I}[S])$$
$$w(F_S) \leqslant 2\rho \text{value}_{\mathcal{I}}(S)$$
$$\bigcup_{S \in \mathcal{L}^*_{\max}} w(F_S) \leqslant 2\rho\,(1-\delta)\,\text{value}(\mathcal{I}) \quad - \text{ As all } S \text{ are disjoint and, from 2 as } B \text{ was a quasi-backbone.}$$

$$\begin{aligned} Cost(\mathcal{I}) &= w(F) + \cup_{S \in \mathcal{L}^*_{\max}} w(F_S) \\ &\leqslant (\kappa + \eta(\alpha_{NW} + 3))\,\text{value}(\mathcal{I}) + 2\rho\,(1-\delta)\,\text{value}(\mathcal{I}) \qquad \text{Substituting } \rho \text{ and } \delta = 0.75. \\ Cost(\mathcal{I}) &\leqslant \rho \text{value}(\mathcal{I}) \end{aligned}$$

□

6 Algorithm for Vertebrate Pairs

The Local Connectivity ATSP Problem

Fix some function $\text{lb} : 2^V \to \mathbb{R}^+$ that obeys the following properties:

1. $\text{lb}(S) = \sum_{v \in S} \text{lb}(\{v\})$ for all $S \subseteq V$
2. $\text{lb}(V) = \text{value}(I)$

One can think of lb as being defined by distributing the LP value among the vertices in some way.

Problem Statement.
Input: I (the instance) and $V_1, \ldots, V_k$ (a partitioning of V) such that V_i is strongly connected.
Output: F, an eulerian set of edges such that for every i, F exits (and hence enters) V_i at least once.

Definition 6.1. An algorithm for local connectivity ATSP is called **α-light** if it outputs F such that for each strongly connected component of F, say $\tilde{G}$, $w(\tilde{G}) \leq \alpha \cdot \text{lb}(\tilde{G})$.

Theorem 6.1 (Svensson '15)**.** *An α-light algorithm for local connectivity ATSP implies a $O(\alpha)$-approximation algorithm for ATSP.*

The paper describes an $O(1)$-light algorithm for local connectivity on vertebrate pairs, which gives us an $O(1)$-approximation algorithm for ATSP on vertebrate pairs, and hence an $O(1)$-approximation algorithm for general ATSP.

We will also assume that $w(B) \leq O(1) \cdot \text{value}(I)$, since in the previous reduction, this inequality is always true.

Definition 6.2. For this algorithm, we will define lb as

$$\textbf{lb(v)} := \begin{cases} \dfrac{\text{value}(I)}{|V(B)|} & \text{if } v \in B \\ 2 \cdot y_v & \text{otherwise} \end{cases}$$

Note that this definition does not guarantee that $\text{lb}(V) = \text{value}(I)$, but $\text{lb}(V)$ is always a constant factor away from $\text{value}(I)$, so we can work with these lb values and suffer a constant factor loss in the approximation ratio.

To make corner cases easier, we add V to $L_{\geq 2}$ for the rest of this write up.

The following is the main technical lemma of the paper.

Lemma 6.1 (Lemma 7.3 in the paper)**.** *Given $U_1, \ldots, U_l$ disjoint subsets of $V \setminus V(B)$ that are strongly connected in G, such that for each $S \in L_{\geq 2}$, either $U_i \in S$ or U_i is disjoint from S, we can find eulerian $F \subseteq E$ such that:*

(1) $w(F) \leq 3 \cdot \text{value}(I)$

(2) $|\delta_F^-(U_i)| \geq 1$ *for all* i

(3) $|\delta_F^-(v)| \leq 4$ *for all* v *such that* $x(\delta^-(v)) = 1$

(4) Any subtour in F that crosses a set in $L_{\geq 2}$ visits a vertex of the backbone

We will use the lemma to prove the following theorem, and prove the lemma later on.

Theorem 6.2. *There is an $O(1)$-light algorithm for local connectivity ATSP on vertebrate pairs.*

Proof. We receive (I, B) and $V_1, \ldots, V_k$ as input. We output $F^* = B \cup P \cup F$, where P and F are defined as:

- P: If B exists entirely within some V_i, then we pick $u \in B$ and $v \notin V_i$, and set P to be the shortest cycle containing u and v. Otherwise we keep P empty.
- F: WLOG, let $V_1, \ldots, V_l$ be the partitions that are disjoint from B. Now let V_i' be the intersection of V_i with a minimal set in $L_{\geq 2}$ that V_i intersects. Let U_i be the first strongly connected component of V_i' (in the topological order). Set F to be the eulerian set guaranteed by running the algorithm of Lemma 6.1 on $U_1, \ldots, U_l$.

Now we need to verify that $B \cup P \cup F$ is feasible, and that it is $O(1)$-light.
For each V_i, there are three possibilities:

(a) V_i completely contains the backbone. The set P guarantees that F^* enters V_i at least once.

(b) V_i partially contains the backbone. In this case, B trivially crosses V_i, so F^* must enter V_i.

(c) V_i is disjoint from the backbone. In this case, (2) tells us that some edge (u, v) must enter U_i. $u \notin V_i'$ since U_i is the source component of V_i'. So either $u \in S \setminus V_i$ or $u \notin S$. If the former is true, then (u, v) enters V_i from outside. If the latter is true, then (u, v) crosses S, and guarantee (4) of Lemma 6.1 tells us that the subtour containing (u, v) visits the backbone, which is outside V_i.

In any case, F^* crosses each V_i, so it is indeed feasible.

Now consider any strongly connected component of F^*, say $\tilde{G}$. We would like to bound $w(\tilde{(G)})$ by $O(\text{lb}(\tilde{G}))$. There are two possibilities for $\tilde{G}$:

(a) $\tilde{G}$ contains the backbone. So

$$w(\tilde{G}) \leq w(B) + w(P) + w(F) \leq O(\text{value}(I)) + \text{value}(I) + 3 \cdot \text{value}(I) = O(\text{value}(I))$$

$$\text{lb}(\tilde{G}) \geq \text{lb}(V(B)) = \text{value}(I)$$

And it follows that $w(\tilde{G}) \leq O(\text{lb}(\tilde{G}))$.

(b) $\tilde{G}$ is disjoint from the backbone. In this case, (4) guarantees that $\tilde{G}$ did not cross any set in $L_{\geq 2}$, so the only sets it might have crossed are singletons. Additionally, (3) guarantees that it crosses each singleton at most four times.

$$w(\tilde{G}) \leq \sum_{v \in V(\tilde{G})} 4 \cdot 2 \cdot y_v = 4 \cdot \text{lb}(\tilde{G})$$

□

Proof of Lemma 6.1: The final ingredient

Order the laminar sets in $L_{\geq 2}$ by size so that $|S_1| \leq |S_2| \leq \cdots \leq |S_l| = |V|$. For each $v \in V$, define $level(v)$ to be the index of the smallest set it is contained in.

Definition 6.3. An edge e is called a **forward edge**, **backward edge**, or **neutral edge** if $e \in E_f$, $e \in E_b$, or $e \in E_n$ respectively, where E_f, E_b, and E_n are defined as:

- $\boldsymbol{E_f} := \{(u, v) \in E \mid level(u) > level(v)\}$
- $\boldsymbol{E_b} := \{u, v) \in E \mid level(u) < level(v)\}$
- $\boldsymbol{E_n} := E \setminus (E_f \cup E_b)$

Definition 6.4. Call $\boldsymbol{G_{sp}}$ the **split graph**, where we define it as follows: for each $v \in V$, create v^0 and v^1 in $V(G_{sp})$. Create the following edges along with weights $\boldsymbol{w_{sp}}$:

- (v^0, v^1) for all $v \in V$ with weight 0. Call these edges $0-1$ edges.
- (v^1, v^0) for all $v \in V(B)$ with weight 0. Call these edges $1-0$ edges.
- (u^1, v^1) for all $(u, v) \in E_f \cup E_n$ with weight $w(u, v)$. Call these edges $1-1$ edges.
- (u^0, v^0) for all $(u, v) \in E_b \cup E_n$ with weight $w(u, v)$. Call these edges $0-0$ edges.

Also for any set $S \subseteq V$, let S^{sp} be its image in G_{sp}.

Consider the lift of some subtour in G_{sp} that crosses at least one set in $L_{\geq 2}$. Let S be the smallest set in $L_{\geq 2}$ that the subtour crosses. Then the edge of the subtour going into S must be a forward edge (i.e. a $1-1$ edge) by definition, and the edge going out of S must be a backward edge (i.e. a $0-0$ edge). This means at some point, the subtour must use a $1-0$ edge, which it can only do if it crosses the backbone.

So if we restrict ourself to lifts of subtours of G_{sp} (instead of all subtours of G), guarantee (4) will always be true.

Lemma 6.2. *We can find x_{sp}, an eulerian vector on G_{sp} that the image of x_{sp} in G is x.*

Proof. If we want the image of x_{sp} to be x, we must necessarily have the following conditions:

(a) $x_{sp}(u^0, v^0) = 0$, for all $(u, v) \in E_f$, since there is no (u^0, v^0) edge in G_{sp}.

(b) $x_{sp}(u^1, v^1) = 0$, for all $(u, v) \in E_b$, since there is no (u^1, v^1) edge in G_{sp}.

(c) $x_{sp}(u^0, v^0) + x_{sp}(u^1, v^1) = x(u, v)$ for all $(u, v) \in E$, since the image of x_{sp} in G must be x.

After defining some x_{sp} values on the $0-0$ and $1-1$ edges, there will be some "potential" on each vertex (i.e. the net outgoing flow), and since x was eulerian, the potential of the 0-copy of a vertex, say v, is exactly the negative potential of the 1-copy. If the potential of the 1-copy is positive, v^0 and v^1 can be brought down to zero potential by routing that amount of flow through (v^0, v^1). But if it is negative, this can only be equalized using a (v^1, v^0) edge, which is impossible unless $v \in V(B)$. In other words, if we want x_{sp} to be eulerian, it must also fulfil the following condition:

(d) $x_{sp}(\delta^-(v_1) \setminus (v^0, v^1)) \geq x_{sp}(\delta^+(v_1) \setminus (v^0, v^1))$ for all $v \notin V(B)$.

It can easily be seen that (a), (b), (c), and (d) are also sufficient for some nonnegative x_{sp} to fulfil the guarantees of Lemma 6.2. This gives rise to the following LP and its dual where f can be interpreted as x_{sp} restricted to $1-1$ edges:

$$\begin{aligned} \max \quad & \sum_{e \in E_f} f(e) & \\ s.t. \quad & f(\delta^+(v)) \geq f(\delta^-(v)) & v \notin V(B) \\ & f(e) = 0 & e \in E_b \\ & 0 \leq f(e) \leq x(e) & e \in E \end{aligned}$$

$$\begin{aligned} \min \quad & \sum_{e \in E_f \cup E_n} x(e)z(e) & \\ s.t. \quad & \pi_v - \pi_u + z(u,v) \geq 1 & (u,v) \in E_f \\ & \pi_v - \pi_u + z(u,v) \geq 0 & (u,v) \in E_n \\ & \pi_v = 0 & v \in V(B) \\ & \pi, z \geq 0 & \end{aligned}$$

Once we have some primal feasible f, we can extend it by defining x_{sp} as f on $1-1$ edges, $x-f$ on $0-0$ edges, and eliminating potential differences between 0 and 1 copies using $0-1$ and $1-0$ edges. The first inequality of the LP guarantees (d) by definition. The second inequality guarantees (b) by definition. And the way we extend f to $0-0$ edges guarantees (c). Now if we know that the primal optimal value is equal to $\sum_{e \in E_f} x_e$, then it guarantees (a) as well. The following claim proves this, so (a), (b), (c), and (d) must all hold. □

Claim 6.1. *The primal optimal value is equal to* $\sum_{e \in E_f} x_e$.

Proof. Note that if there were a dual optimal solution where $\pi = 0$, then the dual optimal (and hence primal optimal) value would be precisely $\sum_{e \in E_F} 1 \cdot x_e$. So consider any dual optimal solution, say (z, π). Our goal is to convert it into a dual optimal solution whose π has strictly smaller support. Running this procedure repeatedly proves that there is a dual optimal where $\pi = 0$.

Let T be the set of vertices in support(π) with the smallest level, and let S be the set corresponding to that level. Also, let $F = \delta(V \setminus S, T)$, and $F' = \delta(T, S \setminus T)$.
To start off, let's decrement the π-values of all vertices in T by the smallest π-value of any vertex in T, say ε. Now π has a smaller support, but it may not be feasible. In order to correct this, we increase $z(e)$ by ε for all $e \in F$. Now the solution might not be optimal. In order to correct this, we decrease $z(e)$ by ε for all $e \in F'$. In order to see why the new dual is feasible, note that an edge (u, v) can be of the following types:

Case 1: $v \notin T$. Since we only decreased the π-value of vertices in T (and left all other vertices unchanged), the left hand side of each of the first two dual inequalities can only increase. So these inequalities are still satisfied.

Case 2: $u, v \in T$. In this case, the decrease in π_v is matched by the increase in $-\pi_u$, so the inequalities are still satisfied.

Case 3: $u \notin S, v \in T$. In this case, the decrease in π_v is matched by the increase in $z(u, v)$, so the inequalities are still satisfied.

Case 4: $u \in S \setminus T, v \in T$. In this case, (u, v) cannot be a forward edge, since if it were, then T would intersect a set in $L_{\geq 2}$ which was smaller than S. But we defined S to be the smallest such set. So we only consider the second inequality, which must be satisfied, since π_u was already 0 and π_v is nonnegative.

The inequality $\pi \geq 0$ must be satisfied simply because we reduced π_v for $v \in T$ by $\min_{u \in T} \pi_u$. Also, since all edges (u, v) in F' are either neutral or forward, we originally satisfied $\pi_v - \pi_u + z(u, v) \geq 0$. Since $u \in T$ and $v \in S \setminus T$, we must have that $0 - \varepsilon + z(u, v) \geq 0$, that is, $z(u, v) \geq \varepsilon$. So the inequality $z \geq 0$ must be satisfied since we only reduced z-value in F', and for all edges in F', we already had ε z-value to begin with.

Now we will see that the new dual value is still optimal. The decrease in dual objective value was precisely $\varepsilon(x(F') - x(F \cap (E_f \cup E_n)) \geq \varepsilon(x(F') - x(F))$. If we can show that this is positive, then the new dual value is still optimal. The proof relies on the fact that S is either an original laminar set or V, so $x(\delta^-(S)) \leq 1$. Now,

$$1 \leq x(\delta^-(S \setminus T)) = \underbrace{x(\delta^-(S)) - x(F)}_{\text{edges from outside } S} + \underbrace{x(F')}_{\text{edges from } T} \leq 1 - x(F) + x(F')$$

It follows that $x(F) \leq x(F')$, so the updated dual solution is still optimal, and there exists a dual optimal solution with $\pi = 0$. □

At this point, it can be shown that (with a few modifications) rounding x_{sp} to an integral solution and taking its image in G would fulfil conditions (1), (3), and (4) of Lemma 6.1. To fulfil (2), the split graph is further modified taking into account the U_is.

Definition 6.5. The modified split graph $\boldsymbol{G'_{sp}}$ is created by modifying G_{sp} according to the following procedure for every U_i:

- Add a new vertex called a_i.
- Pick X_i^- to be a subset of incoming edges to U_i^{sp} such that $x_{sp}(X_i^-) = 1/2$ and X_i^- consists of either solely $1-1$ edges or solely $0-0$ edges. This must be possible since the total x-value coming into U_i^{sp} is at least 1.
- Let X_i^+ be a subset of edges leaving U_i^{sp} that entered through some edge in X_i^-. This can be found using a path decomposition of x_{sp} on U_i and following each edge of X_i^- along its paths. Then $x_{sp}(X_i^+) = 1/2$.
- Redirect every edge in X_i^- to a_i, and every edge in X_i^+ from a_i. In other words, $(u, v) \in X_i^-$ becomes (u, a_i) and $(u, v) \in X_i^+$ becomes (a_i, v). Also for each $u - v$ path we considered in the path decomposition (where $u \in X_i^-, v \in X_i^+$), reroute the x_{sp} flow to the path $(u, a_i), (a_i, v)$. Call this modified flow x'_{sp}. Note that x'_{sp} is still a circulation, and the original paths we considered are effectively deleted in G'_{sp}.