J Comb Optim (2017) 34:931–955
DOI 10.1007/s10878-017-0121-5

Multiple facility location on a network with linear reliability order of edges

Refael Hassin[1] · **R. Ravi**[2] · **F. Sibel Salman**[3]

Published online: 14 March 2017

Abstract We study the problem of locating facilities on the nodes of a network to maximize the expected demand serviced. The edges of the input graph are subject to random failure due to a disruptive event. We consider a special type of failure correlation. The edge dependency model assumes that the failure of a more reliable edge implies the failure of all less reliable ones. Under this dependency model called Linear Reliability Order (LRO) we give two polynomial time exact algorithms. When two distinct LRO's exist, we prove the total unimodularity of a linear programming formulation. In addition, we show that minimizing the sum of facility opening costs and expected cost of unserviced demand under two orderings reduces to a matching problem. We prove NP-hardness of the three orderings case and show that the problem with an arbitrary number of orderings generalizes the deterministic maximum coverage problem. When a demand point can be covered only if a facility exists within a distance limit, we show that the problem is NP-hard even for a single ordering.

Keywords Facility location · Random edge failures · Dependency

This research is supported by a NATO Collaborative Linkage Grant.

✉ F. Sibel Salman
ssalman@ku.edu.tr

Refael Hassin
hassin@post.tau.ac.il

R. Ravi
ravi@cmu.edu

1 Department of Statistics and Operations Research, Tel Aviv University, Tel Aviv, Israel

2 Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA, USA

3 College of Engineering, Koç University, Sariyer, Istanbul, Turkey

1 Introduction

The problem we study aims to locate facilities on a network whose edges are subject to random failure due to a disruptive event such as a natural disaster. Specifically, k facilities should be located at the nodes of an undirected graph whose edges may fail with given probabilities to maximize the expected demand serviced. The expectation is over the possible network realizations and a demand point is serviced/covered in a network realization, if a facility can be reached from it.

Previous work on locating facilities on a network subject to edge failures has remained limited to either single edge failure, or a single facility location on a tree network whose edges may fail independently. Eiselt et al. (1992) have considered a single edge failure while locating p facilities with the objective of minimizing total expected demand disconnected from the facilities, and later the same authors Eiselt et al. (1996) considered the case with an unreliable node or edge. Melachrinoudis and Helander (1996) studied a single facility location problem on a tree with edges that fail independently with given probabilities. The objective was to maximize the expected number of demand nodes reachable by operational paths. A follow-up paper by Xue (1997) gave an improved linear-time algorithm. Colbourn and Xue (1998) gave a linear-time algorithm for this problem on series-parallel graphs. Wolle (2002) addressed the basic problem of calculating the probability of serving all demand points via facilities with given locations, under node and edge failures. In a recent study, Salman and Yucel (2015) studied a multiple facility location problem where edges of the network fail with respect to dependency that takes into account spatial proximity, similar to our study here. In Salman and Yucel (2015), the resulting large number of network realisations are sampled to estimate the objective function in a tabu search heuristic. A case study of Istanbul is analyzed under a likely earthquake scenario, where the edge reliabilities are estimated according to this disaster scenario. This work examplifies an application of the problems studied in the current article.

While independent failures is a common assumption, when edge failures occur due to an exogenous cause such as a disaster, it is often necessary to treat the edge failures as dependent events. We consider the edge failure model defined by Gunnec and Salman (2007) that takes into account both the vulnerability levels of the edge components and the risk of the area in which they reside. For each edge, given the probability that it survives (i.e. its reliability), the edges are ordered linearly starting with the strongest (i.e. the most reliable). Then, it is assumed that given that a particular edge fails, all the edges that are weaker will fail with probability 1. Here, we refer to this dependency structure as *Linear Reliability Order* (LRO). The reliability of an edge is defined under a particular disaster/disruption scenario; hence an LRO is associated with the scenario. When multiple scenarios are under consideration, a reliability vector and a corresponding LRO are associated with each scenario. Then, the facility location problem maximizes the expected demand coverage over all scenarios considering the probability of occurrence of each scenario, and the expected demand serviced in each scenario.

We refer to the problem with d number of distinct LROs as the d-LRO case. We first investigate the case of $d = 1$, i.e. the 1-LRO case. For the 1-LRO case, we provide a transformation to a problem defined on a tree and two exact solution algorithms to the k-facility location problem as well as its generalization with uniform capacities at the facilities. We also provide an exact solution algorithm to the cost minimization version with facility opening costs at the nodes and costs for unmet demand in each network realization. For the 2-LRO case, we prove the total unimodularity of a polynomially-sized Linear Programming (LP) formulation; which implies that the problem is solvable in polynomial time. We show that minimizing the sum of facility opening costs and the expected cost of unserviced demand under 2-LRO reduces to a bipartite matching problem. We prove NP-hardness for the 3-LRO case even when all reliabilities are 0 or 1 (a deterministic problem). We then consider the general d-LRO case and show that the problem is equivalent to a maximum coverage problem with d elements to be covered.

We also investigate the k-facility maximum expected coverage problem with distance limits. When a demand point can be covered only if a facility exists within a specified distance limit, we prove that the problem is NP-hard even for the 1-LRO case since it is equivalent to the well-studied maximum k-facility location problem Cornuejols et al. (1977).

Our results provide tractable problems with edge failures for several facility location problems considering a dependency model based on ordering the edges with respect to their reliabilities. The paper is organized as follows. We introduce the notation and define the k-facility maximum expected coverage problem in Sect. 2. In the case of a single LRO (the 1-LRO case), we present two polynomial time exact algorithms in Sect. 3. In Sect. 4, we give our results for the 2-LRO case. Section 5 presents hardness results on some extensions of the facility location problem. We conclude in Sect. 6.

2 Problem definition

The facility location problem is defined on an undirected input graph $G = (V, E)$, consisting of node set $V = \{v_1, \ldots, v_n\}$ and edge set $E = \{e_1, \ldots, e_m\}$. Each node v_i is a demand point with a given demand value (weight) $w_i \geq 0$, where $\sum_i w_i > 0$. Each edge may exist in either operational or non-operational state after the disaster, which we refer to as the survival or failure of the edge. Let $\xi_i = 1$, if e_i survives and 0, otherwise. The random variable ξ_i takes the value 1 with probability p_i. That is, p_i represents the reliability of edge e_i. We assume that each node will survive and a facility can be located at any node. The problem is to find the locations of at most k facilities that maximize the expected demand covered by the facilities over all network realizations. In a network realization, a demand point is covered if there exists a path from it to one of the open facilities.

After a disruption, the set of edges that have not failed define the *surviving network*, represented by the vector $\xi = (\xi_i)$. In general, the vector ξ has 2^m realizations where each one corresponds to a different surviving network consisting of a number of connected components. If a facility is established at a node, it covers the demand of all nodes that can be reached from it via a path in the surviving network, assuming that a sufficient amount of supply will be available at the facilities. If the locations of the facilities are fixed, in each possible surviving network realization, total demand covered can be evaluated by applying a breadth-first-search starting from each facility node. Note that finding the total expected demand covered requires $O(2^m)$ such calculations. The MAX-EXP-COVER problem is to place at most k facilities to maximize the expected total demand covered.

Let the probability that a particular realization, ξ^q, occurs be $P(q)$ for $q = 0, 1, \ldots, R$. Suppose $F \subseteq V$ represents the selected facility locations. We define an indicator variable to represent the coverage of a demand node in a network realization: $I_v(F, q)$ takes the value 1, if demand of node v can be covered in network realization ξ^q with facilities in F; and 0, otherwise. Note that $I_v(F, q)$ will be 1, if there is a path from v to one of the facilities in F in the subgraph of G induced by the surviving edges of ξ^q. Then, MAX-EXP-COVER can be formulated as follows.

$$\text{MAX-EXP-COVER:} \quad \text{Find} \quad F \subseteq V,\ |F| \le k \quad \text{to}$$
$$\mathit{maximize}\ \sum_{q=0}^{R} \sum_{v_i \in V}\ P(q)\, w_i\, I_{v_i}(F, q).$$

MAX-EXP-COVER problem is NP-hard Feige (1998). Note that it is easy to prove that the objective of MAX-EXP-COVER is a monotone submodular function of the set F, and the problem MAX-EXP-COVER maximizes a submodular function subject to a cardinality constraint.

2.1 Dependent failures with linear reliability order

Under statistical dependency of edge failures, the number of realizations with positive probabilities reduces. We use a concept from the dependency model proposed by Gunnec and Salman (2007). This model first partitions the set of edges such that edges in different sets fail independently. On the other hand, edges in the same set fail according to their order of survival probabilities; this is named VB-dependency in Gunnec and Salman (2007). In the case of failure of road network components under a disaster event, VB-dependency tries to factor in the vulnerability of the components in an edge, such as the strength of a bridge and the soil type on which the edge stands. It is reasonable to assume that edges in close geographic proximity will be prone to a similar disaster magnitude and are expected to show similar behaviour, creating spatial correlation; whereas at the same time their inherent vulnerabilities will create differences in the outcomes.

Definition 2.1 Given two edges i and j with survival probabilities p_i and p_j, edges i and j have *Vulnerability-based dependency (VB-dependency)*, if $p_i \le p_j$ implies probability that i fails given that j fails is 1.

Definition 2.2 If the VB-dependency holds for every pair of edges in E, then E has a *Linear Reliability Order (LRO)* with respect to the reliability vector $p = (p_i)$.

By the above definitions, the failure of a particular edge implies failure of all edges as weak as, or weaker than that one. It was shown in Gunnec and Salman (2007) that under an LRO failure model, only at most $m + 1$ surviving network realizations have positive probability. When the edges are re-indexed such that $p_{i-1} \ge p_i$, for $i = 2, \cdots, m$, index 1 represents the strongest edge and m the weakest. Then, the realizations with positive probability are in the form $\xi^q = (1, 1, 1, \cdots, 1, 0, 0, \cdots, 0)$, where the first strongest q edges survive and the remaining fail, for $q = 0, 1, \cdots, m$. Note that here index q indicates the number of edges that have survived in the corresponding realization. We use this ordering of the edges when we refer to an LRO throughout this article.

If we define $p_0 = 1$ and $p_{m+1} = 0$, then for $q = 0, 1, \cdots, m$ the probability that realization ξ^q occurs is $p_q - p_{q+1}$ (as given in Gunnec and Salman (2007)). Although this failure model is rather restrictive in the number of scenarios when a single set exists, at the same time it enables tractability.

When multiple disruption scenarios exist, a set of LROs, S, is given, where $\Pr(s)$ is the probability that scenario s occurs for $s \in S$, with $\sum_{s \in S} \Pr(s) = 1$. Each scenario s defines an LRO with respect to a given edge reliability vector $p^s = (p_1^s, \cdots, p_m^s)$. Let $P(q, s)$ be the probability that ξ^q occurs in s. We can verify that $\sum_{q=0}^{m} P(q, s) = 1$, since $P(q, s) = p_q^s - p_{q+1}^s$ and $p_0^s = 1$, $p_{m+1}^s = 0$ by definition. Then, the main problem addressed in this paper, MAX-EXP-COVER-LRO, can be formulated as follows.

$$\begin{aligned} &\text{MAX-EXP-COVER-LRO:} \quad \text{Find} \quad F \subseteq V,\ |F| \leq k \quad \text{to} \\ &\qquad\qquad\qquad \mathit{maximize} \quad \sum_{s \in S} \Pr(s) \sum_{q=0}^{m} \sum_{v_i \in V} P(q, s)\, w_i\, I_{v_i}(F, q). \end{aligned}$$

We show the NP-hardness of this problem in Sect. 5. We next investigate the single and two LRO cases due to the complexity of the general case.

3 Algorithms for a single linear reliability order

In this section we assume a single LRO exists with respect to a reliability vector p and omit the scenario index s in the notation. We first show that under 1-LRO, the MAX-EXP-COVER-LRO problem can be reformulated as a facility location problem on a rooted tree network.

3.1 Graph disconnectors and the component tree

Without loss of generality we may assume that the input graph G is connected. The possible network realizations are $\xi^q, q = 0, 1, \cdots, m$, where ξ^m corresponds to G. As we go from ξ^m (all ones) to ξ^0 (all zeros) one by one, each time one more edge fails. Along the way, the number of components in each realization will increase until eventually we get to ξ^0 that consists of n components. We can detect the edges whose failure causes the number of components to go up by 1. Start with the weakest edge. Suppose it fails. Check if the graph is connected. Repeat until the graph has two components. At that point, the last edge that failed is named $e_{[1]}$. Continue in this way. The next edge that increases the number of connected components is named $e_{[2]}$, and so on. By this procedure, we obtain the edges $e_{[1]}, \ldots, e_{[n-1]}$, and call them the *disconnectors* of G. Note that $e_{[1]}$ is the *weakest* and $e_{[n-1]}$ is the *strongest* disconnector. Hence, $[1], \ldots, [n-1]$ defines a new ordering on $n-1$ edges, such that $p_{[i]} \leq p_{[i+1]}$.

The disconnectors form a spanning tree T of G. We can see this by the following reasoning. Suppose the disconnectors form a cycle. Then by removing one of the edges in this cycle, we can still have the same number of components. Hence, the removed edge cannot be a disconnector. Since the disconnectors separate all nodes eventually, T must be a spanning tree of G. In fact, T is a Minimum Spanning Tree (MST) of G when weight of each edge e_i is set to $1 - p_i$, since for any edge not in the tree, its failure probability is at least as much as those in the path between its endpoints in T.

Next, we define a rooted tree that represents how G is disconnected into its components as the disconnectors $e_{[1]}, \ldots, e_{[n-1]}$ fail in order, starting with the weakest one. We call this tree the *Component Tree* of G, and denote it by CT. The leaves of CT are the nodes of G. The intermediate nodes of CT represent the disconnector edges, and the root node of CT is the first disconnector, $e_{[1]}$. Each disconnector has two children in CT, representing both the subtrees of T and the components of G formed with its failure. CT is constructed as follows. Let $e_{[1]}$ be the root node. When $e_{[1]}$ fails, it creates two components in T. If any one of the components is a singleton, then that child is given the name of the singleton node and becomes a leaf node. Otherwise, the child is given the name of the next disconnector to fail in that component (i.e. the one with the minimum index). For each non-leaf node $e_{[k]}$, two children are created by removing $e_{[k]}$ from T in the same way. CT is completed when n leaves are formed. In total, constructing the component tree takes $O(m \log n + T(n))$ time, where $T(n)$ is the time to compute the MST on the input graph G. Sorting the edges of G is followed by an MST algorithm and finally by the removal of each edge of T in order, to obtain the binary structure of CT. The procedure is illustrated by an example in Fig. 1, where the numbers on the edges of the input graph indicate the edge indices after the edges have been sorted according to their reliabilities.

Interestingly, the Component Tree contains the connectivity information for nodes over all network realizations. Each node of CT indicates a set of nodes that are in the same component of G in one or more network realizations. For each node v of CT, let us denote the subtree of CT rooted at v by T_v. Let $L_v \subset V$ be the set of leaf nodes in T_v. If v is a leaf of CT, then it represents the singleton component. Else, v is a disconnector that represents all realizations in which L_v forms one component. Thus, if we pick two leaves, the path between them in CT shows when these nodes will be connected. Namely, when the disconnector with the smallest index on this path survives, these nodes are connected. For example, in Fig. 1, nodes 2 and 6 become connected when [4], i.e. edge 4, survives and continue to be connected if the weaker edges survive.

3.2 Reformulation of MAX-EXP-COVER-LRO

We associate two quantities with each node v of the Component Tree CT: a weight $W(v) = \sum_{v_i \in L_v} w_i$ and a probability $rel(v)$. We set the $rel(v)$ values such that if there is a facility located at a leaf node of T_v, a partial expected demand/weight of $rel(v)W(v)$ accrues in the objective function. We denote this *Partial Expected Weight* as $PEW(v) = rel(v)W(v)$. Next we define the $rel(v)$ values.

Consider a leaf node v of CT that corresponds to some node v_j in V. Let $f(v)$ denote the father of v in CT; $f(v) = e_{[i]}$ for some i. The node v_j is left as a singleton component in G when the disconnector $e_{[i]}$ fails. If $e_{[i]}$ has the original edge index q, then we set $rel(v) = 1 - p_q$.

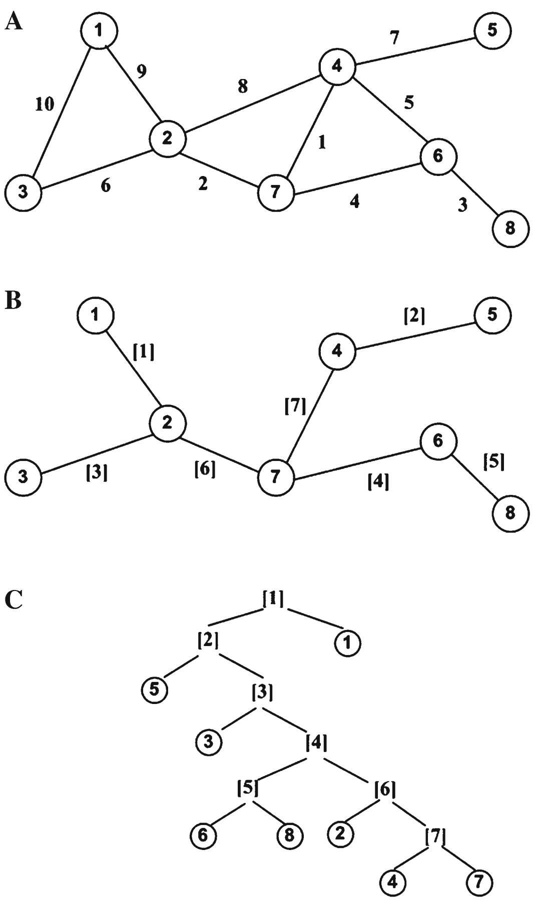

Fig. 1 An example to illustrate the construction of the component tree. **a** Input graph G. Edges are indexed such that $p_1 > p_2 > \ldots > p_{10}$. **b** Tree T consisting of disconnector edges $e_{[1]}, \ldots, e_{[7]}$. **c** Component Tree CT

At any non-leaf node v of CT corresponding to a disconnector edge $e_{[i]}$, we have two children, say L and R. The probability $rel(e_{[i]})$ is set to the sum of the probabilities of all realizations in which all the leaves in subtrees T_L and T_R are exactly the nodes of a single connected component. This connected component is formed when the father of $e_{[i]}$, $f(e_{[i]})$, fails for the first time and it remains to be a single component until $e_{[i]}$ fails. Suppose $e_{[i]} = e_q$ and $f(e_{[i]}) = e_t$. By notation, $q < t$, i.e. e_q is stronger. Then, $rel(e_{[i]})$ is set to $P(q) + P(q+1) + \cdots + P(t-1) = p_q - p_t$.

Finally at the root of the tree, we set $rel(e_{[1]})$, namely the probability of the realizations in which all nodes are in a single component as follows. Let $e_{[1]} = e_q$. Then, $rel(e_{[1]}) = P(q) + P(q+1) + \cdots + P(m) = p_q$.

In the above reformulation, the probabilities are assigned such that in any path from a leaf node to the root, if we sum the $rel(v)$ values, we get 1. For instance in the example given in Fig. 1, $rel([1]) = p_9$, $rel([2]) = p_7 - p_9$, $rel(1) = 1 - p_9$, $rel(5) = 1 - p_7$, $rel([3]) = p_6 - p_7$, $rel(3) = 1 - p_6$, $rel([4]) = p_4 - p_6$, $rel([5]) = p_3 - p_4$, $rel([6]) = p_2 - p_4$, $rel(6) = 1 - p_3$, $rel(8) = 1 - p_3$, $rel(2) = 1 - p_2$, $rel([7]) = p_1 - p_2$, $rel(4) = 1 - p_1$, and $rel(7) = 1 - p_1$.

With the above reformulation, we can view the problem of locating k facilities at nodes as a simple location problem on a rooted tree $T = (V_T, E_T)$ with nonnegative weights $PEW(v)$ at all nodes: the goal is to choose k leaves such that the sum of the *PEW* values of all nodes in the union of paths from the chosen leaves to the root is maximum. We call this the MAX-WT-k-LEAF-SUBTREE problem. Suppose a set F consisting of k leaf nodes of T have been selected. For a given internal node v of the tree, let $I_v(F)$ be 1, if v is contained in the path from l to the root node, for at least one of the leaf nodes l in F; and 0, otherwise. Then, the MAX-WT-k-LEAF-SUBTREE problem is formulated as follows.

MAX-WT-k-LEAF-SUBTREE: Given a rooted tree $T = (V_T, E_T)$ with nonnegative weights $PEW(v)$ for $v \in V_T$, find a subset F of leaf nodes in T such that $|F| = k$ and $\sum_{v \in V_T} PEW(v) I_v(F)$ is maximum.

As a result of this reformulation we have the following proposition.

Proposition 3.1 *Given an undirected graph $G = (V, E)$ whose edges fail according to a single LRO, the MAX-EXP-COVER-LRO problem can be solved by solving a MAX-WT-k-LEAF-SUBTREE problem on the component tree CT of G with weights $PEW(v)$, for v in CT defined as above.*

Proof Suppose F, a subset of k leaf nodes in *CT* are chosen as the facility locations. Let G_F be the subtree of CT that consists of the union of paths from the chosen leaves F to the root. We will show that $\sum_{v \in G_F} PEW(v)$ is the total expected weight covered by the solution F. Recall that a node v of CT represents all realizations where L_v forms a component and the probability of these realizations is $rel(v)$. Let us call these realizations R_v for now. If $v \in G_F$, then L_v must contain a facility node. Since L_v is connected in R_v, all of the demand of L_v, i.e. $W(v)$, is covered with probability $rel(v)$. Hence, $PEW(v)$ should be added to the objective function. On the other hand, if v is not in G_F, then L_v cannot contain a facility. Since nodes in L_v are not connected to any node outside L_v in R_v, their demand cannot be covered in these realizations. Thus, summing $PEW(v)$ over $v \in G_F$ is sufficient to get the total expected weight covered by F. □

Next, we provide a polynomial time exact solution method for the MAX-WT-k-LEAF-SUBTREE problem via a greedy algorithm. By Proposition 3.1, the algorithm is also a polynomial time exact solution method for the MAX-EXP-COVER-LRO problem for the 1-LRO case.

3.3 A greedy algorithm

The greedy algorithm for choosing k leaves to maximize total expected weight of the paths to the root is natural: For k steps, we pick the leaf such that the incremental addition to the total expected weight by adding this node to the solution is as large as possible.

Proposition 3.2 *The greedy algorithm outputs an optimal set of k leaves that maximizes the total node weight of the union of paths from the chosen leaves to the root.*

Proof The proof is by an interchange argument. Suppose that the greedy algorithm results in a solution GREEDY that is sub-optimal and has lower weight than the optimal solution OPT. Let v be the first leaf chosen by GREEDY which is not part of OPT. Let P_v be the path that was first covered by v at the time it was added to GREEDY. If no part of P_v is covered by OPT, then clearly by the greedy choice exchanging v with any leaf in OPT\GREEDY gives a solution at least as good as OPT. Suppose otherwise that some part of P_v is covered by OPT. Let P'_v be the subpath of P_v covered by v but not OPT. Let u be the lowest node of P_v covered by some leaf, say l of OPT. Let P_l be the $l-u$ path covered by l. Then by the greedy choice, the weight of P'_v is at least as large as the weight of P_l. Thus, exchanging v and l gives a solution not worse than OPT. By repeating this exchange step, we see that GREEDY is optimal. □

The above proof generalizes the results of Pardi and Goldman (2005) and Steel (2005) known for a similar unrooted version of the problem (which can be reduced to our rooted version by trying all choices of the root). These results in turn follow also from the Greedoid framework Korte et al. (1991).

We note that the objective function is a submodular function of the set of facilities. Therefore, it follows easily that the greedy algorithm solves the problem optimally in the 1-LRO case.

However, the component tree structure provides insight on the problem structure. The greedy algorithm is $O(kn)$ after the construction of the component tree, as in each step the nodes are traversed at most once. We also note that there exists an alternative algorithm by a dynamic programming formulation. It is presented in "Appendix 1".

4 Algorithms for two LROs

We now consider the case where two disruption scenarios are possible with $S = \{1, 2\}$ and given probabilities Pr(1) and Pr(2) that add up to 1. We have two probability vectors p^1 and p^2 such that each one induces a distinct LRO. The problem is MAX-EXP-COVER-LRO as defined in Sect. 2 except that here $\mid S \mid= 2$.

By a transformation as in the previous section, this problem can be reduced to choosing k nodes among n nodes such that these n nodes are the leaves of two trees T_1 and T_2, namely the Component Trees for $S = \{1, 2\}$. Here, T_s has at its nodes v a weight $PEW^s(v)$, defined as in Sect. 3, with respect to the reliability vector p^s, for $s = 1, 2$. Let V_1 and V_2 denote the *internal* (non-leaf) nodes of the two trees T_1 and T_2, respectively. Then, $T_1 \cup T_2$ has the common set of leaves $L = V$. An illustrative example is given in Fig. 2. The objective is to choose k common leaves so that the total weight of the subtrees induced in both trees by these k leaves is as large as possible. We denote this problem MAX-WT-k-LEAF-2-SUBTREES.

4.1 Polynomial-time solution by total unimodularity of an LP formulation

We next show that MAX-WT-k-LEAF-2-SUBTREES can be formulated as a Linear Program (LP) which can then be shown to have an integral optimal solution.

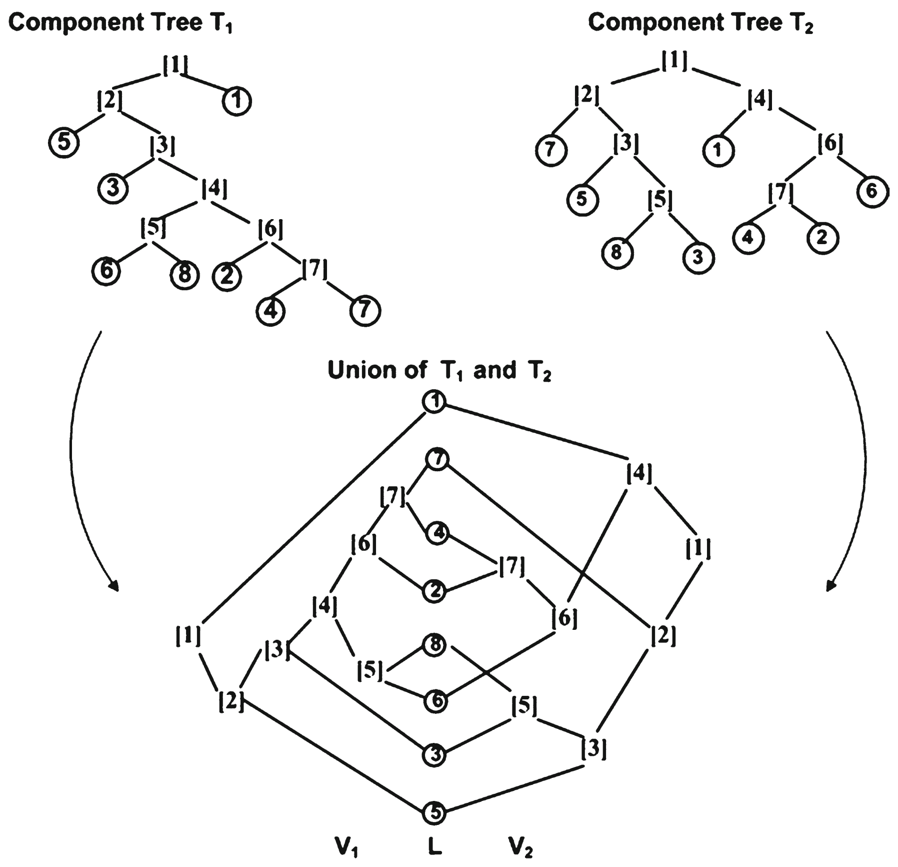

Fig. 2 An example to the 2-LRO case

We use an indicator variable x_l for each leaf node $l \in L$ to denote if it is chosen in the set of k leaves. For every internal node $v \in V_1 \cup V_2$, we have a variable x_v denoting if this node is covered by a facility (i.e. it is on a path from a selected leaf to the root) and thus its weight needs to be counted in the objective function. Let T_v denote the subtree of the tree in which v is an internal node and let C_v denote the children of v in T_v. In addition, the weight of each node $v \in V_s$ is set to $w_v = PEW^s(v)$ for $s = 1, 2$ and for each leaf $l \in L$, $w_l = PEW^1(l) + PEW^2(l)$. We have the following LP relaxation of the 0–1 formulation for the problem.

$$\text{(LP1)} \quad \text{Maximize} \quad \sum_{v \in V_1 \cup V_2} w_v\, x_v + \sum_{l \in L} w_l\, x_l \tag{1}$$

$$s.t. \quad x_v - \sum_{j \in C_v} x_j \le 0 \quad \forall\, v \in V_1 \cup V_2 \tag{2}$$

$$\sum_{l \in L} x_l = k \tag{3}$$

$$0 \le x_v \le 1 \qquad \forall\, v \in V_1 \cup V_2 \cup L \tag{4}$$

Let A be the constraint coefficient matrix of LP1 (excluding the upper bound constraints). We use the Ghouila–Houri Condition for total unimodularity of a constraint matrix A [as given in page 542 of Nemhauser and Wolsey (1988)].

Definition 4.1 *(Ghouila - Houri Condition)* A matrix A with entries $a_{ij} \in \{0, 1, -1\}$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n$ is totally unimodular if and only if for every $J \subset N = \{1, \ldots, n\}$, there exists a partition $J_1,\ J_2$ of J such that $|\sum_{j \in J_1} a_{ij} - \sum_{j \in J_2} a_{ij}| \leq 1$, for $i = 1, \ldots, m$.

Proposition 4.1 *The LP formulation LP1 for the MAX-WT-k-LEAF-2-SUBTREES problem has a totally unimodular constraint coefficient matrix.*

Proof We show that the Ghouila - Houri Condition holds for A^T, where A is the constraint coefficient matrix of LP1. That is, we show that given any subset I of constraints, there exists a partition $I_1,\ I_2$ of I such that $|\sum_{i \in I_1} a_{ij} - \sum_{i \in I_2} a_{ij}| \leq 1$, for $j = 1, \ldots, n$, i.e. for each variable. We have two cases depending on whether constraint (3), i.e. the cardinality constraint, is included in I or not.
Case 1 Constraint (3) is included in I. In this case, let $I_1 = I$ and let I_2 be empty.

For a leaf node variable x_l, I may contain at most two of constraints (2) where the coefficient of x_l is non-zero. Once, for a node v in V_1 for which l is a child and another for a node w in V_2 for which l is a child. In both cases the coefficient of x_l is -1, but x_l also appears in constraint (3) with a +1 coefficient. Hence, the summation of its coefficients across the rows in I is -1, 0 or 1.

A non-leaf node v is either in V_1 or V_2. Suppose without loss of generality, it is in V_1. Consider the constraints where the coefficient of x_v is non-zero. These are constraints (2) where v is a child node with coefficient -1 (except for the root node), and a father node with coefficient +1. Then, I may contain at most two of these constraints so that the summation will be again -1, 0 or 1.
Case 2 Constraint (3) is *not* included in I. In this case, let $I_1 \subseteq I$ contain constraints (2) if $v \in V_1$ and $I_2 \subseteq I$ contain constraints (2) if $v \in V_2$.

Any variable $x_l,\ l \in L$ may appear in constraints (2) as a child of a node in V_1 or V_2. Then, sum of the coefficients of x_l in I_1 is 0 or -1, and the same holds for I_2. Thus the total value of such coefficients of x_l is either -1, 0 or +1.

For a non-leaf node v, x_v appears in constraints (2), which may be in either I_1 or I_2, but not both. Recall that these constraints are where v is a child node with coefficient -1 (except for the root node), and a father node with coefficient +1. Then, I_1 or I_2 may contain at most two of these constraints so that the summation will be again -1, 0 or 1. □

As a result of Proposition 4.1, an integer optimal solution is obtained by solving LP1.

We note that an iterative approach exists to show that an integer solution can be obtained from an alternative LP formulation in polynomial time. The approach is presented in “Appendix 3”.

4.2 Solving the problem with facility opening costs via matching

We next define a version of the MAX-EXP-COVER-LRO problem when fixed facility opening costs exist instead of the requirement that at most k facilities can be open. In this problem, there is no limitation on the number of facilities, and the goal is to maximize the *expected total net value* of the solution, in terms of a weighted combination of the expected demand covered and minus the facility costs. We call this problem MAX-EXP-VALUE. Let f_v be the cost of opening a facility at node v, for $v \in V$. Then the formulation is as follows.

MAX-EXP-VALUE: Find $F \subseteq V$, to

$$maximize \quad \sum_{s\in S} \Pr(s) \left(\sum_{q=0}^{m} \sum_{v_i \in V} \; P(q,s)\, w_i\, I_{v_i}(F,q)\right) - \sum_{v_i\in F} f_{v_i}.$$

We give a polynomial time solution method to solve the case with two LROs, i.e. $|S|$=2 (2-LRO), by constructing a bipartite matching problem.

Proposition 4.2 *MAX-EXP-VALUE with two LROs can be transformed to a maximum weight perfect matching problem in a bipartite graph in polynomial time.*

Proof Given an instance of MAX-EXP-VALUE with input graph $G = (V, E)$, we first transform it into a *subtrees problem* instance with weights $PEW(v)$ at the nodes of the trees defined as before (except that we are not restricted to select k leaves and the objective function is different now). Let T_1 and T_2 be the component trees corresponding to the two LROs, with node sets $V_1 \cup L$ and $V_2 \cup L$. We form a bipartite graph H by modifying $T_1 \cup T_2$ and assigning weights to its edges as follows: Replace each leaf node $l \in L$ by two nodes l_1 and l_2 in H, and add a new edge connecting these nodes with weight $-f_l$. Call these edges *connecting edges*.

For $i = 1, 2$: Replace each edge $(v, f(v))$ in T_i, where $f(v)$ is the father of v, by a two-edge path $f(v) - g(v) - v$ where $g(v)$ is a *new node*. Give the edge $(f(v), g(v))$ the weight $PEW^i(f(v))$ originally attached to $f(v)$ in T_i. Call these edges *upper edges*. The other edge, $(g(v), v)$ obtains zero weight. These are called *lower edges*. Let N_i denote the set of new nodes added to T_i, for $i = 1, 2$. Now T_i contains the $2n - 1$ nodes consisting of leaves L_i and non-leaves V_i, plus the $2n - 2$ new nodes N_i, for both $i = 1$ and $i = 2$. The graph H has the bipartition $V_1 \cup L_1 \cup N_2$ and $V_2 \cup L_2 \cup N_1$, where both node sets have the same cardinality. Figure 3 illustrates the bipartite graph H constructed from $T_1 \cup T_2$.

Now with this setting, we have the following properties in a matching M of H:

1. A leaf node $l_1 \in L_1$ can be matched only to either $l_2 \in L_2$ with weight $-f_l$, or a new node $g(l_1) \in N_1$ with weight zero. This is symmetric for leaves in L_2.
2. A node v in V_1, which is not the root node, that is a father of two nodes u_1 and u_2 can be matched either to $g(u_1)$ or $g(u_2)$ by an upper edge with weight $PEW^1(v)$, or to $g(v)$ by a lower edge with weight zero. A symmetric property holds for any node in V_2.
3. The root node r of T_1 can be matched to one of its two children in H by an upper edge with weight $PEW^1(r)$. This is symmetric for T_2.

We would like to compute in H a maximum weight matching M such that old nodes are matched exactly once, and new nodes are matched at most once. The problem

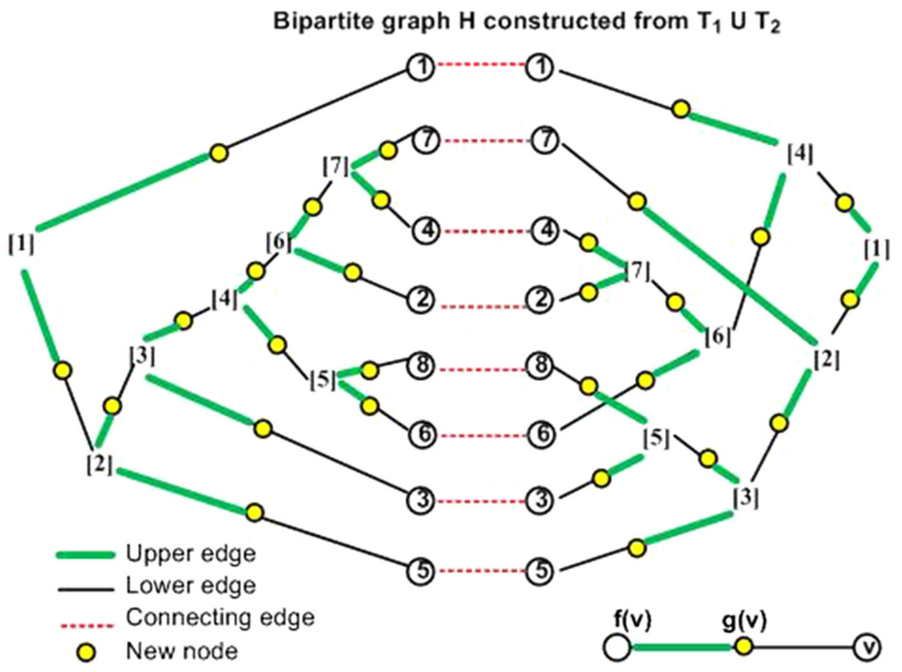

Fig. 3 The bipartite graph construction

having these restrictions can be transformed to an equivalent *perfect matching* problem on an augmented graph H' by adding auxiliary nodes and edges in a routine way as follows. We add $2n-2$ dummy nodes on each side, d_{i1} and d_{i2} for $i = 1, \ldots, 2n-2$. Furthermore, we add the following zero weight edges for each i: (d_{i1}, d_{i2}), $(d_{i1}, g(v))$ for all $g(v) \in N_2$ and $(g(v), d_{i2})$ for all $g(v) \in N_1$.

Let M be a perfect matching in H' with total weight W. Let F be the set of leaves l such that (l_1, l_2) is in M. By the first property, for every leaf node in F, the corresponding opening cost is deducted in W. By the second and third properties, if a weight $PEW^i(v)$ is included in W since some upper edge is included in M, then this means that at least one child of v is also incident to an upper edge in M. This argument follows all the way till a leaf node is reached so that at least one of the leaves that belong to the component defined by v is matched, as required. In other words, a path between a leaf node $l \in F$ and the root node contains node v, as required. Note that all other edges in the matching have zero weight. Hence, we obtain a feasible solution to the *subtrees problem* with weight exactly equal to W.

Next we show that every feasible solution to the *subtrees problem* with value W gives a perfect matching in H' with weight W. Let F be the set of leaves that are selected in the *subtrees problem* and let Q be the set of internal nodes whose weight is included in W. Construct a matching M in H' such that:

1. For any leaf $l \in F$, (l_1, l_2) is in M.
2. For any $v \in Q$, include the upper edge $(f(u), g(u))$ in M for a child u of v such that its subtree contains a leaf in F (as opposed to the lower edge $(v, g(v))$).

3. For a leaf l not in F, include the lower edge $(g(v), l)$ in M, where v is the father of l.
4. For any new node $g(v)$ not matched to an old node so far, include an edge connecting it to a dummy node. That is, add $(d_{i1}, g(v))$ to M if $g(v)$ is in N_2 and add $(g(v), d_{i2})$ to M if $g(v)$ is in N_1.
5. For any dummy node d_{i1} not matched so far, match it with another unmatched dummy node d_{i2} by including (d_{i1}, d_{i2}) in M.

Note that the number of new nodes that are matched to original nodes are equal in both trees. Therefore, the unmatched new nodes are the same in number in both sides, and equal to $k - 1$, if k facilities are opened at the leaves. Hence, they match with an equal number of dummy nodes leaving an equal number of unmatched dummy nodes on either side. These can be finally matched via a zero-weight matching. By this construction M is a perfect matching with weight W. □

5 Extensions and hardness results

In this section, we first show that maximizing expected coverage under three LROs is NP-hard even in the special case when all reliability values are 0 or 1. Then, we investigate the general case with an arbitrary number of scenarios (d-LRO) and show that the problem is as hard as the well-studied maximum coverage problem, even with 0–1 reliability vectors. We also remark that the greedy approach we presented for the 1-LRO case extends to the general d-LRO case but it only gives an $(1 - 1/e)$-approximation for coverage.

5.1 NP-hardness of the case with three LROs

Let us consider MAX-EXP-COVER-LRO with three LROs.

Proposition 5.1 *MAX-EXP-COVER-LRO is strongly NP-Hard for the case of three scenarios each with an LRO, even with 0–1 reliability vectors.*

Proof The proof is by reduction from the three dimensional matching problem (3-DM) which is known to be strongly NP-hard Ausiello et al. (1999). In 3-DM, three disjoint sets A, B, C s.t. $|A| = |B| = |C| = n$, and a set of triples $T, T \subseteq A \times B \times C$ with $|T| = t$, are given. The problem is to find n triples $M \subseteq T$ such that their union is $A \cup B \cup C$ and they form a matching (no elements in M agree in any coordinate).

Given an instance of 3-DM, we define an instance of MAX-EXP-COVER-LRO with three scenarios and hence three LROs; the probability that each scenario occurs is 1/3. Let the input graph G be a complete undirected graph on t nodes, each representing a triple in T. A 0–1 reliability vector $p^s = (p^s_1, \ldots, p^s_m)$ is defined for each scenario $s = A, B, C$ as follows. For scenario A, for each element a_i of A, we consider all triples in T that contain a_i. We set the reliability of all edges between such triple nodes to 1, so that a complete graph called the ith block of A, is formed by the edges with probability 1. For the rest of the edges in G, the reliability is set to zero in p^A. Thus, n connected components (blocks) exist for scenario A. Furthermore, the weight of each node is such that a total weight of 1 is distributed evenly among the nodes of each block so that the total weight in scenario A is n. This construct is repeated for scenarios B and C. In this instance, we need to locate n facilities, i.e. $k = n$.

Now suppose there is a 3-DM solution with n triples $t_1, \ldots, t_n$. In G, we select the n nodes corresponding to these triples as the facility locations. Since the triples in the 3-DM solution match every element of A, B, C, the total weight collected (i.e. the expected coverage) in the facility location problem should be n. Note that this is the maximum coverage possible since each scenario has probability 1/3 and the total weight of each scenario is n.

If MAX-EXP-COVER-LRO has a solution with expected demand value n, then we can construct a 3-DM solution. In each of the three scenarios, the facility locations F in this solution should cover every block in G to get coverage value n as each block has a weight of 1. This implies that a facility exists in each block of each scenario and that facility covers all the triples in each block. Note that a block consists of all triples that contain the same element from A, or B or C. Hence every element in $A \cup B \cup C$ is covered in the 3-DM solution. □

5.2 The case with an arbitrary number of LROs

We had defined the MAX-EXP-COVER-LRO problem in Sect. 2. In this problem we have d-LRO for an arbitrary number d. Note that each LRO is associated with a disruption scenario $s \in S$ (hence, $|S| = d$). We first discuss how this problem is reformulated in terms of component trees.

For each scenario $s \in S$, its LRO defines a component tree T_s whose leaves are common in all the trees. Let N denote the common leaf node set and V_s denote the non-leaf nodes in T_s, for $s \in S$. By assigning weights to the nodes of the trees as in Sect. 3, that is by defining $EW(v, s)$, $v \in V_s$, we can reformulate the MAX-EXP-COVER-LRO problem as a *subtrees* problem.

MAX-WT-k-LEAF-s-SUBTREES: Given rooted trees T_s, with weights $EW(v, s)$ for each node of v of T_s for $s \in S$, with a common leaf node set N, find a subset L of N such that $|L| \leq k$ and $\sum_{v \in V_s} EW(v, s) I_v(L, s)$ is *maximum*, where $I_v(L, s)$ equals 1 if the subtree of T_s rooted at v contains a leaf node in L; and 0, otherwise.

This problem is NP-hard since we proved NP-hardness for the 3-LRO case. We next show that the MAX-EXP-COVER-LRO problem is equivalent to the *maximum coverage* problem (MAX-COVERAGE) (Hochbaum 1982), which is defined as follows. Given a set U of n elements, subsets $S_1, S_2, \ldots, S_m$ of U, and integer k, $1 \leq k \leq m$, choose k subsets to maximize the number of elements of U that are covered (i.e. contained in the selected subsets).

Proposition 5.2 *MAX-COVERAGE reduces to MAX-EXP-COVER-LRO with a 0–1 reliability vector for each LRO.*

Proof Given an instance of MAX-COVERAGE with m sets and n elements, we construct an instance of MAX-EXP-COVER-LRO with a complete input graph G on m nodes such that each node v_i represents the set S_i, for $i = 1, \ldots, m$. We define n scenarios such that each scenario s corresponds to an element u_s in U, for $s = 1, \dots, n$ and the probability of each scenario is $1/n$. A 0–1 reliability vector p^s is defined for each scenario s by selecting the edges with reliability 1 as follows. For element u_s, consider the sets among $S_1, S_2, \dots, S_m$ that contain this element. All the nodes in G corresponding to these sets are connected to each other with edges having reliability 1. All of the remaining edges have zero reliability. That is, in scenario s the graph G reduces to one completely connected component corresponding to element s, called block B_s, and the remaining nodes are just singletons. The weights of the nodes in B_s are all equally distributed to sum up to 1, whereas the weights of the singleton nodes are zero. As a result, in each scenario the total weight is 1 and the expected total weight over all scenarios is also 1. In this instance, we select k nodes to locate facilities.

Suppose the MAX-COVERAGE solution covers w elements with k sets. Then, we select the k nodes in G corresponding to these sets as the facility locations. Let F represent these nodes. In each scenario s corresponding to a covered element, the block B_s must contain a node in F (in other words, the element u_s is covered by at least one of the sets represented in B_s). Thus, a total weight of 1 is covered in each such scenario, leading to total expected coverage of w/n.

Now, suppose MAX-EXP-COVER-LRO has a solution with expected demand value w with k facilities located at the nodes F. We will show that w/n elements are covered in MAX-COVERAGE by selecting the k sets corresponding to the facility nodes in F. The given facility location solution should cover w blocks in G to get the coverage value w/n as each block has a weight of 1 and occurs in a scenario with probability $1/n$. This implies that a facility exists in each such block. Since a block consists of nodes representing the sets that cover a specific element, the element is covered by the sets we selected (each one corresponding to a node in F), for a total of at least w elements. □

The greedy algorithm presented in Sect. 3 for the single scenario case generalizes to the multiple scenario case and has a $(1 - 1/e)$-approximation ratio for MAX-EXP-COVER-LRO, as the proof in Hochbaum (1982) is also valid here.

Proposition 5.3 *A $(1 - 1/e)$-approximation algorithm exists for MAX-EXP-COVER-LRO with an arbitrary number of LROs.*

5.3 Maximizing expected demand served within a distance limit in the d-LRO case

In a more general version of the facility location problem MAX-EXP-COVER-LRO, we define distances or travel times on the edges. In case of a disaster, delivering relief aid from the facilities to the demand points in shortest time is of high priority. Considering such a setting, we aim to satisfy as much demand as possible within a specified time/distance limit. The time/distance limit can be incorporated into MAX-EXP-COVER-LRO by defining edge lengths in the input graph and allowing a demand node to be covered only if a facility exists within distance R to itself, where R is a specified parameter. Assuming that a sufficient amount of supply will be available at the facilities, if a facility is established at a node, it covers the demand of all nodes that can be reached from it via a path of length R in the surviving network. If the locations of the facilities are fixed, in each possible surviving network realization, total demand covered can be evaluated by applying a shortest-path algorithm starting from each facility node. The location problem is to place at most k facilities to maximize the expected total demand covered within distance R. We denote this problem by MAX-EXP-COVER-LRO-R.

Definition 5.1 Let d_{ij}^q denote the length of a shortest path between nodes i and j in the graph defined by the network realization ξ^q. For a set $F \subseteq V$ of selected facility locations, let $I_v(R, F, q)$ be an indicator variable that takes the value 1, if there exists j in F such that $d_{vj}^q \leq R$; and 0, otherwise.

Let $P(q, s)$ be the probability that ξ^q occurs in scenario s, as before. Then, MAX-EXP-COVER-LRO-R is formulated as given below.

MAX-EXP-COVER-LRO-R: Find $F \subseteq V,\ |F| \leq k$ to

$$maximize\ \sum_{s \in S} \Pr(s) \sum_{q=0}^{m} \sum_{v_i \in V} P(q, s)\, w_i\, I_{v_i}(R, F, q).$$

Proposition 5.4 *MAX-EXP-COVER-LRO-R is strongly NP-Hard even for the case of a single scenario with a linear reliability order of edge failures (1-LRO).*

Proof The proof is by reduction from the maximum k-facility location problem (defined in Cornuejols et al. (1977)), which is known to be strongly NP-hard. In the maximum k-facility location problem, a set of clients I and a set of potential facility locations J are given with profits $c_{ij} \geq 0$ for each pair $i \in I$, $j \in J$. At most k facilities are located at a subset F of J to maximize $\sum_{i \in I} \max_{j \in F} c_{ij}$. Given an instance of this problem, let c_{max} be maximum c_{ij} value over all pairs $i \in I$, $j \in J$. Define an instance of MAX-EXP-COVER-LRO-R by taking the complete bipartite graph $I \times J$ as the input graph G. For each edge (v_i, v_j), let us set its reliability to c_{ij}/c_{max} (so that it is between zero and one) and its length to 1. Set $R = 1$ and $w_i = 1,\ \forall v_i \in I$, $w_j = 0,\ \forall v_j \in J$. Then, any solution that maximizes expected total demand covered locates the facilities at a subset of J due to the distance limit $R = 1$. Furthermore, facilities will be selected to maximize the total reliability of the edges connecting each node v_i, $i \in I$ to a facility node v_j, $j \in F$ with maximum c_{ij}/c_{max}. Hence, this solution also maximizes the profits in the k-facility location problem. □

We next show that MAX-EXP-COVER-LRO-R reduces to the maximum k-facility location problem; hence, any solution algorithm developed for the latter can be used to solve the former by means of the transformation in the proof.

Proposition 5.5 *MAX-EXP-COVER-LRO-R under 1-LRO for edge failures can be reduced to the maximum k-facility location problem in polynomial time.*

Proof Suppose we are given an instance of MAX-EXP-COVER-LRO-R with a single LRO defined by a reliability vector p for the input graph $G = (V, E)$. We define a complete bipartite graph $V \times V'$ by duplicating the node set V as V'. The set V corresponds to the set of clients and V' to the set of potential facility locations. For edges (v_j, v_j'), we set $c_{v_j, v_j'} = w_j$, for $v_j \in V$. We next define the profit of the pair (v_i, v_j') for $i \neq j$. For a pair of nodes v_i and v_j in V, let d_{ij}^q be the distance between the two nodes in a given network realization ξ^q. Note that if the distance d_{ij}^q exceeds R, then it will remain so in all of ξ^{q-1}, ..., ξ^1, ξ^0. Therefore, if $d_{ij}^m > R$ (the limit is exceeded even when all edges survive), then we set the profit c_{v_i,v'_j} to 0, since a facility in v_j cannot serve v_i in any realization. Otherwise, let s be the smallest index such that $d_{ij}^q \leq R$ for all $q \leq s$. Then, we set $c_{v_i,v'_j} = w_j \sum_{q=s}^m P(q)$ since a facility in v_j can serve v_i in realizations s to m. Recall from Sect. 2 that $\sum_{q=s}^m P(q) = p_s$, as the probability that realization ξ^q occurs is $p_q - p_{q+1}$. Thus, $c_{v_i,v'_j} = w_j p_s$.

Suppose $F \subset V'$, $|F| \leq k$, is an optimal solution to this instance of the maximum k-facility location problem. Then, F gives the maximum value of $\sum_{v_i \in V} \max_{v'_j \in F} c_{v_i,v'_j}$ by definition. As the problem is uncapacitated, each client is serviced by one facility. If a facility is located at v'_j, it services v_j with profit equal to w_j as any other facility in F will provide a smaller profit. For a node v_i such that v'_i is not in F, it is serviced by some $v'_j \in F$, $i \neq j$, such that c_{v_i,v'_j} is maximum over all facilities. Note that for any fixed F, for any i such that $v'_i \notin F$, we can assign it to some $v'_j \in F$ such that $d_{ij}^s \leq R$ for the maximum possible s over all such v'_j. In this way, the demand at i will be serviced for the most number of realizations.

Under this reduction, the set of facilities F^* that are optimal for the maximum k-facility location problem will also form a solution to MAX-EXP-COVER-LRO-R with the same objective value. Therefore, this reduction is also approximation-preserving. $\square$

Cornuejols et al. (1977) showed that a greedy algorithm has a worst-case bound of $(1 - 1/e)$ for the maximum k-facility location problem. Later, Ageev and Sviridenko (2004) improved the worst-case bound to $(1 - (1 - 1/q)^q)$, where q is the maximum size of the subsets. Since the reduction in the proof of Proposition 5.5 is approximation-preserving, the same ratios will be also valid for MAX-EXP-COVER-LRO-R.

6 Conclusions

We studied the problem of locating facilities to maximize the expected demand serviced in a network with unreliable edges. As opposed to similar problems in the literature, in this problem edges do not fail independently. Given the reliability of each edge, we assume a linear reliability ordering of edges such that failure of an edge implies the failure of all edges with the same or lower reliability. Under this reliability model, the possible network realizations become polynomial in number and a favorable problem structure exists. We showed that the surviving network components can be represented by a binary tree; hence a transformation to a tree problem follows. We presented polynomial time exact algorithms or hardness results for several variations of the problem in terms of the number of linear orderings, different objectives, capacity limits and distance limits for coverage. Our findings represent new results on finding tractable models of edge failure for facility location planning.

Appendix 1: A dynamic programming algorithm for 1-LRO

We are given a rooted tree T with weights $PEW(v)$ for all nodes v of T. T_v denotes the subtree of T rooted at v, and L_v the set of its leaves. For every nonnegative integer $t \leq k$, we denote $EW(v,t)$ as the maximum expected weight at T_v obtained by placing exactly t facilities in L_v. We have $EW(v,0) = 0$ at all nodes v of T. If v is a leaf, then $EW(v,t) = PEW(v)$, for all $1 \leq t \leq k$. For a non-leaf node v, let v_l and v_r be the two children of v. For $t \geq 1$, we have the following recursive relation.

$$EW(v,t) = \max_{0 \leq t' \leq t} \{EW(v_r, t') + EW(v_l, t - t')\} + PEW(v). \tag{5}$$

The recurrence corresponds to allocating t' of the t facilities optimally in the right subtree and the remaining in the left and counting in the expected weight of the root node v as long as $t \geq 1$ since in this case some leaf will have a facility and allow the expected weight at the root node to be counted in the objective. The recursion proceeds bottom-up from leaves to the root node.

Proposition 6.1 *The dynamic programming algorithm solves the MAX-WT-k-LEAF-SUBTREE problem in $O(kn)$ time.*

Proof As the component tree has $2n - 1$ nodes, the recursive equations in the DP algorithm are calculated in $O(kn)$ time. □

Corollary 6.1 *The MAX-EXP-COVER-LRO problem is solved in $O(m \log n + T(n) + kn)$ time, where $O(T(n))$ is the time complexity of finding an MST of the input graph.*

Proof First the component tree is constructed in $O(m \log n + T(n))$ time. Then the DP algorithm is applied in $O(kn)$ time to the component tree CT with its weights defined as $PEW(v)$, for v in CT. By Propositions 3.1 and 6.1, we obtain an optimal solution to the MAX-EXP-COVER-LRO problem. □

The dynamic programming approach can be generalized to handle a version of the problem with uniform capacities at the facilities and another cost minimization version with penalties for unmet demand.

Solving the capacitated 1-LRO problem by dynamic programming

When the supply quantities at the facilities are limited and a shortage can possibly occur after the disruption, a capacitated problem can be defined. We assume that k facilities each with supply quantity C will be open and the objective is to maximize the expected total demand served. The capacitated k-facility problem of maximizing expected coverage with a single LRO is formulated as follows.

MAX-EXP-COVER-LRO-CAP: Find $F \subseteq V,\ |F| = k$ to

$$\mathit{maximize}\ \sum_{q=0}^{m} \sum_{v_i \in V} P(q)\, w_i\, D_{v_i}(F, C, q),$$

where $D_{v_i}(F, C, q)$ (taking values between 0 and 1) is the percentage of w_i, namely demand of node v_i, that can be covered by the facilities F, each with capacity C, in the network realization where the q strongest edges survive.

We next give a dynamic programming algorithm to solve this problem. The recursion is on the subtrees of the component tree CT and the number of facilities. For an interior node v of the component tree CT, let T_v be the subtree of CT rooted at node v. Associated with T_v, we define $ECC(T_v, t) =$ *Maximum Expected Coverage* in T_v when t facilities are located at the leaves of T_v. Recall that when a disconnector edge fails, it disconnects T_v into two components and this is represented by its two children in CT. Let the corresponding two subtrees be $T_{v,L}$ and $T_{v,R}$ so that if a facility is located in T_v, it will either be located in $T_{v,L}$ or $T_{v,R}$.

If $t = 0$ we have $ECC(T_v, 0) = 0$ at all nodes v in the tree. If v is a leaf, we then have $ECC(v, t) = \min\{Ct, w_v\} \cdot rel(v)$ for all $1 \le t \le k$.

For a non-leaf node v and for $t > 0$, we have the following relation.

$$\begin{aligned} ECC(T_v, t) = \max_{1\le t'\le t} \{&ECC(T_{v,L}, t') + ECC(T_{v,R}, t - t') \\ &+ rel(v)\min\{Ct, \sum_{l\in L_v} w_l\}\} \end{aligned} \tag{6}$$

The recurrence corresponds to allocating t' of the t facilities optimally in the left subtree and the remaining in the right. As a result of this recursion we obtain a solution with two properties: (1) the demand assignment in a component will be kept the same when this component is contained in a bigger component, in some network realization where more edges survive, and (2) any component which has a deficit will try to satisfy its deficit from the first surplus component that it connects to (the earliest) along the component tree (bottom-up).

Property 6.1 *If node $v_i \in V$ is assigned to some facility at node v_j which is a leaf in T_v, it will also be assigned to the facility at v_j for any of the nodes v' whose subtree $T_{v'}$ contains T_v. We call this property the* Monotonicity of Demand Assignment *(MDA) property.*

Property 6.2 *A solution has the* Bottom-up Demand Assignment *(BDA) property, if the following holds:*

For each component $H \subset G$ with a deficit, and each node v_j with unsatisfied demand in this component, let H' be the first component that contains H with respect to the CT with surplus. Then, v_j is assigned to a facility in H'.

By virtue of the above properties, an optimal solution can be constructed by satisfying the demands of all nodes at the first instance (going bottom-up) when there is enough capacity, leading to the recursion presented.

Lemma 6.1 *There exists an optimal solution to the MAX-EXP-COVER-LRO-CAP problem with the MDA and BDA properties.*

Proof Take an optimal solution to the MAX-EXP-COVER-LRO-CAP problem such that the MDA property does not hold. Then, there must be some subtree T_v of CT

representing a component $H \subset G$ in a network realization such that some node v_i is assigned to some facility at node v_j of H, but in another component H' that contains H, v_i is assigned to a facility at a node outside H. Consider the edge whose failure leaves H as a component. We can reroute flow on this edge to be directed from a surplus component to a deficit component, satisfying the same set of demands with the MDA property.

Take an optimal solution to the MAX-EXP-COVER-LRO-CAP problem such that the BDA property does not hold. By satisfying the deficit in a component by the earliest surplus-component, we get the most partial expected weight. □

Proposition 6.2 *The dynamic programming algorithm solves the MAX-EXP-COVER-LRO-CAP problem in $O(kn)$ time after the construction of the component tree.*

Proof As the component tree has $2n-1$ nodes, the recursive equations in the algorithm are calculated in $O(kn)$ time. □

Appendix 2: Solving the cost minimization 1-LRO problem by dynamic programming

We consider a cost minimization version of the facility location problem for the 1-LRO case. We are given a fixed cost f_v for opening a facility at node v, a variable capacity cost c_v per unit of demand served from a facility at node v, for $v \in V$. Furthermore, a variable shortfall cost s per unit of unserved demand is given. The objective is to minimize a weighted sum of the facility opening costs and the expected costs of capacity used and demand unserved. In this problem up to n facilities can be opened. We refer to the problem as MIN-EXP-COST, and formulate it below.

MIN-EXP-COST: Find $F \subseteq V$, to

$$maximize\ \sum_{v_i \in F} f_{v_i} + \sum_{q=0}^{m} P(q)\,\{\sum_{v_i \in F} c_{v_i} x_{iq} + s\ \sum_{v_i \in V} y_{iq}\}$$

where x_{iq} is the amount of capacity used at the facility located at v_i, and y_{iq} is the part of w_i that is not covered by the facilities F in the network realization where the q strongest edges survive.

This problem can be solved by a dynamic programming algorithm. The recursion is on the subtrees of the component tree CT. We use the same definitions of T_v, $T_{v,L}$, $T_{v,R}$, L_v and $PEW(v)$ as before. Associated with T_v, we define $MEC(T_v, UC)$ as the minimum expected cost in T_v when at least one facility is located at the leaves of T_v and the minimum unit capacity cost of the open facilities in T_v is at most UC. This is defined for each UC value that is equal to c_l for $l \in L_v$. Let $CV_{min}(v) = \min_{l \in L_v} c_l$. Then, $MEC(T_v, CV) = \infty$ for all $0 < CV < CV_{min}(v)$. Similarly, we define $MEC(T_v, 0) =$ as the minimum expected cost in T_v when *no facility* is located at the leaves of T_v.

For a leaf node v, we have the following relations.

$$MEC(v, CV) = \begin{cases} f_v + c_v\, w_v\, rel(v), & \text{for all } CV \geq c_v \\ \infty, \text{ for all } 0 < CV < c_v. \end{cases} \tag{7}$$

$$MEC(v, 0) = s\, w_v\, rel(v). \tag{8}$$

For a non-leaf node v, we have the following two relations.

$$MEC(T_v, 0) = MEC(T_{v,L}, 0) + MEC(T_{v,R}, 0) + s\, PEW(v) \tag{9}$$
$$MEC(T_v, UC) = \min\{\, MEC_1, MEC_2, MEC_3 \,\} \tag{10}$$

where

$$MEC_1 = MEC(T_{v,L}, 0) + \min_{UC' \le UC} \{MEC(T_{v,R}, UC') + \min\{UC', s\}\, PEW(v)\}$$
$$MEC_2 = MEC(T_{v,R}, 0) + \min_{UC' \le UC} \{MEC(T_{v,L}, UC') + \min\{UC', s\}\, PEW(v)\}$$
$$MEC_3 = \min_{UC_L, UC_R \text{ s.t. } UC \ge \min\{UC_L, UC_R\}} \{MEC(T_{v,L}, UC_L) + MEC(T_{v,R}, UC_R) + \min\{UC_L, UC_R, s\}\, PEW(v)\}$$

Equation (9) considers the case when no facility exists in T_v so that none of the demand of T_v can be serviced and the corresponding shortfall cost is incurred. In Eq. (10) the minimum expected cost in T_v is calculated when a facility is located in one of the leaves of T_v. We take the minimum of three terms that correspond to three cases: (1) A facility exists in $T_{v,R}$ but not in $T_{v,L}$; (2) A facility exists in $T_{v,L}$ but not in $T_{v,R}$; (3) A facility exists in both $T_{v,R}$ and $T_{v,L}$. Let us consider the first case. Here, at node v of CT, the partial expected weight is covered if the minimum unit capacity cost UC in T_v is less than or equal to the unit shortfall cost s. The second case is symmetric. In the third case, at least one of the two subtrees contains a facility with unit capacity cost at most UC (hence the condition $UC \ge \min\{UC_L, UC_R\}$). All nodes in T_v are either served by the smallest unit cost facility or the shortfall cost is incurred in the last term.

Proposition 6.3 *The dynamic programming algorithm solves the minimum expected cost facility location problem MIN-EXP-COST in $O(n^4)$ time after the construction of the component tree.*

Proof The component tree has $2n - 1$ nodes. For each node the recursive equations in the DP algorithm are calculated for each of the possible UC values. There are at most n possible UC values, one for each node where a facility can be placed. For any particular UC value, there are $O(n^2)$ choices to check to compute the best combination for the minimum. □

Appendix 3: Polynomial-time solution using an iterative argument

We next give an alternative LP formulation for the MAX-WT-k-LEAF-2-SUBTREES problem and show that an integral optimal solution can be obtained from an optimal extreme point solution of the LP using an iterative argument similar to the technique presented in Singh and Lau (2007).

To do this, we further reduce the problem to one of picking a set of k leaves such that the weight of the internal nodes *not* in the paths from these leaves to the roots in the two trees is as small as possible. We use an indicator variable x_l for each leaf to denote if it is chosen in the set of k leaves, for $l \in L$. For every internal node v in each of the two trees, we have a variable y_v denoting if this node is *not* in a path from a chosen leaf to the corresponding tree's root, and thus needs to be counted in the objective function to be minimized. We have the following LP relaxation for the problem.

$$\text{(LP2)} \qquad Minimize \qquad \sum_{v \in V_1 \cup V_2} w_v\, y_v + \sum_{l \in L} w_l\,(1 - x_l) \tag{11}$$

$$s.t. \qquad y_v + \sum_{l \in T_v \cap L} x_l \geq 1 \qquad \forall\, v \in V_1 \cup V_2 \tag{12}$$

$$\sum_{l \in L} x_l = k \tag{13}$$

$$y_v \geq 0 \qquad \forall\, v \in V_1 \cup V_2 \tag{14}$$

$$0 \leq x_v \leq 1 \qquad \forall\, v \in L \tag{15}$$

We prove the following results for a slightly more general LP2 where T_1 and T_2 may be forests rather than trees, with common leaf set L.

First observe that if we find a variable y_v that is set to 1 in an optimal solution to LP2, we can simplify the problem as follows. For such an internal node v, all its non-leaf descendants also have y-value 1 and all its leaf descendants have x-value 0; thus we pick the highest such node and delete its subtree with the corresponding constraints (12) from the problem.

On the other hand, if we find a variable x_l that is set to 1 in an optimal solution to LP2, then we simplify as follows. The y-value of all nodes on the path from l to the root in both forests to zero (their constraints are satisfied). We delete this leaf l and these internal nodes with the corresponding constraints (12). We now have a problem with at least one less leaf; hence constraint (13) is also modified with a right-hand side of $k - 1$.

In either case, the resulting problem has the same type of constraints and so we can continue to recursively find another variable set to 1 to finally find an integer solution. We summarize this self-reducibility property in the lemma below.

Lemma 6.2 *If every extreme point solution to LP2 has a variable set to 1, then LP2 always has an integral optimal solution.*

We can now get an integral solution to LP2 by proving the following key claim.

Proposition 6.4 *Any extreme point solution to LP2 always has a variable set to 1.*

Proof We prove this by contradiction. Suppose at an extreme feasible point (or basic feasible or vertex solution) x', y' all positive variables are fractional and without loss of generality they are < 1. Consider a maximal set of *independent tight* constraints from LP2 corresponding to this extreme point (namely, a maximal set of constraints that are tight at this solution and are linearly independent of one another). The set of tight constraints (excluding nonnegativity) can be indexed by $V_1' \cup V_2' \cup k$ where V_1' and V_2' are internal nodes in the forests T_1 and T_2 whose corresponding constraints (12) are tight and k is the index of the cardinality constraint (13).

Since we have a basic solution, the number of linearly independent tight constraints equals the number of fractional valued variables. We argue that if all variables are fractional, every tight constraint has many nonnegative variables involving it. However we can show that the total number of independent tight constraints cannot be as many as the number of such fractional variables contradicting the fact that this is an extreme point. To carry out the proof, we use a token argument as in Jain (1998), Singh and Lau (2007).

We give one token to each positive variable. We then redistribute these tokens among the independent tight constraints so that each tight constraint gets at least one token and there are some leftover tokens which will show a contradiction.

Here is how the redistribution works. If any tight constraint in V_1' or V_2' corresponding to an internal node v has $y_v' > 0$, we assign the token of v to this constraint. Next, we look at tight constraints in V_1' or V_2' corresponding to nodes v such that $y_v' = 0$: we call these constraints *leaf-tight*. Note that any leaf can be only in one leaf-tight constraint in each tree since a set of independent leaf-tight constraints in a tree correspond to disjoint sets of leaves. In particular, if there are two leaf-tight constraints in the same tree containing a leaf, they correspond to two ancestors of the leaf; subtracting the constraint of the lower ancestor from that of the higher leaves a set of leaf variables which are all set to zero. However, these tight zero constraints and the lower ancestor's constraint added together give the higher ancestor's constraint (a contradiction). Therefore, for every leaf l such that $x_l' > 0$, we assign half its token to a leaf-tight constraint in V_1' that contains x_l', if one exists, and the other half to a leaf-tight constraint in V_2' that contains x_l', if one exists. Also note that any leaf-tight constraint involves at least two leaves (since it is tight and no leaf has x'-value 1). Thus every leaf-tight constraint in each tree also gets at least one token.

Now consider the set of all leaf-tight constraints among $V_1' \cup V_2'$ in one of the forests, say T_1. If their union contains all leaves that have fractional x'-value, then their sum is the same as the cardinality constraint (on all leaves) which must be linearly independent of them by our initial choice. Thus there is at least one leaf that is not spanned by these maximal leaf-tight constraints in V_1'. However, note that the difference between the cardinality constraint and these leaf-tight sets in T_1 is an integer which is not zero and since there is no leaf with x'-value 1 there are actually *two* leaves that are not spanned by leaf-tight constraints in T_1 and thus their token assignments of $\frac{1}{2}$ to constraints in T_1 is unused. By a symmetric argument, there are two leaves that are not spanned by leaf-tight constraints in T_2 and thus their token assignments of $\frac{1}{2}$ each are not used in T_2. We thus get a total of 2 unassigned tokens of which we can assign one to the cardinality constraint and the other extra one gives the desired contradiction. □

As a result of Proposition 6.4 and Lemma 6.2, an integer optimal solution is obtained from LP2 recursively.

References

Ageev AA, Sviridenko MI (2004) Pipage rounding: a new method of constructing algorithms with proven performance guarantee. J Comb Optim 8(3):307–328

Ausiello G, Crescenzi P, Gambosi G, Kann V, Marchetti-Spaccamela A, Protasi M (1999) Complexity and approximation. Springer, Berlin

Colbourn CJ, Xue G (1998) A linear time algorithm for computing the most reliable source on a series-parallel graph with unreliable edges. Theoret Comput Sci 209:331–345

Cornuejols G, Fisher M, Nemhauser G (1977) Location of bank accounts to optimize float: an analytic study of exact and approximate algorithms. Manag Sci 23:789–810

Eiselt HA, Gendreau M, Laporte G (1992) Location of facilities on a network subject to a single-edge failure. Networks 22:231–246

Eiselt HA, Gendreau M, Laporte G (1996) Optimal location of facilities on a network with an unreliable node or edge. Inf Process Lett 58(2):71–74

Feige U (1998) A threshold of ln n for approximating set cover. J ACM 45(4):634–652

Gunnec D, Salman FS (2007) Assessing the reliability and the expected performance of a network under disaster risk. In: Proceedings of the international network optimization conference (INOC), April 22–25, Spa, Belgium

Hochbaum DS (1982) Approximation algorithms for the set covering and vertex cover problems. SIAM J Comput 11:555–556

Jain K (1998) A factor 2 approximation algorithm for the generalized Steiner network problem. Combinatorica, 21:39–60, 2001. Preliminary version in Proceedings of 39th IEEE FOCS

Korte B, Lovasz L, Schrader R (1991) Greedoids, algorithms and combinatorics. Springer, Berlin

Melachrinoudis E, Helander ME (1996) A single facility location problem on a tree with unreliable edges. Networks 27(3):219–237

Nemhauser GL, Wolsey LA (1988) Integer and combinatorial optimization. Wiley, New York

Pardi F, Goldman N (2005) Species choice for comparative genomics: being greedy works. PLoS Genet 1(6):e71

Salman FS, Yucel E (2015) Emergency facility location under random network failures: insights from the Istanbul case. Comput Oper Res 62:266–281

Singh M, Lau LC (2007) Approximating minimum bounded degree spanning trees to within one of optimal

Steel M (2005) Phylogenetic diversity and the greedy algorithm. Syst Biol 54(4):527–529

Wolle T (2002) A framework for network reliability problems on graphs of bounded treewidth. In: Proceedings of the 13th international symposium on algorithms and computation, LNCS, vol 2518, pp 137–149

Xue G (1997) Linear time algorithms for computing the most reliable source on an unreliable tree network. Networks 30:37–45