1 Proof for Theorem 2.1

Theorem 2.1 The budget constrained density weighted connectivity optimization problem is NP-hard. The density weighted connectivity function is neither submodular nor supermodular.

Proof. First, we prove that the BCDWC-Opt problem is NP-hard. It can be proved by a reduction from the Knapsack problem. In a Knapsack problem, we have $n$ items, each having a cost $c_i$ and a value $val_i$. Given a budget $B$, we select a subset of items to maximize their total value with their total cost less than $B$.

To build an instance of our BCDWC-Opt problem to model the Knapsack problem, we build a graph $G = (V,E)$ with $2n$ nodes indexed by $(i_j)_{i=1:n,j=0:1}$ with all $D_{i_0} = 0$. Edges only exist between $i_0$ and $i_1$ or $E = \{(i_0, i_1) | \forall i = 1:n\}$. Define $D_{i_0}, r_{i_0,i_1}, r_{i_1,i_0}$ arbitrarily as long as $D_{i_0}(\exp(-r_{i_0,i_1}) - \exp(-r_{i_1,i_0})) = val_i$. The cost of protecting edge $(i_0, i_1)$ is $c_i$. Also, we set $p_0 = 1$ and $\alpha = 1$.

That is, in this BCDWC-Opt problem, by protecting an edge with cost $c_i$, we can increase the objective by and only by $val_i$. So, with a budget $B$, the optimal strategy of edge protection corresponds to the optimal solution of the Knapsack problem.

Now, we use an example in Fig. 1 to show that the objective function $DWC(\cdot)$ is not submodular. We have

- $DWC(\emptyset) = 0$
- $DWC(\{e_1\}) = 100 \times 0.9$
- $DWC(\{e_1,e_2\}) = 2(100 \times 0.9 + 100 \times 0.9 \times 0.9)$

where $DWC(\{e_1\})$ means $e_1$ protected and the same meaning extends to any set of edges.

Let $A = \{e_1\}, B = \emptyset$ with $B \subseteq A$ and $x = e_2$. We have

$$DWC(A \cup \{x\}) - DWC(A) > DWC(B \cup \{x\}) - DWC(B)$$

which violates the property of submodularity. The objective function is not submodular.

A different example can be built to show that the objective function is not super-modular. Briefly, we build a graph in which there are multiple paths between two nodes with very high densities and all other nodes have 0 density. If one path has been protected and has the highest probability, efforts to protect edges on other paths do not increase the objective effectively.

2 Proof for Theorem 3.2

Theorem 3.2 Suppose $\left| \hat{P}(s \sim t | l) - P(s \sim t | l) \right| < \epsilon$ for any $l$. Let the optimal solution to the original problem be OPT. Let the optimal solution with $\hat{P}$ as the objective function be OPT* (measured in the original objective function). We must have

$$OPT \geq OPT^* \geq OPT - \left( \sum_{s,t} D_{s} \right) \epsilon.$$
The definition of \( \hat{P} \) guarantees that
\[
\hat{P}(s \to t|l) \leq P(s \to t|l).
\]
Because \(|\hat{P}(s \to t|l) - P(s \to t|l)| < \epsilon\), we must have:
\[
P(s \to t|l) - \epsilon \leq \hat{P}(s \to t|l) \leq P(s \to t|l)
\]
for any \( s, t \) pair, and any \( l \). Suppose \( x_0 \) is the optimal solution, Equation (1) guarantees that
\[
OPT - (\sum_{s,t} D_s)\epsilon = \sum_{s,t} D_s P_{x_0}(s \to t|l_{st}) - (\sum_s D_s)\epsilon
\]
\[
\leq \sum_{s,t} D_s \hat{P}_{x_0}(s \to t|l_{st})
\]
\[
\leq \sum_{s,t} D_s P_{x_0}(s \to t|l_{st}).
\]
Here, \( P_{x_0} \) presents probabilities for a protection plan \( x_0 \). Now suppose there is an algorithm which optimizes for objective function \( \hat{P} \) and produces the optimal solution \( x_1 \). By the definition of the optimal solution, we have:
\[
\sum_{s,t} D_s \hat{P}_{x_0}(s \to t|l_{st}) \leq \sum_{s,t} D_s \hat{P}_{x_1}(s \to t|l_{st}).
\]
By Equation (1),
\[
\sum_{s,t} D_s \hat{P}_{x_1}(s \to t|l_{st}) \leq \sum_{s,t} D_s P_{x_1}(s \to t|l_{st}).
\]
By the Definition that \( x_0 \) is the optimal solution for objective function \( P \), we have
\[
\sum_{s,t} D_s P_{x_1}(s \to t|l_{st}) \leq \sum_{s,t} D_s P_{x_0}(s \to t|l_{st}) = OPT.
\]
This is one side of the Inequality to prove. Combining the Equations (3), (4) and (2), we can have the other side of the Inequality.

3 Proof of Theorem 4.2

Theorem 4.2 If \( w(x; s, t, u) \) is an indicator function that returns one if and only if \( (x, s, t, u) \) satisfies constraint (15) and (16) in the main text, we have
\[
\frac{1}{2} \max_x \sum_{s,t} D_s P(s \to t) - \frac{D_{\text{max}}}{2^{j+1-2w}} \leq \frac{D_{\text{max}}}{2^j} \max_{s,t,u} w(x; s, t, u) \leq \sum_{s,t,u} D_s P(s \to t).
\]

Proof. For \( j = 1, \ldots, l - 1 \), when \( \frac{1}{2^{j+1-2w}} < \frac{D_s P(s \to t)}{D_{\text{max}}} \leq \frac{1}{2^j} \), \( u_j, u_{j+1}, \ldots, u_{l-1} \) are all fixed to 0 (Equation (13) in main text). \( u_1, \ldots, u_{j-1} \) are not constrained, so
\[
\frac{1}{2} \sum_u w(x; s, t, u) = \frac{D_{\text{max}}}{2^{j+1}}.
\]
We have
\[
\frac{1}{2} \frac{D_s P(s \to t)}{D_{\text{max}}} \leq \frac{D_{\text{max}}}{2^j}.
\]
Equation (6) holds.

When \( \frac{D_s P(s \to t)}{D_{\text{max}}} > \frac{1}{2^j} \), all \( u_1, \ldots, u_{j-1} \) are free, so
\[
\frac{1}{2} \sum_u w(x; s, t, u) = \frac{1}{2} \text{ (note } \frac{D_s P(s \to t)}{D_{\text{max}}} < 1\text{.) Equation (6) holds as well.}
\]

When \( \frac{D_s P(s \to t)}{D_{\text{max}}} < \frac{1}{2^j} \), Equation (14) in the main text guarantees that \( \frac{1}{2} \sum_u w(x; s, t, u) = 0 \). Equation (6) holds as well.

\[
\frac{1}{2} \sum_{s,t} D_s P(s \to t) - \frac{D_{\text{max}}}{2^{j+1-2w}} \leq \frac{D_{\text{max}}}{2^j} \sum_{s,t,u} w(x; s, t, u)
\]
\[
\leq \sum_{s,t,u} D_s P(s \to t).
\]

Proof. Add the left and right hand side of Equation (6) together for different \( s \) and \( t \). Note: \( w = \log_2 |V| \).

Proof. (Theorem 4.2) From Lemma 3.2,
\[
\frac{D_{\text{max}}}{2^j} \sum_{s,t,u} w(x; s, t, u) \leq \sum_{s,t,u} D_s P(s \to t).
\]
holds for all fixed \( x \). Note that the probability \( P(s \to t) \) depends on \( x \). Let \( x_0 \) be the one that attains maximal value of the objective function
\[
\frac{D_{\text{max}}}{2^j} \sum_{s,t,u} w(x; s, t, u),
\]
then we have
\[
\frac{D_{\text{max}}}{2^j} \sum_{s,t,u} w(x_0; s, t, u) \leq \sum_{s,t,u} D_s P(s \to t; x_0).
\]
Further because
\[
\sum_{s,t,u} D_s P(s \to t; x_0) \leq \max_x P(s \to t; x),
\]
we have the second inequality of Theorem 4.2. The first inequality is similar.
5 MIP encoding for XOR \( k \)

For replication (i) in XOR\( k \) algorithm, we need establish the following flow constraints:

\[
\sum_{e \leftarrow v} \left( I_{e}^{p(i)} + I_{e}^{u(i)} \right) - \sum_{e \rightarrow v} \left( I_{e}^{p(i)} + I_{e}^{u(i)} \right) = \mathbf{1}(\text{bin}(v) = s^{(i)}) - \mathbf{1}(\text{bin}(v) = t^{(i)}), \forall v \in V. \tag{8}
\]

Here, \( I_{e}^{p(i)} \) and \( I_{e}^{u(i)} \) are flow variables for replication (i). \( \mathbf{1}(\text{bin}(v) = s^{(i)}) \) is an indicator variable that is one iff the binary representation of the index of node \( v \) is \( s(i) = (s_{w-1}^{(i)}, \ldots, s_{0}^{(i)}) \). We introduce \( s_{w-1}^{(i)}, \ldots, s_{0}^{(i)} \) as binary variables in the MIP encoding. We use a continuous variable \( \beta_{v}^{(i)} (0 \leq \beta_{v}^{(i)} \leq 1) \) to represent this indicator variable. For example, when \( v = 1 \), we can enforce the following constraints:

\[
\beta_{v}^{(i)} \leq 1 - s_{k}^{(i)}, \quad \forall k = 1, \ldots, w - 1
\]

and

\[
\beta_{v}^{(i)} \leq s_{0}^{(i)}\]

and

\[
\beta_{v}^{(i)} \geq s_{0}^{(i)} + \sum_{k=1}^{w-1} (1 - s_{k}^{(i)}) - w + 1.
\]

This guarantees that \( \beta_{v}^{(i)} \) is the indicator variable for \( \mathbf{1}(\text{bin}(v) = s^{(i)}) \). Similarly, we introduce \( \gamma_{v}^{(i)} \) as the indicator for \( \mathbf{1}(\text{bin}(v) = t^{(i)}) \), and we use similar constraints to enforce it. Finally, we can enforce the flow constraint as:

\[
\sum_{e \leftarrow v} \left( I_{e}^{p(i)} + I_{e}^{u(i)} \right) - \sum_{e \rightarrow v} \left( I_{e}^{p(i)} + I_{e}^{u(i)} \right) = \beta_{v}^{(i)} - \gamma_{v}^{(i)}. \tag{9}
\]

Equation (15) (16) in the main text are essentially indicator constraints, which can be encoded using big-M notation.

6 Further Details about the Dataset

**Synthetic Instances** The density matrices are designed, with population centers located far away from each other. The edges are defined to connect adjacent parcels from top, bottom, left and right. The resistance matrix is a randomly located subregion of the resistance matrix for the realistic instance (see below). The price of purchasing an edge \( c_{e} \) is always 1. \( r_{e}^{p} \) – the resistance of an edge under protection is the average of the resistances of the two parcels it connects, and \( r_{e}^{u} \) is set to a large value.

**Realistic Instances** In order to generate realistic size instance we use a a kriging algorithm to create a 32 x 32 pixel spatially correlated surface of covariate values (resistance), simulating differential cost of movement among landscape pixels in a patchy environment. We scaled the covariate values between 0 and 1 to allow for only positive estimates of cost.