Computational Approaches for Optimal Load Shedding Control of DC Networks under Cascading Failure

Qin Ba  Ketan Savla

Abstract—We consider discrete-time dynamics, for cascading failure in DC power networks, whose map is composition of failure rule with control policy. Under feasible control actions, supply-demand at the nodes is monotonically non-increasing in magnitude. Under the failure rule, a link is removed permanently if the flow on it exceeds given thermal capacity constraints. We consider a finite horizon optimal control problem to steer the network from an arbitrary initial state, defined in terms of active link set and supply-demand vector at the nodes, to a feasible state, i.e., a state which is invariant under the failure rule. There is no running cost and the value associated with a feasible terminal state is the associated cumulative supply-demand. We propose two novel computational frameworks for control synthesis. The first is a network decomposition approach which can be implemented in two iterations for tree reducible networks, and leads to a semi-analytical solution for the unit time horizon case. The second approach interprets optimal control synthesis as an optimal search problem. An algorithmic procedure is provided to compute the one-stage reachable set using arrangement of hyperplanes to facilitate this search. We outline a approximation strategy to reduce computations in the search.

I. INTRODUCTION

Cascading failure in physical networks can be modeled via discrete-time dynamics, where the time epochs correspond to component failures. The map of the dynamical system is described in terms of composition of a failure rule with a control policy. A common failure rule is permanent removal of a link if its physical flow exceeds capacity constraints. Analysis of such dynamics under a given control policy has attracted considerable attention, primarily through simulations, e.g., see [1], [2]. However, control design is relatively less well understood, see, e.g., [3] and our previous work in [4] for few such examples. In this paper, we consider such an optimal control problem for power networks.

The network state is described in terms of active links, i.e., links that have not been removed so far, and demand-supply vector at the nodes. We consider the class of load shedding control policies, i.e., control actions under which the absolute values of demand-supply at all the nodes are monotonically non-increasing. Under the failure rule, at a given network state, links are permanently removed if their power flow, as determined by the DC model, exceeds thermal capacity constraint. A network state is called feasible if it is invariant under the failure rule. Given an initial state, the optimal control problem is to steer the network to a terminal feasible state within a specified time horizon. The running cost is zero, and the value associated with a terminal feasible state is equal to the associated cumulative demand-supply. This problem was formulated in [3], where low-complexity control policies were also considered. However, a formal framework for control synthesis or performance analysis is lacking in the literature. The objective of this paper is to develop rigorous computational approaches for (approximately) optimal control synthesis that significantly generalize the class of control policies in [3]. While we provide specific control policies with performance guarantees, we believe that the computational frameworks suggested in this paper provide a much needed formalism to this important class of control problems, and hence are significant contributions.

We provide two frameworks for control synthesis. The first is a decomposition approach for tree reducible networks. The underlying algorithm can be implemented over two iterations as follows. In the first iteration, starting from leaves, every node computes, via local optimization, local control action parameterized by the flow on link to its parent node. The first iteration ends with root node computing a specific local control action which is broadcast to its children nodes, and thereafter in the second iteration, all the nodes sequentially compute their respective local control actions. Moreover, if one restricts to the class of constant (over all stages) control actions, and if the network satisfies a certain monotonicity property, then we provide a semi-analytical solution to the optimal controller. In particular, for the unit horizon case, this implies that the optimal solution can be found in time linear in number of nodes.

The second approach is inspired by interpretation of optimal control synthesis as search for an optimal terminal feasible state. We provide an exact finite representation of the hybrid reachable set by aggregating control actions corresponding to the same reachable active link set. This finite representation is shown to be arrangement of hyperplanes corresponding to link capacity constraints, e.g., see [5]–[8]. We define a novel operation on the incidence graph representation of the arrangement for efficient usage in the search process. To the best of our knowledge, this presents a new engineering application of these computational geometric tools beyond path planning in robotics, e.g., see [9].

The size of the finite representation increases exponentially with the number of non-transmission nodes, and hence can be prohibitive for large networks. We address this an approximation strategy which performs aggregation in a projected space. A particular instance of such a projection yields the setting of the optimal scaling problem in [3, Section 6.1.1].
We conclude this section by defining a few notations. \( \mathbf{R}, \mathbf{R}_{\geq 0} \) and \( \mathbf{R}_{>0} \) respectively denote the set of real, non-negative real, and positive real numbers. 0 and 1 denote vectors of all zeros and ones of proper size; \( \mathbf{e}_i \) is the unit vector on \( i \)th axis with 1 on \( i \)th component and 0 on others. For an integer \( n \), \( [n] := \{1, 2, \ldots, n\} \). For a matrix \( M \in \mathbb{R}^{S_1 \times S_2} \) and \( S_1' \subseteq S_1 \) and \( S_2' \subseteq S_2 \), \( M_{S_1' S_2'} \in \mathbb{R}^{S_1' \times S_2' \times \mathbb{R}} \) is the submatrix of \( M \) containing entries corresponding to rows in \( S_1' \) and columns in \( S_2' \). In addition, \( M_{S_2} := M_{S_2 S_2} \). Similarly, given sets \( S \) and \( S' \subseteq S \), \( S_1 := S \setminus S_1 \), \( |S| \) denotes the cardinality of \( S \), and \( x_S \) and \( f_S(\cdot) \) denote the appropriate sub-vector of vector \( x \in \mathbb{R}^d \) and function \( f : X \rightarrow \mathbb{R}^d \), respectively. The range of a matrix \( M \) is denoted by \( \text{range}(M) \).

We start by recalling the DC power flow model

**A. DC power flow model**

In this model, it is assumed that the transmission lines are lossless and the voltage magnitudes are constant at 1.0 unit. The graph topology of the power network is described by a directed graph \( \mathcal{G} = (\mathcal{V}, \mathcal{E}) \), where the direction of every link in \( \mathcal{E} \) is decided arbitrarily. Let \( \mathcal{V}_c \subseteq \mathcal{V} \) and \( \mathcal{V}_d \subseteq \mathcal{V} \) be the set of supply and demand nodes respectively. A node is called transmission node if it is neither a supply or demand node. Let \( \mathcal{V}_t := \mathcal{V}_c \cup \mathcal{V}_d \) denote the set of nontransmission node. Since our objective is to study cascading dynamics which could possibly lead to loss of connectivity in the network, we let \( \mathcal{V}_c \cup \mathcal{V}_d \cup \mathcal{V}_t \) denote the partition of the original graph among the \( \mathcal{r} \) weakly connected components. \( \mathcal{V}_e^{(i)} := \mathcal{V}_c \cap \mathcal{V}_d^{(i)} \), for all string \( e \in \{l, s, d\} \). These sets \( \mathcal{V}_c, \mathcal{V}_d, \mathcal{V}_t \) are implicitly associated with a network \( \mathcal{G} = (\mathcal{V}, \mathcal{E}) \) and we use them without explicitly mentioning in the paper. The graph \( \mathcal{G} \) is associated with a node-link incidence matrix \( \mathbf{A} \in \mathbb{R}^{\mathcal{V} \times \mathcal{E}} \), where \( A_i \) corresponds to link \( i \) and has +1 and −1 respectively on the tail and head node of link \( i \), and 0 on other nodes. The links are associated with a flow vector \( f \in \mathbb{R}^d \); The signs of elements of \( f \) are to be interpreted as being consistent with the directional convention chosen for links in \( \mathcal{E} \). We also associate a diagonal matrix \( \mathbf{W} \in \mathbb{R}^{\mathcal{E} \times \mathcal{E}} \) whose diagonal elements give the negative of susceptances, or weights, of the corresponding links. For brevity, \( w_i \) shall denote the \( i \)th diagonal element of \( \mathbf{W} \). The nodes are associated with phase angles \( \phi \in \mathbb{R}^d \) and the supply and demand nodes are associate with a supply-demand vector \( p \in \mathbb{R}^d \); \( p_i > 0 \) for \( i \in \mathcal{V}_c \) and \( p_i < 0 \) for \( i \in \mathcal{V}_d \).

The quantities defined above are related similarly by Kirchhoff’s law and Ohm’s law in DC circuit as follows:

\[
Af = p; \quad f = WA^T \phi
\]  

where \( p \) is required to be balanced, \(^\text{1}\) to be feasible, i.e., \( p \in B_{\mathcal{E}} := \{ u \in \mathbb{R}^{\mathcal{V}} \mid \sum_{v \in \mathcal{V}(i)} p_v = 0, \forall i \in [\mathcal{E}] \} \). For a given network \( \mathcal{G} = (\mathcal{V}, \mathcal{E}) \) with balanced supply and demand \( p \), unique flow \( f \) exists to satisfy \( (1) \) and it is given by \( [10] \)

\[
f = W A^T L^p =: f(\mathcal{E}, p)
\]

where \( L = AW \) is the weighted Laplacian matrix of the \( \mathcal{G} \) and \( L^p \) is its pseudo-inverse. As indicated in the function \( f(\mathcal{E}, p) \), flow \( f \) depends on the link set \( \mathcal{E} \). Nevertheless, for a given \( \mathcal{E}, f(\mathcal{E}, p) \) is a linear function of \( p \).

**B. Cascading dynamics**

Let \( \mathcal{E}_0 \) and \( p_0 \in B_{\mathcal{E}_0} \) be the initial link set and supply-demand vector of a network. The link flow \( f \) is uniquely determined by \( \mathcal{E}_0 \) or \( [10] \). We associate each link \( i \in \mathcal{E}_0 \) with a thermal capacity \( c_i > 0 \). Once the magnitude of flow on a link \( i \in \mathcal{E}_0 \) exceeds its thermal capacity \( c_i \), that is, \( |f_i| > c_i \), link \( i \) fails and is removed from the network irreversibly. This changes the topology of the network and leads to flow redistribution and possibly more line outages. The continuing line outages constitute the cascading failure of the network. Note here we assume the link outage rule is deterministic and depends solely on the instantaneous flow through a link. This is to be contrasted with other deterministic outage rules based on moving average of the successive flows, or stochastic line outage rules, e.g., see [3], [11]. While these other variations can be easily incorporated into our formulation, the technical results in this paper are specific to the deterministic version.

We now formally specify the cascading failure dynamics in discrete-time, where we assume the node set remain the same and each time epoch corresponds to failure of some links. Let \( (\mathcal{E}_t, p_t) \) be the state of the network at time \( t \), with \( \mathcal{E}_t \subset 2^{\mathcal{E}_0} \) and \( p_t \in B_{\mathcal{E}_t} \) denote the active link set and supply-demand vector at time \( t \), respectively. We consider load shedding as the control and, for convenience, employ control variable \( u \in \mathbb{R}^{\mathcal{V}} \) to be supply-demand vector after load shedding. The controlled cascading dynamics, for \( t = 0, 1, \ldots \), and starting from the initial state \( (\mathcal{E}_0, p_0) \), is given by:

\[
(\mathcal{E}_t+1, p_t+1) = F(\mathcal{E}_t, p_t, u_t), \quad u_t \in U_{\mathcal{E}_t}(\mathcal{E}_t, p_t)
\]

where the two components are \( F(\mathcal{E}, p, u) \) and \( p(\mathcal{E}, p, u) := \mathcal{F}(\mathcal{E}, p, u) := \{ i \in \mathcal{E} \mid -c_i < f_i(\mathcal{E}, u) \leq c_i \} \) and \( p(\mathcal{E}, p, u) := \mathcal{F}(\mathcal{E}, p, u) := u \). \( \mathcal{F} \) is the set of feasible links in \( \mathcal{E} \) under the supply-demand vector \( u \); and the control input \( u_t \) at time \( t \) becomes the next stage supply-demand vector \( p_t+1 \). The control space \( U_{\mathcal{E}}(\mathcal{E}, p) \) at state \( (\mathcal{E}, p) \) is equal to the its action set \( U(\mathcal{E}, p) \) for all \( [10] \)

\[
U(\mathcal{E}, p) := B_{\mathcal{E}} \cap \text{cube } p
\]

where cube \( p := \{ u \in \mathbb{R}^{\mathcal{V}} \mid 0 \leq \text{sign}(p_v) u \leq |p_v| \forall v \in \mathcal{V} \} \), with \( \text{sign}(x) \) is 1 for \( x \geq 0 \) and -1 for \( x < 0 \), characterizes the load shedding requirement. By definition, when all the supply and demand node are disconnected from each other at a state \( (\mathcal{E}, p) \), \( U(\mathcal{E}, p) = \{0\} \).

The evolution of cascading dynamics proceeds as follows. Suppose the initial flow \( f(\mathcal{E}_0, p_0) \) at state \( (\mathcal{E}_0, p_0) \) is infeasible. To respond, a controller picked \( u_0 \in U(\mathcal{E}_0, p_0) \) to be

\(^{1}\text{The reason to have two notions } U(\mathcal{E}, p) \text{ and } U_{\mathcal{E}}(\mathcal{E}, p) \text{ will become clear later.}\)
the next state supply-demand vector \( p^1 \). If the resulting flow 
\( f(E^0, p^1) \) was within the capacity, the cascading dynamics would stop and the system would stay at state \((E^0, p^1) = F(E^0, p^0) = (E^1, p^1)\) (cf. (3)). Otherwise, the over capacitated links would fail and a new state \((E^1, p^1)\) would be reached with \( E^1 = F_E(E^0, u^0) \neq E^0 \) and \( p^1 = u^0 \). If \( f(E^1, p^1) \) was within the capacity, the controller would not respond by simply picking \( u^2 = p^1 \). The cascading dynamics would stop in this case as \( f(E^1, p^1) = E^1 \). Otherwise, the controller would respond with a control \( u^2 \in U(E^1, p^1) \) and the cascading dynamics would evolve in this way and so on.

It is clear that the cascading dynamics terminates at a state \((E, p)\) if all active links are feasible, that is, \( F_E(E, p) = E \). Let the set of feasible state be \( S := \{(E, p) \mid F_E(E, p) = p, p \in B_E\} \). \( S \) is not empty since \((E, 0) \in S \) for every \( E \in 2^\mathbb{E} \). Because \( \mathcal{E}^t \) decreases, that is, \( \mathcal{E}^{t+1} \subseteq \mathcal{E}^t \) and \( E^0 \) is finite, the dynamics converges to a feasible state within finite episodes. At the same time, one may wish to terminate cascading failure in limited number of episodes. Therefore, without loss of generality, we consider \( N \) (finite) horizon for the load shedding problem. That is to say, at stage \( N - 1 \) and state \((E^{N-1}, p^{N-1})\), we assume the controller must take an action \( u^{T-1} \) such that \((E^{N-1}, u^{T-1})\) is feasible. Hence, the control space at \( t \) is redefined as \( \mathcal{U}(E, p) := U(E, p) \) for \( 0 \leq t \leq N - 2 \) and \( \mathcal{U}_{N-1}(E, p) := \{ u \in U(E, p) \mid F_E(E, u) = E \} \).

C. Searching for solutions

The objective is to find an action sequence \( u^{[N]} := (u^0, \ldots, u^{N-1}) \) which steers the system from the initial state \((E^0, p^0)\) to a feasible state \((E^N, p^N) \in S \) associated with maximal amount of remaining load. Formally, we consider the following problem:

\[
\sup_{u^{[N]}} d^pu^N \quad \text{s.t.} \quad (E^N, p^N) \in S \tag{5}
\]

where \( d \in \mathbb{R}^V \) is constant and defined as: \( d_v := 1 \) for \( v \in \mathcal{V}_s \), \( d_v := -1 \) for \( v \in \mathcal{V}_d \), and \( d_v := 0 \) otherwise. \( d^pu^N \) equals to the sum of magnitude of supply and demand at the final stage \( N \). Supremum, rather than maximum, is used in (5) because in general the feasible set of \( u^{[N]} \), denoted by \( \mathcal{D} \), is not necessarily closed. An example is provided as follows.

Example 1: Consider two stage load shedding problem \((N = 2)\) for the triangular network shown in Fig. 1a where node 1 is the supply node and node 2 and 3 are the demand nodes and the initial supply-demand vector is \( p^0 = [30, -10, -20]^T \). The weight is \( w = [2, 1, 1, 1]^T \) and the capacity is \( c = [6, 7, 4, 5]^T \). At \( t = 0 \), we control the flow \( u^0 = [21 + 2/k, -7 - 1/k, -14 - 1/k]^T \) for some \( k \geq 1 \). The resulting flow is \( [8 + 6/(7k), 4 + 3/(7k), 9 + 5/(7k), 5 + 2/(7k)]^T \). Link \( e_1 \) and \( e_2 \) fail due to excessive flow and the network becomes the one shown in Fig. 1b at \( t = 1 \). We employ the control \( u^1 = [21, -7, -14]^T \) and the network becomes feasible. This is to say, \( u^{[N]}(k) := [21 + 2/k, -7 - 1/k, -14 - 1/k]^T \), \([21, -7, -14]^T \) \( \notin \mathcal{D} \) for all \( k \geq 1 \). However, one can verify that \( \hat{u}^{[N]} = \lim_{k \to \infty} u^{[N]}(k) = ([21, -7, -14]^T, [21, -7, -14]^T) \notin \mathcal{D} \). In that case, only \( e_3 \) fails after the control action \( u^0 = [21, -7, -14]^T \) and the resulting network at \( t = 1 \) is shown as in Fig. 1c. With the new supply-demand vector \( \hat{u}^1 = [21, -7, -14]^T \) at \( t = 2 \), the flow is \( [0, 28/3, 35/3, 7/3]^T \). \( e_2 \) fails and is following by the failure of \( e_3 \) and \( e_4 \) if no more load shedding is taken. This demonstrates that \( \mathcal{D} \) is not closed for this network.

![Fig. 1: A triangular network](image)

We solve the problem by searching [12, Chap 3]. A directed tree is constructed, where a node corresponds to a network state \((E, p)\) and an arc corresponds to a control action \( u \) and the associated state transition according to (3). The initial state \((E^0, p^0)\) is the root and the set \( S \) contains the goal states that the system is to reach. Hence each node \((E, p)\) is assigned a value \( r(E, p) \) defined as: \( r(E, p) = d^pu \) if \((E, p) \in S \) and \( r(E, p) = -\infty \) otherwise. As can be seen, only goal states have finite value. The problem is to find a path from the initial state to a goal state that is within the range of \( N \) stages and has the maximal value.

Let \( J_t(E, p) \) be the maximal value at a state \((E, p)\) with \( t \) stages to go. It equals to the maximal value among the goal states within \( t \) hops of \((E, p)\). Solving (5) is equivalent to computing \( J_N(E^0, p^0) \). By definition: \( J_0(E, p) = r(E, p) \), and, for all \( t \in [N] \):

\[
J_t(E, p) = \sup_{u \in \mathcal{U}_{t-1}(E, p)} J_{t-1}(F_E(E, u), u) \tag{6}
\]

where supremum is used in (6) because \( F_E(E, u) \) is not continuous w.r.t. \( u \). In particular, for \( t = 1 \),

\[
J_1(E, p) = \max_{u} d^pu \quad \text{s.t.} \quad -c_e \leq f(E, u) \leq c_e; u \in U(E, p) \tag{7}
\]

where maximization is used because \( F_E(E, u) = E \) for \( u \in U_{N-1}(E, p) \). (7) is a linear program, commonly referred to as LP power redispatch, e.g., see [13]. It is straightforward to see that \( J_1(E, p) \leq J_t(E, p) \leq d^pu \) for all \((E, p)\) and \( t \in [N] \). These bounds can be used to trim the tree in the searching process. These bounds are later used in Section IV-C.

Computing a solution to (5) by searching based on (6) is prohibitive. One has to search through nearly the entire (reachable) state space\(^2\) in order to obtain an optimal solution. But the action set \( U(E, p) \) is a continuum for most of the interesting state \((E, p)\) (e.g., \((E, 0)\) excluded) and consequently the directed search tree has infinite branching.

\(^2\)This tree is not to be confused with the network topology of the power network, especially in Section IV-C where we focus on tree reducible power networks.

\(^3\)Since reachable set is the relevant state space for searching, by state space, we shall be implicitly referring to reachable set.
factor. A natural strategy is to discretize action set \( U(\mathcal{E}, p) \) in order to proceed with numerical implementation of search. Using dimension to roughly quantify of the size of the search space, it is \( N(|\mathcal{V}| - r) \).

In order to address the computational challenge, we propose the following two methods: (i) by focusing on tree reducible networks, \( G^i \) is decomposed into problems depending solely on local network components and consequently the problem dimension is greatly reduced. (see Section III); (ii) by aggregating the state in a special way, we obtain a finite representation of the state space (see Section IV), thereby facilitating application of classical search algorithms.

### III. DECOMPOSITION FOR TREE REDUCIBLE NETWORKS

We consider the class of tree reducible networks in this section.

#### A. Tree reducible networks

![Fig. 2: (a) A tree reducible network \( G \); and (b) reduced tree \( T \) of \( G \)](image)

A tree reducible network \( G = (\mathcal{V}, \mathcal{E}) \) is a generalization of a tree network \( T = (\mathcal{V}_T, \mathcal{E}_T) \) with each link \( i \in \mathcal{E}_T \) replaced independently by an arbitrary connected network \( G^i = (\mathcal{V}^i, \mathcal{E}^i) \); \( G^i \) joins the incident nodes of link \( i \). In the special case, when \( G \) is a tree, each \( G^i \) corresponds to a single link. Fig. 2 illustrates an example of tree reducible network, where each shape (denoted by \( G^1, \ldots, G^5 \)) in Fig. 2a corresponds to a subnetwork. \( T \) and \( G^i \), for all \( i \in \mathcal{E}_T \), are referred as the reduced tree and reducible components of \( G \), respectively. We use standard terminologies for tree (see [14]) when referring to \( T \). The root is the top node and indexed as \( v_0 \). A child node of a (parent) node is another node directed connected to the node when moving away from the root. For node \( i \in \mathcal{V} \), let \( C_i \) denote the set of its children nodes and \( \overline{C}_i \) denote the node set including its descendants and itself, e.g., in Fig. 2b \( C_2 = \{5, 6\} \) and \( \overline{C}_2 = \{2, 5, 6, 7, 8\} \). With this definition, \( \mathcal{V}_T \equiv \overline{C}_0 \). A leaf is a node \( i \) without any children, that is, \( C_i := \emptyset \). As a convention for tree networks, the direction of a link is from the child to its parent and the link is labeled using the same index as the child node, e.g., in Fig. 2b the link connecting \( v_7 \) with \( v_5 \) is labeled as \( e_7 \).

As \( G \) evolves from an initial state \( (\mathcal{E}^0, p^0) \) under an admissible action sequence \( u^{[N]} \in \mathbb{R}^{V \times N} \) according to \( \mathcal{E}^0 \), each subnetwork \( G^i \) can be independently treated as a network containing the two incident nodes of link \( i \) in \( \mathcal{E}_T \) as the only supply and demand node, respectively. For example, \( v_5 \) and \( v_2 \) is treated as the supply and demand node, respectively, for \( G^5 \) in Fig. 2. The flow going through these two nodes by \( G^i \) at time \( t \), denoted by \( z_{it} \), is considered as the supply and demand imposed on \( G^i \). \( z_{it} := z_{i[N]} \in \mathbb{R}^N \) hence acts as an action sequence for \( G^i \). Let \( E_i \) be the set of all action sequences \( z_{it} \), not necessarily increasing nor decreasing, under which \( G^i \) reaches a feasible state.

**Remark 1:** The set \( E_i \) is very similar to the feasible set \( D \) for \( G^i \) (cf. discussion after (5)), except that the action sequences in \( E_i \) may not come from load shedding. Therefore, \( D \subseteq E_i \). Nevertheless, since thermal capacities are bounded, every sequence in \( E_i \) must be bounded. While we assume \( E_i \) to be well-defined, its explicit computation follows from the discussion in Section IV-C (cf. Remark 3).

On the other hand, we define the set of load shedding sequence for an initial supply or demand \( x \in \mathbb{R} \): \( S(x) := \{u^{[N]} \in \mathbb{R}^N | u^0 \in \text{cube } x, u^{t+1} \in \text{cube } u^t, \forall t \in [N - 1]\} \). For every node \( i \in \mathcal{V}_T \) with initial supply/demand \( p_i^0 \), we use the shorthand notation \( S_i := S(p_i^0) \). Moreover, let \( y_i := (u_{i1}^0, u_{i2}^i, \ldots, u_{iN}^i) \), for all \( i \in \mathcal{V}_T \).

We consider the following problem for each node \( i \in \mathcal{V}_T \) given \( z_t \in E_i \): \( J_i(z_t) := \sum_{k \in C_i} d_k y_{kN} \sup_{z_k \forall k \in C_i \setminus i, y_k \forall k \in C_i} \) s.t. \( z_{kt} = y_k + \sum_{j \in C_i} y_j \) \( z_k \in E_k \forall k \in C_i \setminus i \) \( y_k \in S_k \forall k \in C_i \)

where \( y_{jN} \) is the \( N \)th component of \( y_j \in \mathbb{R}^N \). We use the same letter \( J \) as in (6) – the use of different argument should avoid confusion. \( J_i(z_t) \) gives the maximal utility over the subtree rooted at node \( i \in \overline{C}_0 \) provided that the history of flow out from node \( i \) is \( z_t \in \mathbb{R}^N \). Therefore, (5) is equivalent to the problem for \( J_0(0) \).

Consider the following transformation of (8) into a nested form: \( J_i(z_t) = \sup_{z_j \in E_j} \sum_{j \in C_i} d_j y_{jT} + \sum_{j \in C_i} J_j(z_j) \) s.t. \( z_i = y_i + \sum_{j \in C_i} z_j \)

It is straightforward that (8) and (9) are equivalent. In particular, for a leaf node \( i \in \mathcal{V}_T \), \( C_i = \emptyset \) and (9) reduces to \( J_i(z_i) = \max_{z_i} y_i \in S_i \) \( d_i y_{iN} \). It has solution \( J_i(z_i) = d_i z_{iN} \) for all \( z_i \in E_i \cap S_i \) and is infeasible for all \( z_i \in E_i \setminus S_i \). In general, (9) is feasible only for \( z_i \in Z_i := E_i \cap \left(S_i + \sum_{j \in C_i} Z_j\right) \). For all \( i \in \mathcal{V}_T \), \( 0 \in S_i \) and \( 0 \in E_i \), and hence \( 0 \in Z_i \). Hereafter, \( Z_i \) is implicitly taken to be the domain of \( J_i(z_i) \).

Using (9), (8) is solved in two iterations over \( T \). In the first iteration, starting from leaf nodes, every node computes \( J_i \) from (9) as a function of \( z_j \in Z_i \), given \( J_j(z_j) \), \( z_j \in Z_j \), computed already from previous iterations for all \( j \in C_i \).

The first iteration ends with root node computing a solution of \( y_0^* \) and \( z_j^* \), \( j \in C_0 \), for \( z_0 = 0 \). The values of \( z_j^* \) are then broadcast to its children nodes \( j \in C_0 \), and thereafter in the second iteration, from the root to leaves, all the nodes \( i \) sequentially compute the solution \( y_i^* \) and \( z_i^* \), \( j \in C_i \) using (9), given the value of \( z_i^* \) received from its parent node.
The recursion in (9) decomposes (8) into sub-problems of smaller sizes and decrease the computational complexity to a great extent. While the original problem (5) (and hence (6)) has a dimension of \( N(|V| - r) \), (9) relates to the local star network \((C_i \cup \{i, C_i\})\) and has a dimension of \( N(|C_i| + 1)\). Note \(|C_i| + 1\) is in fact the degree of node \(i\) in the reduced tree \(T\). Therefore, the overall time complexity of the recursive procedure is the summation of these subproblems for all nodes in \(T\), which is a great improvement compared with the tree search in [1, C].

B. Constant control for tree reducible networks

The goal in this section is to find an optimal within the class of constant load shedding control actions for tree reducible networks. A \(N\) stage constant control action \(u^{(N)} \in \mathbb{R}^{V \times N}\) for a network \(G = (V, E)\) is one such that \(u^{t-1} = u\) for all \(t \in [N]\) and some admissible \(u \in \mathbb{R}^V\). With constant control, one performs load shedding in one shot. While failures could happen in subsequent stages, under a feasible constant control action, the dynamics is guaranteed to arrive at a feasible state at the end of \(N\) stages.

Consider a tree reducible network \(G = (V, E)\) with initial supply-demand vector \(p^0 \in \mathbb{R}^V\) and reduced tree \(T = (V_T, E_T)\). When focusing on constant control, all the \(z_i\) and \(y_i\) variables in (8) and (9) can be equivalently transformed into one dimensional variables, and consequently, \(S_i \subset \mathbb{R}\) and \(E_i \subset \mathbb{R}\) are one dimensional. It is straightforward that \(S_i = \text{cube } p^0_i\) and \(E_i = -E_i\) (since the link capacities are assumed to be symmetric). However, in general, \(E_i\) can contain multiple disconnected intervals that are possibly half open. For simplicity, we assume every reducible component \(\tilde{G}^i\) satisfies Assumption 1 below with the incident nodes of link \(i \in E_T\) being the only nontransmission nodes. In this case, there exists a scalar \(c_i > 0\) such that \(E_i = [-c_i, c_i]\).

**Assumption 1:** For a network \(G = (V, E)\) with single supply and single demand, consider the \(N\) stage load shedding problem, let \(E_i^t\) be the active link set at time \(t\) according to (8) under the constant supply-demand vector \(zp_0\), where \(z \in \mathbb{R}\) and \(p_0\) is the “unit” supply-demand vector that has 1 and −1 on the supply and demand node, respectively. Network state \((E_i^N, zp_0)\) is feasible only if \((E_i^N, z'p_0)\) is feasible for all \(z' \in \text{cube } z\).

**Remark 2:** Assumption 1 is a type of monotone condition. It is trivially satisfied for (i) all networks if \(N = 1\); and (ii) parallel networks with sufficiently large \(N\) (see [15]). For other networks, the condition can be checked efficiently, e.g., by checking if the set computing \(E_i\) for \(G\) is connected. (see also Remark 1 for computation of \(E_i\)).

It is sufficient to work directly with reduced tree \(T\) and hence, for simplicity, let \(T = (V_T, E_T)\) initial supply-demand vector and capacity be \(p^0 \in \mathbb{R}^V\) and \(c \in \mathbb{R}^E\), respectively. We first introduce some notations. For \(i \in V\), let \(p_i^0 := 0\), \(p_i^n := p_i^0\) for \(p_i^0 \geq 0\), \(p_i^0 := p_i^0\) for \(p_i^0 < 0\), hence cube \(p_i^0\); set \(Z_i\) is an interval, as will be shown, let \(z_i^L \leq z_i^U\) be the left and right end of \(Z_i\), respectively; Furthermore, define a piecewise linear function: \(l_x : [a, b] \rightarrow \mathbb{R}_{\geq 0}\) over domain \([a, b] \subset \mathbb{R}\) and for \(x = (x_1, x_2) \in \mathbb{R}^2\):

\[
I_x(z) := \begin{cases} z - x_1 + x_2 & a \leq z \leq x_1 \\ -z + x_1 + x_2 & x_1 < z \leq b \end{cases}
\]

(10)

For \(a < x_1 < b\), as shown in Fig 3, \(I_x(z)\) contains two line segments joining at the top point \(x\). For \(x_1 \geq b\) (or \(x_1 \leq a\)), \(I_x(z)\) contains only one increasing (or decreasing) line segment and the top point would be \(x' := (b, b-x_1+x_2)\) (or \(x' := (a, x_1+x_2-a)\)). In this case, \(I_x(z) = I_{x'}(z)\) for all \(z \in [a, b]\). Moreover, for every node \(i \in V\), let \(x_i := (p_i^0 + p_i^n, p_i^n - p_i^0)\), then \(l_x(u) = d_iu\) over domain \([p_i^0, p_i^n]\).

We further write \(I_x(z) = I_{x'}(z)\) for simplicity.

We now show how to obtain an explicit solution to (9).

For every leaf node \(i \in V\), it is straightforward that \(J_i(z_i) = d_i z_i = l_i(z_i)\) has domain \(Z_i = S_i \cap E_i = [p_i^0, p_i^n]\), \(Z_i' = [z_i^L, z_i^U]\). Substituting into (9), we get the following problem for the parent node:

\[
J_i(z) = \max_{u \in \mathbb{R}^{E_i \cup \{i\}}} \sum_{j \in C_i \cup \{i\}} l_j(u_j) \quad \text{s.t.} \quad z = 1^Tu; q^j \leq u \leq q^J
\]

(11)

where maximum is used because the feasible set is closed and \(q^j, q^J \in \mathbb{R}^{E_i \cup \{i\}}\) are defined as: \(q_i^j := p_i^j\) and \(q_i^J := p_i^J\); \(q_i^j := z_i^L\) and \(q_i^J := z_i^U\) for all \(j \in C_i\). By this definition, \(q_i^J \leq 0 \leq q_i^j\). (11) is to find a supply-demand \(u \in [q^J, q^J]\) satisfying \(z = 1^Tu\) to maximize certain utility over the star network \((C_i \cup \{i\}, C_i)\); each node \(j\) has utility given by function \(l_j(u_j)\). The explicit solution to (11) is provided as follows.

**Proposition 1:** For a star network \(T = (V, E)\) rooted at \(v\), \(J_v(z_v)\) is defined in (11) for given function \(l_j\) with domain \([q^j, q^J]\) and top point \(x_j = (x_{j,1}, x_{j,2})\), \(x_{j,1} \in \{q^j, q^J\}\), for all \(j \in V\). Consider \(z \in [-c_0, c_0]\) for some scalar \(c_0 > 0\), then \(J_v(z) = l_v(z)\) has domain \(Z_v = [z_v^L, z_v^U]\), where \(z_v^L := \max\{1^Tq^J - c_0, z_v^L\}\), \(z_v^U := \min\{1^Tq^J + c_0, z_v^U\}\), and top point \(x_v\) which is \((1^Tx_v, 1^Tx_v)\) if \(1^Tx_v \in [-c_0, c_0]\), \((-c_0 + 0 + 1^Tx_v, x_{v,2})\) if \(1^Tx_v < -c_0\), and \((c_0 + 1^Tx_v + 1^Tx_v, x_{v,2})\) if \(1^Tx_v > c_0\); \(x_{v,1} := x_{v,1} \in \mathbb{R}^1\) and \(x_{v,2} := x_{v,2} \in \mathbb{R}^1\). Furthermore, for given \(z \in Z_v\), any point in \(z\) is an optimal solution, where \(\chi(z) : Z_v \rightarrow \mathbb{R}^V\) is a set function defined as:

\[
\chi(z) = \begin{cases} \{u \in [p_i^0, x_{i,1}] | 1^Tu = z\} & z_{i,1} \leq z \leq x_{i,1} \\ \{u \in [x_{i,1}, p_i^n] | 1^Tu = z\} & x_{i,1} \leq z \leq x_{i,1} \end{cases}
\]

Proof: As the coupling happens only on the last constraint \(z = 1^Tu\) in (11), we introduce a Lagrange multiplier...
$\mu \in \mathbb{R}$ for it and obtain the following dual function:

$$
\phi(\mu) = -\mu z + \max_{q \leq v \leq u} \sum_{v \in V} l_v(u_v) + \mu u_v \\
= -\mu z + \sum_{v \in V} \max_{q \leq v \leq u} l_v(u_v) + \mu u_v \\
= \left( \sum_{v \in V} u^*_v(\mu) - z \right) \mu + \sum_{v \in V} l_v(u_v(\mu))
$$

where $u^*_v(\mu) := \arg\max_{q \leq v \leq u} l_v(u_v) + \mu u_v$ and can be obtained swiftly for a given $\mu$ due to the piecewise linear objective and closed interval feasible set. For $\mu > 1$, the function $l_v(u_v) + \mu u_v$ is strictly increasing for all $v \in V$ and hence the optimal solution is $u^*_v = q^*_v$. It follows $\phi(\mu)$ is linear over $(1, +\infty)$. In order for the primal to be feasible, $\phi(\mu)$ has to be nondecreasing over $(1, +\infty)$ and consequently we have $z \leq 1^T q^a$. Similarly, $\phi(\mu)$ is linear and nonincreasing over $(-\infty, -1)$ and $z \geq 1^T q^f$. For $-1 \leq \mu \leq 1$, $l_v(u_v) + \mu u_v$ is piecewise linear, first nondecreases with slope $(1 + \mu)$ over $(-\infty, x_{v,1})$ and then nonincreases with slope $(\mu - 1)$ over $(x_{v,1}, +\infty)$. As a result, one solution is $u^*_v = x_{v,1} \in \{q^r_v, q^l_v\}$. It follows in the same way that $\phi(\mu)$ is also linear over $[-1, 1]$ for any $z$.

To summarize, the dual function $(\mu)$ is piecewise linear with possible break point at $\mu = -1$ or $\mu = 1$; and $(\mu)$ is feasible for $z \in [1^T q^r, 1^T q^a]$ and infeasible otherwise. Hence $J_1(z) = \min \{\phi(-1), \phi(1)\}$. Substituting $-1$ and $1$ into $\phi(\mu)$, we get

$$
J_1(z) := \begin{cases} 
-1^T x_1 + 1^T x_2 & \text{if } 1^T q^f \leq z \leq 1^T x_1 \\
-1^T x_1 + 1^T x_2 & \text{if } z \leq 1^T x_1 \\
-1^T x_1 + 1^T q^a & \text{if } z > 1^T x_1 
\end{cases}
$$

Compare the above equation with (10) and since $z \in [-c_t, c_t]$, it is clear that $J_1(z) = l_x(z)$ over the domain $[1^T q^f, 1^T q^a] \cap [-c_t, c_t] = [z_t, z^*_t]$ and the top point is $(1^T x_1, 1^T x_2)$ if $1^T x_1 \in [z_t, z^*_t]$ and $(z_t, 0)$ if $1^T x_1 > z_t$. Combining with the fact that $1^T q^f \leq 1^T x_1 \leq 1^T q^a$, the claim above is obtained. Due to the special shape of the function, the solution set is easy to verify.

Proposition 1 implies that for all $i \in V$, $J_1(z_i)$ is $l_x$ function as long as $J_1(z_i)$ is $l_x$ function for all $j \in C_i$. This invariance of function properties over nodes enables us to apply Proposition 1 repeatedly on $\beta$. On the other hand, Proposition 1 also implies that $z_t^i \leq 0 < z^*_t^i$ if $q^f_t \leq 0 \leq q^a_t$ for all $i \in C_j$. Hence $z_t^i \leq 0 < z^*_t^i$ for all $i \in I$. Given $\beta_i$ for all $i$, the original problem $J_0(0)$ can be then solved in two iterations over the tree. The first iteration from bottom to top provides function $J_2$ and $\chi$ and the second iteration from top to bottom gives values of an solution.

### IV. An Equivalent State Aggregation Approach

In this section, we develop a state aggregation approach to facilitate implementation of search algorithms to solve (5) optimally.

### A. A State Aggregation Approach

The key idea in developing finite representation of the state space is a finite consistent partitioning of the action set to get the set of aggregated control actions. In general, a cover of a set $U$ is a collection of nonempty subsets $\{U_i\}_{i \in I}$ of $U$ such that $U = \bigcup_{i \in I} U_i$ and a partition is a cover with pairwise disjoint elements. We call a cover or partition finite if it contains finitely many elements. Furthermore, for a network state $(E, p)$, a partition $\{U_i\}_{i \in I}$ of set $U$ is said to be consistent if $\mathcal{F}_E(E, u) = \mathcal{F}_E(E, \hat{u})$ for all $u, \hat{u} \in U_i$ and $i \in I$. Consistency implies that it is valid to write $\mathcal{F}_E(E, u) \equiv \mathcal{F}_E(E, U_i)$ for all $u \in U_i$ and $i \in I$. Note here set $U$ is not necessarily the action set $U(E, p)$ and can be arbitrary set of balanced supply-demand vector. At the same time, we extend the notion of the action set defined in (4) as follows: for a link set $\mathcal{E}$ and supply-demand vector set $P \subset \mathbb{R}^V$,

$$
U(\mathcal{E}, P) := \bigcup_{p \in P} U(\mathcal{E}, p) = \mathcal{E} \cap \mathcal{P}
$$

where cube $P := \bigcup_{p \in P} \text{cube } p$.

We now show how finite consistent partition of action set induces a natural finite cover of the state space at each stage in the cascading dynamics. Initially, the state space $(\mathcal{E}^0, P^0)$ is a singleton, $(\mathcal{E}^0, P^0)$ forms a trivial partition with $P^0 := \{0^0\}$. Let $(U^0)^{\mathcal{E}}$ be a finite consistent partition of $(\mathcal{E}^0, P^0)$. Then at $t = 1$, the state space $\{\mathcal{F}_E(E^0, u) \mid u \in U(E^0, P^0)\}$ is covered by $\{E_1^1, P_1^1\}_{i \in I_1}$, where $E_1^1 := \mathcal{F}_E(E^0, U^0_1)$ and $P_1^1 := U_1^1$ for all $i \in I_1$. Let $(U^1)^{\mathcal{E}}_{i \in I_1}$ be a finite consistent partition of $(E_1^1, P_1^1)$ for all $i \in I_1$. Then at $t = 2$, $\{\mathcal{F}_E(E_1^1, u) \mid u \in U(E_1^1, P_1^1)\}$ is covered by $\{E_2^1, P_2^1\}_{j \in I_2}$ for all $i \in I_1$, where $E_2^1 := \mathcal{F}_E(E_1^1, U^1_2)$ and $P_2^1 := U^1_2$ for all $j \in I_2$. Thus the state space $\{\mathcal{F}_E(E_2^1, u) \mid u \in U(E_2^1, P_1^1)\}$ at $t = 2$ is covered by $U(E_2^1, P_2^1)$. Repeated application of this procedure to all the subsequent stages then gives the desired finite representation. We thus employ the elements $(E^t, P^t)$ of the cover as the aggregated states at $t$ and obtain the following dynamics over the aggregated states as an extension of (2), for $0 \leq t \leq N - 1$:

$$
(E^{t+1}, P^{t+1}) = \mathcal{F}(E^t, P^t, U^t), U^t \in \mathcal{U}(E^t, P^t) \quad (13)
$$

where $\mathcal{F}_p(E, P, U) \equiv \mathcal{F}_p(U) := U$ and $\mathcal{F}_E(E, P, U) \equiv \mathcal{F}_E(E, U) := \{i \in E \mid -c_i \leq f_i(E, u) \leq c_i \text{ for all } u \in U\}$. $\mathcal{U}(E^t, P^t)$ is defined by a consistent partition of $(E^t, P^t)$ and serve as the aggregated control space at $t$ in (13).

We now describe the partition $\mathcal{U}(E, P)$ used in this paper. We associate the link set $E$ with a vector $\beta \in \{1, 0, -1\}^E$. For a given state $(E, P)$ and $\beta$, let $U(E, P, \beta) := \{u \in U(E, P) \mid f_i(E, u) < -c_i \text{ for } \beta_i = 1 \}$ and $U(E, P, \beta) := \{u \in U(E, P) \mid f_i(E, u) > c_i \text{ for } \beta_i = 1 \}$ and $U(E, P, \beta) := \{u \in U(E, P) \mid f_i(E, u) = c_i \text{ for } \beta_i = 1 \}$. With this, we define

$$
U(E, P) := \{U(E, P, \beta^i) \mid \beta \in \{1, 0, -1\}^E \} \quad (14)
$$

where $I(E, P) := \{j \mid U(E, P, \beta^j) \neq \emptyset\}$. It is straightforward to see that $U(E, P)$ defined in (14) is a consistent
By aggregating the states, one gets a finite aggregated search tree (cf. Section I-C), with each node being an aggregated state and each arc being an aggregated control action. The set of goal states or steady state is $\tilde{S} := \{(E, p) \mid (E, p) \in S, \forall p \in P\}$. Each state $(E, p)$ is assigned the value: $r(E, p) := \max_{p \in cl(P)} d^T p$ for $(E, p) \in \tilde{S}$ and $r(E, p) := -\infty$ else, where $cl(P)$ denotes the closure of $P$. $P$ may not be closed, but since it is bounded and the function $d^T p$ is continuous, we have $\max_{p \in cl(P)} d^T p = \sup_{p \in P} d^T p$.

One can conduct tree search over the aggregated states to solve the $N$ stage load shedding problem. Similarly, let $J_t(E, P)$ be the value one could get from state $(E, P)$ with $t$ stages to go, then for $J_0(E, P) = r(E, P)$ and for $t \in [N]$:

$$J_t(E, P) = \max_{U \in U_{t-1}(E, P)} J_{t-1}(F_E(E, U), U) \tag{15}$$

Maximization is used in (15) because set $U(E, P)$ is finite. For $t = 1$, $J_1(E, P) = J_0(E, U(E, cl(P), 0)) = r(E, U(E, cl(P)), 0)$. Since by definition $U(E, cl(P), 0) = \{u \in U(E, cl(P)) \mid -c_e \leq f(E, u) \leq c_e\}$, $J_1(E, P) = \max_{u} d^T u$

$$\text{s.t. } -c_e \leq f(E, u) \leq c_e; u \in U(E, cl(P)) \tag{16}$$

where a linear function is to be maximized over a bounded closed set and hence the optimal value is achievable.

Likewise, the problem is to find an aggregated action sequence $(U_0, \ldots, U_{N-1})$ that starts from the initial aggregated state $(E^0, \{p^0\})$ and gives the value $J_N(E^0, \{p^0\})$. The next result shows that the iterations in (15) give the same value as that in (6).

**Theorem 1:** For a network with initial active link set $E^0$ and initial supply-demand vector $p^0$, let $J_t(E, P)$ and $J_t(E, P)$, for $t \in [N]$, be the value functions determined by (6) and (15), respectively. Then $J_t(E, P) = \sup_{p \in P} J_t(E, P)$ for all $t \in [N]$.

**Proof:** The statement is proved by induction. For $t = 1$,

$$\sup_{P \in P} J_1(E, P) = \sup_{p \in P} \max_{u} d^T u \text{ s.t. } -c_e \leq f(E, u) \leq c_e; u \in U(E, P)$$

which is equal to $J_1(E, P)$ as shown in (16).

Suppose the claim is true for all $t \leq k$, we now prove it is true for $t = k + 1$.

$$J_{k+1}(E, P) = \max_{U \in U(E, P)} J_k(F_E(E, U), U) = \max_{U \in U(E, P)} \sup_{u \in U} J_k(F_E(E, u), u) = \sup_{u \in U} J_k(F_E(E, u), u) = \sup_{u \in U} J_{k+1}(E, u)$$

where the first equality is due to (15), the second equality is due to the induction assumption; the third and forth equalities are due to $\cup_{U \in U(E, P)} U = U(E, P) = \cup_{p \in P} U(E, P)$, as implied by the definitions of $U(E, P)$ and $U(E, P)$; the last equality is due to (6).

**B. Optimal control action**

The actual numerical implementation of (15) would be shown in Section IV-C. For the moment, we assume the optimal aggregated action sequence $U^{[N]}$ is known. We want to derive from $U^{[N]}$ an optimal action sequence of supply-demand vectors $u^{[N]}$, which can then be implemented to control the actual (i.e., unaggregated) cascading dynamics. However, $u^{[N]}$ may not exist because $D$ may not be closed, as explained in Section II-B. The following result shows that one can always find a solution $\tilde{u}^{[N]}$ whose cost is arbitrarily close to that of $u^{[N]}$.

**Proposition 2:** For a network with initial state $(E^0, p^0)$, let $J_N(E^0, p^0)$ be as defined in (15). Then, for every $\epsilon > 0$, there exists $\tilde{u}^{[N]} \in D$ (cf. (5)) such that $J_N(E^0, p^0) \geq d^T \tilde{u}^{[N]} - \epsilon$.

**Proof:** Theorem 1 implies that $J_N(E^0, p^0) \geq d^T \tilde{u}^{[N]} - \epsilon$ for all $\tilde{u}^{[N]} \in D$. We only show the other half of the inequality.

Let $U^{[N]}$ be an optimal aggregated control sequence that satisfies (15) and $\tilde{E}^{[N]}$ be the induced active link set sequence. Let $u^{N-1,*} \in cl(U^{N-1,*})$ and $J_N(\tilde{E}^{[N]}, p^0) = d^T u^{N-1,*}$. We now show that for arbitrary $\epsilon > 0$, there exists $\tilde{u}^{[N]} \in D$ (cf. (5)) such that $d^T \tilde{u}^{[N]} \geq d^T u^{N-1,*} - \epsilon$.

Let $M(u, \epsilon)$ be the open ball centered at $u \in \mathbb{R}^\ell$ and of radius $\epsilon$. Since $u^{N-1,*} \in cl(U^{N-1,*})$, $U^{N-1,*} \cap M(u^{N-1,*}, \epsilon/|V_i|) \neq \emptyset$ for all $\epsilon > 0$. Then it is possible to pick $\tilde{u}^{N-1} \in U^{N-1,*} \cap M(u^{N-1,*}, \epsilon/|V_i|)$ so that $\tilde{u}^{N-1} \neq u^{N-1,*}$ and $d^T \tilde{u}^{N-1,*} > d^T u^{N-1,*} - \epsilon$. It is then sufficient to show that there exists $\tilde{u}^{0}, \tilde{u}^{1}, \ldots, \tilde{u}^{N-2}$ such that $\tilde{u}^{t+1} \in \text{cube } \tilde{u}^t$ and $\tilde{u}^t \in U^{[t]}$, for all $0 \leq t \leq N - 2$. We now show how to find $\tilde{u}^{N-2}$ satisfying the conditions. As $u^{N-1,*} \in U(\tilde{E}^{[N-1]}, cl(U^{N-2,*}))$, there exist $u^{N-2,*} \in cl(U^{N-2,*})$ such that $u^{N-1,*} \in \text{cube } u^{N-2,*}$. Hence we can pick $\tilde{u}^{N-2}$ to be an interior point of $U^{N-2,*} \cap M(u^{N-2,*}, \|u^{N-1,*} - \tilde{u}^{N-1}\|_2)$ so that $\tilde{u}^{N-2} \neq u^{N-2,*}$ and $\tilde{u}^{N-1} \in \text{cube } \tilde{u}^{N-2}$, where the special choice of $\tilde{u}^{N-1} \neq u^{N-1,*}$ ensures
that } M(u_{N-2,*}, \| u_{N-1,*} - u_{N-1,1} \|_2) \text{ has positive radius, } u_{N-3}, \ldots, u_0 \text{ can be obtained in the same way. }

Proposition 2 implies that, in order to find } u_{\mathcal{N},*} \text{, it is sufficient to solve } U(\mathcal{N},*) \text{ to } (15) \text{ and } u_{N-1,*} \text{ to } (16). \text{ To find } U^{\mathcal{N},*} \text{, for sake of convenience in the numerical implementation, we use } (E, e, P) \text{ and } cl(U) \text{ to replace each } (E, P) \text{ and } U \text{ as the new aggregated state and control action. It does not have considerable influence on the optimal aggregated control sequences. Equivalently, we use the following variant of definition of } U(E, P, z) := \{ u \in U(E, P) | f_1(E, u) \leq -c_1 \text{ if } \beta_1 = -1; -c_1 \leq f_3(E, u) \leq c_1 \text{ if } \beta_1 = 0; f_1(E, u) \geq c_1 \text{ if } \beta_1 = 1; \forall i \in E \} \text{.}

To find } u_{\mathcal{N},*} \text{, the next result implies that } (16) \text{ is a linear program by showing that, for every state } (E, P), \text{ the set } U(E, P) \text{ is a polytope.}

Lemma 1: For an initial state } (E^0, P^0), \text{ let } (E', P') \text{ be an arbitrary aggregated state at } t \in [N-1] | \cup \{0\} \text{ in the dynamics } (15). \text{ Then both } P^t \text{ and } U(E', P') \text{ are polytopes.}

Proof: The claim is proved by induction. It is clear that at } t = 0, P^0 = (p^0) \text{ and } U(E^0, P^0) = U(E^0, p^0) \text{ are polytopes. Suppose for any aggregated state } (E^{t-1}, P^{t-1}) \text{ at } t - 1, P^{t-1} \text{ and } U(E^{t-1}, P^{t-1}) \text{ are polytopes, it is sufficient to show that for arbitrary aggregated control action } U \in U(E^{t-1}, P^{t-1}), \text{ the resulting } P^t \text{ and } U(E^t, P^t) \text{ are polytopes. } P^t = U(t - 1) = U(E^{t-1}, P^{t-1}, \beta) \text{ for some } \beta. \text{ By the definition of } U(E^{t-1}, P^{t-1}, \beta) \text{ and the induction assumption that } U(E^{t-1}, P^{t-1}) \text{ is a polytope, } P^t \text{ is a polytope. It follows then by definition } U(E, P) \text{ is a polytope as well.}

C. Aggregated tree search and implementation using incidence graph

Iterations in (15) describe the process of constructing the aggregated tree to find a goal state, i.e., leaf nodes in the aggregated tree, with the maximal value. Each path corresponds to a possible network topology sequence in cascading dynamics. Iterations in (15) provide a procedure to systematically evaluate all the possible scenarios. While any classical tree search algorithm, e.g., in [12, Chap 3], can be used, we wish to make the most of the following bounds for tree pruning: } J_1(E, P) \leq J_r(E, P) \leq r(E, P) \forall t \in [N]. \text{ To this end, iterative deepening depth-first search algorithm [12, Chap 3] is preferable for our problem. Firstly, with limited memory consumption, it leads to an efficient transversal over the goal states which contributes to the lower bound, as each state that is not a goal state includes a goal state as child. Secondly, the search can be stopped anytime in the process of computation but still provide feasible control actions that is reasonably well-performed, since the search over the first } t < N \text{ layer provides an optimal } t \text{-stage load shedding scheme. At the same time, the upper bound provides an estimate of performance gap when search is terminated early.}

In order to proceed with numerical implementation of the tree search algorithm, proper numerical representation is required for both } U \text{ and } U(E, P). \text{ As Lemma 1 shows, } U \text{ is polytope and } U(E, P) \text{ is a partition of polytope } U(E, P). \text{ We hence use the construct of incidence graph, e.g., see [5] [6, Chapter 24], to represent these objects. Incidence graph is often used to conveniently represent an arrangement of hyperplanes, which is closely related to } U(E, P), \text{ as we describe next.}

Given a finite collection } \mathcal{H} \text{ of hyperplanes in } \mathbb{R}^d \text{, the arrangement } \mathcal{A}(\mathcal{H}) \text{ is the dissection of } \mathbb{R}^d \text{ into connected pieces of various dimensions induced by } \mathcal{H} [5] [6, Chapter 24]. \text{ In our problem, the constraints for flow capacity, balanced condition and load shedding requirement form hyperplanes. Fig. 4a provides an illustration for a network with initial supply-demand vector } p^0, \text{ } \forall s = \{1, 2\} \text{ and } \forall d = \{3\}, \text{ where the box } ode, \text{ the projection of action set } cube p^0 \text{ on the plane } u_3 = u_1 + u_2. \text{ Hence, } d = 2 \text{ and the hyperplanes are lines. The three solid lines are the flow constraints for the initial over-capacitated links and they separate the plane into seven 2-faces, nine 1-faces and three 0-faces, where we call a } k \text{ dimensional connected piece in an arrangement a } k \text{-face. In particular, 0-face, 1-face, } (d-2)-\text{face, } (d-1)-\text{face and } d \text{-face are called vertex, edge, ridge, facet and cell, respectively. Each of the nine cells inside the box in Fig. 4a corresponds to an aggregated control action. } U(E, P) \text{ is seen as a collection of the cells inside the polytope } U(E, P). \text{ In order to represent this collection, incidence graph uses a node to denote a face and an edge between two nodes to denote their incidence relationship, where two faces are called incident if one is contained in the boundary of the other and their difference in dimensions is one. Fig. 4b shows the substructure of the incidence graph that is associated with the cell } abc \text{ (or action } U_1 \text{). As can be seen, } acb \text{ is incident to three edges and each of the three edges is incident to two vertices.}

In order to implement the tree search using incidence graph, one needs to execute the following two steps at a state } (E, P): \text{ (i) construct the incident graph of } U(E, P), \text{ As it is intersection of } cube P \text{ with hyperplanes, it is sufficient to focus on constructing } cube P; \text{ and (ii) obtain } U(E, P) \text{ by constructing the arrangement of hyperplanes associated with the current capacity and load balance constraints.}

Initially, } cube p^0 \text{ is a hypercube and its incidence graph can be obtained conveniently, as shown in Fig. 4c. Existing algorithms, e.g., in [5] can then be used to compute the arrangement by incrementally adding hyperplanes to } cube p^0.\text{ In addition, } 4u_3 = u_1 + u_2 \text{ represents load balance condition, and the projection is } cube (p^0_1, p^0_2) \subset \mathbb{R}^2.\text{ Fig. 4: (a) Projection of arrangement on hyperplane } u_3 = u_1 + u_2; \text{ (b) Incidence graph of } U(E, p); \text{ (c) Incidence graph of } U_1.\text{
$U(\mathcal{E}^0, p^0)$ is obtained as a result. Then tree search picks one action, say $U_1$, whose incidence graph is at hand already as cell substructures, see Fig. 4c. From $U_1$, one has to construct cube $U_1$. This construction is more complex than cube $p^0$, especially in high dimensions. We provide a novel recursive algorithm for doing this.

In general, for a polytope $P \subset \mathbb{R}^d_{\geq 0}$ and $k \in [d]$, we define: $\text{proj}_k P := \{x \in P \mid x - x_k e_k\}$ and $\text{sweep}_k P := \{x - \theta_k x_k e_k \mid x \in P, \theta_k \in [0, 1]\}$. As shown in Fig. 5a, the projection of cell $abc$ along $e_2$ is the edge $c'b'$ and the trajectory of this projection is $\text{sweep}_2(U_1)$. When conducting projection of a polytope, its facets are classified as three types: top (face $ca$ and $ab$), vertical and bottom (face $cb$) facets. Only the ridges between the top and vertical/bottom facets matters to the sweep. They correspond to vertex $b$ and $e$ in Fig. 5b. One first removes the substructure of incidence graph for the bottom and vertical facets of $P$, then forms the projection and hence sweep of these ridges, and finally use them to construct $\text{sweep}_k P$. Fig 5b shows the incidence graph of $\text{sweep}_2(U_1)$, where the black substructure comes from $P$ and the gray part is added in the algorithm. The sweep operation provides a numerical algorithm to construct the incidence graph of the polytope $\text{sweep}_k P$ from that of $P$. While it seems to be straightforward in $\mathbb{R}^2$, it shows power in high dimensional space for which intuition lacks.

![Fig. 5: Illustration of the operation sweep: (a) sweep of $U_1$; (b) the incidence graph of sweep$_2(U_1)$; (c) cube $U_1$ as sweep of sweep$_2(U_1)$.](image)

Recursive application of sweep operation on $P$ gives cube $P$ as: $cube P = sweep_1(sweep_2(\ldots sweep_d(P)))$. For $U_1$, cube $P = sweep_1(sweep_2(P))$ is shown in Fig. 5c. With the incidence graph of cube $P$ at hand, one again incrementally adds hyperplanes which correspond to the updated capacities and load balance constraints, constructs the relevant arrangement, and obtain $U(F_c(\mathcal{E}, U_1), U_1)$. This process goes on in the aggregated tree search algorithm and we are able to obtain numerically a solution.

Remark 3: (i) For a network with single supply and single demand, the hyperplanes are single points and control actions are associated with line intervals. Both the arrangement construction and sweep operation can be conducted very efficiently. (ii) The set $E_i$ used in (8) and (9) can be obtained similarly (see Remark 1).

D. Approximation algorithm via projection

The number of nodes in the aggregated search tree, in general, increase exponentially with the dimension $d := |\mathcal{V}|$, which could be prohibitive for large networks. We now outline a strategy for projection of the state space. Aggregation and search in this projected space then gives an approximation approach.

For a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with supply-demand vector $p$, let $\tilde{p} \in \mathbb{R}^d$ be the collection of nonzero components of $p$, and $u \in U'(\mathcal{E}, \tilde{p}) \subset \mathbb{R}^d$ be the associated action set. It is equivalent to work with $\tilde{p}$ and $U'(\mathcal{E}, \tilde{p})$. Let $\Phi = [\Phi_1, \ldots, \Phi_d] \in \mathbb{R}^{d \times d}$ be an orthonormal (transformation) matrix, and let index set $B$ and $B^c$ be a partition of $[d]$. If one wishes to search within the column space of $\Phi_B$, that is, search among those supply-demand vectors that can be expanded as linear combination of $\Phi_i, i \in B$, then one should only consider the actions in the new action set $U'(\mathcal{E}, \tilde{p}) \cap \mathcal{R}(\Phi_B) = \{u \in U'(\mathcal{E}, \tilde{p}) \mid \Phi^T u = 0, \forall i \in B^c\}$, which is the intersection with $\{B^c\}$ hyperplanes. The dimension of action set is decreased by $|B^c|$. The implementation of this approximation is straightforward. We add these $|B^c|$ additional hyperplanes into the arrangement and only keep the substructure that is inside their intersection. The procedure can be extended to $\Phi$ being arbitrary transformations.

Example 2: Consider a network with initial supply-demand vector $p^0$. By choosing $B^c = \{1\}$ and $\Phi_1 = p^0/\|p^0\|_2$, we get proportional control policies [3, Section 6.1.1], i.e., a class of control policies whose action set at state $(\mathcal{E}, p)$ is $\{\lambda p \mid 0 \leq \lambda \leq 1\}$. The one-dimensional search space resulting from proportional control policy, as shown in Example 2, is favorable for computational purposes. However, the projection-based approximation strategy implies that one could possibly find better control actions, using the same computational budget, by using different projections.

V. Conclusions and Future Work

In this paper, we provided novel computational approaches for optimal control of a particular discrete time dynamics corresponding to cascading failure in DC power networks under controlled load shedding. These approaches make connections with and build upon tools from network optimization and combinatorial geometry. While we provide specific results on optimal and approximate control synthesis within these frameworks, the extensive literature in these other disciplines opens up several avenues for future work.

We plan to extend network decomposition (Section III) to loopy networks. A good starting point is a simple extension of the algorithm that yields, at the minimum, a feasible control action for an arbitrary network. We plan to relax conditions under which the semi-analytic solution approach outlined in Proposition 1 holds true. Finally, we believe that the aggregation method and the associated search methods (Section IV) can be extended to a more general setup, for dynamics evolving over hybrid state space.

REFERENCES