Maximum Expected Value Partial Failure Path Problem

Noam Goldberg, Michael Poss

Maximum Expected Value Partial Failure Path Problem

Noam Goldberg Michael Poss

September 16, 2020

1 Introduction

Consider a setting in which each of m items has an associated profit and probability of success. In this paper we consider a maximum expected value partial failure (s,t)-path problem, where each (s,v) subpath of an (s,t) path, where v is some vertex on a path from s to t, has a value which is the product of probabilities of arcs along the path s-to-v, times its profit that is the sum of arc profits.

Our problem bears similarities with the probabilistic all-or-nothing problem, which is to select a subset of items so as to maximize the expected total profit – the product of probabilities of selected items times the sum of profits of these items. In particular, the all-or-nothing probabilistic path problem [**?**], has the subsets of items corresponding to paths in a graph, each of which has a value that is equal to the product of its underlying arc probabilities times the sum of its arc profits. Similar all-or-nothing problems are also considered for different settings such as unrestricted subsets of a ground set and matchings in a graph [**?**].

While all-or-nothing path problems have applications in designing series-connected subsystems, among others, the partial failure variant studied in the current paper has important applications in kidney-exchange problems. Paired-kidney donation graphs indicate donor-recipient compatibility using directed arcs. Each vertex in such a graph corresponds to a pair of a recipient (patient) and donor – typically a family member who is willing to donate a kidney in exchange for the related patient receiving one [**?**]. Because exchanges must occur simultaneously, there is significant uncertainty with respect to the success of a particular donation to take place as a part the an exchange. For example, a particular donor may not show up on a given day due to illness or simply reneging on the agreement. This setting with uncertainty gives rise to partially successful probabilistic paths and the objective considered in this paper; see [**?**].

The kidney-exchange problem described in [**?**] and [**?**] is a market clearing problem where all (recipient) vertices must be covered by paths or cycles. Cycles are initiated by paired donors and paths are initiated by yet-to-be-paired altruistic donor vertices (vertices that have only outgoing arcs). The problem of determining a maximum expected-value partial failure path problem can be considered as the pricing problem of the market clearing problem. Also, the maximum expected-value partial failure path problem can be considered independently; as soon as an altruistic donor arrives he/she can be used to initiate a path of maximum expected value. The current paper is focused on improving and developing new exact solution techniques and heuristics for this problem.

Next we formally describe the problem and relate our mixed-integer nonlinear programming (MINLP) formulation to the optimization problem as it appears in the literature. We then propose a linearization and reformulation of the MINLP, a dynamic program and several heuristics.

Finally, we conclude with computational experiments to evaluate the proposed techniques along with different attempts to strengthen our proposed formulation.

2 Problem definition

We now formally state the probabilistic partial-failure elementary path problem. Let $G = (V, E)$ be a directed graph, $s, t \in V$ be the source and the sink, and let $\delta^+(v) = \{v \in V : (u, v) \in E\}$ and $\delta^-(u) = \{v \in V : (v, u) \in E\}$ be sets of direct successors and predecessors of node $i \in V$, respectively. One can readily consider the cases of multiple sources S and sinks T by adding a supersource s and a supersink t that are connected only to nodes from S and T, respectively. For each edge $e = (u, v) \in E$, we are given a positive (integer) profit $c_e = c_{uv} \geq 0$ and a probability of success $p_e = p_{uv} \in [0, 1]$. Let $|E| = m$ and $|V| = n$. The partial-failure path problem is to find a path π from s to t such that the objective function

$$z(\pi) = \sum_{(u,v)\in\pi} c_{uv} \prod_{f\in\pi(v)} p_f. \tag{1}$$

First, the following proposition relates the simplified form of the objective (**??**) to the objective function considered in for determining kidney exchange paths initiated by altruistic donors in [**?**].

Proposition 1. *The partial failure objective*

$$\sum_{(u,v)\in\pi} (1 - p_{uv}) \sum_{e\in\pi(u)} c_e \prod_{f\in\pi(u)} p_f + \sum_{e\in\pi} c_e \prod_{f\in\pi} p_f, \tag{2}$$

*considered in [**?**] is equivalent to* (**??**).

Proof. We rewrite (**??**) as

$$\begin{aligned} z(\pi) &= -\sum_{(u,v)\in\pi} p_{uv} \sum_{e\in\pi(u)} c_e \prod_{f\in\pi(u)} p_f + \sum_{(u,v)\in\pi} \sum_{e\in\pi(u)} c_e \prod_{f\in\pi(u)} p_f + \sum_{e\in\pi} c_e \prod_{f\in\pi} p_f \\ &= -\sum_{(u,v)\in\pi} \sum_{e\in\pi(u)} c_e \prod_{f\in\pi(v)} p_f + \sum_{(u,v)\in\pi} \sum_{e\in\pi(v)} c_e \prod_{f\in\pi(v)} p_f \\ &= \sum_{(u,v)\in\pi} c_{uv} \prod_{f\in\pi(v)} p_f. \end{aligned}$$

□

Then, the maximum expected value partially successful path problem is

$$\max\{z(\pi) : \pi \in \mathcal{P}(t)\}, \tag{3}$$

where $\mathcal{P}(t)$ is the set of all *elementary* paths from s to t. Note that in the case that $p_e = 1$ for all $e \in E$ the objective z resembles that of a classical traveling repairman and minimum latency problems [**?**, **?**] where the problem is to determine a tour of a graph such that this objective is minimized (in contrast to the maximization in (**??**)). Also related is the orienteering problem: an orienteering path problem is to maximize the number or weight of vertices visited subject to a time limit on all visits [**?**, **?**]. In particular a discounted orienteering problem is a special case with $p_e = P$ for all $e \in E$ and some $0 < P < 1$ (the problem is introduced in [**?**] for undirected graphs). Further the problem (**??**) with $p_e = 1$ for all $e \in E$ is a longest path problem that is strongly NP-hard in general directed graphs.

3 MILP Formulation

Although the problem (**??**) appears to be highly nonlinear in fact it can be formulated as a MILP. In the following let $0 < M_{uv} \leq 1$ be a sufficiently large probability constant. Let the path flow polytope in the given graph $G = (V, E)$, with $s, t \in V$, be denoted by

$$\Pi = \left\{ x \in \mathbb{R}^{|E|} \;\middle|\; \begin{array}{c} \sum_{i \in \delta^-(j)} x_{ij} = \sum_{i \in \delta^+(j)} x_{ji} \text{ for } v \in V \setminus \{s,t\} \\ \sum_{i \in \delta^+(s)} x_{si} = \sum_{i \in \delta^-(t)} x_{it} = 1 \end{array} \right\}.$$

and consider the following MILP formulation.

$$\text{max} \quad \sum_{(u,v) \in E} c_{uv} q_{uv} \tag{4a}$$

$$\text{subject to:} \quad \sum_{w \in \delta^-(u)} q_{wu} = \sum_{v \in \delta^+(u)} \frac{1}{p_{uv}} q_{uv} \quad u \in V \setminus \{s,t\} \quad (u,v) \in E : u, v \neq s, u \neq t \tag{4b}$$

$$q_{uv} \leq M_{uv} x_{uv} \qquad (u,v) \in E \tag{4c}$$

$$x \in \Pi \qquad (u,v) \in E \tag{4d}$$

$$x_{uv} \in \{0,1\}, q_{uv} \geq 0 \qquad (u,v) \in E \tag{4e}$$

For each $(u, v) \in E$ the decision variable x_{uv} is used to indicate whether (u, v) is selected as a path edge, and the auxiliary variables q_{uv} is used to define $q_{uv} = \prod_{(u,v) \in \pi(v)} p_{uv}$ as in $z(\pi)$. The constraints corresponding to (**??**) are the $s - t$ path-flow conservation constraints. Constraint (**??**) enforces that $q_{uv} = 0$ for each $(u, v) \in E$ with $x_{uv} = 0$. Finally, (**??**) implements the recursion to accumulate the probability product for each pair of selected incident edges (w, u) and (u, v) with $x_{uv} = x_{wu} = 1$, it requires that $q_{wu} p_{uv} = q_{uv}$.

To strengthen the formulation (**??**) it is desirable to select M_{uv} for each $(u, v) \in E$ as small as possible. It can be observed it suffices to set $M_{uv} = \bar{p}(u) p_{uv}$ where $\bar{p}(v)$ denotes the maximum probability of a path from s to v for all $v \in S \setminus \{s\}$ and $\bar{p}(s) = 1$.

Proposition 2. *$\bar{p}(u)$ can be computed in polynomial time.*

Proof. We consider the directed graph G with weight $\omega_e = -\ln p_e \geq 0$ for each edge $e \in E$ and denote by $d(u)$ be the value of the shortest path from s to u. We see that $\bar{p}(u) = e^{-d(u))}$. □

Nothing prevents x to form cycles in the optimal solution of (**??**), so we must add cycle-breaking inequalities. We consider two alternative ways of breaking the cycles. First, following [**?**], we consider the *generalized cutset inequalities*, which can be defined as

$$\sum_{(u,v) \in \delta^+(S)} x_{uv} \geq \sum_{(w,v) \in \delta^+(w)} x_{wv} \qquad w \in S, S \subseteq V \setminus \{s,t\}, |S| \geq 2. \tag{5}$$

These constraints break any cycle by enforcing that the number of arcs leaving S be not smaller than the number of arcs outgoing from any node k of S.

Our second type of inequalities ensures acyclic optimal solutions exist whenever a technical condition is satisfied.

$$\sum_{v \in \delta^+(u)} x_{uv} \leq 1 \qquad u \in V \setminus \{t\}. \tag{6}$$

Proposition 3. *The problem* (**??**) *with* (**??**) *has an acyclic optimal solution if* $\prod_{f\in c} p_f < 1$ *for each cycle* $c \subseteq E$.

Proof. Consider (x, q) that is a feasible solution of (**??**) with (**??**). Note that by the flow-conservation constraints and the integrality of x, the support of x is a union of one path π from s to t and a set of cycles $C \subset 2^E$. Suppose that there is some $u \in V$ that is incident to edges in two different cycles or an edge in a cycle and and edge in π. Then, by the integrality of x, $\sum_{v\in\delta^+(u)} x_{uv} > 1$, thereby violating (**??**). It follows that the cycles $c \in C$ and the path π are all vertex disjoint. Now further suppose for the sake of deriving a contradiction that (x, q) is an optimal solution with minimal $|C| > 0$ and consider a cycle $c \in C$. If $q_e = 0$ for each $e \in c$, then a new solution (x', q) can be defined (using (x, q)) by setting $x'_e = 0$ for each $e \in c$. The objective value of (x', q) is equal to that of (x, q) and has one less cycle, thereby establishing a contradiction. Otherwise, if $q_e > 0$ for some $e \in c$, constraints (**??**) written for all $(u, v) \in c$ imply that $\prod_{f\in c} p_f = 1$. □

4 Heuristic algorithms rounding the probabilities

We present in the this section two heuristic algorithms inspired by a dynamic programming algorithm that considers rounded probabilities and assumes that G is acyclic. Let $Z(P, u)$ denote the maximum profit among the paths from s to u having a probability of P, formally

$$Z(P,u) = \max\left\{ z(\pi) : \pi \in \mathcal{P}(u), \prod_{f\in\pi} p_f = P \right\}, \tag{7}$$

and let $\pi(P, u)$ denote the path that reaches the maximum in (**??**). One readily verifies using definition (**??**) that

$$z(\pi(P/p_{vu}, v) \cup \{(v, u)\}) = z(\pi(P/p_{vu}, v)) + c_{vu} \times P. \tag{8}$$

Assuming G is acyclic, we naturally obtain

$$Z(P,u) = \begin{cases} \max\limits_{v\in\delta^-(u)} \{Z(P/p_{vu}, v) + c_{vu} \times P\} & P \le 1 \text{ and } u \neq s \\ 0 & P = 1 \text{ and } u = s \\ -\infty & \text{otherwise,} \end{cases} \tag{9}$$

which can be solved by following the order of a topological sort starting with s. The number of states involved in recursion (**??**) cannot, in general, be bounded by a pseudo-polynomial function of the input data. Therefore, we assume in what follows that there exist $\alpha > 0$ and $k_e \in \mathbb{Z}$ such that $p_e = \alpha^{k_e}$ for each $e \in E$. In that case, (**??**) can be reformulated as

$$\tilde{Z}(L,u) = \begin{cases} \max\limits_{v\in\delta^-(u)} \{Z(L - k_{vu}, v) + c_{vu} \times \alpha^L\} & L \ge 1 \text{ and } u \neq s \\ 0 & L = 0 \text{ and } u = s \\ -\infty & \text{otherwise.} \end{cases} \tag{10}$$

Proposition 4. *If* G *is acyclic and there exists* $\alpha > 0$ *such that for all* $e \in E$, $p_e = \alpha^{k_e}$ *for some* $k_e \in \mathbb{Z}$, *then DP* (**??**) *can be applied to solve problem* (**??**) *with a polynomial running time.*

Proof. The time complexity of computing all values of $Z(P,u)$ is $O(-n^2 \sum_{f\in E} \log_\alpha p_f)$. □

We discuss next how to use DP (**??**) to provide heuristic solutions whenever the two assumptions of Proposition **??** do not hold. Consider first the case where the probabilities cannot be expressed as the integer powers of a common base α. In that case, we can artificially choose a base α and define

$$k_e = \lceil \log_\alpha p_e \rceil.$$

If G is acyclic, then DP (**??**) can be used to compute a path π which is hopefully close to the optimal solution of problem (**??**). Intuitively, the closer from 1 is α, the closer π is from the optimal solution. This intuition is made more precise in Appendix **??** when we turn DP (**??**) into a Fully-Polynomial Time Approximation Scheme.

The presence of cycles in G needs more significant changes to DP (**??**), and we present next two extensions of DP (**??**) that return elementary paths. Given path π from s to some node u, our first algorithm disregards any arc e that leads to a cycle in $\pi \cup \{e\}$. Denoting by $\Pi(L,u)$ the set of vertices along a path with value $\tilde{Z}(L,u)$, we consider appending (the vertices incident to) arc (v,u) to path $\Pi(L,v)$ only if $i \notin \Pi(L,v)$. We modify (**??**) as follows:

$$\tilde{Z}(L,u) = \begin{cases} \max\limits_{v\in\delta^-(u): u\notin\Pi(L-k_{vu},v)} \{Z(L-k_{vu},v) + c_{vu} \times \alpha^L\} & L \geq 1 \text{ and } u \neq s \\ 0 & L = 0 \text{ and } u = s \\ -\infty & \text{otherwise.} \end{cases} \tag{11}$$

********** Let $v_1, \ldots, v_n$ be an ordering of vertices in V by hop distance from s (for example output of a breadth-first search).

Algorithm 1 "Michael's DP Heuristic"

Initialize $\mathcal{Z}_{0,s} = 0$ and $\mathcal{Z}_{L,s} = -\infty$ for $L > 0$.
Initialize $\mathcal{Z}_{L,v} = -\infty$ for all $L = 0, 1, \ldots, L_{ub}$ and $v \in V \setminus \{s\}$.
Initialize $\Phi_{L,v} = \emptyset$ for all $L = 1, \ldots, L_{ub}$ and $v \in V$.
for $L = 1, \ldots, L_{ub}$ **do**
  for $v = v_1, \ldots, v_n$ **do**
    $u^* \leftarrow \text{argmax}_{u\in\delta^+(v): v\notin\Phi_{L-k_{uv},u}} \{\mathcal{Z}_{u,L-k_{uv}} + c_{uv} \cdot \alpha^L\}$
    $\mathcal{Z}_{L,v} \leftarrow \mathcal{Z}_{u,L-k_{u^*v}} + c_{u^*v} \cdot \alpha^L$
    $\Phi_{L,v} \leftarrow \Phi_{L,v} \cup \{u^*\}$
  end for
end for

****

Our second approach considers the flow formulation of DP (**??**) and prevents cycles through appropriate linear constraints. Let us define $K = \{n \min_{e\in E} k_e, \ldots, n \max_{e\in E} k_e\}$ and define G^{DP} as the extended acyclic graph coming from the dynamic programming algorithm. The set of all paths in $G^{DP} = (V^{DP}, E^{DP})$ corresponds to the extreme points of the following polytope

should define more clearly

$$\Pi^{DP} \equiv \left\{ x \in [0,1]^{m\times|L|} \;\middle|\; \begin{array}{c} \sum_{(v,u)\in E} x_{vu}^{k+k_e} = \sum_{(u,v)\in E} x_{uv}^k, \quad v\in V\setminus\{s,t\}, k\in K \\ \sum_{(s,v)\in E} x_{sv}^{k_e} = 1, \quad \sum_{(u,t)\in E, k\in K} x_{ut}^k = 1 \end{array} \right\}.$$

While any path $\pi^{DP} \subset E^{DP}$ is elementary, its projection $\pi \subset E$ might contain a cycle. We

should define

forbid such paths π^{DP} by limiting to 1 the number of outgoing arcs from any $u \in V \setminus \{s\}$, see constraints (**??**) in the formulation below:

$$\text{max} \quad \sum_{(u,v)\in E} \sum_{k\in K} \alpha^k c_{uv} x^k_{uv} \tag{12a}$$

$$\text{subject to:} \quad \sum_{\substack{v\in V\setminus\{s\}:\\ (u,v)\in E}} \sum_{k\in K} x^k_{uv} \leq 1 \qquad u \in V \setminus \{t\} \tag{12b}$$

$$x \in \Pi^{DP} \qquad (u,v) \in E \tag{12c}$$

$$x^k_{uv} \in \{0,1\} \qquad (u,v) \in E, k \in K \tag{12d}$$

Both (**??**) and (**??**) are heuristic algorithms without performance guarantee. However, if α is chosen close enough to one, the worst-case performance of the optimal solution of (**??**) can be bounded from the theoretical viewpoint as established in the following proposition.

Proposition 5. *For $\alpha \leq (1-\epsilon)^{1/n}$, then each optimal solution to* (**??**) *is an $(1-\epsilon)$-approximate solution to* (**??**) *and* (**??**).

4.1 Label setting algorithm

Each path π from the source s to node u corresponds to a label $\ell = (u, z, \sigma, \pi) \in L[k]$, where $L[k]$ is a set that contains the labels as logarithms of the probability products that equal k, z denotes the path objective value and $\sigma \in \{0,1\}^n$ is a binary indicator vector of the intersection of some subset $S \subseteq V$ that should be repeated and the vertices along the path π. In the following let $V(\sigma) \subseteq V$ denote the vertex set corresponding to indictor vector σ.

At each iteration, the label-setting algorithm takes a label from $L(k)$, where k is the smallest value for which $L(k)$ is non-empty, sets it as marked, and extends it through all successors of u in the graph.

The newly created labels are then compared to the existing labels at these nodes. Specifically, new label ℓ at node u is kept only if is undominated by existing labels at that node, $\ell' = (u, z', \sigma', \pi') \in L(k')$, where ℓ' dominates ℓ if $z' \geq z$, $k' \geq k$, and $V(\sigma') \subseteq V(\sigma)$.

In [**?**], they suggest do add the resources corresponding to S dynamically to the labels, starting with $S = \emptyset$ and gradually growing S over the algorithm iterations. Their results indicate the clear superiority of Highest multiplicity on the optimal path (HMO), which is the procedure we implemented.

In addition to the above steps, we also introduce a new step that filters labels by optimality. Specifically, let us define:

- LB: be a lower bound on the optimal solution (typically obtained by the previous heuristic),
- $UB(u, q)$: an upper bound on the optimal path from u to t using no more than q hops. That value is obtained by relaxing the elementary condition from our problem **(note: maybe something stronger could be done here, e.g. ng-path [?] or removing 2-cycles [?].)**.

Then, label ℓ' is maintained only if

$$z' + \alpha^{k'} \times UB(u, n - |\pi'|) > LB.$$

I also tried pre-processing. What about simple preprocessing - removing nodes that are not reachable from s or t is not reachable from them as well as 1-degree nodes. The following is actually stronger and leads to no removal at all. Let $Z(u, v, q, k)$ be the maximum profit from u to v using q hops and a probability log of k. Then node u can be removed if

k is fixed or not in the above? k'? In the code there is α^{k-1}... k is not fixed, there is a maximum, above formula removed to keep only yours (with brackets)

$$\max_{q,k} \left\{ Z(s, u, q, k) + \alpha^k \times \max_{q' \leq n-q, k'} Z(u, t, q', k') \right\} \leq LB$$

The results indicate no node can be removed.

5 Computational Results

5.1 Kidney Exchange Graph Experiments

In this subsection we experiment with kidney compatibility graphs that are randomly generated following the structure and data given in kidney exchange literature:

Graph topology simulated according to [?] (graph[n]a[A] datasets). We generate a "sparse-dense" graph with the node connectivity generated according to the panel reactive antibodies (PRA) levels as suggested in [**?**], and we make use of additional data given in [**?**, **?**]. In particular we generate graph vertices with a highly sensitized recipient with probability 0.27 these recipients are connected with donors (corresponding to an incident incoming edge) with probability 0.03. The failure probabilities are assigned also based on PRA levels and highly sensitized patients are matched with a success probability of 0.5. Otherwise, low PRA recipients are compatible with donors with probability 0.5, 57% of them are matched with a success probability of 0.75 (cross-match failure rate of 0.25) and 43% of them are matched with a success probability of 0.95 (cross-match failure rate of 0.05). The values of matching PRA recipients are generated from a normal distribution with mean 10 and standard deviation 2. The values of matching highly sensitized recipients are a factor of 1.5 greater (so the mean is 15) as suggested in [**?**].

Graphs based on simulated donor data instances of [?] and [?] (MD[n] datasets). These instances were generated using a simulator based on [**?**]. Graph arcs were constructed based on blood-type compatibility and we set the weights to be either unitary in datasets MD[n]unitval or normally distributed with mean 10 and standard deviation 2 in datasets MD[n]-stochval. For patients that are considered highly sensitized with PRA levels exceeding 0.73 in the original data, the arc values considered were twice as much as initially generated. The success probabilities are set to 0.5 for highly sensitized patients (determined by the PRA levels). Otherwise, a success probability of 0.75 is set with probability 0.57 and success probability of 0.95 is set with probability 0.43 (similar to the graph[n]a[A] datasets).

5.2 Implementation details

All algorithms have been coded in Julia 1.3.1 on a platform using a (Intel i7-10510U) 1.80GHz

CPU with 4 cores and an 8MB cache and 16GB of RAM memory. The mathematical programs are modeled using the JuMP v0.21.1 [?] and Gurobi v0.7.6 packages and solved using with the Gurobi 9.0.1 solver running 4 threads. All graph algorithms rely on the package LightGraph v1.3.1 [?]. A time limit of 1800 seconds has been set for each instance.

Evidently, none of our instances contain no edge with a probability equal to 1. Although, if an instance has multiple altruistic donors, we then connect supersources and supersinks to all nodes with edges having probability 1, but this may not lead to cycles c such that $\prod_{f \in c} p_f = 1$. Hence, the condition of Proposition ?? holds so inequalities (??) are enough to guarantee the existence of optimal acyclic solutions. This being said, inequalities (??) are not dominated by inequalities (??), so the separation of the latter through callbacks might possibly improved the overall performance of the formulation.

Following [?], we thus tested the separation of these constraints at both fractional and integer solutions by checking whether they are satisfied on each strongly connected component of the graph induced by the positive components of x. Our results indicated a worsening of the overall performance: the relaxation improvement, often nonexistent, is too little to compensate for the time spent in the separation procedure. Therefore, we do not use inequalities (??) in the results presented next.

5.3 Description of the results

We report solution times in seconds on Tables ?? and ??, where T stands for a time limit hit while M stands for a memory hit. Several observations can be made from these tables. First, heuristic DP (??) runs very quickly, solving most instances within less than a second. The extended formulation (??) scales reasonably well, as long as its execution does not exceeds the memory available. For some of the large MD instances, namely 84unitval, 84stochval, 125unitval, it converges faster than formulation (??). Third, we see that the exact formulation (??) behaves much better with instances MD than with instances graph. To further investigate this behavior, we present on Table ?? some of the instances characteristics, namely, their number of nodes, number of edges, number of altruistic donors, the longest path from the supersource to any node in the graph, and the value of the root relaxation expressed in %. These tables show that graph30a2 and instances MD43 – MD70 have comparable numbers of nodes and edges. However, graph30a2 cannot be solved in 1800 seconds, while these MD instances are all solved within a few seconds, except instance 66unitval which requires roughly 72 seconds. A partial explanation of this behavior could lie in the strongest linear programming relaxation for these MD instances, ranging from 6 to 20% while the one of graph30a1 is equal to 31%. These stronger relaxations are probably linked to the value of the longest paths in these graphs (directly impacting M_{uv}): these paths range from 17 to 22 hops, contrasting with the 30 hops of the longest path in graph30a1.

Next, Tables ?? and ?? provide the objective values of the two heuristics, relatively to the best solution found by the exact formulation (??) and expressed in %. All instances solved exactly by the exact formulation (??) can only lead to negative values. We see that while using α smaller than 0.8 leads to arbitrarly bad solutions, increasing its value above 0.8 yields only marginal improvements. What is more, on most instances the two heuristic algorithms provide solutions having roughly the same quality. Finally, we remark that for the instance 125unitval that cannot be solved to optimaity, the heuristic formulation (??) even provides slightly better solutions than the best one returned by the exact formulation.

does it? or is it %?

name	$\|V\|$	$\|E\|$	|sources|	LP	rootgap (%)
graph20a1	22	178	1	20	25
graph30a2	32	329	2	30	31
graph40a2	42	681	2	41	30
graph50a3	52	981	3	51	23
graph60a3	62	1387	3	61	17
graph70a4	72	1851	4	71	13
graph80a4	82	2247	4	81	10
MD12stochval	18	96	1	9	9
MD12unitval	18	96	1	9	10
MD19stochval	18	150	1	14	7
MD19unitval	18	150	1	14	4
MD21stochval	18	114	2	10	7
MD21unitval	18	114	2	10	1
MD23stochval	18	121	2	13	17
MD23unitval	18	121	2	13	28
MD43stochval	34	402	1	19	14
MD43unitval	34	402	1	19	13
MD44stochval	34	337	1	17	15
MD44unitval	34	337	1	17	11
MD51stochval	34	405	3	19	6
MD51unitval	34	405	3	19	16
MD60stochval	34	371	3	19	10
MD60unitval	34	371	3	19	9
MD61stochval	34	422	4	19	16
MD61unitval	34	422	4	19	17
MD66stochval	34	403	4	22	20
MD66unitval	34	403	4	22	16
MD70stochval	34	332	4	18	13
MD70unitval	34	332	4	18	18
MD81stochval	66	2205	3	55	15
MD81unitval	66	2205	3	55	20
MD84stochval	66	1512	3	36	29
MD84unitval	66	1512	3	36	27

Table 1: Instances main characteristics.

	Dynamic Programming (??)					Extended Formulation (??)					Formulation (??)
instance/α	0.75	0.8	0.85	0.9	0.95	0.75	0.8	0.85	0.9	0.95	
20a1	0.3	0.00	0.00	0.00	0.01	3.4	5.6	7.0	10	16	11
30a2	0.01	0.01	0.01	0.01	0.01	14	32	40	79	25	T
40a2	0.01	0.01	0.01	0.02	0.03	119	345	635	115	608	T
50a3	0.02	0.02	0.02	0.03	0.04	1198	128	407	590	114	T
60a3	0.02	0.03	0.03	0.04	0.07	233	386	534	1456	216	T
70a4	0.04	0.04	0.05	0.05	0.10	T	1521	1784	729	662	T
80a4	0.04	0.05	0.06	0.07	0.14	T	T	T	492	T	T

Table 2: Solution times in seconds for instances graph.

A FPTAS for directed acyclic graphs

TO DO: rewrite

In the special case that G is a DAG we develop an approximation scheme that is described by Algorithm **??**. For each $i \in V$, let $Z'(P, u)$ be defined as $\bar{Z}(P, u)$ with p' in place of p, and define z' similarly. The following proposition establishes that Algorithm **??** is a fully polynomial-time

Algorithm 2 Approximate Partial Failure Path

Input: G, p, c

1: For each $e \in E$ let $p'_e = (1-\epsilon)^{\lceil \log_{(1-\epsilon)^{1/n}} p_e \rceil / n}$.

2: $P' \leftarrow (1-\epsilon)^{\lceil \log_{(1-\epsilon)^{1/n}} P \rceil / n}$

3: $z^* \leftarrow Z'(P', t)$.

Output: z^*

approximation scheme (FPTAS).

Proposition 6. *Algorithm* **??** *is an FPTAS for problem* (**??**).

Proof. Consider π that is the output of Algorithm **??** and π^* that is optimal to (**??**). By the definition of p', the maximum number of arcs that form an elementary path and the optimality of π and π^* for their respective problems, it follows that

$$(1-\epsilon)z(\pi^*) \leq z'(\pi^*) \leq z'(\pi) \leq z(\pi).$$

Letting $p_{\min} = \min_{e \in E} p_e$, the complexity of the algorithm is $O(n^2 \sum_{f \in E} \log_{(1-\epsilon)^{1/n}} p_f) \subseteq O\left(n^4 \frac{\ln(p_{\min})}{\ln(1-\epsilon)}\right)$, which is polynomial in n and ϵ (recalling that for positive x, $\ln x \leq x - 1$). □

B Worst-case performance guarantee

Merge the two appendices?

Actually I thought it fits best in the paper if at all also I don't understand the previous as an appendix to be honest

Add a proof.

I think that it is obvious

	Dynamic Programming (**??**)					Extended Formulation (**??**)					Formulation (**??**)
instance/α	0.75	0.8	0.85	0.9	0.95	0.75	0.8	0.85	0.9	0.95	
12unitval	0.2	0.01	0.01	0.01	0.01	1.5	0.9	1.1	1.0	4.0	1.12
12stochval	0.01	0.01	0.01	0.01	0.01	0.6	1.0	0.9	1.5	4.0	0.03
19unitval	0.01	0.01	0.01	0.01	0.01	0.6	1.1	1.1	1.9	6.3	0.17
19stochval	0.01	0.01	0.01	0.01	0.01	0.5	1.1	1.4	3.0	17.8	0.06
21unitval	0.01	0.01	0.01	0.01	0.01	0.5	0.9	1.3	1.7	5.6	0.05
21stochval	0.01	0.01	0.01	0.01	0.01	0.6	0.9	1.0	1.5	4.1	0.01
23unitval	0.01	0.01	0.01	0.01	0.01	0.8	1.1	1.2	1.6	2.5	0.49
23stochval	0.01	0.01	0.01	0.01	0.01	0.6	1.2	1.0	1.2	2.5	0.38
43unitval	0.01	0.01	0.01	0.01	0.01	8	28	43	57	37	1.71
43stochval	0.01	0.01	0.01	0.01	0.01	10	27	37	54	53	2.07
44unitval	0.01	0.01	0.01	0.01	0.01	4	10	13	17	25	2.59
44stochval	0.01	0.01	0.01	0.01	0.01	4	11	13	17	13	1.19
51unitval	0.01	0.01	0.02	0.01	0.02	15	38	49	107	46	1.74
51stochval	0.01	0.01	0.01	0.01	0.02	11	39	57	114	73	0.75
60unitval	0.01	0.01	0.01	0.01	0.01	8	26	34	81	55	2.08
60stochval	0.02	0.01	0.01	0.01	0.01	10	21	31	59	70	0.59
61unitval	0.01	0.01	0.01	0.01	0.01	15	33	36	72	54	1.37
61stochval	0.01	0.01	0.01	0.01	0.02	11	29	35	74	44	1.65
66unitval	0.01	0.01	0.01	0.01	0.01	12	30	38	72	39	72.85
66stochval	0.01	0.01	0.01	0.01	0.01	18	64	121	170	62	2.12
70unitval	0.01	0.01	0.01	0.01	0.01	7	19	28	60	27	6.73
70stochval	0.01	0.01	0.01	0.01	0.01	6	21	37	58	45	1.34
81unitval	0.03	0.04	0.06	0.06	0.14	564	731	210	207	557	T
81stochval	0.03	0.04	0.6	0.06	0.13	T	T	T	T	T	T
84unitval	0.02	0.03	0.03	0.01	0.08	367	1469	1524	T	T	T
84stochval	0.9	0.02	0.03	0.04	0.09	332	603	715	521	T	T
125unitval	0.6	0.2	0.2	0.3	0.7	409	1200	1343	M	M	T
125stochval	0.1	0.3	0.2	0.3	0.5	M	M	M	M	M	T
203unitval	7.1	6.7	8.0	12.1	25.1	M	M	M	M	M	T
203stochval	5.6	6.8	8.0	12.0	24.8	M	M	M	M	M	T
215unitval	6.2	7.6	8.5	13.1	27.0	M	M	M	M	M	T
215stochval	6.3	7.7	8.6	12.3	26.2	M	M	M	M	M	T

Table 3: Solution times in seconds for instances MD.

	Dynamic Programming (**??**)					Extended Formulation (**??**)				
instance/α	0.75	0.8	0.85	0.9	0.95	0.75	0.8	0.85	0.9	0.95
20a1	-31	-6	-3	-3	-3	-31	-4	-4	-1	0
30a2	-36	-3	-3	-3	-3	-44	-3	-1	0	0
40a2	-55	-1	-1	-1	-1	-55	-1	0	1	1
50a3	-50	0	0	0	0	-49	0	1	1	1
60a3	-71	0	1	0	0	-71	0	1	1	1
70a4	-71	0	0	1	0	-72	0	0	1	1
80a4	-73	-1	-1	-1	0	-74	-1	-20	0	–
Average	-55.2	-1.7	-1.0	-0.9	-0.9	-56.6	-1.3	-3.3	0.4	0.6

Table 4: Heuristic method solution quality relative to the best solution available [??].

	Dynamic Programming (**??**)					Extended Formulation (**??**)				
instance/α	0.75	0.8	0.85	0.9	0.95	0.75	0.8	0.85	0.9	0.95
12unitval	-9	0	0	0	0	-9	0	0	0	0
12stochval	-7	0	0	0	0	-7	0	0	0	0
19unitval	-39	0	0	0	0	-27	0	0	0	0
19stochval	-26	0	0	0	0	-26	0	0	0	0
21unitval	-16	0	-1	0	0	-1	0	-1	0	0
21stochval	-24	0	0	0	0	-24	0	0	0	0
23unitval	-34	-1	-1	-1	-1	-46	0	0	0	0
23stochval	-37	-17	-3	-3	-3	-37	-2	-3	-2	0
43unitval	-62	-10	-10	-10	-10	-43	0	0	0	0
43stochval	-36	-14	-14	-10	-6	-44	-10	0	0	0
44unitval	-51	-3	-3	-3	-3	-62	0	0	0	0
44stochval	-52	-1	-1	-1	0	-52	-1	-1	-1	0
51unitval	-51	-2	-2	-2	-2	-37	0	0	0	0
51stochval	-30	0	0	0	0	-30	0	0	0	0
60unitval	-48	0	0	0	0	-48	0	0	0	0
60stochval	-30	-2	-2	-2	0	-34	-2	-2	0	0
61unitval	-61	-5	-5	-5	-5	-54	0	0	0	0
61stochval	-46	0	0	0	0	-42	0	0	0	0
66unitval	-55	0	0	0	0	-24	0	0	0	0
66stochval	-57	-7	-7	-3	-3	-59	-3	-3	0	0
70unitval	-41	0	0	0	0	-58	0	0	0	0
70stochval	-50	-3	-3	-3	-3	-50	-3	-2	-2	0
81unitval	-54	-2	-2	-2	-2	-77	0	0	0	0
81stochval	-60	-4	-3	-3	-3	-63	–	-27	-8	-1
84unitval	-65	-5	-5	-5	-5	-54	-1	0	-1	-1
84stochval	-49	-7	-5	-4	-4	-49	-7	-5	-1	0
125unitval	-82	0	0	0	0	-64	1	1	–	–
125stochval	-75	-1	-1	0	0	–	–	–	–	–
203unitval	-79	2	2	2	2	–	–	–	–	–
203stochval	-70	1	1	1	1	–	–	–	–	–
215unitval	-79	5	5	5	5	–	–	–	–	–
215stochval	-73	1	1	1	1	–	–	–	–	–
Average	-48.4	-2.3	-1.9	-1.5	-1.3	-41.4	-1.1	-1.7	-0.5	-0.1

Table 5: Heuristics solutions relatively to the best exact solution for instances MD.