Budget-Constrained Demand-Weighted Network Design for Resilient Infrastructure

Amrita Gupta
School of Computational Science & Engineering
Georgia Institute of Technology
Atlanta, GA 30332, USA
agupta375@gatech.edu

Bistra Dilkina
Department of Computer Science
University of Southern California
Los Angeles, CA 90089, USA
dilkina@usc.edu

Abstract— **Our work is motivated by an important network design problem in climate adaptation. As floods become more frequent and severe due to climate change, it is increasingly crucial that road infrastructure be strategically upgraded to support post-disaster recovery efforts and normal functionality. We focus on the problem of allocating a fixed budget towards restoring edges to maximize the satisfied travel demand between locations in a network, which we formalize as the budget-constrained prize-collecting Steiner forest problem. We prove that the satisfiable travel demand objective exhibits restricted supermodularity over forests, and utilize this property to design an iterative algorithm based on maximizing successive modular lower bounds for the objective that finds better solutions than a baseline greedy approach. We also propose an extremely fast heuristic for maximizing modular functions subject to knapsack and graph matroid constraints that can be used as a subroutine in the iterative algorithm, or as a standalone method that matches the greedy baseline in terms of quality but is orders of magnitude faster. We evaluate the algorithms on synthetic data, and apply them to a real-world instance of retrofitting the Senegal national road network against flooding.**

I. Introduction

Natural and man-made disasters damage transportation networks, telecommunications and the power grid, leading to loss of service and hindering disaster response. As extreme weather events become increasingly frequent under climate change, damage to these critical infrastructures is projected to increase dramatically–e.g. six-fold in the EU by 2050 [1]. This has spurred governments, policymakers and researchers to seek out technologies and strategies to reduce the consequences of critical infrastructure system failures. Possible actions include structural adaptation measures like changing road surface composition to better withstand weather-related stress, as well as management adaptation measures like strategically positioning repair crews to quickly restore lost power [2]. The associated network design, planning and scheduling problems have also inspired AI researchers to develop effective and scalable techniques that can be applied to these critical, real-world problems.

In this paper, we focus on the problem of upgrading a subset of key components in a road network to minimize service disruption in the event of an earthquake, flood, or other natural hazard. Road infrastructure plays a vital role immediately before, during, and shortly after disasters in planning and conducting zone-based evacuations and enabling first responders to access affected populations. This has motivated a great deal of research on variations of the pre-disaster transportation network preparation problem [3], e.g. to strategically upgrade roads such that evacuation paths are protected [4], or such that the average travel time of emergency response vehicles to service points is minimized [5]. There is also a growing body of work in post-disaster road network restoration, where the goal is to optimize the order in which to clear roads and the positioning of equipment to enable evacuations [6] or distribute emergency supplies [7].

However, far less attention has been paid to systematically fortifying national roads to support the large-scale, normal functioning of the network after a disaster. Road networks enable the distribution of goods and resources and facilitate overland trade flows that are integral to economic activity. Furthermore, restoration times for road infrastructure are typically longer than for other infrastructure systems, meaning that vulnerable areas connected to the larger network by only a few roads may remain inaccessible for months if those roads fail. For example, roads between the Mozambican capital city Maputo and the rest of the country remained unusable for nearly a year after devastating flooding in 2000, causing economic growth to come to a halt [8].

We attempt to fill this gap in the literature and focus on the problem of finding a pre-disaster road fortification plan that ensures that a regional population's travel needs are still met as much as possible under likely disaster scenarios. In the post-disaster period, affected populations begin to return to normal activity, using alternative travel routes if necessary even if they are longer or more congested than original routes. First, we model this problem as an instance of *Budget*-Prize-Collecting Steiner Forest (Budget-PCSF): given a graph, pairs of vertices that need to be connected and travel demand between them, and edge repair costs, we select a subset of edges to repair in order to maximize the satisfied travel demand while respecting a budget constraint[1]. In the regular Prize-Collecting Steiner Forest (PCSF) problem [9], the goal is to minimize the combination of edge purchasing costs and incurred penalties for failing to connect designated pairs of vertices. However, in the disaster preparedness

[1]All code can be found at `https://github.com/amritagupta/budget-pcsf-semigradient-ascent`.

setting, it may be undesirable to combine the financial costs of upgrading roads and the socioeconomic costs of failing to connect certain location pairs into a single optimization objective. Moreover, government agencies and development initiatives typically must operate within strict budget plans, and hence it is necessary to include a hard budget constraint.

Next, we prove that the objective function of Budget-PCSF exhibits the property of *restricted supermodularity*, with positive implications for algorithm design. Specifically, we show that a modular profit function for connecting pairs of vertices exhibits the property of compounding gains when restricted to the set of forests in the graph (the graph matroid). We then demonstrate how to adapt recent work on constrained supermodular maximization [10, 11] to the budget-constrained, graph matroid-restricted supermodular setting using a tree-sampling approach. This allows us to extend ideas from [12] to develop an iterative algorithmic approach where at each step we derive a modular lower bound to the Budget-PCSF objective that we maximize while respecting the budget and matroid constraints. We also propose a novel heuristic algorithm (`Knapsack-Repair`) for maximizing a modular lower bound subject to a budget and a matroid constraint, based on optimally solving a knapsack integer linear program (ILP) followed by a greedy repair step to fix any cycles introduced into the solution.

We demonstrate experimentally that iteratively maximizing successive modular lower bounds either greedily or with our proposed heuristic consistently finds solutions of much better quality than a greedy baseline (the fastest algorithm previously applied to Budget-PCSF known to the authors), while also typically being faster. We also find that running a single iteration of our approach using `Knapsack-Repair` performs as well as the greedy baseline, but it is 100-5000 times faster. Finally, we apply our approaches for solving Budget-PCSF to a large-scale network design problem for retrofitting the national highway system in Senegal in preparation for flooding scenarios of different severities.

We first describe Budget-PCSF as it relates to road infrastructure resilience, review the literature for algorithmic approaches to solving this problem, and prove the restricted supermodularity of its objective. Section III describes how to compute modular lower bounds for the Budget-PCSF objective, how to maximize these using either a greedy or our proposed ILP-based algorithm, and how to wrap this step into an iterative procedure. Section IV presents our experimental results on synthetic and real-world instances.

II. Budget Prize-Collecting Steiner Forest

A. Problem Definition

We are given an undirected, uncapacitated graph $G = (V, E)$, where edges represent road segments, and vertices represent junctions or endpoints of the road segments. We also have an OD (origin-destination) matrix whose entries (i, j) contain the expected number of trips from vertex i to vertex j over the road network, which we refer to as the travel demand from i to j. Travel demands need not be symmetric, and we assume that as long as a path exists in the network between vertices i and j, the travel demand between them in both directions is satisfied. This constitutes a profit function $p(u, v) : V \times V \to \mathbb{R}_+$ for connecting pairs of vertices. We emphasize that in our formulation of Budget-PCSF, we use these demands to set pairwise profits on vertices to be maximized, rather than setting penalties to be minimized as is typical in the PCSF problem. We are also given a cost function $c : E \to \mathbb{R}_+$ on the edges, and a fixed budget B. These edge costs reflect the projected cost of satisfactorily upgrading a given road segment to ensure it withstands a given flood scenario. The planner's task is to decide which road segments to upgrade through the allocation of the budget B, such that the maximum travel demand is satisfied. This leads to the following problem:

Given: Graph $G = (V, E)$, a non-negative edge cost function $c : E \to \mathbb{R}_+$, a budget B, pairs of vertices $\mathcal{P} = \{(u_1, v_1), (u_2, v_2), \dots, (u_k, v_k)\}$ to be connected, and a non-negative profit function $p : \mathcal{P} \to \mathbb{R}_+$ for successfully connecting vertex pairs in $\mathcal{P}$

Find: A forest $F \subseteq E$ such that $\sum_{e \in F} c(e) \leq B$ and $\sum_{(u,v) \in \mathcal{Q}} p(u, v)$ is maximized, where $\mathcal{Q} \subseteq \mathcal{P}$ is the set of vertex pairs connected by edges in F.

We specify that the selected road segments should form an acyclic subgraph (a forest). Although a cyclic subgraph would be a feasible solution with respect to our goal of maximizing satisfied travel demand, we note that the presence of cycles in the solution means there are multiple edge-disjoint paths between the same pairs of vertices, providing no benefit in terms of additional connectivity but consuming more of the budget than needed. Hence, we can restrict our search to solutions that form a forest in G.

B. Related Work

Closely related to our graph optimization problem, the prize-collecting Steiner forest (PCSF) problem asks the following question: given an undirected graph $G = (V, E)$, a non-negative edge cost function $c : E \to \mathbb{R}_+$, pairs of vertices $\mathcal{P} = \{(u_1, v_1), (u_2, v_2), \dots, (u_k, v_k)\}$, and a non-negative *penalty* function $\pi : \mathcal{P} \to \mathbb{R}_+$, which subgraph F of G minimizes the cost of edges in F plus the sum of penalties for pairs in $\mathcal{P}$ that are not connected by F? This problem has been studied extensively and applied in domains as diverse as molecular biology for discovering signaling pathways in a cellular interactome [13], and backbone discovery in transportation networks [14]. The PCSF problem was shown to be APX-hard [15], and has a 3-approximation algorithm based on the primal-dual method [9]. [16] devised an $O(|V|^{2/3} \cdot 2 \cdot log|V|)$-approximation algorithm for the *Quota*-PCSF with uniform profits, where the goal is to find a minimum cost forest such that the satisfied demand is at least Q. However, despite the practical relevance of the budget-constrained variant of PCSF, it has received relatively little attention, and there is no known approximation algorithm for Budget-PCSF known to the authors.

Existing approaches for solving Budget-PCSF largely rely on computationally heavy approaches. [17, 18] propose

$$S = \{\emptyset\}$$
$$T = \{e_{AB}\}$$
$$U = \{e_{AB}, e_{BC}\}$$

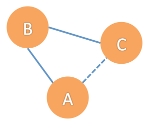

Fig. 1: Restricted supermodularity. $f(S \cup \{e_{BC}\}) - f(S) \leq f(T \cup \{e_{BC}\}) - f(T)$, showing compounding gains. However, $f(T \cup \{e_{AC}\}) - f(T) \nleq f(U \cup \{e_{AC}\}) - f(U) = \mathbf{0}$, showing restricted supermodularity.

mixed integer programming formulations for this problem and demonstrate results on a graph with 23 vertices and 34 edges representing the Ohio interstate system. [19] propose a bi-objective integer programming model to solve a closely related problem to Budget-PCSF, in which the goal is to maximize the satisfied demand (where demand at an origin is satisfied if a path exists to *at least one* of a set of predefined destinations) while minimizing travel time between origin-destination pairs as a second objective with application to retrofitting bridges along critical routes for earthquake response. In contrast, we consider the setting in which the demand is specified between each pair of vertices. In general, flow-based integer programming formulations are known to encounter challenges when scaling to larger instances of these families of network design problems and can be particularly sensitive to the number of OD-pair flow variables. Instead, in this work we focus on developing scalable heuristic approaches. For example, in [20] the authors employed a simple greedy algorithm to solve Budget-PCSF.

C. Restricted Supermodularity

In recent work, results about the modularity of objective functions have played an instrumental role in designing effective algorithmic approaches to solving network design problems in domains such as robotic motion control [21] and sensor placement [22]. We now turn our attention to an analysis of the objective function in Budget-PCSF, and specifically prove that it is *restricted supermodular* over subsets of edges that form forests. A set function $f(S) : 2^E \to \mathbb{R}$ is supermodular if it exhibits the property of compounding gains, or formally if $\forall S \subset T \subset E$ and $\forall e \in E \setminus T$:

$$f(S \cup \{e\}) - f(S) \leq f(T \cup \{e\}) - f(T) \tag{1}$$

However, when the above property holds only over a collection of subsets of E, f is referred to as a *restricted* supermodular function. The analogous property of restricted submodularity for functions with diminishing returns was first described by [23] who used it to analyze a greedy algorithm for the Steiner tree problem.

Let $S \subseteq 2^E$ denote a set of edges selected from graph G. We aim to maximize the following objective function subject to a knapsack (budget) constraint:

$$f(S) = \sum_{(u,v) \in \mathcal{Q}} p(u, v) \tag{2}$$

where $\mathcal{Q} \subseteq \mathcal{P}$ is the set of vertex pairs connected by edges in $G(S)$, the graph induced by edges in the set S, and $p(u, v)$ is the profit function described earlier.

Proposition 1: *$f(S)$ is monotone non-decreasing.*
This is trivial since augmenting the set S with another element (edge) can never reduce the number of pairs of vertices connected in G; and the profit function p is non-negative.

Proposition 2: *$f(S)$ is supermodular when restricted to the set of forests F on G (the graph matroid).*
A first intuition regarding the relationship between f and the structure of subsets on which it is defined, is that there must exist a set S that maximizes f and contains no cycles– i.e. $G(S)$ is a forest. As described earlier, any solution containing a cycle would contain multiple edge-disjoint paths between the same pair of vertices, providing no additional connectivity but consuming more of the budget than strictly necessary. Therefore, we can restrict our search to forests.

A second intuitive notion is that connectivity builds upon itself. See, for instance, the graph in Figure 1. If edge A–B is restored, then demand between A and B can be satisfied. The same holds for edge B–C. However, when *both* A–B and B–C are restored, in addition to restoring connectivity between A and B and between B and C, we get the *extra* benefit of connecting A and C. This phenomenon of *compounding gains* or *increasing differences* is the characteristic feature of supermodular functions, and has been observed and leveraged in various other settings, such as influence maximization under the linear threshold model with edge addition [11]. Figure 1 also illustrates the concept of restricted supermodularity. When attempting to augment set U by adding edge e_{AC} that forms a cycle with respect to the other edges already in U, we have $f(U \cup \{e_{AC}\} - f(U) = 0$, although adding e_{AC} to a subset of U could have strictly increased f; in this case the gain associated with e_{AC} was diminished, violating the requirement for general supermodularity. A formal proof that f is restricted supermodular function follows.

Proof: For notational convenience, let Let $G(u; S)$ be the connected component in $G(S)$, the subgraph induced by edges in the set S, that contains vertex u. Let $f_G(u; S)$ be the total profit between pairs of vertices in the connected component $G(u; S)$. Consider a subset of edges $S \subset E$ and two different edges $e_1 = (u_{e_1}, v_{e_1})$ and $e_2 = (u_{e_2}, v_{e_2})$ in $E \backslash S$. Let $\Delta_e f(S) := f(S \cup \{e\}) - f(S)$. We want to show that $\Delta_{e_2} f(S \cup \{e_1\}) \geq \Delta_{e_2} f(S)$ whenever $G(S \cup \{e_1, e_2\})$ contains no cycles. (We already provided a counterexample in Figure 1 showing that this is not true when S contains a cycle.) There are 3 possible cases:

a) 1. $\mathbf{e_2}$ has no endpoints in $\mathbf{G(S \cup \{e_1\})}$.: Then e_2 also has no endpoints in $G(S)$, so

$$\begin{aligned}
&\Delta_{e_2} f(S) = f(S \cup \{e_2\}) - f(S) \\
&= f_G(u_{e_2}; S \cup \{e_2\}) - f_G(u_{e_2}; S) \\
&= f_G(u_{e_2}; S \cup \{e_2\}) - 0 = f_G(u_{e_2}; \{e_2\})
\end{aligned}$$

$$\begin{aligned}&\Delta_{e_2} f(S \cup \{e_1\}) = f(S \cup \{e_1, e_2\}) - f(S \cup \{e_1\})\\&= f_G(u_{e_2}; S \cup \{e_1, e_2\}) - f_G(u_{e_2}; S \cup \{e_1\})\\&= f_G(u_{e_2}; S \cup \{e_1, e_2\}) - 0\\&= f_G(u_{e_2}; \{e_2\}) = \Delta_{e_2} f(S)\end{aligned}$$

b) 2. $\mathbf{e_2}$ *has exactly 1 endpoint in* $\mathbf{G(S \cup \{e_1\})}$*.:* Let u_{e_2} be the endpoint of e_2 in $G(S \cup \{e_1\})$. Either u_{e_2} is in the same connected component as the endpoints of e_1, or it is in a different connected component of $G(S \cup \{e_1\})$. If $u_{e_2} \in G(u_{e_1}; S \cup \{e_1\})$, there are two possibilities–either u_{e_2} is a vertex in $G(S)$ or $u_{e_2} = v_{e_1}$. In the former case:

$$\begin{aligned}&\Delta_{e_2} f(S) = f(S \cup \{e_2\}) - f(S)\\&= f(u_{e_2}; S \cup \{e_2\}) - f(u_{e_2}; S)\\&= \{\text{flow } G(S) \leftrightarrow v_{e_2}\}\end{aligned}$$

$$\begin{aligned}&\Delta_{e_2} f(S \cup \{e_1\}) = f(S \cup \{e_1, e_2\}) - f(S \cup \{e_1\})\\&= f_G(u_{e_2}; S \cup \{e_1, e_2\}) - f_G(u_{e_2}; S \cup \{e_1\})\\&= \{\text{flow } G(S) \leftrightarrow v_{e_2}\} + \{\text{flow } v_{e_1} \leftrightarrow v_{e_2}\}\\&\geq \{\text{flow } G(S) \leftrightarrow v_{e_2}\} = \Delta_{e_2} f(S)\end{aligned}$$

In the latter case:

$$\begin{aligned}&\Delta_{e_2} f(S) = f_G(u_{e_2}; S \cup \{e_2\}) - f_G(u_{e_2}; S)\\&= \{\text{flow } u_{e_2} \leftrightarrow v_{e_2}\}\end{aligned}$$

$$\begin{aligned}&\Delta_{e_2} f(S \cup \{e_1\}) = f_G(u_{e_2}; S \cup \{e_1, e_2\}) - f_G(u_{e_2}; S \cup \{e_1\})\\&= \{\text{flow } G(u_{e_1}; S \cup \{e_1\}) \leftrightarrow v_{e_2}\}\\&\geq \{\text{flow } u_{e_2} \leftrightarrow v_{e_2}\} = \Delta_{e_2} f(S)\end{aligned}$$

If, on the other hand, $u_{e_2} \notin G(u_{e_1}; S \cup \{e_1\})$ then:

$$\begin{aligned}&\Delta_{e_2} f(S) = f(S \cup \{e_2\}) - f(S)\\&= f_G(u_{e_2}; S \cup \{e_2\}) - f_G(u_{e_2}; S)\end{aligned}$$

$$\begin{aligned}&\Delta_{e_2} f(S \cup \{e_1\}) = f(S \cup \{e_1, e_2\}) - f(S \cup \{e_1\})\\&= f_G(u_{e_2}; S \cup \{e_1, e_2\}) - f_G(u_{e_2}; S \cup \{e_1\})\\&= f_G(u_{e_2}; S \cup \{e_2\}) - f_G(u_{e_2}; S) = \Delta_{e_2} f(S)\end{aligned}$$

c) 3.$\mathbf{e_2}$ *has both endpoints in* $\mathbf{G(S \cup \{e_1\})}$*.:* Since we restrict ourselves to subsets of E that do not form cycles, the endpoints of e_2 must be in two separate connected components both in $G(S)$ and $G(S \cup \{e_1\})$. Either e_2 links two connected components that do not contain the endpoints of e_1—in this case $\Delta_{e_2} f(S \cup \{e_1\}) = \Delta_{e_2} f(S)$. Otherwise, e_2 links one connected component containing the endpoints of e_1 with another connected component–in this case $\Delta_{e_2} f(S \cup \{e_1\}) \geq \Delta_{e_2} f(S)$. ■

III. Budget-Constrained Restricted Supermodular Maximization

With the results of Section II-C, we can approach Budget-PCSF as maximizing a monotone non-decreasing, restricted supermodular function over the independent sets of a graph matroid, subject to a knapsack constraint. There is very limited work to date on maximizing restricted submodular or supermodular functions subject to additional constraints. In [23], the authors studied the implications of restricted submodularity over forests for a greedy algorithm for the minimum Steiner tree problem, in which there are no cardinality or knapsack constraints for acquiring edges. In [24], the authors studied maximizing a function that is submodular when restricted to the set of solutions satisfying a cardinality constraint. In contrast to these works, in Budget-PCSF the supermodularity of the objective function holds over the independent sets of a matroid, while there is an additional, separate knapsack constraint on edge selection.

In this work, we treat both the graph matroid structural restrictions and the knapsack constraint as hard constraints. Even though a solution violating the matroid restriction is not strictly infeasible, the violations degrade the supermodularity property that serves as the basis of our algorithm design. Therefore, we aim to maximize a supermodular function subject to a knapsack and a matroid constraint. [10] provide bounds for maximizing a supermodular function subject to a *cardinality* constraint (equivalent to knapsack constraints with unit or uniform edge costs) **or** a matroid constraint, but not both. They showed that a simple greedy algorithm that adds items in order of maximum benefit with respect to the current set achieves a $(1 - \kappa^f)$ approximation bound, where κ^f is the supermodular curvature of function f. It is unclear whether the same bound holds for maximizing supermodular functions subject to a knapsack constraint, rather than a cardinality constraint; or for the combination of a knapsack and a matroid constraint, as we have here. [20] employed a similar greedy algorithm to [10] to solve Budget-PCSF, in which items are selected in order of benefit-cost ratio. However, as we will show, this greedy algorithm scales poorly with problem size.

A. Modular Lower Bounds for Restricted Supermodular Functions

Instead of relying on a simple cost-benefit greedy criterion, we implement an iterative heuristic based on *semigradient ascent* [12], which is based on computing modular lower bounds (MLB) for the objective function f and then efficiently maximizing the MLB. This theoretical framework was recently adapted by [11] to supermodular maximization with a cardinality constraint for influence maximization in graphs using edge addition.

Supermodular functions have discrete subdifferentials that can be used to construct tight MLBs [10]. Given a set function f and a set S, the subdifferentials of f at S are all vectors y such that $f(S) - y(S) + y(S') \leq f(S')\ \forall S' \in E$; a subgradient is one such vector y. Therefore, a subgradient essentially provides a lower bound for the function f evaluated on set S'. In [10], the following two *discrete* subgradients for supermodular functions are proposed:

$$\breve{y}(j) = \begin{cases} f(j|S \setminus \{j\}), & \text{if } j \in S \\ f(j|\emptyset), & \text{otherwise} \end{cases} \tag{3}$$

and,

$$\hat{y}(j) = \begin{cases} f(j|E \setminus \{j\}), & \text{if } j \in S \\ f(j|S), & \text{otherwise} \end{cases} \tag{4}$$

These lead to the following two modular lower bounds for the function f at a new solution S' using current solution S:

$$\check{m}_S(S') = f(S) - \sum_{j \in S \setminus S'} f(j|S \setminus j) + \sum_{j \in S' \setminus S} f(j|\emptyset) \leq f(S') \tag{5}$$

$$\hat{m}_S(S') = f(S) - \sum_{j \in S \setminus S'} f(j|E \setminus j) + \sum_{j \in S' \setminus S} f(j|S) \leq f(S') \tag{6}$$

The terms $f(j|S \setminus j)$ and $f(j|S)$ depend on the current solution S and thus need to be computed on the fly through calls to a function evaluator for f. However, $f(j|\emptyset)$ in Eq. 5 is simply the travel demand between the endpoints of edge j, so this term can be looked up from the demand matrix.

For general supermodular functions, the $f(j|E \setminus j)$ terms in Eq. 6 can be precomputed for each $j \in E$. These terms quantify the largest reduction in objective value that could occur as a result of removing element j from the current solution. However, for our restricted supermodular function, we need to find $\max_{S \in \mathcal{C}} f(j|S \setminus j)$, where $\mathcal{C}$ is the collection of sets over which the supermodularity property holds, or the graph matroid in our case. The largest objective value reduction for removing a given edge j occurs when S is a specific spanning tree on G such that j is the cut-set for a weighted max-cut on the spanning tree. We adopt a sampling approach to evaluate the maximum possible impact of each edge j, sampling a set $\mathcal{T}_j$ of N random spanning trees of G in which j is a member and setting $f(j|E \setminus j) \approx \max_{\mathcal{T} \in \mathcal{T}_j} \{f(j|\mathcal{T} \setminus j)\}$.

Another simple modular lower bound that was successfully used for constrained submodular maximization [11] is the following:

$$\sum_{j \in S'} f(j|\emptyset) \leq f(S') \tag{7}$$

where $f(j|\emptyset)$ is again the demand between the endpoints of edge j. Eq. 7 does not require a current solution or any expensive function evaluations to compute, and is in fact equivalent to Eq. 5 or Eq. 6 when the initial solution S is the empty set.

B. *Semigradient Ascent*

The modular lower bounds in Eqs. 5 and 6 are computed with respect to a solution S, and can be used to iteratively find solutions of increasing objective value using Algorithm 1 proposed in [12] and adapted to our restricted supermodularity setting. A current solution to Budget-PCSF (E_{add}) is used to compute the coefficients for one of the subgradient-based modular lower bounds (lines 9 and 12). Then, we find a new solution that maximizes this modular lower bound subject to budget and matroid constraints (lines 10 and 13). We alternate between the two bounds, terminating when no further changes to the solution are made under either of them.

Algorithm 1 SEMIGRADIENT-ASCENT

```
 1: function SEMIGRAD($E, c, B, p, \epsilon$)
 2:     current_mlb ← $\check{m}$
 3:     $\check{m}$_converged←False
 4:     $\hat{m}$_converged←False
 5:     converged←False
 6:     $E_{\text{add}} \leftarrow \emptyset$
 7:     while not converged do
 8:         if current_mlb = $\check{m}$ then
 9:             $\check{m}_{E_{\text{add}}}(\cdot)$ ←GETMLBCOEFFS($E_{\text{add}}$)
10:             $E'_{\text{add}} \leftarrow \arg\max_{S \in \mathcal{C}} \check{m}_{E_{\text{add}}}(S)$
11:         else
12:             $\hat{m}_{E_{\text{add}}}(\cdot)$ ←GETMLBCOEFFS($E_{\text{add}}$)
13:             $E'_{\text{add}} \leftarrow \arg\max_{S \in \mathcal{C}} \hat{m}_{E_{\text{add}}}(S)$
14:         if $E'_{\text{add}} = E_{\text{add}}$ then
15:             if current_mlb=$\check{m}$ then
16:                 $\check{m}$_converged←True
17:                 current_mlb ← $\hat{m}$
18:                 $E_{\text{add}} \leftarrow E'_{\text{add}}$
19:             else
20:                 $\hat{m}$_converged←True
21:                 current_mlb ← $\check{m}$
22:                 $E_{\text{add}} \leftarrow E'_{\text{add}}$
23:             if $\check{m}$_converged and $\hat{m}$_converged then
24:                 converged←True
25:         else
26:             $E_{\text{add}} \leftarrow E'_{\text{add}}$
27:             $\check{m}$_not_converged←True
28:             $\hat{m}$_not_converged←True
29:     return $E_{\text{add}}$
```

C. *Maximizing Modular Lower Bounds*

Maximizing any of the above modular lower bounds corresponds to maximizing a modular function subject to *both a knapsack and a matroid constraint*. There is relatively little work on solving the 0-1 knapsack problem (whose objective is the modular sum of the item values), subject to additional constraints beyond the budget constraint. However, maximizing *submodular* functions subject to combinations of different numbers and types of constraints is an active area of research. We leverage these approaches based on the fact that modular functions are in fact submodular (as well as supermodular), with the inequality satisfied with equality.

Maximizing a monotone submodular function subject to a knapsack *and* a matroid constraint is NP-hard to approximate to within a factor better than $1 - \frac{1}{e}$ [25], and only a handful of relatively recent works have proposed algorithmic techniques for addressing this problem. One of the first approaches for submodular maximization suitable for a combination of matroid and knapsack constraints was the randomized swap rounding algorithm proposed by [26]. More recently, [25] presented a simpler $\frac{1-e^{-2}}{2}$-approximation algorithm that also handles the simultaneous application of a matroid constraint and k knapsack constraints. The algorithm proceeds by greedily building the solution set by considering local search moves (either adding a single element or by swapping a previously added element for a new one) that respect the matroid constraints. The best move is chosen greedily according to benefit-cost ratio. Despite the attractive guarantees regarding the quality of this solution set, the algorithm is not guaranteed to terminate in polynomial time.

Algorithm 2 KNAPSACK-REPAIR

```
function KR($E, c, B, p, \epsilon$)
    add_edges←True
    $E_{\text{add}} \leftarrow \emptyset$
    while add_edges do
        $E_{\text{K}} \leftarrow$ KNAPSACKILP($E, c, B, p, \epsilon$)
        $E_{\text{add}} \leftarrow E_{\text{add}} \cup E_{\text{K}}$
        $E_{\text{prune}} \leftarrow$ REPAIR($E_{\text{add}}$, $c$)
        $E_{\text{add}} \leftarrow E_{\text{add}} \setminus E_{\text{prune}}$
        $B \leftarrow B - \sum_{e \in E_{\text{add}}} p(e_u, e_v)$
        $E \leftarrow E \setminus (E_{\text{add}} \cup E_{\text{prune}})$
        if $B < \min\{c(e)\}_{e \in E}$ OR $E = \emptyset$ then
            add_edges←False
    return $E_{\text{add}}$
function REPAIR($E_{\text{add}}, c$)
    cycles_fixed←False
    $E_{\text{prune}} \leftarrow \emptyset$
    $E_{\text{cycles}} \leftarrow$ FINDEDGESINCYCES($E_{\text{add}}$)
    if $E_{\text{cycles}} = \emptyset$ then
        cycles_fixed←True
    while not cycles_fixed do
        $e_{\text{expensive}} \leftarrow \arg\max_{e \in E_{\text{cycles}}} c(e)$
        $E_{\text{prune}} \leftarrow E_{\text{prune}} \cup \{e_{\text{expensive}}\}$
        $E_{\text{cycles}} \leftarrow$ FINDEDGESINCYCES($E_{\text{add}}$)
        if $E_{\text{cycles}} = \emptyset$ then
            cycles_fixed←True
    return $E_{\text{prune}}$
```

We adopt simpler heuristic strategies for maximizing the modular lower bounds subject to the knapsack and matroid constraints. Our first approach, GreedyMLB is similar to [25] in that edges are chosen in order of benefit-cost ratio, but swaps are not allowed. Each edge is added to the solution only if it can be purchased with the remaining budget and if it does not introduce a cycle into the solution. We also propose a second algorithm, Knapsack-Repair, shown in Algorithm 2. This algorithm alternates between phases of edge addition, in which an integer linear program for the 0-1 knapsack problem is solved to allocate the available budget towards edges, and repair, in which matroid constraint violations are corrected by greedily removing the most expensive edge participating in a cycle in the solution, recovering the cost of the edge, and continuing until all cycles are repaired. The algorithm alternates between the knapsack and repair phases until either the budget is exhausted or all edges have been either added or discarded.

In both GreedyMLB and Knapsack-Repair, each edge is added to the solution at most once (rather than allowing edges to be swapped out and then potentially swapped back in), and so the algorithms terminate quickly and are good candidates for solving the subproblems in lines 10 and 13 of Algorithm 1.

IV. EXPERIMENTS AND RESULTS

We compare the performance of four algorithms: 1) a baseline Greedy algorithm that iteratively constructs a solution by adding the edge with the best benefit to cost ratio without violating the knapsack and matroid constraints ([20], implemented with lazy evaluation); 2) KR maximizing f by applying Knapsack-Repair (Algorithm 2) to the modular lower bound in Eq. 7; 3) Semigrad-GreedyMLB (Algorithm 1) where at each iteration the modular lower bound is optimized greedily; and 4) Semigrad-KR (Algorithm 1) where at each iteration the modular lower bound is optimized by calling Knapsack-Repair. We report results on a real road network from Senegal, a country where flood resilience is of particular concern, and on synthetic instances to more fully characterize the algorithms' performance. For each problem instance, we compute the cost of the minimum spanning tree, which is the minimum cost at which all the available profit can be obtained. We then vary the budget allocated for Budget-PCSF as a fraction of the MST cost. The knapsack ILPs within KR were solved using Gurobi v8.0. All experiments were run on a cluster of five 32-core machines with 2.10GHz processors and 256GB of RAM.

A. Synthetic Instances

We synthetically generate instances of Budget-PCSF on which to evaluate our proposed methods. Specifically, we generate random planar graphs with mean degree close to 3.8, the mean degree of the Senegal road network retrofitting instance under the 50-year flood scenario. We vary the size of the instances parameterized by number of edges, generating 20 random instances of each graph size. We also created a similar dataset with random Erdos-Renyi graphs, for which results can be found on our GitHub page. We generated costs for edges and demand between pairs of vertices following a random uniform distribution.

Figure 2 shows the relative quality of solutions found by KR, Semigrad-GreedyMLB and Semigrad-KR compared to the Greedy method at different budget levels for instances varying from 500 to 1500 edges. Solution quality is reported as % improvement in objective value compared to the Greedy baseline. The results show that for budget fractions 0.1-0.2 and 0.7-0.9, all 3 algorithms achieve results on par with the Greedy baseline. *At intermediate budget levels, the two iterative semigradient ascent-based methods deliver dramatic improvements over the Greedy solutions.*

Figure 3 shows the mean runtime for each algorithm for Budget-PCSF at different budget levels and instance sizes. KR is the fastest method by far, with runtimes under 1 second in nearly all cases. This means that *KR can provide solutions with quality on par with Greedy with a speed-up of 100 to 5000 times*. The runtimes for the two semigradient ascent-based methods also include the time taken to estimate the maximum value of each edge in the graph instance, as

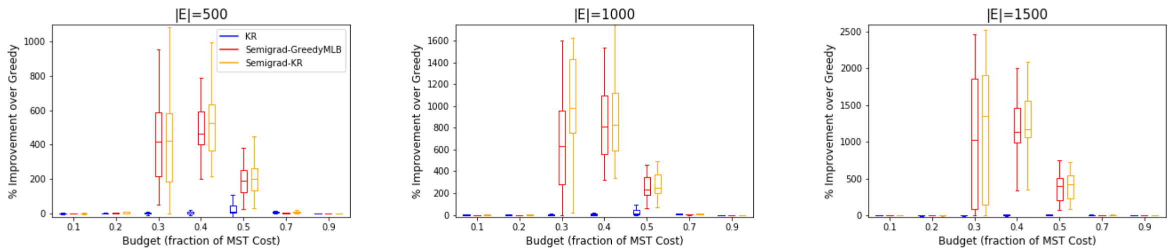
Fig. 2: Solution quality relative to `Greedy` versus budget level on random planar graphs of different sizes; `KR` is on average as good as `Greedy`, while `Semigrad-GreedyMLB` and `Semigrad-KR` find solutions 3-15 times better than `Greedy`.

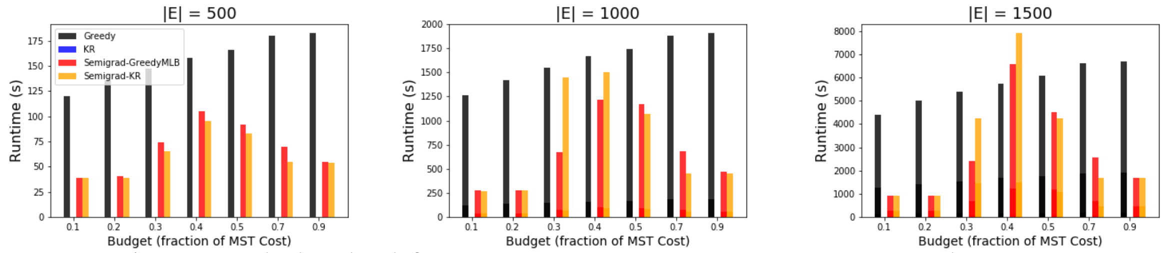
Fig. 3: Mean runtime versus budget level for `Greedy`, `KR`, `Semigrad-GreedyMLB` and `Semigrad-KR` on random planar graphs of 3 sizes; `Knapsack-Repair` (KR) is extremely fast across all instance sizes and budget levels.

described in Section III-A. Semigradient-based methods are typically faster than `Greedy` across different budgets; hence, at intermediate budget levels, semigradient-based methods beat `Greedy` in both solution quality and time.

B. *Road Upgrades for Flood Resilience in Senegal*

Road infrastructure plays a key role in socioeconomic development, and consequently infrastructure expansion initiatives are a core focus of many countries' economic plans. However, extreme weather events cause damage to essential infrastructure, costing billions of dollars to repair and potentially setting back economic development and exacerbating the existing vulnerabilities faced by the population. Flooding is of major concern in Senegal, due to both a recent increase in severe floods and the susceptibility of the largely unpaved road network to damage from precipitation. Preserving connectivity via the national and regional road network in the event of frequently-occurring (e.g. 5- or 10-year) floods is an important goal towards enabling recovery efforts as well as normal activity to continue in the face of these risks.

TABLE I: Budget-PCSF instance sizes resulting from floods of different return periods on the Senegal road network.

Return Period	Vertices	Edges	MST Cost (km)
5	161	178	60.399
10	285	321	122.593
20	444	518	211.250
50	654	758	353.884

We apply the Budget-PCSF problem to retrofitting the Senegal regional road network against 5, 10, 20 and 50-year flooding scenarios. The full road network consists of 6917 vertices corresponding to road intersections or endpoints and 7175 edges corresponding to road segments forming a single connected component. The flood risk of each road can be estimated from predicted or historical flood data. The costs of the edges are assumed to be proportional to the length of the flooded portion of the road segment, which would need to be fortified. We construct a compact representation of the flooded road network, where each vertex represents a connected component and each edge is the least cost edge between a pair of connected components. The Budget-PCSF graph sizes resulting from the 4 flood scenarios are summarized in Table I. Travel demand over the network can be estimated from population data, or from fine-grained mobility data mined from call detail records, for example [20]. Unlike our synthetic instances with random uniform demands, the pairwise travel demands between vertices in the Senegal road network graphs were highly skewed, with very low travel demand between all but a few vertex pairs.

We compare the performance of the `Greedy` algorithm, `KR`, `Semigrad-GreedyMLB` and `Semigrad-KR` with a budget of $B = 0.1 \times$MST cost in Table II. The four methods perform more similarly compared to our experiments on synthetic instances; this is due to the fact that the highly skewed pairwise demands enable the `Greedy` baseline to perform reasonably well. Nevertheless, `KR` still produces solutions as good as or better than `Greedy` in a fraction of the time, providing a $140\times$ speedup on the smallest instance (5-year flood scenario) and a $> 8000\times$ speedup on the largest instance (50-year flood scenario). The semigradient-ascent based methods improve solution quality beyond either `Greedy` or `KR` while still being 5 to 20 times faster than `Greedy`. Although the higher objective values attained by `Semigrad-GreedyMLB` and `Semigrad-KR` come at the cost of more iterations and longer runtime compared to `KR`, these algorithms terminate within a few seconds on the real-world road instances, e.g. in under 3 minutes on even the largest instance corresponding to the 50-year flood scenario.

V. Conclusion

We address the problem of strategically fortifying edges in an infrastructure network against failures in order to maximize satisfied demand between vertices in the network. Unlike previous work on network design that relies on integer

TABLE II: Best found solution objective value (number of feasible trips) and runtime (s) on the Senegal road network for 5-, 10-, 20- and 50-year floods with budget 0.1*MSTCost. Values in parentheses are gap percentages for sub-optimal solutions.

Method	5-year		10-year		20-year		50-year	
	Objective	Time (s)	Objective	Time (s)	Objective	Time (s)	Objective	Time (s)
Greedy	1195531703 (0.02%)	14.03	1182448994 (0.75%)	84.00	1177026417 (0.08%)	332.30	1161809642 (0.83%)	1140.11
KR	1195531703 (0.02%)	0.10	1182534472 (0.74%)	0.05	1177389779 (0.05%)	0.07	1161974353 (0.82%)	0.13
Semigrad-GreedyMLB	**1195800142** (0.00%)	2.97	**1191291479** (0.00%)	7.06	**1177993964** (0.00%)	32.17	1171044232 (0.04%)	85.84
Semigrad-KR	1195798624 (0.00%)	2.06	1183590105 (0.65%)	5.04	1177948435 (0.00%)	16.63	**1171484174** (0.00%)	155.19

programming-based methods, we show that our optimization objective exhibits the property of restricted supermodularity, connecting budget-constrained prize-collecting Steiner forest to the vast literature on submodular/supermodular optimization. We demonstrate how to extend recent work on constrained supermodular maximization to our *restricted* supermodular setting. We also propose a novel, fast algorithm for maximizing modular functions subject to a knapsack and a budget constraint. Empirically, we show that our proposed algorithms perform as well as a greedy baseline on both synthetic and real-world networks, while typically being significantly faster. Importantly, we show that supermodularity-based algorithms have the potential to scale well to solve large practical network design problems in this family.

Acknowledgments

Gupta and Dilkina were partially supported by National Science Foundation Grant No. CCF-1522054 (COMPUST-NET: Expanding Horizons of Computational Sustainability). Dilkina was partially supported by the U.S. Department of Homeland Security under Grant Award No. 2015-ST-061-CIRC01. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S Department of Homeland Security.

References

[1] G. Forzieri, A. Bianchi, F. B. e Silva, M. A. M. Herrera, A. Leblois, C. Lavalle, J. C. Aerts, and L. Feyen, "Escalating impacts of climate extremes on critical infrastructures in europe," *Global environmental change*, vol. 48, pp. 97–107, 2018.

[2] OECD, "Climate-resilient infrastructure," *OECD Environment Policy Papers*, no. 14, 2018. doi: `10.1787/4fdf9eaf-en`.

[3] S. Peeta, F. S. Salman, D. Gunnec, and K. Viswanath, "Pre-disaster investment decisions for strengthening a highway network," *Comput Oper Res*, vol. 37, no. 10, pp. 1708–1719, 2010.

[4] K. Kumar, J. Romanski, and P. Van Hentenryck, "Optimizing infrastructure enhancements for evacuation planning.," in *AAAI*, pp. 3864–3870, 2016.

[5] X. Wu, D. Sheldon, and S. Zilberstein, "Optimizing resilience in large scale networks.," in *AAAI*, pp. 3922–3928, 2016.

[6] D. T. Aksu and L. Ozdamar, "A mathematical model for post-disaster road restoration: Enabling accessibility and evacuation," *Transportation Research Part E: Logistics and Transportation Review*, vol. 61, pp. 56–67, 2014.

[7] F. Liberatore, M. T. Ortuño, G. Tirado, B. Vitoriano, and M. P. Scaparra, "A hierarchical compromise model for the joint optimization of recovery operations and distribution of emergency goods in humanitarian logistics," *Comput Oper Res*, vol. 42, pp. 3–13, 2014.

[8] P. Chinowsky and C. Arndt, "Climate change and roads: A dynamic stressor–response model," *Review of Development Economics*, vol. 16, no. 3, pp. 448–462, 2012.

[9] M. T. Hajiaghayi and K. Jain, "The prize-collecting generalized steiner tree problem via a new approach of primal-dual schema," in *Proc. of the 17th annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 631–640, Society for Industrial and Applied Mathematics, 2006.

[10] W. Bai and J. Bilmes, "Greed is still good: Maximizing monotone submodular+supermodular (bp) functions," in *ICML*, pp. 314–323, 2018.

[11] E. B. Khalil, B. Dilkina, and L. Song, "Scalable diffusion-aware optimization of network topology," in *Proc. of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1226–1235, ACM, 2014.

[12] R. Iyer, S. Jegelka, and J. Bilmes, "Fast semidifferential-based submodular function optimization," in *ICML*, pp. 855–863, 2013.

[13] N. Tuncbag, A. Braunstein, A. Pagnani, S.-S. C. Huang, J. Chayes, C. Borgs, R. Zecchina, and E. Fraenkel, "Simultaneous reconstruction of multiple signaling pathways via the prize-collecting steiner forest problem," *J Comput Biol*, vol. 20, no. 2, pp. 124–136, 2013.

[14] S. Chawla, K. Garimella, A. Gionis, and D. Tsang, "Backbone discovery in traffic networks," *International Journal of Data Science and Analytics*, vol. 1, no. 3-4, pp. 215–227, 2016.

[15] M. Bateni, C. Chekuri, A. Ene, M. T. Hajiaghayi, N. Korula, and D. Marx, "Prize-collecting steiner problems on planar graphs," in *Proc. of the 22nd annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1028–1049, Society for Industrial and Applied Mathematics, 2011.

[16] D. Segev and G. Segev, "Approximate k-steiner forests via the lagrangian relaxation technique with internal preprocessing," in *European Symposium on Algorithms*, pp. 600–611, Springer, 2006.

[17] Y.-S. Myung and H.-j. Kim, "A cutting plane algorithm for computing k-edge survivability of a network," *European Journal of Operational Research*, vol. 156, no. 3, pp. 579–589, 2004.

[18] T. C. Matisziw and A. T. Murray, "Modeling s–t path availability to support disaster vulnerability assessment of network infrastructure," *Comput Oper Res*, vol. 36, no. 1, pp. 16–26, 2009.

[19] K. Viswanath and S. Peeta, "Multicommodity maximal covering network design problem for planning critical routes for earthquake response," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1857, pp. 1–10, 2003.

[20] A. Gupta, C. Robinson, and B. Dilkina, "Infrastructure resilience for climate adaptation," in *Proc. of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies*, p. 28, ACM, 2018.

[21] I. Shames and T. H. Summers, "Rigid network design via submodular set function optimization," *IEEE Transactions on Network Science and Engineering*, vol. 2, no. 3, pp. 84–96, 2015.

[22] N. Mehr and R. Horowitz, "A submodular approach for optimal sensor placement in traffic networks," in *2018 Annual American Control Conference (ACC)*, pp. 6353–6358, IEEE, 2018.

[23] D.-Z. Du, R. L. Graham, P. M. Pardalos, P.-J. Wan, W. Wu, and W. Zhao, "Analysis of greedy approximations with nonsubmodular potential functions," in *Proc. of the 19th annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 167–175, Society for Industrial and Applied Mathematics, 2008.

[24] A. Aouad, R. Levi, and D. Segev, "Greedy-like algorithms for dynamic assortment planning under multinomial logit preferences," 2015.

[25] K. K. Sarpatwar, B. Schieber, and H. Shachnai, "Constrained submodular maximization via greedy local search," *arXiv preprint arXiv:1705.06319*, 2017.

[26] C. Chekuri, J. Vondrak, and R. Zenklusen, "Dependent randomized rounding via exchange properties of combinatorial structures," in *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pp. 575–584, IEEE, 2010.