# Predisaster Preparation of Transportation Networks

**Hermann Schichl**[*]
Faculty of Mathematics
University of Vienna, Austria

**Meinolf Sellmann**
IBM Research
Yorktown Heights, NY, U.S.A.

## Abstract

We develop a new approach for a pre-disaster planning problem which consists in computing an optimal investment plan to strengthen a transportation network, given that a future disaster probabilistically destroys links in the network. We show how the problem can be formulated as a non-linear integer program and devise an AI algorithm to solve it. In particular, we introduce a new type of extreme resource constraint and develop a practically efficient propagation algorithm for it. Experiments show several orders of magnitude improvements over existing approaches, allowing us to close an existing real-world benchmark and to solve to optimality other, more challenging benchmarks.

## Earthquake Preparation

We consider a real-world pre-disaster planning problem that was introduced in (Peeta et al. 2010). Given a transportation network, where links are subject to probabilistic failure, the task is to use a limited budget to decrease the failure probability on selected links such that the overall expected shortest-path distance between a number of origin/destination pairs (OD-pairs) is minimized. The original model assumes independence between the failures of links. Consequently, a priori an exact standard stochastic programming approach would need to consider an exponential number of scenarios. Therefore, the approach in (Peeta et al. 2010) solves the problem without guarantees on the quality of the solution. Exploiting value-interchangeability, a symmetry-notion from constraint programming, only very recently (Prestwich, Laumanns, and Kawas 2013; 2014) were able to reduce the number of scenarios massively to a point where the benchmark introduced in (Peeta et al. 2010) could be solved to optimality within a number of minutes.

The idea of combining scenarios that all lead to the same value for the random variable considered (in our case: the sum of all expected shortest path lengths between all origin/destination pairs) is very interesting for scenario generation in general. We build on this idea and formulate the predisaster planning problem as an integer programming problem with a very simple constraint structure and a heavily non-linear objective. Due to the binary nature of investment decisions the problem is obviously not convex. Furthermore, we will show that it is hard to approximate and also neither sub- nor super-modular. We then introduce a new type of extreme resource constraint and develop a practically effective propagator which allows us to solve the existing benchmark instances in milliseconds, two to three orders of magnitude faster than existing methods. This then paves the way for tackling harder benchmark instances which consider several orders more non-interchangeable scenarios as well as correlated failure probabilities.

## Problem Definition and Complexity

**Definition 1.** *We are given an undirected graph $G = (N, E)$ with a node set $N = \{1, \dots, n\}$ and an edge set $E = \{\{i,j\} \mid i,j \in N, i < j\}$. Furthermore, we are given four functions. An a priori survival probability function $s : E \to [0,1]$, a survival probability after investment function $v : E \to [0,1]$, an edge length function $l : E \to \mathbb{N}$, and a cost of investment function $c : E \to \mathbb{N}$. Finally, we are given a budget limit $C \in \mathbb{N}$, as well as node pairs $(o_1, d_1), \dots, (o_k, d_k) \in N^2$ and penalties $M_{o_h,d_h} \in \mathbb{N}$ for disconnecting the OD pair $(o_h, d_h)$.*

*Given a set $I \subseteq E$, we denote with $p_I : E \to [0,1]$, with $p_I(i,j) = v(i,j)$ if $\{i,j\} \in I$ and $p_I(i,j) = s(i,j)$ otherwise, the survival probabilities after investing in links in $I$. We consider the probability space $(\Omega, P_I)$ where $\Omega = \{F \mid F \subseteq E\}$ and $P_I$ is the unique probability measure with $P_I(F) = \prod_{\{i,j\}\in F} p_I(i,j) \prod_{\{i,j\}\notin F}(1 - p_I(i,j))$. Finally, for all $o, d \in N$ let us denote with $\mathrm{SPL}^{\mathrm{I}}_{\mathrm{o,d}} : \Omega \to \mathbb{N}$ a random variable over $(\Omega, P_I)$ such that $\mathrm{SPL}^{\mathrm{I}}_{\mathrm{o,d}}(F)$ returns the shortest-path length from $o$ to $d$ in the weighted graph $(N, F, l)$, and $M_{o,d}$ if no path from $o$ to $d$ exists in $(N, F)$ (we omit a formal definition of shortest paths due to limited space).*

*The Predisaster Transportation Network Preparation Problem (PTNP) consists in selecting a subset $I \subseteq E$ of links to invest in such that $\sum_{\{i,j\}\in I} c(i,j) \le C$ and $\sum_{h=1}^{k} \mathrm{E}(\mathrm{SPL}^{\mathrm{I}}_{\mathrm{o_h,d_h}})$ is minimized.*

The PTNP is a probabilistic version of the deterministic problem class of survivable network design problems (SNDPs). Node- and link-survivable network design prob-

[*]This research was supported through grant P22239-N13 by the Austrian Science Foundation FWF.

lems have important applications in supply chain design, telecommunication network design (Kerivin and Mahjoub 2005), and wildlife preservation (LeBras et al. 2013). The SNDP helps to design networks that keep their functionality in case of the failure of a small number of links or nodes. The PTNP, on the other hand, optimizes networks for maximal survivability in case of a disaster that may simultaneously affect many links and nodes at the same time.

Later we will also study variants of this problem where we consider correlations between link failures (change in definition of $P_I$) or where we change the objective and minimize the expected maximum relative length increase $\max_{h=1}^{k}\{\mathrm{E}(\mathrm{SPL}^{\mathrm{I}}_{\mathrm{o_h,d_h}})/\mathrm{SPL}^{I}_{o_h,d_h}(E)\}$ or where we minimize the sum of squares of these values $\sum_{h=1}^{k}\left(\frac{\mathrm{E}(\mathrm{SPL}^{\mathrm{I}}_{\mathrm{o_h,d_h}})}{\mathrm{SPL}^{I}_{o_h,d_h}(E)}\right)^2$.

**Theorem 1.** *The PTNP is APX-hard and, unless $P = NP$, cannot be approximated for any constant relative error guarantee.*

*Proof.* Reduction from Steiner trees (omitted). □

Thus, not only is the PTNP NP-hard, it cannot even be approximated. Moreover:

**Theorem 2.** *The PTNP is neither sub- nor super-modular.*

*Proof.* Assume there is just one OD-pair, connected by two paths, the first consists of two edges $a$ and $b$, the second has just one edge $c$ which directly connects origin and destination. All edges have length 1, a priori survival probability 0, and survival probability 1 after investment. Investing in $a$ by itself results in no improvement, yet after investing in $b$ investing in $a$ helps. This is a case of increasing returns. However, after investing in $b$ and $c$, investing in $a$ is again useless as the shortest path, going directly via edge $c$, is already guaranteed to exist. This is a case of diminishing returns. □

## Formulation as Non-linear Integer Program

Therefore, pre-disaster planning is hard and does not exhibit some of the standard structural backdoors which would lead to reasonably practically efficient algorithms. Consequently, (Peeta et al. 2010) used a best-effort approach to tackle the problem. (Prestwich, Laumanns, and Kawas 2013), on the other hand, considered scenario bundles which led to the first provably optimal solutions for the Istanbul benchmark introduced in (Peeta et al. 2010).

We propose a non-linear integer programming model. There is only one constraint in the problem, and that is the limit on the budget which is exhausted linearly by the binary investment decisions. Our objective is to state the problem in the form

$$\text{minimize } f(x_1,\ldots,x_m) \text{ such that } \sum_{i=1}^{m} x_i c_i \le C,$$

where $m$ is the number of links in the graph, $x_1,\ldots,x_m$ are binary variables with $x_i = 1$ if and only if we invest in the link $i$, $c_i$ is the investment cost on link $i$, and $C$ is our total available budget. To achieve this simple structure we need to formulate the objective function $f$ which, given the investment decisions, computes the sum of the expected shortest path lengths over all OD-pairs.

Assume that, for each OD-pair $(o,d)$, we know the list $P_1,\ldots,P_r$ of all simple paths from $o$ to $d$, ordered by increasing path lengths. Then, we can compute the expected shortest path length for this pair by summing up, over all paths $P_i$ in the list, the product of the length of $P_i$ times the probability that all links in $P_i$ survive while, for all paths $P_j$ with $j < i$, at least one edge in $P_j$ does not survive. That is, rather than considering scenarios and computing the shortest paths given a scenario, we directly compute the probability that a given path determines the shortest-path length, thus bundling all scenarios that lead to the same shortest path (compare with (Peeta et al. 2010)).

The objective becomes simpler if we reorder events. Let us denote with $l(P_i)$ the length of path $P_i$ and with $p_i(x_1,\ldots,x_m)$ the probability, given the investment decisions $x_1,\ldots,x_m$, that at least one edge in each path $P_1 \ldots P_{i-1}$ does *not* survive. To simplify the notation, let us also set $l(P_0) = 0$ and $l(P_{r+1}) = M_{o,d}$, where $l(P_r) \le M_{o,d} \in \mathbb{N} \cup \{\infty\}$ is the penalty for losing all connections between $o$ and $d$. Then the expected shortest-path length for the pair $(o,d)$ is

$$\mathrm{E}(\mathrm{SPL}^{\{i \mid x_i=1\}}_{o,d}) = \sum_{i=1}^{r+1}\big(l(P_i) - l(P_{i-1})\big)p_i(x_1,\ldots,x_m),$$

and it holds $f(x_1,\ldots,x_m) = \sum_{(o,d)} \mathrm{E}(\mathrm{SPL}^{\{i \mid x_i=1\}}_{o,d})$. Our task thus reduces to

- computing the $r$ shortest simple paths between $o$ and $d$ which have length lower than $M_{o,d}$, and
- computing the terms $p_i(x_1,\ldots,x_m)$.

For the first task we conduct a simple best-first search in the given graph, whereby we prune paths that have no chance of obeying the threshold $M_{o,d}$ using shortest-path distances to the destination (see (Sellmann 2003b)) as well as those that are not simple (meaning that no node is visited more than once).

Our second task is a bit more involved. In $p_i(x_1,\ldots,x_m)$, to assess the probability that the first $i-1$ paths are all blocked, we enumerate all disjoint events where this is the case. To keep the objective somewhat reasonable in size, we will need to find mutually disjoint blocking events which quickly partition all possible cases.

Assume we have to compute the term $p_{i+1}(x_1,\ldots,x_m)$ and we have the list of paths $P_1,\ldots,P_i$. We can view these paths as sets of links. Then, each blocking event will select at least one element in each set, one link for each path that is failing. Therefore, we are actually looking for set covering solutions: The sets to be covered are the paths, and the elements in the sets are links in the graph. Once we arrive at this view it is easy to devise a very simple yet surprisingly effective greedy heuristic to enumerate all mutually exclusive events which will block paths $P_1,\ldots,P_i$: In each step we select a link that will cover as many paths as possible that are currently not covered, with ties broken randomly.

Once all paths are covered, we store the respective set of links. Then we backtrack our last decision and decide *not* to include this link this time. Then, we try to add other links to achieve a full cover. If we find a new cover, we store the set of all links in the cover, *as well as the set of links we actively decided not to include*. The latter is necessary to ensure that the probabilistic events, whose cumulated probability we are aiming to assess, are mutually disjoint. Only then can we simply add their probabilities to arrive at the joint probability $p_{i+1}(x_1, \dots, x_m)$.

Of course, we can improve this greedy tree-search enumeration by adding early pruning whenever it is clear that a full cover can no longer be achieved down a branch. Moreover, the independence of link failures allows us to stop the generation of covers early when each remaining link can only cover at most one path currently not covered. Then, for each such path, we can assess the probability of its failing as 1 minus the product of all survival probabilities of remaining links on this path. The probability of all remaining paths being covered is then simply the product of these values.

Despite these improvements, the entire procedure may appear prohibitively expensive. However, it turns out that it is extremely efficient in practice. For the real-world Istanbul scenario introduced in (Peeta et al. 2010), for example, we find 24 shortest paths total for five OD-pairs. To formulate the entire objective, using the greedy strategy described above, we need to consider a mere 156 covers in total. Not only is this number of covers less than the 341 remaining scenarios which are eventually considered in (Prestwich, Laumanns, and Kawas 2013), but also the entire computation of these scenarios takes milliseconds compared to 5 minutes reported in (Prestwich, Laumanns, and Kawas 2013).[1] Most importantly, we arrive at a non-linear integer program with an extremely simple constraint structure which we are going to exploit in the next section.

## Extreme Constraints

We established that our pre-disaster planning problem is neither sub- nor super-modular. However, we make a simple observation: In order to achieve a minimal shortest path length it never hurts to invest in another link if the budget allows us to do so. Consequently, we only need to consider selections of links to invest in which *fully exhaust our budget.* By that we do not mean that there is necessarily no money left, but that the remaining money would not suffice to invest in any additional link. Formally, we introduce the following extreme constraint for linear resources.

**Definition 2.** *Given $n$ binary variables $x_1, \dots, x_n$, costs $c_1, \dots, c_n \in \mathbb{N}$, and a budget $C \in \mathbb{N}$, the $\text{ExtremeLinearResource}(x_1, \dots, x_n)$ constraint is true if and only if $\sum_i x_i c_i \leq C$ and $\forall i : \; x_i = 0 \rightarrow c_i > C - \sum_j x_j c_j$.*

Before we study this constraint and develop a propagator, let us briefly compare it to other constraints. For optimization problems, in CP so-called global optimization constraints are considered. Prominent examples are the allDifferent constraint with costs (Régin 1999; Focacci, Lodi, and Milano 1999; Régin 2002; Sellmann 2002), the knapsack constraint (Trick 2001; Sellmann 2003a; Trick 2003), the shorter path constraint (Sellmann 2003b; Sellmann, Gellermann, and Wright 2007), or the not-too-heavy spanning tree constraint (Dooms and Katriel 2006; 2007).

Typically these constraints consider a particular constraint in conjunction with a bounded (often linear) objective. That is, apart from enforcing the basic constraint (allDifferent, bounded linear resource, directed path, spanning tree, etc) these optimization constraints merely consider a bound on the objective, yet they do not require an *optimal* assignment. This also explains some rather unusual names, such as shorter path instead of shortest path constraint, or not-too-heavy spanning tree instead of minimum spanning tree constraint. The reason why bounds are considered rather than optimal assignments is that there may be other constraints in the problem which prohibit solutions that would be optimal with respect to the basic constraint in isolation. For the problem at hand, we can enforce the extreme variant of the linear resource constraint. We believe that the same idea can also be applied in many other applications and for classical combinatorial optimization problems, e.g., in bin packing.

**Theorem 3.** *Achieving generalized arc-consistency (GAC) for the ExtremeLinearResource Constraint is NP-hard.*

*Proof.* We reduce Number Partitioning to GAC on ExtremeLinearResource. Given $n$ natural numbers $a_1, \dots, a_n$, we need to decide whether there exists a set $I \subseteq \{1, \dots, n\}$ such that $\sum_{i \in I} a_i = S$ with $S = \left\lfloor \frac{\sum_{i=1}^n a_i}{2} \right\rfloor$. We construct an ExtremeLinearResource constraint over $n+1$ binaries with weights $a_1, \dots, a_n$ associated with $x_1, \dots, x_n$ and 1 associated with $x_{n+1}$. The budget capacity is $S$.

Now, we must filter 0 from the domain of $x_{n+1}$ if and only if the number partitioning instance has no solution: Assume that 0 must be filtered from the domain of $x_{n+1}$. This means that there is no way of fully exhausting the resource (without overfilling!) using the first $n$ items with weights $a_1, \dots, a_n$. Consequently, there is no solution to the number partitioning problem. On the other hand, if there is no way of exhausting the budget exactly, then in any selection of items that obeys the budget constraint there is always room to add item $x_{n+1}$. Consequently, GAC must infer that $x_{n+1} = 1$ and thus filter value 0 from the domain of $x_{n+1}$. □

**Theorem 4.** *GAC for the ExtremeLinearResource constraint is achievable in pseudo-polynomial time $O(n^2 C)$, where $n$ is the number of items and $C$ the available budget.*

*Proof.* We set up a three-dimensional dynamic program (DP): the first dimension is the item index (size $n+1$), the second is the index of a skipped item with smallest weight (size $n+1$) and the last dimension is the total weight of the selection (size $C$). Then, node $N_{i,m,k}$ is reached by branching on items $1, \dots, i$ such that ($m = 0$ or ($x_m = 0$ and $c_m \leq \{c_j \mid 1 \leq j \leq i, \; x_j = 0\}$)) and $\sum_{j=1}^{i} x_j = k$. The construction starts at node $N_{0,0,0}$. Now, we iterate through

[1]All experiments were conducted on 1.6 GHz Intel Core i7 Q170 machines with a 4-core CPU, 6 MB cache, and 12 GB RAM, running 64-bit Fedora 20. The code was compiled with g++ 4.8.3 and full optimization. No external software was used.

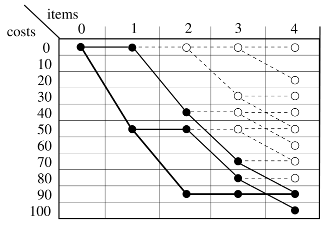

Figure 1: Illustration of the two-dimensional dynamic program with four items with costs 50, 40, 30, and 20, the budget limit is 100. Dashed lines and hollow nodes depict nodes and edges which are removed. In the situation shown no values can be filtered from variable domains. However, after deciding, e.g., not to invest in link 2, it is immediately inferred that we must invest in links 1, 3 and 4.

layers $i = 1, \dots, n$: For all existing nodes $N_{i-1,m,k}$, if 0 is in the domain of $x_i$, we add a directed edge to (the potentially new) node $N_{i,h,k}$ where $h = m$ if $m > 0$ and $c_h \geq c_m$, and $h = i$ otherwise. This edge corresponds to branching on $x_i = 0$. If 1 is in the domain of $x_i$, and if $k + c_i \leq C$, we add an edge from $N_{i,m,k}$ to (the potentially new) node $N_{i+1,m,k+c_{i+1}}$. This edge corresponds to branching on $x_i = 1$. Finally, we remove all nodes and adjacent edges from where we cannot reach any node $N_{n,m,k}$ with $c_m + k > C$. As a result, we only retain nodes and edges on paths from $N_{0,0,0}$ to some node $N_{n,m,k}$ with $c_m + k > C$. These paths directly correspond to feasible solutions to the ExtremeLinearResource constraint. Following the filtering based on dynamic programs introduced in (Trick 2001), if and only if there is no edge that corresponds to branching on $x_i = b$ (with $b \in \{0, 1\}$) on layer $i$, then and only then $b$ can and must be removed from the domain of variable $x_i$. □

In summary, we have found that the ExtremeLinearResource constraint is NP-hard to filter exactly, but it is hard in the nicest possible way, allowing us to filter the constraint in pseudo-polynomial time.

In practice, we may wish to limit the size of the dynamic program a bit further. To eliminate a factor $n$ in the complexity, we can keep a record on an upper bound on the smallest cost skipped on any path to a node and a lower bound on the slack left in the budget on any path to the leaf-level. Then, whenever we are guaranteed that the smallest item skipped has cost lower or equal the remaining slack in the budget we can remove a node and analogously also edges in the dynamic program. Filtering variable domains then works exactly as before (see Figure 1).

To achieve this, at each node we record two values, SKIP and SLACK. At level $i \in \{0, \dots, n-1\}$ and budget $k \in \{0, \dots, C\}$ the following recursion equations govern: $\text{SKIP}_{0,k} = \infty$, $\text{SLACK}_{n,k} = C - k$,

$$\text{SKIP}_{i+1,k} = \max\{\text{SKIP}_{i,k-c_i}, \min\{\text{SKIP}_{i,k}, c_i\}\}, \text{ and}$$

$$\text{SLACK}_{i,k} = \min\{\text{SLACK}_{i+1,k+c_i}, \text{SLACK}_{i+1,k}\}\},$$

whereby we discard values from consideration which do not belong to feasible nodes and edges in the dynamic program.

| Instance | Solution | Value | ELR | LR |
|---|---|---|---|---|
| $B_1$-Low | {10,17,21,22,23,25} | 83.0801 | 0.03 | 0.14 |
| $B_2$-Low | {4,10,12,17,20,21,22,25} | 66.1877 | 0.05 | 2.22 |
| $B_3$-Low | {3,4,10,16,17,19–22,25} | 57.6802 | 0.11 | 16.51 |
| $B_1$-High | {10,17,21,22,23,25} | 212.413 | 0.03 | 0.11 |
| $B_2$-High | {4,10,12,17,20,21,22,25} | 120.083 | 0.08 | 2.18 |
| $B_3$-High | {3,4,10,16,17,19–22,25} | 78.4017 | 0.11 | 17.1 |

Table 1: Istanbul Benchmark Results

This heuristic filtering algorithm comes with no theoretical guarantees, but it runs in $O(nC)$ and it may only miss very few filtering opportunities in practice, thus making it a favorable choice within a practical solution approach. We will use this algorithm in the following experiments.

## Experimental Results

Using the ExtremeLinearResource constraint, we conduct a plain branch-and-bound to find the optimal investment for our pre-disaster planning problem: At each search node, we first perform constraint filtering. Then, we compute a dual (since we are minimizing: lower) bound on the objective simply by assuming that we could invest in all links undecided so far. As branching heuristic, we branch on highest cost investments first.

### Istanbul Earthquake Preparation

We test this algorithm on the Istanbul benchmark introduced in (Peeta et al. 2010): We are expected to move casualties after an earthquake from highly affected areas, as determined in the Japan International Cooperation Agency Report (2002), to areas with large medical support capacities. The task is to minimize the sum of the expected shortest-path distances for five OD-pairs. In the first set of three instances $B_1$-Low, $B_2$-Low, and $B_3$-Low, the penalty for experiencing a disconnect for an OD-pair is 31, 31, 28, 19, and 35, respectively. This is to model the costs of airlifting injured people to regions with properly equipped medical facilities.

In the corresponding instances with the suffix -High, the penalty for a disconnect is 120, for all five OD-pairs. It is important to note that neither (Peeta et al. 2010) nor (Prestwich, Laumanns, and Kawas 2013) consider paths that exceed the low penalties, even when the penalty of a disconnect is high. The total number of paths considered is, in both papers, 24. In $B_1$ the total budget is 10% of the budget needed to invest in all links in the network, in $B_2$ it is 20%, and in $B_3$ it is 30%. The exact survival probabilities before and after investment are given in (Peeta et al. 2010). These values were determined by structural engineers using domain-specific information, as summarized in the 2003 Master Earthquake Plan of the Istanbul Municipality. The independent failure probabilities on the network links as well as the fact that investment decisions affect probabilities itself render this problem a very hard stochastic optimization problem.

In Table 1 we show the results. We give the optimal solutions and corresponding objectives for all six benchmark instances, as well as the run-times in seconds based on non-linear integer programs, once using the ExtremeLinearResource constraint (ELR), and once using a standard

LinearResource constraint (LR) which limits the budget but does not enforce that we exhaust it to a point where we could not afford any other investment. In both cases we use the same dual bound and the same branching strategy.

The new extreme constraint results in speed-ups of up to two orders of magnitude. Moreover, overall our approach massively outperforms the previously published approaches. Both algorithms in (Peeta et al. 2010) and (Prestwich, Laumanns, and Kawas 2013) run for ca. 5 minutes on each instance, whereby the first gives approximate results only. The AI-based non-linear optimization approach, on the other hand, finds six solutions and proves their respective optimality in less than half a second for all six instances *together*. Note that this time includes everything, the computation of the shortest paths up to the penalty values, the coverings and set-up times of the non-linear integer programs, as well as their solution based on the ExtremeLinearResource constraint. The new approach thus runs three orders of magnitude faster than the previous state-of-the-art.

Comparing with the solutions given in (Prestwich, Laumanns, and Kawas 2013), we find that our solution to $B_2$-High has a slightly higher expected path length than the one given in that paper. The reason for this is that the solution given there actually violates the 20% budget constraint. Indeed, our solutions show that the optimal investment decisions for the low and high penalties are *exactly the same.* The objective value differs, of course, but the links in which to invest are identical.

To challenge the new method, we considered three other problem variants. In the first *all paths* up to the penalty of 120 may serve as shortest paths. Doing so results in tripling the number of paths considered, which jumps from 24 to 72, roughly 15 different evacuation routes for each OD-pair. The number of scenarios that our approach considers also jumps, from 156 to 8,491. For $B_1$ the new approach now requires 8 seconds, for $B_2$ 22 seconds, and for $B_3$ 26 seconds. Considering more paths obviously increases the complexity of the problem, as the scenarios which result in the same shortest routes become harder and harder to represent compactly. This confirms the same finding in (Prestwich, Laumanns, and Kawas 2013). Curiously, looking at the solutions, we find that the optimal investment decisions are again the very same as the ones given in Table 1. In practice this justifies simplifying problem instances by considering paths up to a relatively low threshold only as budget constrained investment decisions are hardly affected by considering large numbers of sub-optimal paths.

In the second and third variants of the benchmark we consider different objectives: First, instead of considering the sum of all expected shortest-path lengths we minimize the maximum path length. Since some OD-pairs are much closer together than others, we normalize the penalty for each OD-pair by dividing by the length of the shortest path for this pair. Therefore, we minimize the maximum expected percent increase relative to the shortest path for each OD-pair. The final variant minimizes the sum of squares of these expected ratios rather than their maximum.

Run-times in seconds and optimal solutions are given in Table 2. We observe that these objectives do not make the overall problem harder for the AI-based approach. This is interesting since the sum of squares would be much harder to model as a traditional two-stage stochastic program. On the larger instances, the use of the extreme version of the linear resource constraint leads to over 500-fold speed-ups over the traditional constraint. Overall, the new approach does not only work significantly faster than existing approaches, it also offers much greater flexibility regarding the exact choice of objective function.

| Instance | Solution | Value | ELR | LR |
|---|---|---|---|---|
| $B_1$-max | {19–22,28} | 1.99401 | 0.01 | 0.11 |
| $B_2$-max | {9,16,20,21,22,25} | 1.57278 | 0.01 | 3.4 |
| $B_3$-max | {4,7,10,12,16,17,19–22,25} | 1.30401 | 0.03 | 20.9 |
| $B_1$-squ | {10,17,21,22,23,25} | 15.0528 | 0.03 | 0.14 |
| $B_2$-squ | {4,10,12,17,20,21,22,25} | 9.29439 | 0.06 | 2.33 |
| $B_3$-squ | {3,4,10,16,17,19–22,25} | 6.67251 | 0.08 | 18.3 |

Table 2: Istanbul Benchmark Variants

## Bay Area Earthquake Preparation

With the introduction of the ExtremeLinearResource constraint, the Istanbul benchmark instances, which were impossible to solve to optimality 18 months ago, no longer pose much of a problem. We therefore introduce a new, even more challenging benchmark set. Based on the Highway Evacuation Plan of the Federal Highway Administration (FHWA) we consider the emergency mass transportation highway evacuation network for the California bay area (Vásconez and Kehrli 2010)).

The scenario again considers the possibility of an earthquake, whereby in this case we assume that most urban infrastructure, housing as well as utilities (water, gas, and electricity), would be destroyed, thus requiring large numbers of citizens to leave the bay and flee to lesser populated areas to the North and East (see Figure 2). The challenge is to mass evacuate people from San Francisco (node 2), Berkeley/Oakland (node 7), Palo Alto/Mountain View (node 16) and Santa Clara/San Jose (node 18), each to any one of five possible evacuation points: Petaluma (node 0), Vallejo (node 19), Benicia (node 20), Antioch (node 6), or Tracy (node 15). Failure probability and cost models follow a similar logic as in the Istanbul benchmark, with highest costs and highest failure probabilities for bridges, whereby we assume that some positive failure probability persists even after in-

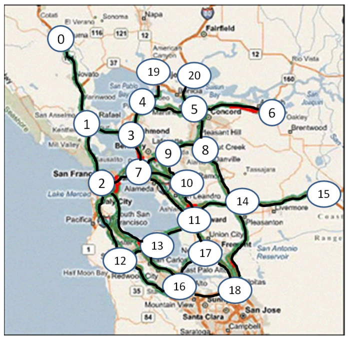

Figure 2: Bay Area Highway Mass Evacuation Network

vesting in a link. Budgets are again limited to 10%, 20%, and 30% of the cost of investing in all links in the network, denoted by subfixes, $E_1$, $E_2$, and $E_3$.

Note that budgets that would allow investment in 30% of all possible investments are improbable, we consider them here to test the limits of the approaches developed. In practice, less than 10% are much more realistic. Moreover, while the size of the networks considered in these and the Istanbul instances may appear small at first, it is important to stress that real-world evacuation networks are fortunately small: Roads with intersections and one lane roads are not suitable for mass evacuations. Cars break down, accidents happen, and even via a multi-lane freeway we can maximally evacuate 2,000 cars per lane and hour. Consequently, in areas with millions of people investment decisions focus on designated mass evacuation routes only. Nevertheless, computationally these problems are still extremely hard, because the search space grows exponentially in the number of network links, and the objective function is highly non-linear which massively increases the complexity even for evaluating a given investment plan.

For each of the four areas to be evacuated we now face roughly 20 possible evacuation routes. The model considers 81 paths and 16,196 covers total. That is over 100 times the number of scenarios than we needed to consider for the Istanbul benchmark. In Table 3 we minimize again three different objectives as before: sum of all expected shortest path lengths (-sum), maximum percent increase (-max), and sum of squared percent increases in expected shortest path length (-squ). We observe similar trends as before, minimizing the maximum percent increase in expected shortest path length is easiest, the other two objectives are roughly equally hard, whereby the Bay Area instances are clearly harder than the Istanbul instances. Overall, even when having to consider two orders of magnitude more scenarios, we can still solve each instance in a few seconds.

We wanted to see how far we could push this approach and considered a -High variant for this benchmark as well. In these instances, we consider 351 evacuation routes, on average more than 85 per evacuation area. This leads to over 3 million scenarios that need to be considered, roughly 20,000 times (!) the number of scenarios considered for the Istanbul benchmark. Just computing the non-linear objective function for this problem takes over 80 seconds. For the 10% budget limit we can solve this problem to optimality in 2 minutes, for 20% in 4 minutes, and for 30% in 6 minutes.

Since improved upper bounds on the objective result in better pruning, we used the ExtremeLinearResource constraint and the monotonicity of the objective function to derive a primal heuristic for finding good feasible points before we start the branch and bound procedure. Using this heuristic, solution times for the most difficult Bay Area -High variants reduce to 2.1, 4.5, and 12.8 seconds, respectively, plus 80 seconds each for constructing the objective function. After constructing the objective, roughly equal time is spent between the primal heuristic and the branch and bound.

| Instance | Solution | Value | ELR |
|---|---|---|---|
| $E_1$-sum | {3,6,8} | 58.0981 | 1.85 |
| $E_2$-sum | {3,6,8,9} | 55.9286 | 3.5 |
| $E_3$-sum | {3,6,7–10,12,27,28} | 54.7743 | 4.5 |
| $E_1$-max | {3,6,8} | 1.94013 | 1.21 |
| $E_2$-max | {3,6,8,9} | 1.67574 | 1.35 |
| $E_3$-max | {1,3,6,8,9} | 1.59876 | 1.55 |
| $E_1$-squ | {3,6,8} | 8.4507 | 1.72 |
| $E_2$-squ | {3,6,8,9} | 7.51121 | 3.4 |
| $E_3$-squ | {0,3,5,6,7–10,12,27} | 7.1024 | 4.4 |

Table 3: Bay Area Benchmark

| Instance | Solution | Value | ELR |
|---|---|---|---|
| $E_1$-low | {6,8,12,27} | 57.921 | 3.2 |
| $E_2$-low | {3,6,8,9} | 55.935 | 4.45 |
| $E_3$-low | {3,6–10,12,27,28} | 54.719 | 6.97 |
| $E_1$-high | {6,8,12,27} | 61.518 | 140 |
| $E_2$-high | {6,8,9,10,27} | 58.614 | 180 |
| $E_3$-high | {3,6–12,27} | 57.172 | 400 |

Table 4: Bay Area Benchmark with Correlation

## PTNP With Failure Correlation

Next, we considered scenarios where link failures are *correlated*. We believe this is a very important aspect in practice: Bridges of the same construction type may exhibit similar resistance depending on the strength and frequency of shock waves. Links of same orientation may show some correlation, as shock waves hit them equally longitudinally or crosswise. Low ocean drives may sink under the water level or could be flooded by Tsunamis which would affect them equally. All these effects may correlate failure probabilities.

We model correlated link failure distributions by means of Bayesian networks. The variables modeling the investment decisions are binary value nodes $X_i, i = 1, \ldots, k$ of the network, as are the nodes $L_j, j = 1, \ldots, n$ that describe the failure probabilities of the links. The Bayesian network is required to be *monotone* in the sense that for all variable vectors $X$ and $\overline{X}$ with $X \leq \overline{X}$ and all pairwise disjoint subsets $I$, $J$, $K$ all conditional link failure probabilities satisfy $\mathrm{P}(L_K|L_I, \hat{L}_J, X) \geq \mathrm{P}(L_K|L_I, \hat{L}_J, \overline{X})$, i.e., that the probability for simultaneous failure of the links in $K$, given that the links in $I$ fail and those in $J$ survive, reduces if additional investments are made. This is a generalization of the concept of monotonicity of Bayesian networks as, e.g., discussed in (van der Gaag, Bodlaender, and Feelders 2004).

The monotonicity of the Bayesian network implies that the monotonicity of the objective is retained. This and the budget constraint are the two primary sources for inference in our approach. Consequently, the AI-based approach handles these problems as well. Table 4 shows that we can efficiently handle such correlated distributions as well, both with low and high cutoff thresholds.

## Best-Effort Solutions

To assess how the approach scales with increased network sizes we took the Istanbul benchmark and successively refined the network by splitting every edge into $m$ sub-edges, simulating that instead of a single investment decision on every edge $m$ investment decisions can be taken independently of each other. Of course, the investment costs are chosen in such a way that they add up to the original cost for investing into the unsplit link, and the survival probabilities of the sub-links are ensured to be consistent with the original survival probabilities for the unsplit link. We expect that solution times show some form of exponential increase with increasing number of investment decisions to be taken. In Table 5 in three columns we show the solution times in seconds of the full branch and bound proving global optimality, the primal heuristic, which usually finds a solution deviating not more than 5% from the global optimum, and a simplified primal heuristic, which usually finds a solution within 15% of the global optimum. Due to the stochastic nature of the PTNP the size of the sample space for computing the expected values increases exponentially with the number of links, so in the biggest case just the evaluation of the objective function at a single point takes more than one second.

We find that the full branch and bound scheme works well up to about 150 decision variables, the expensive heuristic up to about 600 decision variables, and the simplified heuristic up to about 900 decision variables. For investment decision problems like ours runtimes of about 4 hours of computation time are completely acceptable in practice. These experiments thus show that we can effectively solve real-size PTNP problems, especially when keeping in mind that in reality instances hardly reach sizes bigger than the Bay Area instances.

| $\#edges$ | Branch and Bound | Heuristic | Simple Heuristic |
|---|---|---|---|
| 30 | 0.03 | 0.01 | 0.01 |
| 60 | 0.54 | 0.21 | 0.01 |
| 90 | 25.8 | 1.73 | 0.07 |
| 150 | 13777 | 26 | 0.67 |
| 210 | — | 129 | 2.45 |
| 300 | — | 1227 | 10.7 |
| 600 | — | 13695 | 253 |
| 900 | — | — | 1628 |

Table 5: Istanbul Subdivision Benchmark Results

## Conclusion

We developed a novel AI-based solution method for the pre-disaster network preparation problem. We formulated the problem as a highly non-linear integer programming problem and developed a practically efficient propagator for the extreme linear resource constraint. Based on this approach we were able to effectively close the Istanbul benchmark, with run-time improvements over the best existing methods of three orders of magnitude. We then considered an even more challenging set of benchmark instances with five-orders of magnitude more scenarios, larger network sizes, and with and without link failure correlation. We showed that the AI approach is flexible and efficient enough to also solve these instances in affordable time.

## References

Dooms, G., and Katriel, I. 2006. The minimum spanning tree constraint. In *Principles and Practice of Constraint Programming-CP 2006*. Springer. 152–166.

Dooms, G., and Katriel, I. 2007. The "not-too-heavy spanning tree" constraint. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. Springer. 59–70.

Focacci, F.; Lodi, A.; and Milano, M. 1999. Cost-based domain filtering. In *Principles and Practice of Constraint Programming–CP'99*. Springer Berlin Heidelberg. 189–203.

Kerivin, H., and Mahjoub, A. R. 2005. Design of survivable networks: A survey. *Networks* 46(1):1–21.

LeBras, R.; Dilkina, B. N.; Xue, Y.; Gomes, C. P.; McKelvey, K. S.; Schwartz, M. K.; Montgomery, C. A.; et al. 2013. Robust network design for multispecies conservation. In *AAAI 2013*.

Peeta, S.; Sibel Salman, F.; Gunnec, D.; and Viswanath, K. 2010. Pre-disaster investment decisions for strengthening a highway network. *Computers & Operations Research* 37(10):1708–1719.

Prestwich, S. D.; Laumanns, M.; and Kawas, B. 2013. Value interchangeability in scenario generation. In *Principles and Practice of Constraint Programming*, 587–595. Springer.

Prestwich, S. D.; Laumanns, M.; and Kawas, B. 2014. Symmetry breaking for exact solutions in adjustable robust optimisation. In *ECAI 2014*, 741–746.

Régin, J.-C. 1999. Arc consistency for global cardinality constraints with costs. In *Principles and Practice of Constraint Programming–CP'99*. Springer Berlin Heidelberg. 390–404.

Régin, J.-C. 2002. Cost-based arc consistency for global cardinality constraints. *Constraints* 7(3-4):387–405.

Sellmann, M.; Gellermann, T.; and Wright, R. 2007. Cost-based filtering for shorter path constraints. *Constraints* 12(2):207–238.

Sellmann, M. 2002. An arc-consistency algorithm for the minimum weight all different constraint. In *Principles and Practice of Constraint Programming-CP 2002*, 744–749. Springer.

Sellmann, M. 2003a. Approximated consistency for knapsack constraints. In *Principles and Practice of Constraint Programming–CP 2003*, 679–693. Springer.

Sellmann, M. 2003b. Cost-based filtering for shorter path constraints. In *Principles and Practice of Constraint Programming–CP 2003*, 694–708. Springer.

Trick, M. A. 2001. A dynamic programming approach for consistency and propagation for knapsack constraints. In *3rd international workshop on AI and OR techniques in Constraint Programming for Combinatorial Optimization Problems (CP-AI-OR)*, 113–124.

Trick, M. A. 2003. A dynamic programming approach for consistency and propagation for knapsack constraints. *Annals of Operations Research* 118(1-4):73–84.

van der Gaag, L. C.; Bodlaender, H. L.; and Feelders, A. 2004. Monotonicity in bayesian networks. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, 569–576. AUAI Press.

Vásconez, K. C., and Kehrli, M. 2010. Highway evacuations in selected metropolitan areas: Assessment of impediments. Technical Report FHWA-HOP-10-059, Federal Highway Administration, Office of Transportation Operations, Washington, DC 20590.