Single-Commodity Stochastic Network Design with Multiple Sources and Sinks

Biju K. Thapalia and Crainic, Teodor G. and Michal Kaut and Wallace, Stein W.

in: INFOR: Information Systems and Operational Research. See also BibTeX entry below.

BibTeX:

```
@article{ThapWalKaut09b,
  author = {Biju K. Thapalia and Crainic, Teodor G. and Michal Kaut and Wallace, Stein W.},
  title = {Single-Commodity Stochastic Network Design with Multiple Sources and Sinks},
  journal = {INFOR: Information Systems and Operational Research},
  volume = {49},
  year = {2012},
  number = {3},
  pages = {193--211},
  doi = {10.3138/infor.49.3.193}
}
```


Document created on August 2, 2013.
Cover page generated with CoverPage.sty.

Single-Commodity Stochastic Network Design with Multiple Sources and Sinks

Biju K. Thapalia* Teodor Gabriel Crainic† Michal Kaut‡
Stein W. Wallace§

30 November 2009

Abstract

This paper examines the single-commodity stochastic network design problem with multiple sources and sinks. We characterize the structures of the optimal designs and compare with the deterministic counterparts. We do this primarily to understand what constitutes good robust network designs, but hope that the understanding can also be used to develop better heuristic algorithms than those available today.

Keywords: single-commodity network design, multiple sources and sinks, stochastic, correlations, robustness

1 Introduction

Many operations-research (OR) applications, as well as problems in computer science, applied mathematics, and many fields of engineering and management are based on network formulations with an underlying design problem, see for example Ahuja et al. (1995). Today's complex supply chains require goods and information to be distributed in many layers and in integrated ways. Increased competition force decision-makers to study the whole supply chain, all the way from suppliers to end consumers, trying to achieve overall optimality.

Network design has been a major area of research for the last four or five decades and shows great diversity in methodology, see for example Scheibe and Ragsdale (2009). But still we know very little about the structural characteristics of the optimal designs. We are interested in revealing structural properties of the designs that can be used to understand / evaluate designs even without solving the corresponding design problems.

It is evident that in most cases, at the time when a network is designed (or expanded), the demand or supply that it will later face is uncertain. Traditionally this is not taken into account during the design phase, but rather, the handling of uncertainty is postponed to the operational phase of the problem at hand. While it is true that the actual handling of uncertainty – meaning the reaction to revealed information – by definition, must take place when it occurs, it is equally clear that different designs offer different opportunities for *how* the uncertainty is handled, in particular, how costly the handling might be. This principle is

*Molde University College, biju.k.thapalia@himolde.no
†University of Quebec at Montreal
‡Norwegian University of Science and Technology
§Lancaster University Management School

well explained in for example Yen and Birge (2006). A discussion may also be found in Ball et al. (2007). So apart from understanding designs in general we are particularly interested in understanding how designs stemming from assuming deterministic demands differ from designs where uncertainty is included already in the design phase of a project. We ask: Does it matter? Are there recognizable differences between the two designs? Technically speaking, we shall compare, in different ways, designs coming from two-stage stochastic programs (where the design is stage 1 and the commodity flows stage 2) and their deterministic counterparts (where random demand is replaced by expected demand).

A common way to handle this situation is to perform single- or multi-parameter sensitivity analysis in order to understand how the optimal solution changes as a function of demand. This approach might seem appropriate, but in fact it is not. This is outlined in detail in Wallace (2000) and Higle and Wallace (2003). Logically, when performing sensitivity analysis, one is assuming that the design can be postponed until after demand has become known. So, whether sensitivity analysis is performed or not, we end up with a solution not created to handle uncertainty, and hence, we may have to face difficult operational decisions when demand is revealed.

It is old news that a deterministic solution might perform very badly in a stochastic environment. The reason is simply that it is not made to handle variation in parameters such as price or demand in a good way. This argument often follows the logic of "The value of the stochastic solution", see Birge (1982). In the network design case, good designs stem from flexibility in the commodity flows, i.e., the ability to utilize installed capacity across very different demand realizations. We have illustrated this in Thapalia et al. (2012): the deterministic solution is itself badly suited to handle stochastic demand for the single commodity, single source, multiple sink network design problem. However, in the same paper we also observed that the structure (i.e., which edges to open) might be similar in the deterministic and stochastic cases, albeit with rather different capacities installed. Even this kind of similarity is unusual.

Lium et al. (2009) found consolidation to be a way to hedge against uncertain demand in their multi-commodity stochastic service network design model. This cannot (of course) be observed in the solution to the corresponding deterministic model, as the model has no reason to hedge against uncertainty. The deterministic design might contain volume-related consolidation, but that is not enough to cater properly for uncertainty. So in that case, not only is the expected behavior of the deterministic design bad, but also the structure (i.e. information about which edges to open) is of limited value. A question for this paper is therefore: As we pass to the case of multiple sources for the single-commodity case, shall we observe that the structure of the deterministic solution is good (as we observed in the single-source single-commodity case) or bad as in the multi-commodity case?

Hence, while we focus on comparing stochastic and deterministic designs, it is not primarily to (once again) show the weaknesses of the deterministic solution, but to really understand in what ways (if any) the deterministic solution is good and in what ways it is bad. We also hope that this can be used not only to obtain a deeper understanding of the effects of uncertainty on design, but also to develop heuristics for the stochastic case.

2 Problem description

Given a set of nodes (divided into source nodes, demand nodes, and transshipment nodes) and a set of potential edges connecting these nodes, the single-commodity stochastic network design problem with multiple sources and sinks (MSSND) is the problem of determining a subset of the edges to open (including the edges' capacities), so as to fulfill the demand at the demand nodes at minimal cost, taking into account capacities of the source nodes.

In general, the stochastics in this problem arises in the form of demand uncertainties at the demand nodes, supply uncertainties at the source nodes, and failure of connections (or failure of certain proportion of capacities in the edges) between the nodes. Demand uncertainties and edge failures are observed in most real life problems, as it is rare that demand is fully known when the design is determined or that edges never fail. In this paper, we discuss only random demand. The design is based on minimizing the sum of the fixed costs of selecting edges connecting the nodes; linear costs to open capacities in the edges; per unit flow costs of flows on the edges; and per unit penalty costs for not satisfying demand. Not satisfying demand can have many interpretations, such as sending the flow at a later point in time, with another mode, or a straightforward rejection. In any case, in the model, it takes the form of a penalty cost per unit of unsatisfied demand.

It is important to include the possibility of flow being rejected in the model. The main reason is that in real life, except in extremely particular situations, it is prohibitively costly to build a network that can meet any possible demand—however unlikely it might be. Deterministic models, operating on expected demand, may reasonably operate under the assumption that (average) demand *must* be met. But even there, there will normally be an understanding that some demand may end up being turned down in reality. When working with stochastic demand, there is also the problem that requiring demand to be met turns the model into a worst-case model, where the worst-case in most cases is not even well understood. So, in total, we find it crucial to include the possibility of not satisfying all the demand. We use the same formulation also in the deterministic models, to make the results comparable.

We shall let all source nodes have the same capacity so that our focus will be on the random demand. Hence, our assumption is that a set of demand nodes will have their random demands satisfied from a set of equally-sized source nodes.

In the deterministic case, the demand in each node is fixed at the expected demand from the stochastic case. This corresponds to the classical case of a single-commodity multiple source network design problem. The first stage decisions in this problem are to decide which edges to open and what capacities to install. The second stage decisions are the flow decisions in the given network. The recourse action here is described by a penalty cost incurred for not satisfying demand.

2.1 Mathematical formulation

Let $G = (\mathcal{N}, \mathcal{E})$ be a network defined by a set $\mathcal{N}$ of n nodes and set $\mathcal{E}$ of m edges (undirected arcs), where

$$\mathcal{E} \subset \{k = (i, j) : i \in \mathcal{N}, j \in \mathcal{N} \text{ and } i < j\}.$$

Each edge is indexed either by i, j or by k.

The random demand is described by a set of scenarios $\mathcal{S}$, where each individual scenario $s \in \mathcal{S}$ has one demand realization for each demand node. We shall discuss in Section 3.1 how

the scenarios were generated. The notations for the sets, parameters, and variables associated with this problem are as follows:

Sets:

$\mathcal{C}$	set of all source nodes;
$\mathcal{D}$	set of all demand nodes;
$\mathcal{T}$	set of all nodes with zero demand (transshipment nodes); $\mathcal{T} = \mathcal{N} \setminus (\mathcal{C} \bigcup \mathcal{D})$;
$\mathcal{S}$	set of all scenarios s.

Parameters:

M	maximal arc capacity; used for linking capacities and open arcs in (5);
R	unit cost of unsatisfied demand;
P^s	probability of scenario $s \in \mathcal{S}$;
C_k	flow cost on edge $k \in \mathcal{E}$;
G_k	fixed setup cost for edge $k \in \mathcal{E}$;
H_k	variable setup cost; the cost for adding one unit of capacity to edge $k \in \mathcal{E}$;
V_k	initial/ existing capacity on edge $k \in \mathcal{E}$, if any;
D_i^s	demand ($D_i^s < 0$) in node $i \in \mathcal{D}$ in scenario $s \in \mathcal{S}$;
D	supply in each source node, $D > 0$.

Variables:

$x_k^s = x_{ij}^s$	flow on edge $k = (i,j) \in \mathcal{E}$ going in direction $i \to j$, in scenario $s \in \mathcal{S}$;
$z_k^s = z_{ij}^s$	flow on edge $k = (i,j) \in \mathcal{E}$ going in direction $j \to i$, in scenario $s \in \mathcal{S}$;
u_k	new capacity that is developed on edge $k \in \mathcal{E}$;
e_i^s	for $i \in \mathcal{D}$, this is the unsatisfied/lost demand in node i in scenario $s \in \mathcal{S}$; for $i \in \mathcal{C}$, this is the unused capacity of source node i in scenario $s \in \mathcal{S}$;
y_k	1 if edge $k \in \mathcal{E}$ is developed, 0 otherwise.

We assume that total supply from equally-sized source nodes equals maximal demand in the network, so that

$$D = \max_s \{\sum_{j \in \mathcal{D}} \{D_j^s\}\} / |\mathcal{C}| \tag{1}$$

where $|\mathcal{C}|$ is the number of source nodes.

Our overall problem is hence:

$$\min \sum_k G_k y_k + \sum_k H_k u_k + \sum_s P^s \left\{ \sum_k C_k \left(x_k^s + z_k^s \right) + R \sum_{i \in \mathcal{D}} e_i^s \right\} \tag{2}$$

Subject to:

$$\sum_{j:\,(ij) \in \mathcal{E}} \left(x_{ij}^s - z_{ij}^s \right) - \sum_{j:\,(ji) \in \mathcal{E}} \left(x_{ji}^s - z_{ji}^s \right) = \begin{cases} 0 & \forall i \in \mathcal{T}\,, \forall s \in \mathcal{S} \\ D - e_i^s & \forall i \in \mathcal{C}\,, \forall s \in \mathcal{S} \\ D_i^s + e_i^s & \forall i \in \mathcal{D}\,, \forall s \in \mathcal{S} \end{cases} \tag{3}$$

$$x_k^s + z_k^s \leq u_k + V_k \quad \forall k \in \mathcal{E} \quad \forall s \in \mathcal{S} \tag{4}$$

$$u_k \leq M y_k \qquad \forall k \tag{5}$$

$$0 \leq e_i^s \leq -D_i^s \qquad \forall i \in \mathcal{D}; \forall s \tag{6}$$

$$x_k^s, z_k^s, u_k, e_i^s \geq 0 \text{ and } y_k \in \{0,1\} \qquad \forall k; \forall i; \forall s \tag{7}$$

The objective function (2) minimizes the total costs of the network. The first part is the costs of constructing all new edges, the second part the costs of building all the new capacities, the third part the expected flow costs through all the edges and the fourth part is the expected penalty costs of not fulfilling demand. Constraints (3) model conservation of flow at nodes. The left-hand side is the net outflow from node i, which must be zero for all transshipment nodes $i \in \mathcal{T}$ and is equal to the unused capacity for source node $i \in \mathcal{C}$. For the demand nodes, the net outflow must be equal to the satisfied demand; since D_i^s is negative in this case, the right-hand side is the a difference between the scenario demand D_i^s and the (positive) unsatisfied demand e_i^s.

Constraints (4) represent the flow limit in each edge. The left hand side of the equation is the net flow on edge k which should be less then or equal to the total capacity of the edge. Since we do not start with any initial/existing capacity in our test cases, we always have $V_k = 0$. Note that in an optimal solution, an edge will never have flow in both directions. Constraints (5) show that new capacity u_k can be developed only if edge k is built. Constraints (6) give bounds for the rejection amount and finally, (7) insure that all variables are non-negative and the edge constructions binary.

For the deterministic counterpart we replace the stochastic demand by its expectation.

We model the problem in AMPL and solve it to optimality using CPLEX 9.0. The solution time varies from few seconds to 5 hours depending on the case, on a PC with 3 GHz Intel® CPU and 8 GB of RAM.

3 Experimentation and Computational Results

The tests have two related goals. First we primarily focus on the quality of the deterministic designs by trying to understand how they differ from their stochastic counterparts, and to what extent solving a deterministic problem will guide us toward a good design in a stochastic environment. Does the deterministic design contain useful information, or is it totally misleading if the real setting is that of stochastic demand? In the second part we try more directly to characterize good designs, assuming that the stochastic design model is the appropriate one. Our goal is to make qualitative statements about what characterizes of a good design in light of random demand. These two questions are of course related, but we find it useful to have these two focuses.

In order to answer these problems we have constructed a number of test cases. These are now described together with our scenario generation approach.

3.1 Test instance generation

We have used seven different network instances. The first four instances, namely Germany, Nobel-EU, NY, and US are telecommunication examples from the SNDlib library (Orlowski

et al., 2010), with some modification to suit our problem's needs. The fifth case was generated by us and named Molde and the last two, Montreal_r06.1 and Montreal_r10.1 were obtained from CIRRELT (Interuniversity Research Center on Enterprise Networks, Logistics and Transportation), Montreal. The names of the instances do not mean anything particular in our computational setup.

It is worth noting that in all cases from SNDlib and Montreal, the test instances are multi-commodity network design problems, so not all parameters can be used directly by us. We only kept the coordinates (where available) for the nodes and the fixed setup cost G_k for the edges. The values for the other parameters – variable setup costs H_k and flow costs C_k – are all chosen proportional to the Euclidean distance between the node pairs. The cost of unfulfilled demand R is derived for each test case using some multiple of the highest value of the fixed plus variable setup cost for an edge in the network. We made sure that R is not driving the solution. The results in the first part of Section 3.3 are based on test cases with these cost structures. In the second part of the section, we changed the fixed costs to understand their relative importance.

The Montreal test instance does not have node coordinates, so we used Graphviz (Gansner and North, 2000) to draw the graph using fixed setup cost as distance measure. The graphs of the test instances Nobel-EU, US and Molde are planar whereas the graphs of the test instances Germany, NY, Montreal_r06.1 and Montreal_r10.1 are non-planar.

For each of the seven problems, we picked 3 sets of nodes (2 in the case of Montreal_r06.1) as possible source node sets, thus creating in total 20 base test instances. These 20 different versions of the problem instances are presented in Table 1. A set of source nodes contains three or four nodes depending upon the test instances; the number of source nodes for each test instance is listed in the fifth column of Table 1.

Given the difficulty of solving the stochastic network design problem to optimality we kept n (the number of nodes) below 30 and m (the number of edges) below 50 for the first six cases, while for the Montreal_r10.1 cases we have up to 87 edges.

Table 1: The different test cases. Test case Molde is generated by us. For the others, the names have been kept, even though the cases are adjusted to our needs.

Problem name	# nodes	# edges	# demand nodes	# sources	# source sets
Germany	29	48	9	3	3
Nobel-EU	28	41	8	4	3
NY	16	41	7	4	3
US	26	42	9	3	3
Molde	22	45	8	4	3
Montreal_r06.1	10	37	5	3	2
Montreal_r10.1	20	87	6	4	3

We know from the work of Lium et al. (2007) that correlations might be important in shaping the structure of the network. Hence, we further create 3 cases for each problem instance: one with uncorrelated demands, one with positively correlated demands (all correlations are set to 0.7), and one with mixed correlated demands: the demand nodes are put into groups such that each group contains about half of the total number of nodes. All correlations *within* a group are set to 0.7, while *between* groups we use -0.7. All of this leads to positive definite correlation matrices. Thus we have in total 60 test cases.

As stochastic programs need discrete distributions to represent the stochastics, we discretized the chosen distributions (discussed below) by creating scenarios each having equal probabilities to occur using the moment-matching method from Høyland et al. (2003). In the absence of reference to a particular distribution representing the random demand we chose to use truncated normal distributions with mean equal to the deterministic demand (from the underlying cases) and standard deviation equal to 25% of the mean to represent its stochasticity.

The decision on the number of scenarios used to represent the stochastics is critical as we want to be sure we study the effects of randomness on our model, and not some random effect of the scenario generating procedure. There is a trade-off between the quality of scenarios representing the underlying distribution reasonably well and the time needed to solve the stochastic program to optimality. As we increase the number of scenarios, we increase the quality of the representation of the distribution, but also decrease the chance to solve the model to optimality within a manageable time. In our case, we generated 100 scenarios to represent the distributions as this gives us in-sample stability and manageable solution times. The in-sample stability is checked by solving the same problem repeatedly with different 100-scenario trees. This lead to a coefficient of variation (the standard deviation divided by the mean) of less than 1%, except for the cases of 'NY', 'Molde' and 'Nobel-EU' where it was 1.2%, 3.9%, and 4.8% respectively. The cases of 'Molde' and 'Nobel-EU' have very high rejection costs, so even a minor change in rejection volume results in a large change in objective function value.

With these values we are satisfied that we have in-sample stability for the problems at hand. Since this is a necessary, but not sufficient, property of a satisfactory scenario generation procedure, we also check out-of-sample stability. Out-of-sample stability is checked by creating scenario trees with 1000 scenarios using sampling and then evaluating the solutions over those scenarios. The evaluation is performed by re-optimizing the flow in the network with given designs i.e., fixing the first stage variables of problem (2) to (7), representing the network design under evaluation. The procedure is repeated 10 times for a given network design and the objective values are compared by again calculating the coefficient of variation. For our problem, out-of-sample stability was achieved as the coefficient of variation for all the test instances was less then 0.2%, except for the cases 'NY', 'Molde' and 'Nobel-EU' where it was 2.9%, 1.4%, and 2.6% respectively. For more discussion on this subject we refer to Kaut and Wallace (2007).

Finally, we have to reconsider the definition of the supply D given in (1): since it depends on the actual realizations of the stochastic demands D_j^s, it would be different for the three versions (with different correlations) we generate for each test case, making it difficult to compare the results. To avoid this problem, we calculated D for each of the three correlation versions of a given case and used the median of the three D's as our demand size in all three versions.

3.2 Comparison Tests

As outlined in the Introduction, the deterministic solution, by its nature, has a worse expected behavior than its stochastic counterpart. However, we would like to understand more about why this is the case, and in what sense it is worse.

In order to check the quality of the deterministic designs, as well as comparing them to the stochastic ones, we have set up three tests, named *comparisons*. Whenever a comparison

is performed, we take the deterministic and stochastic designs – or parts thereof – (i.e. the first-stage solutions) and evaluate them using reference trees – in our case trees with 1000 scenarios, to make sure we have good approximations of the true distributions. The costs from the design and evaluation phases are added up, making the reported costs comparable across all tests.

A word of warning might be worthwhile here. If a stochastic programming problem, as well as its deterministic counterpart, use hard constraints in the formulation, the deterministic solution will normally be infeasible in the stochastic formulation (caused by capacity problems when demand is high), and hence, its expected cost will be infinitely large. On the other hand, if soft constraints are used, the deterministic solution will normally be feasible in the stochastic model, but its expected performance can be made arbitrarily bad by choosing large penalties on the soft constraints. This way, it is always possible to make the deterministic solution look bad. We shall, however, set the penalties at reasonable levels, and our goal is to *understand* how the deterministic solutions relate to their stochastic counterparts. So, we shall certainly present numbers, and we do believe the numbers are informative. But there will never be really objective numerical results in this setting.

The three comparisons are:

A The classical test where the whole first-stage solution is evaluated out-of-sample. This amounts to solving a 1000-scenario stochastic program with all first-stage variables (designs and capacities) fixed, so in fact this equals the solution of 1000 independent second-stage problems. Since the second stage does not involve any integer variables, this is very fast.

B Only edge information is imported from the first stage. So, in a 1000-scenario stochastic program, all discrete variables y describing opened and closed edges—we call it a *skeleton*—are fixed and the stochastic program is run. So the model is allowed to install any capacity on the opened edges (also lower than in the deterministic case), but not to open new ones.

C The whole design (both the skeleton and its capacities) is taken as input to the 1000-scenario stochastic program. The stochastic program can then add new capacities on already opened edges (paying only variable setup costs) and new edges (paying both fixed and variable setup costs). Hence, all capacities opened in the deterministic case add cost to the objective function, even if these are not needed in the final design.

The purpose of Comparisons B and C is to check if the design from the deterministic solution really is good for the stochastic case, and if it bad, in what way it is bad. By making edges from the deterministic case "free" in two different ways, the stochastic programs (as defined in the comparisons) are guided toward the deterministic solution. This way we compare if stochastic programs solved with input from the deterministic solutions behave much worse than stochastic programs which have no deterministic input (they will never behave better).

So, Comparison A is the classical test of the quality of the deterministic solution. Comparison B, on the other hands, checks if we can use a deterministic method to determine the skeleton and then solve a stochastic *linear* program to set the capacities. If Comparison B comes out with good results for the deterministic solution, it points to an alternative solution procedure that avoids solving a stochastic mixed integer program: First use a deterministic

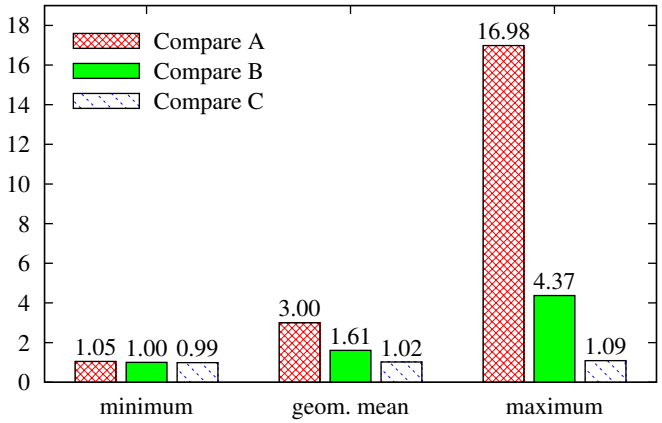

Figure 1: Results of the Comparison Tests. Ratios between the expected objective values, obtained from importing the deterministic solution into the stochastic setting, and the expected objective function value of the stochastic solution.

method to find the skeleton, then a stochastic linear program to set capacities. This represents a severe saving in computation (if it works well, of course).

Comparison C can be seen as testing what happens if we first solve the deterministic design problem and implement the solution, but then discover that it is not very good, and wish to update it. If Comparison C comes out well for the deterministic solution, a deterministic design can be corrected and become almost optimal for the stochastic case provided setup costs must not be paid again. If Comparison C comes out badly, the costs of updating a deterministic design in light of uncertainty in demand will be high. Note that Comparison C is itself a stochastic mixed integer program, so in most cases it does not represent an alternative solution approach. In our tests, though, Comparison-C with 1000 scenarios is faster than the original stochastic problem with 100 scenarios. But for large problems, both are unsolvable.

In what follows of this section, we discuss the major findings, details are given in the Appendix. Our first need is to understand the relationship between the stochastic and deterministic solutions. We therefore perform Comparisons A, B and C as discussed in Section 3.2. That is, for all our 20 deterministic cases, we solve the corresponding network design problem. Also, we solve all 60 stochastic cases, representing the stochastic versions of the deterministic cases (each with three different correlation structures).

Then each of the 20 deterministic designs are imported into its three stochastic counterparts (the three different correlation matrices). This is done for all three comparisons. In all cases, as outlined earlier, the evaluations are done out-of-sample using 1000 scenarios. Figure 1 shows the results, where also the stochastic designs are evaluated out-of-sample.

3.3 Inheritance from the deterministic solutions

The deterministic solution is bad in the stochastic environment, and inheriting the structure (the skeleton) is also not good, see Figure 1. For Comparison A, the deterministic solutions have expected objective function values which are from five up to almost 1600% higher than that of their stochastic counterparts with mean value of 200%. For Comparison B errors are from 0% up to little over 300% with mean value of 61%. On the other hand, for Comparison C

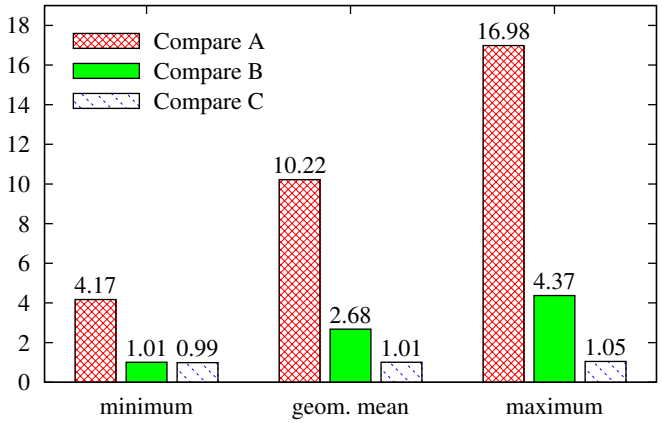

Figure 2: Results for cases with higher cost of first stage decision. Ratios between the expected objective value, obtained from importing the deterministic solution for the test cases with high fixed and variable setup cost into the stochastic setting, and the expected objective function value for the stochastic solution.

the errors are very low, from 0% up to just 9% with mean value of 2%. This shows that when we allow to add new edges and open new capacities, the deterministic design can be updated to become almost as good as the stochastic design. (Note that in these tests we might observe that the deterministic design is better than the one coming from a 100-scenario stochastic model, since both designs are evaluated out-of-sample with 1000 scenarios. We have observed one single case which can be seen in Figure 1.)

Now, if we look at Figure 2, which compares the cases where the costs of the first stage decisions (the fixed and/or variable setup costs) are highest (the test instances of Molde, Nobel-EU, and NY), then we see that the *best* design for Comparison A is 4.17 times more costly than its stochastic counterpart. And on average, the deterministic design produces expected costs that are 10.22 times what a stochastic model would produce. If we further look only at certain components of the overall costs for these cases, which are presented in Figure 3, we find that on average, the costs for opening edges are just 74%, and capacity built is just 62%, of that in the stochastic designs. This indicates that the solutions are far from the optimal structure both in terms of edges opened and capacities built.

So what do we see? Not very surprisingly we observe that the deterministic solution behaves rather badly in a stochastic environment, implying that the common practice of creating a design based on expected values and then handling randomness operationally is not a very good idea—the costs can be astronomical. One reason for this is simply that the deterministic design does not install enough capacity – even in our case where the penalty is set at a very reasonable level. So what if only the skeleton – the edges to be opened – is imported from the deterministic design, and the capacities are set as in Comparison B? This is computationally effective, as solving the stochastic program of Comparison B is very simple even for huge problems (it has no integer variables). It helps, of course, but we can still be several hundred percent off, which in most cases is not acceptable. There are situations where Comparison B does well, but it isn't easy to know upfront if a given case is of that type.

The results of Comparison C are worth an extra comment as they do not represent a very common situation: if the deterministic design is taken as a starting point, a very good (even if not optimal) design can be found by adding extra edges and capacities on top of the

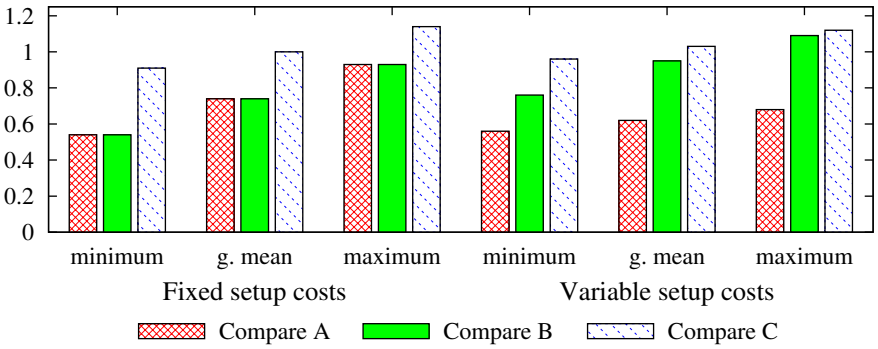

Figure 3: Components of result. Ratios of the fixed and variable setup costs obtained from importing the deterministic solution, for the test cases with high fixed and variable setup cost, into the stochastic setting divided by the fixed and variable setup costs for the stochastic solution.

deterministic one. In our test cases, we never lost more than 9% that way. This is still a large number in many cases, but given the uncertainty in the model (which of course is always there) this is not bad. Computationally, we must then solve a deterministic design problem first, and then a stochastic one (Comparison C). This stochastic program has substantially fewer integer variables than the original one, but can still be expected to be as unsolvable as the original one for practical problems. This is not the main point, though. The main observation is that the deterministic solution can be updated to become very good even in a stochastic environment.

It is worth mentioning that the results in Comparison C are not in line with the general observation that a stochastic solution is normally *not* the deterministic one "plus something"—see Wallace (2010) for reasons why. In our case that is exactly what we observe (genuinely or as a good approximation): the stochastic design equals the deterministic design plus "something".

For the single-source case, as described in Thapalia et al. (2012), Comparison B came out rather well contrary to what we observe here. This difference can mainly be attributed to the fact that we now have many source nodes with limited supply capacity. As we minimize costs, the deterministic skeletons have many short paths, we term them *arms*, connecting individual source nodes to nearby demand nodes. If not generally, this typically gives us a forest of small trees. Just adjusting the capacities of these trees is not enough to find good designs. When there is only one supply node, the deterministic skeleton is a tree, and hence, all nodes are connected, even if the connections are not optimal. When the skeleton is a forest, there are simply too few connections.

We know from Lium et al. (2007) that correlations may play important roles in shaping the solution structure of the stochastic problem in terms of sharing capacity and taking benefit from variation in demand. By importing the deterministic solution into the stochastic problem, the deterministic solution structure with short arms from each source node cannot benefit from this variation in demand. Hence we see poor performance in Comparison A. This is more evident in the cases where we have higher setup costs (fixed, variable, or both) as these result in deterministic designs with particularly short arms. Consider Figure 4, where we observe that it is worse in the uncorrelated and mixed correlated cases as compared to the positively correlated cases. This is natural since with strong positive correlations there is

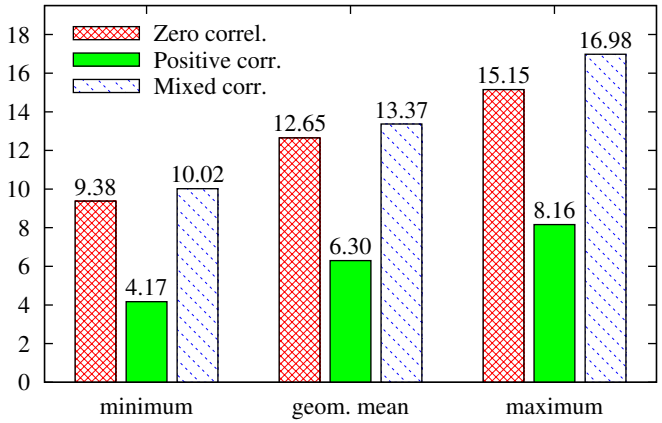

Figure 4: Results for Comparison A for the cases with higher cost of first stage decision. Ratios between the expected objective value, obtained from importing the deterministic solution into the stochastic setting, and the expected objective function value for the stochastic solution.

less to be gained from joint use of edges in any case.

Relation between the number of source nodes and inheritance

The extreme case—one source node—was covered in Thapalia et al. (2012). In that case both Comparisons B and C were rather good. We have already seen that for three or four source nodes - as in this paper - Comparison B is no longer very good, while Comparison C remains very strong. With high setup costs this is even more evident, as that will cause the skeleton to be as minimalistic as possible in terms of the number of edges. We wonder if also Comparison C will become weak as we get more source nodes.

Some of this we already understand: With one source node, the deterministic skeleton is a tree (not necessarily spanning because of the transshipment nodes), while as the number of source nodes increases, we tend to get several trees, in the extreme case, one for each source node, and the ability to share capacity when randomness hits becomes steadily lower. Comparisons A and B, limited by the deterministic skeleton, suffer from this lack of connectedness – as it prevents sharing of supply capacity – and hence they do not do very well.

In order to better understand the effect of the number of source nodes, we have increased the number of source nodes for a few cases. What we observe is that Comparison C gets steadily worse as the number of source nodes increases. However, be aware that it is not easy to define what "these two cases are the same except that one has more source nodes than the other" means. The reason is that as the number of source nodes increases, the whole network design problem changes, and comparisons become unclear. In this paper we are limited to cases we can solve to optimality. That prevents checking the fate of Comparison C for really large cases. Within what we could check, we found that Comparison C got worse as the number of source nodes increased, but remained very good throughout, the deterministic solution never being more than 10% worse than the stochastic one.

So, it seems, taking the deterministic design and adding edges and capacities produces good solutions. Note again, however, that since Comparison C is also a stochastic integer program, it it likely to be as unsolvable as the original stochastic program for large realistic cases.

Fixed costs

We want to make sure that the observations of how the stochastic solutions differ from the deterministic ones do not depend on the cost structures we have used. So we turn to testing these results when we vary the way setup costs are distributed between fixed setup cost G_k and variable setup cost H_k. Let L be some large positive number, selected conveniently. Here we take it to be 25% of M. For each of the 20 test cases, we calculate for each edge $C_k = G_k + LH_k$. Then we redistribute C_k in five different ways:

a Fixed setup cost 0.1% of C_k and variable setup cost 99.9% of C_k/L

b Fixed setup cost 5% of C_k and variable setup cost 95% of C_k/L

c Fixed setup cost 25% of C_k and variable setup cost 75% of C_k/L

d Fixed setup cost 50% of C_k and variable setup cost 50% of C_k/L

e Fixed setup cost 99.9% of C_k, and variable setup cost 0.1% of C_k/L.

All tests described earlier are now performed for each of these five cases and results are shown in Figure 5. The tests are performed with 100 scenarios. Some test instances which CPLEX could not solve within 15 days are ignored.

We find that the deterministic design is, as before, rather bad in the stochastic environment (with errors up to nearly 2700%), while Comparisons B and C are getting somewhat worse, in particular when the fixed setup cost is low and the capacity cost is high. This is natural since in that case the cost of opening one edge with a given capacity costs basically the same as opening two edges with the same total capacity. Hence, the structures enforced upon the solutions in Comparisons B and C become more costly. We see errors up to nearly 700% in the case of Comparison B, and up to 11% in the case of Comparison C—still quite good. The results seems little dependent on correlations. The details of the results are presented in the Appendix.

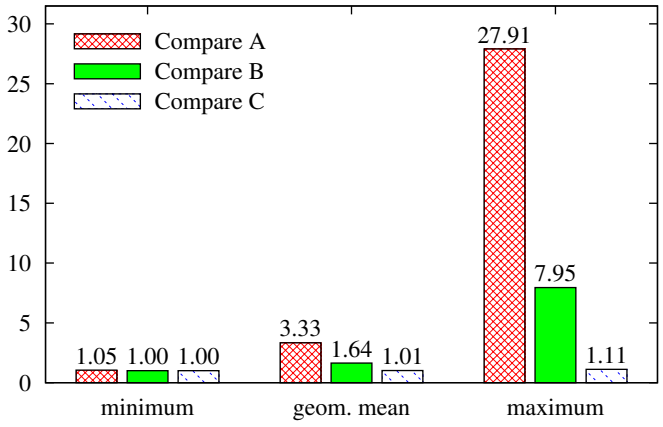

Figure 5: Results of the Comparison tests for different distributions of setup costs. Ratios between the expected objective value, obtained from importing the deterministic solution into the stochastic settings, and the expected objective function value for the stochastic solution.

3.4 Structural differences

So far we have discussed different ways to bring the deterministic design into the stochastic environment to understand to what extent the deterministic design is useful in creating good solutions. Now we shall pass to a more direct comparison of the the structures of the stochastic and deterministic designs, instead of just looking at their expected costs.

The optimal structures in single-commodity network design seem to be more complicated to *understand* than those of the corresponding multi-commodity cases. The main reason is that in the multi-commodity case the commodities only share edge capacities whereas in the single-commodity case there is also the phenomenon of flow cancellation. So while it is easier to *solve* single-commodity flow problems (as standard network flow theory can be applied directly), the optimal design is harder to characterize.

As already pointed out, the stochastic designs tend to have more capacity and more edges than the deterministic counterparts. For the cases of uncorrelated and mixed correlated demands, the extra edges and capacities are mainly there to cater for capacity sharing. This results in loop formations, leave connections, and connections between different clusters of nodes. The trees with few short arms, typical of the deterministic skeletons, will generally not allow sharing based on some demands being large when others are small simply because there are few demand nodes in each tree, and there is no particular reason why demand nodes with negatively correlated demand end up in the same tree.

In the cases of positive correlations, the edges and capacities are mostly there to cater for the high-demand scenarios. Two phenomena occur: The high demand scenarios (which now have high probabilities attached to them) need much more capacity than the deterministic (expected value) case, and the limited capacity of the individual source nodes makes it necessary to connect them so that all supply is used well. These connections are simply to few (if at all) in the deterministic designs. In other words, even though the demands are positively correlated, there is some variation, and connections are needed to utilize overall supply.

Let us now turn to a more direct study of the stochastic designs rather than primarily *comparing* stochastic and deterministic designs to understand the qualities of the deterministic ones. We know that good designs stem from flexibility in the routing of flow, and the goal is then to understand how this is achieved. We shall do this by studying the problems from Section 3.1. Some of the results are "obvious" meaning that good robust designs are, at least structurally, not so difficult to understand. We consider this a strength.

Similarity in designs

The deterministic skeletons are trees (not spanning trees) with arms emerging from the different source nodes. If the supply capacities are tight, the trees might be connected to each other so that the supply nodes can help each other satisfy demand in "their" demand nodes. These skeletons are contained within the stochastic designs under certain conditions. Remember that the trees from the deterministic designs represent the cheapest way to connect to the demand nodes in the average case. Hence, if the capacities of the source nodes are high enough to handle high demand scenarios, then we often see that the deterministic skeletons are part of (or even the same as) the stochastic skeletons. Figure 6 compares the deterministic design with two stochastic designs, one with high demand variation and one with low. (Solid edges (blue) are installed with the given capacities, light (grey) edges are not installed. The dark (yellow) square nodes are the supply nodes, the shaded (green) circular nodes are demand

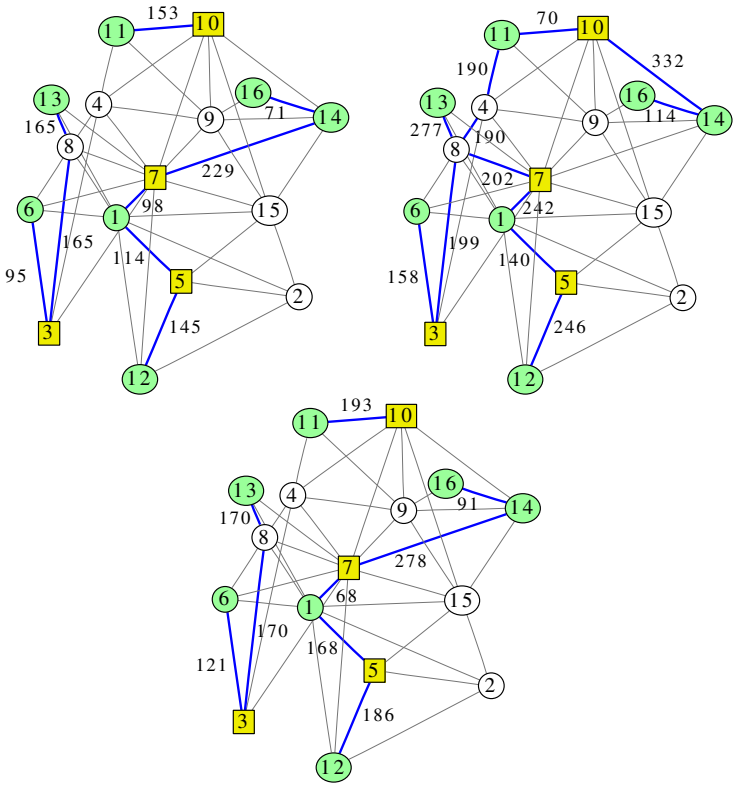

Figure 6: Comparison between deterministic and stochastic structure. Deterministic (top left), stochastic with high variance (top right) and low variance (bottom) solutions of the NY_02 test case showing that the stochastic solution structures are similar to the deterministic ones in the low variation demand case, but rather different in the case of high variation.

nodes, and the white circular ones transshipment nodes. This color scheme is followed in all subsequent figures.) The low variation case has the same skeleton as the deterministic design. This is because with lower variation the source nodes still have sufficient capacities to fulfill the demand in most scenarios (there is some unsatisfied demand). The higher variation case results in high demand scenarios, where some source nodes may have insufficient capacity to fulfill the demands of "their" demand nodes and hence we see a re-alignment of the distribution patterns to fulfill more demand than would be possible from the deterministic skeleton.

Also when setup costs are substantially lower than flow costs, the deterministic skeletons are contained in the stochastic skeletons. This is because the deterministic design represents the cheapest way to transport the bulk of the demand. Longer routes will result in substantially higher flow costs. On top of the deterministic solution, new edges needed to facilitate high demand scenarios and coordination of source nodes are cheap to add. In Figure 7 we see that both the mixed correlated and uncorrelated cases contain the deterministic skeleton, and add a few extra edges to meet higher demands. For example, the demand node 15 connects to source node 2 (for uncorrelated case) and node 15 & 3 get connected to source node 12 (for mixed correlated case) in the stochastic solution. This helps satisfying the higher demands. But the deterministic skeleton is fully used in the stochastic designs.

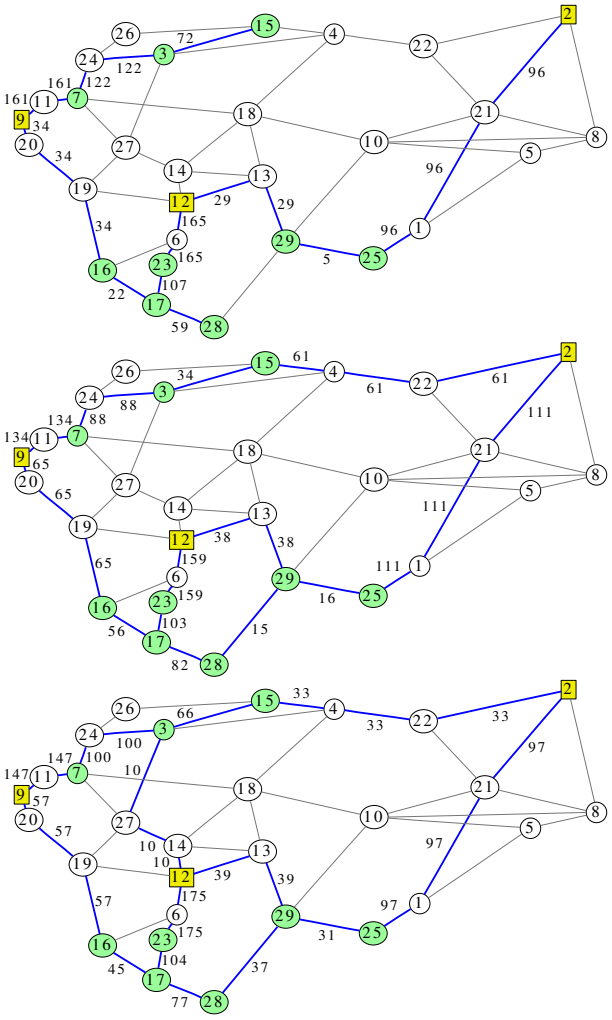

Figure 7: Similar deterministic and stochastic structure. Deterministic demand (top), uncorrelated stochastic demand (middle) and mixed correlated stochastic demand (bottom) solutions of Germany_01 test cases showing that the stochastic solution structures are similar to the deterministic ones when setup costs are low.

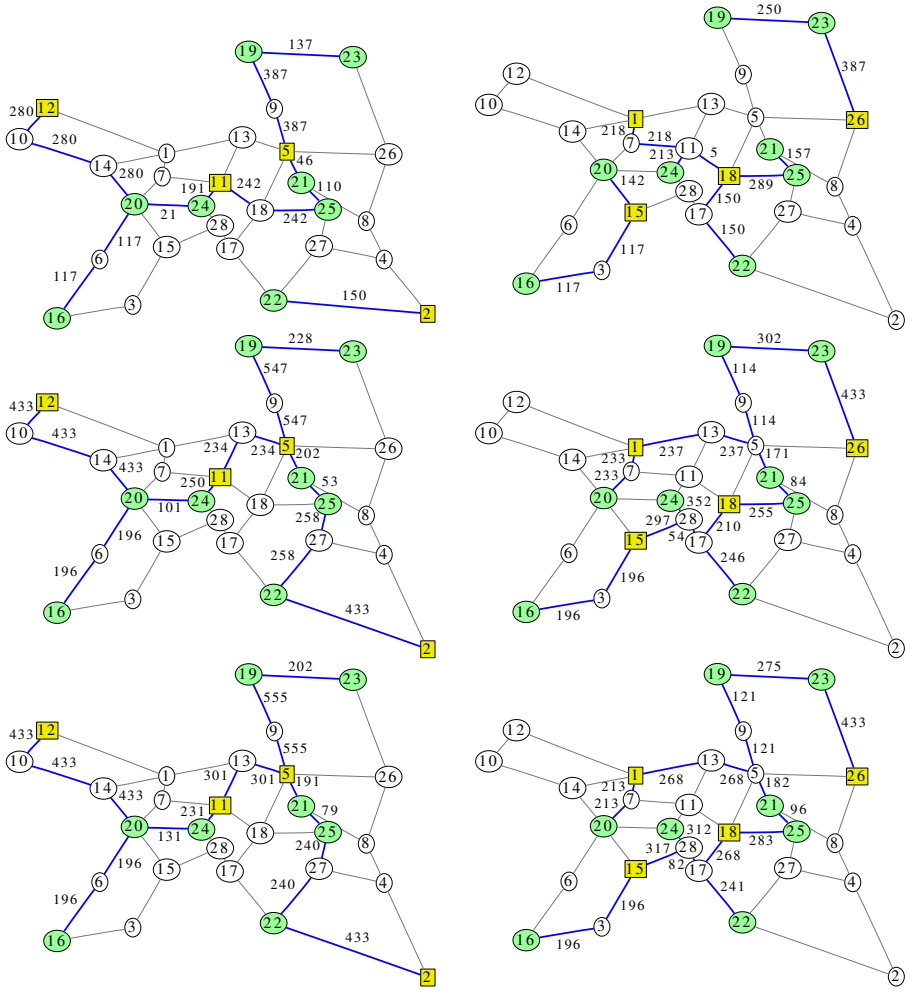

Figure 8: Deterministic skeletons in stochastic solution. Deterministic demand (top), uncorrelated stochastic demand (middle) and positively correlated stochastic demand (bottom) solutions of Nobel-EU_01 (left) and Nobel-EU_02 (right) test cases showing that the stochastic solution contain the deterministic skeletons when source nodes are far from demand nodes and not otherwise.

When source nodes are far from the demand nodes, we also find the deterministic skeletons, more or less fully, within the stochastic ones. Consider Figure 8. In the first column we see a case where the deterministic skeleton is almost kept. This is because source nodes 2 and 12 are in the corners of the graph and both are far from demand nodes 22 and 20. Hence, the paths needed to reach those demand nodes are long. In that case it is usually better to keep these cheapest connections even in the stochastic case. But even so, the mixed correlated case is different. This is caused by another feature of the stochastic solution which we shall discuss later in the section on negative correlations. For the case in the second column, the skeleton is not retained, as here the source nodes are very near to the demand nodes and hence have the possibilities to utilize variation of demand and re-align the design.

Consolidated paths

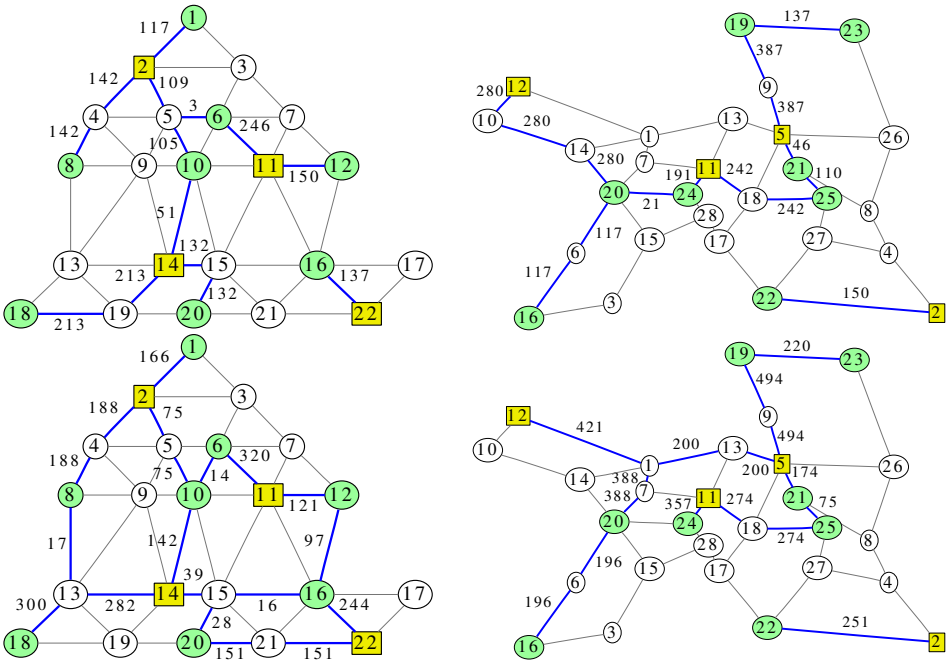

Figure 9: Consolidated paths. Deterministic (top) and stochastic (bottom) solutions of Molde and Nobel_EU test cases, showing that the stochastic solution structures have consolidated paths which are not in the deterministic solutions.

In the stochastic design we sometimes observe that there are paths which act like highways carrying supply and demand for many nodes. This is more evident in the cases where the setup costs are very high, especially when variable setup costs are proportionally very high. This can be explained by the fact that as the setup costs are high, it becomes beneficial to consolidate demand of different nodes in order to reduce the total installed capacity. In Figure 9, we show two cases where variable setup costs are comparatively much higher than other costs. The first set of figures shows a consolidated path 22–21–20–15–14–13 efficiently connecting nodes on the path and one edge away. The path is also part of a loop (see later). This is better according to the out-of-sample evaluation, even though most individual demand nodes now have longer paths to the source nodes. Here it is possible, for example, to use free capacity available on the path to reach demand nodes 16, 18 and 8 from source node 22.

In the second set of figures we see that demand nodes 20 and 16 are connected with source node 12 via transshipment node 1 instead of the shorter (in terms of fixed and variable setup costs) path via node 10, as in the deterministic design. If we look at the stochastic design in more detail, we observe that edge 1–13 has 200 units of capacity installed, while 1–7 and 7–20 have 388 units of capacity. These add to 588 units of capacity usable for source node 12. But only 421 units of capacity are installed on edge 12–1. This makes sense since by consolidating the flow from source node 12 to nodes 20 and 16 with the flow to node 13 (and onward), the capacity into node 1 can be set 167 units lower than the outgoing capacity (this statement makes sense even though the edges are not directed). Thus, consolidating flow saves costs in total even though the paths used may be longer and costlier than in the deterministic case.

But this feature may not be attractive when the setup costs becomes low i.e., when flow costs matter much for the optimal design. Then sharing does not create sufficient savings as flow costs more than offset the savings in setup costs.

Loops

The stochastic design sometimes has loops. That will never happen in the deterministic case as long as there are no effective upper bounds on edge capacities. Two types of loops are seen, one where the source node(s) are part of them and the other is where the loops are formed with nodes excluding source nodes. In Figure 10 we see both types of loops. In the second chart of the figure, we see that loops 3–5–4–2–3, 11–12–7–9–11, and 11–16–14–17–7–9–11 are formed having a source node in them. The third chart shows a loop 16–13–15–18–14–16 without a source node in it.

The first kind of loop takes advantage of free capacity available in one of the arms of the deterministic skeleton to fulfill demand of some demand node lying on a different arm. An extra edge, connecting the two arms, makes a loop and helps satisfy demand. For the second chart of the figure, demand node 4 has a maximal demand of 1803 in one of the scenarios, whereas the path serving it (edges connecting 3, 2, and 4) has 1421 units of capacity. Here, the free capacity available on the path 3–5–12 is utilized by building an extra edge 5–4 to form a loop to serve most of this higher demands of node 4. In the third chart of the figure, we see a loop created by adding an extra edge 16–14 to the deterministic solution. Here the loop is formed to fulfill higher demand scenarios of node 14. This loop provides more supply to demand node 14 than the capacity we see in path 11–16–14, as extra capacity of the path 16–13–15 is utilized to serve node 14.

And if we look back at the second network in Figure 7, we see that node 28, a leaf in the deterministic design, gets connected to the path 12–13–29–25 by an edge 28–29 to form a loop. This is because it is cheaper to add the extra edge 28–29 with some capacity than increase capacities all the way on the paths 12–6–23–17–28 and 12–13–29 to satisfy demand of nodes 28 and 29 respectively. This edge is valuable as it can be use in both directions, to supplement demand needs of nodes 28 and 29.

Negative correlations

A very basic hedging principle is seen when source and sink nodes are linked. This principle does not show in the deterministic solutions. A source node is typically linked with demand nodes with negatively correlated demands, so that variation in demand can be utilized. In Figure 11, we see that in the mixed correlation case demand node 16 is connected to source

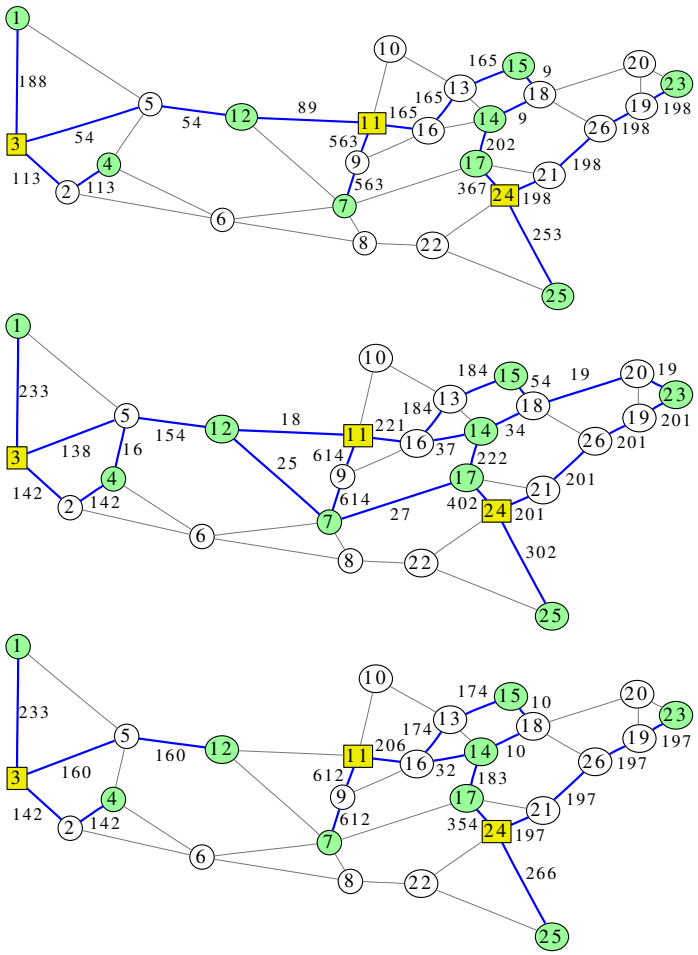

Figure 10: Loop formation in stochastic solution. Deterministic (top), uncorrelated (middle) and positively correlated (bottom) stochastic solutions of US_02 showing that the stochastic solution structures have loops which are not in the deterministic solutions. Note that the values in the figures are one-tenth of actual.

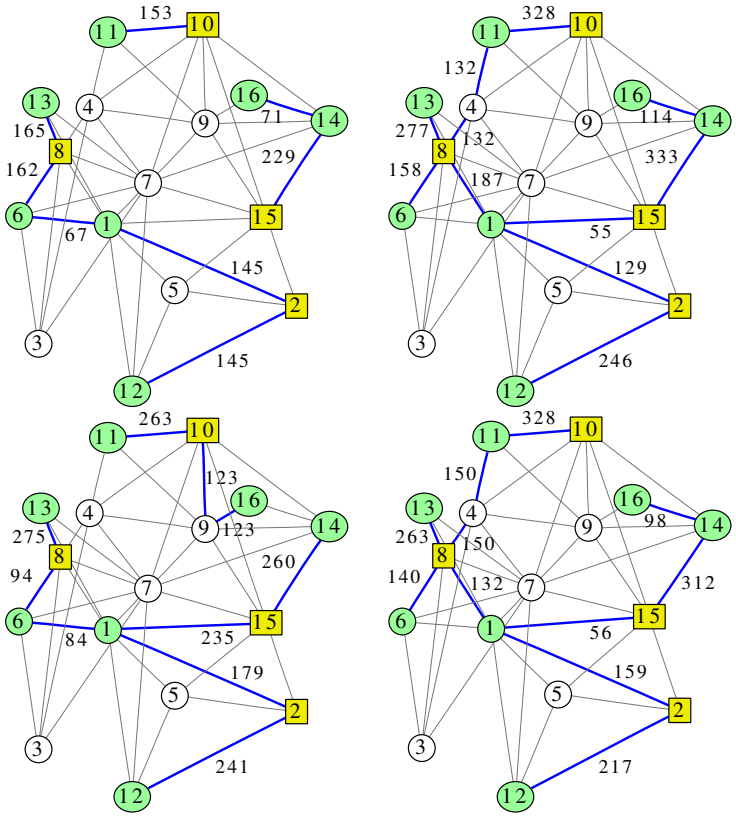

Figure 11: Pairing of negatively correlated demand nodes. Deterministic (top left) and three correlated cases of stochastic solutions–zero (top right), mixed (bottom left) and positive (bottom right)–of NY_04 test case showing that in the stochastic solution negatively correlated demand nodes 11 and 16 get connected to same source node.

node 10 which is also supplying demand node 11. This is because demand nodes 11 and 16 have negatively correlated demands. In the other cases, demand node 16 is served by source node 15. Similarly, we observe that positively correlated demand nodes are disconnected. Refer to the bottom-right chart of Figure 9 of Nobel_01 mixed correlated case where positively correlated demand nodes 20 and 24 are no longer connected the way they were in the uncorrelated case.

4 Conclusion

The purpose of this paper has been to understand what constitutes a good robust design for a single-commodity stochastic network design problem with multiple sources and sinks. This paper discusses only randomness in demand.

We observe that the deterministic solution can be very bad with respect to expected behavior. But still we see certain structural patterns re-emerging in the stochastic solutions. First we observe that the deterministic solution behaves worse in the stochastic environment as the number of source nodes increases. With many source node, the deterministic solution, which is a forest, typically of many trees, decreases the ability to share capacity in a stochastic

environment. Also, as correlations are of no concern in a deterministic setting, the assignment of demand nodes to source nodes may be rather far off what is optimal.

If the variation in demand is moderate or low, the deterministic skeleton (being a forest) can be used to carry a major portion of the flow, needing very few additional edges in the stochastic environment. As the variation increases, the source nodes will have insufficient capacity to fulfill demand using the deterministic skeleton (irrespective of installed capacities), and hence will need more edges. Therefore, a re-alignment of distribution patterns will emerge. However, when the fixed and variable setup costs are low compared to the flow costs, the deterministic skeleton is contained in the stochastic one, with only few extra edges added, even in cases of higher variation in demand.

For uncorrelated and positively correlated demands existing far from the source nodes, we keep the portion of the deterministic skeleton that contains paths leading up to clusters of demand nodes. However, within a cluster of demand nodes we observe changes from that of the deterministic solution, in order to benefit from demand variations.

With high variable setup cost, we see consolidation of capacities in paths reaching downstream demand nodes. These paths will emerge more so between demand nodes which are negatively correlated. However, with increasing proportion of flow cost in deciding the optimal solution, this consolidation will be weaker.

Networks with all types of possible correlations among demands show loops in the stochastic solution. The loop formation gets stronger with increasing variable setup costs.

Source nodes choose demands to serve according to the possibility for hedging among them. Hence, everything else being equal, negatively correlated demand nodes are most likely to be served from the same source node. This also results in the breaking or weakening of links between positively correlated demand nodes relative to the deterministic solution, which has no such concerns.

In total, the main observations of optimal designs are therefore as follows. Note that the observations are connected, and to some extent see the same phenomena from different perspectives.

- Especially with high setup costs, the deterministic design tends to be a forest of small trees. This is particularly bad in a stochastic environment. The good robust designs will contain many more connections than the deterministic counterpart, and the design will contain loops.

- From a demand node perspective: When negative correlations in demand are present, the design should be such that nodes with negatively correlated demands share paths to one or more source nodes. If there are no negative correlations, it is still important to utilize variation in demand to reduce investments by looking for as small (albeit positive) correlations as possible.

- From a source node perspective: If supply capacity is limited, a source node needs to be connected to several demand nodes, preferably nodes with as small correlations (negative if feasible) in demand as possible, so as to be able to use its supply capacity well in all scenarios.

- Especially with high setups costs and source and demand nodes concentrated in different areas, it is good to build a high capacity path – a highway – from the area of the

source nodes to the area of the demand nodes. The source and demand nodes are then connected to the highway using the principles of the previous two items.

- A single loop may be seen as two paths plus a crossover edge (or path). The crossover must be placed such that demand along either the two upstream or the two downstream sub-paths are negatively correlated. If no negative correlations are possible, the same correlations should be as small as possible. This will reduce the overall investments. The same logic applies if the crossover edge (path) connects two disjoint paths.

Future work. We plan to follow up this work by studying the case of random arc capacities.

Acknowledgments

This project was supported in part by grant 171007/V30 from The Research Council of Norway. While working on this project, T.G. Crainic was the NSERC Industrial Research Chair on Logistics Management, ESG UQAM, and Adjunct Professor with the Department of Computer Science and Operations Research, Université de Montréal, and the Department of Economics and Business Administration, Molde University College, Norway. Partial funding for this project has been provided by the Natural Sciences and Engineering Council of Canada (NSERC), through its Industrial Research Chair and Discovery Grants programs.

References

R. K. Ahuja, T. L. Magnanti, J. B. Orlin, and M. R. Reddy. Applications of networks optimization. In M. Ball, T.L. Magnanti, C.L. Monma, and G.L. Nemhauser, editors, *Network Models*, volume 7 of *Handbooks in Operations Research and Management Science*, pages 1–83. North-Holland, Amsterdam, 1995.

M. Ball, C. Barnhart, G. Nemhauser, and A. Odoni. Air transportation: Irregular operations and control. In C. Barnhart and G. Laporte, editors, *Transportation*, number 14 in Handbooks in Operations Research and Management Science, chapter 1, pages 1–67. Elsevier, 2007.

J. R. Birge. The value of the stochastic solution in stochastic linear programming with fixed recourse. *Mathematical Programming*, 24:314–325, 1982.

Emden R. Gansner and Stephen C. North. An open graph visualization system and its applications to software engineering. *Software: Practice and Experience*, 30(11):1203–1233, 2000. doi: 10.1002/1097-024X(200009)30:11⟨1203::AID-SPE338⟩3.0.CO;2-N.

J. L. Higle and S. W. Wallace. Sensitivity analysis and uncertainty in linear programming. *Interfaces*, 33:53–60, 2003.

Kjetil Høyland, Michal Kaut, and Stein W. Wallace. A heuristic for moment-matching scenario generation. *Computational Optimization and Applications*, 24(2–3):169–185, 2003. doi: 10.1023/A:1021853807313.

ILOG. *CPLEX 9.0 User's Manual*. ILOG, S. A., 2003.

Michal Kaut and Stein W. Wallace. Evaluation of scenario-generation methods for stochastic programming. *Pacific Journal of Optimization*, 3(2):257–271, 2007.

A.-G. Lium, T. G. Crainic, and S. W. Wallace. Correlations in stochastic programming: A case from stochastic service network design. *Asia-Pacific Journal of Operational Research*, 24(2):161–179, 2007. doi: 10.1142/S0217595907001206.

A.-G. Lium, T. G. Crainic, and S. W. Wallace. A study of demand stochasticity in stochastic network design. *Transportation Science*, 43(2):144–157, 2009. doi: 10.1287/trsc.1090.0265.

S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly. SNDlib 1.0 – Survivable Network Design library. *Networks*, 55(3):276–286, 2010. doi: 10.1002/net.20371.

Kevin P. Scheibe and Cliff T. Ragsdale. A model for the capacitated, hop-constrained, per-packet wireless mesh network design problem. *European Journal of Operational Research*, 197(2):773–784, 2009. doi: 10.1016/j.ejor.2008.07.020.

Biju K. Thapalia, Teodor G. Crainic, Michal Kaut, and Stein W. Wallace. Single source single-commodity stochastic network design. *Computational Management Science*, 9(1): 139–160, 2012. doi: 10.1007/s10287-010-0129-0. Special issue on 'Optimal decision making under uncertainty'.

Stein W. Wallace. Decision making under uncertainty: Is sensitivity analysis of any use? *Operations Research*, 48(1):20–25, 2000.

Stein W. Wallace. Stochastic programming and the option of doing it differently. *Annals of Operations Research*, 177(1):3–8, 2010. doi: 10.1007/s10479-009-0600-x.

Joyce W. Yen and John R. Birge. A stochastic programming approach to the airline crew scheduling problem. *Transportation Science*, 40(1):3–14, 2006.

A Results of the numerical tests

This appendix provides detailed results from the tests in Section 3. Table 2 provides the numbers used to generate Figure 3, in Section 3.3. The analysis in Section 3.4 is also based on these computations, but the individual cases cannot be reproduced from these tables. Tables 3 to 5 present the full computational results for Figure 1, while those of Figure 5 are found in Tables 6 to 8.

Table 2: The numbers corresponding to Figure 3

	Fixed setup cost			Variable setup cost		
	A	B	C	A	B	C
Minimum value	0.54	0.54	0.91	0.56	0.76	0.96
Geometric mean	0.74	0.74	1.00	0.62	0.95	1.03
Maximum value	0.93	0.93	1.14	0.68	1.09	1.12

Table 3: Results of Comparison A corresponding to Figure 1, split by correlation structure.

	Deterministic solution			**Stochastic solution**		
Test Name	$\rho = \mathbf{0}$	$\rho > \mathbf{0}$	$\rho \gtrless \mathbf{0}$	$\rho = \mathbf{0}$	$\rho > \mathbf{0}$	$\rho \gtrless \mathbf{0}$
Germany_01	17165	18525	17504	13410	14838	16038
Germany_02	18077	18749	18047	13774	15728	17210
Germany_04	17372	18048	17711	12182	13569	14822
Molde_01	89318	89814	89844	6968	21110	6794
Molde_02	90920	90791	90831	7470	21767	7032
Molde_04	90746	90618	90657	6987	21248	6769
Montreal_r06.1_01	157972	157974	157969	112619	111764	113638
Montreal_r06.1_02	147886	147892	147884	102292	104391	102577
Montreal_r10.1_01	116165	117029	115614	103571	105251	102774
Montreal_r10.1_02	80913	81512	81325	59662	61636	59078
Montreal_r10.1_03	92081	92666	92496	71747	73474	71545
Nobel-EU_01	1209410	1332870	1237110	128951	192455	123499
Nobel-EU_02	1269300	1339410	1243730	110903	172271	108688
Nobel-EU_03	1262480	1332590	1236910	100219	163244	97482
NY_01	24860500	26023300	26012800	1851100	3366850	1786500
NY_02	25761800	26256300	26239100	1726640	3423220	1545050
NY_04	24693000	25856600	25846100	1629720	3268480	1553470
US_01	414391	414559	414378	371013	388984	360429
US_02	348460	353104	345384	314188	333334	303797
US_04	394493	400384	390647	354348	373575	345213

Table 4: Results of Comparison B corresponding to Figure 1, split by correlation structure.

	Deterministic solution			**Stochastic solution**		
Test Name	$\rho = \mathbf{0}$	$\rho > \mathbf{0}$	$\rho \gtrless \mathbf{0}$	$\rho = \mathbf{0}$	$\rho > \mathbf{0}$	$\rho \gtrless \mathbf{0}$
Germany_01	13772	16130	13552	13388	14812	13077
Germany_02	14246	15922	13565	13732	15710	13190
Germany_04	12448	13566	12645	12139	13556	12213
Molde_01	28210	63165	17824	6449	20898	6156
Molde_02	28727	63623	18177	7046	21340	6574
Molde_04	28643	63527	18102	6595	21088	6278
Montreal_r06.1_01	118847	121655	116083	101285	105398	95922
Montreal_r06.1_02	108009	110727	105391	87818	93341	85182
Montreal_r10.1_01	104465	108639	103366	103536	105220	102748
Montreal_r10.1_02	62477	66216	60969	59605	61577	59002
Montreal_r10.1_03	74936	78331	73921	71676	73418	71505
Nobel-EU_01	163710	517543	117573	122131	189166	116968
Nobel-EU_02	204476	305544	130275	105638	168372	103345
Nobel-EU_03	300252	639796	143485	96720	160822	94153
NY_01	5813210	10931900	5899210	1675610	3281980	1700720
NY_02	3850040	8315310	2995700	1629340	3225060	1501700
NY_04	5586750	10731400	5691810	1516610	3128730	1483130
US_01	379657	389969	373204	369994	388887	360025
US_02	318592	334225	308236	313412	333110	303356
US_04	358255	373970	349697	353351	373159	344944

Table 5: Results of Comparison C corresponding to Figure 1, split by correlation structure.

	Deterministic solution			**Stochastic solution**		
Test Name	$\rho = \mathbf{0}$	$\rho > \mathbf{0}$	$\rho \gtrless \mathbf{0}$	$\rho = \mathbf{0}$	$\rho > \mathbf{0}$	$\rho \gtrless \mathbf{0}$
Germany_01	13588	15807	13180	13406	14836	13101
Germany_02	13752	16053	13172	13751	15717	13186
Germany_04	12114	13684	11968	12170	13563	11992
Molde_01	6582	21104	6175	6520	20911	6216
Molde_02	7168	21616	6667	7034	21352	6607
Molde_04	6908	21491	6380	6606	21076	6321
Montreal_r06.1_01	97268	102469	94633	97280	102753	94745
Montreal_r06.1_02	86172	91202	84293	86232	91237	84299
Montreal_r10.1_01	107376	109207	106848	103546	105228	102765
Montreal_r10.1_02	64556	66634	64130	59659	61593	59018
Montreal_r10.1_03	71833	73624	71604	71718	73431	71534
Nobel-EU_01	124432	191719	117573	125304	190475	116983
Nobel-EU_02	106971	170755	103847	105794	169633	103365
Nobel-EU_03	101393	165388	98930	97405	162022	94310
NY_01	1738870	3326650	1761180	1676470	3286030	1705290
NY_02	1632510	3246760	1534730	1629740	3225230	1501700
NY_04	1524290	3143840	1472440	1521280	3128920	1483130
US_01	377252	401183	366298	370814	388883	360311
US_02	317048	341646	306368	313987	333100	303691
US_04	353590	373386	345223	353987	373186	345195

Table 6: The ratios corresponding to Figure 5 split by correlation structure.

		Comparison A			**Comparison B**			**Comparison C**		
Test Name	$\mathbf{g_k/C_k}$	$\rho\mathbf{=0}$	$\rho\mathbf{>0}$	$\rho\gtrless\mathbf{0}$	$\rho\mathbf{=0}$	$\rho\mathbf{>0}$	$\rho\gtrless\mathbf{0}$	$\rho\mathbf{=0}$	$\rho\mathbf{>0}$	$\rho\gtrless\mathbf{0}$
Germany_01	0.001	1.32	1.29	1.36	1.04	1.01	1.08	1.01	1.06	1.01
	0.05	1.27	1.27	1.33	1.02	1.09	1.03	1.01	1.07	1.01
	0.25	1.30	1.25	1.33	1.05	1.06	1.12	1.02	1.03	1.01
	0.5	1.33	1.29	1.35	1.07	1.07	1.15	1.00	1.00	1.00
	0.999	1.40	1.35	1.39	1.03	1.01	1.00	1.00	1.00	1.00
Germany_02	0.001	1.37	1.23	1.42	1.07	1.02	1.09	1.00	1.03	1.00
	0.05	1.32	1.20	1.36	1.03	1.01	1.03	1.00	1.03	1.00
	0.25	1.31	1.24	1.38	1.01	1.04	1.05	1.00	1.01	1.00
	0.5	1.31	1.26	1.34	1.03	1.06	1.09	1.01	1.03	1.01
	0.999	1.37	1.33	1.37	1.03	1.01	1.00	1.00	1.00	1.00
Germany_04	0.001	1.47	1.36	1.49	1.05	1.01	1.07	1.00	1.01	1.00
	0.05	1.39	1.34	1.45	1.05	1.11	1.11	1.00	1.02	1.00
	0.25	1.39	1.34	1.43	1.04	1.10	1.07	1.01	1.03	1.01
	0.5	1.40	1.35	1.44	1.02	1.09	1.05	1.02	1.04	1.03
	0.999	1.45	1.38	1.45	1.03	1.01	1.00	1.00	1.00	1.00

Table 7: Ratios corresponding to Figure 5 split by correlation structure, cont. The asterisk (*) denotes cases where the solver did not finished within 15 days.

		Comparison A			**Comparison B**			**Comparison C**		
Test Name	$\mathbf{g_k/C_k}$	$\rho=\mathbf{0}$	$\rho>\mathbf{0}$	$\rho\gtrless\mathbf{0}$	$\rho=\mathbf{0}$	$\rho>\mathbf{0}$	$\rho\gtrless\mathbf{0}$	$\rho=\mathbf{0}$	$\rho>\mathbf{0}$	$\rho\gtrless\mathbf{0}$
Molde_01	0.001	16.10	4.57	16.11	2.19	2.01	3.71	1.04	1.01	1.01
	0.05	13.69	4.41	14.38	2.53	1.11	1.79	1.02	1.01	1.01
	0.25	12.04	4.25	12.06	4.16	1.40	3.00	1.01	1.00	1.00
	0.5	10.45	4.04	10.55	3.66	1.35	2.67	1.01	1.00	1.00
	0.999	8.52	3.72	9.14	3.05	1.27	2.37	1.00	1.00	1.02
Molde_02	0.001	14.96	4.50	15.10	3.16	3.18	4.77	1.05	1.02	1.02
	0.05	13.67	4.40	14.03	2.84	3.10	4.44	1.03	1.02	1.02
	0.25	11.49	4.19	11.90	2.45	2.96	3.82	1.03	1.02	1.03
	0.5	9.22	3.91	9.64	1.93	1.33	1.86	1.02	1.00	1.04
	0.999	8.02	*	*	3.02	*	*	1.00	*	*
Molde_04	0.001	15.92	4.55	15.87	3.27	3.20	5.01	1.08	1.03	1.02
	0.05	14.60	4.46	14.57	3.03	3.14	4.62	1.07	1.03	1.03
	0.25	11.84	4.20	11.93	4.13	1.39	3.01	1.02	1.00	1.04
	0.5	10.15	3.99	10.44	3.59	1.34	2.68	1.02	1.00	1.05
	0.999	8.17	3.68	8.05	2.30	1.33	2.41	1.03	1.01	1.04
Montreal_r06.1_01	0.001	1.40	1.36	1.43	1.05	1.05	1.02	1.00	1.00	1.00
	0.05	1.39	1.36	1.43	1.04	1.04	1.01	1.00	1.00	1.00
	0.25	1.39	1.35	1.42	1.03	1.03	1.01	1.00	1.00	1.00
	0.5	1.38	1.34	1.42	1.02	1.03	1.00	1.00	1.00	1.00
	0.999	1.38	1.34	1.44	1.01	1.02	1.00	1.00	1.00	1.00
Montreal_r06.1_02	0.001	1.50	1.46	1.53	1.06	1.06	1.02	1.00	1.00	1.00
	0.05	1.50	1.45	1.53	1.05	1.06	1.02	1.00	1.00	1.00
	0.25	1.49	1.45	1.52	1.04	1.05	1.01	1.00	1.00	1.00
	0.5	1.48	1.44	1.52	1.03	1.04	1.00	1.00	1.00	1.00
	0.999	1.48	1.43	1.53	1.02	1.03	1.00	1.00	1.00	1.00
Montreal_r10.1_01	0.001	1.14	1.13	1.14	1.03	1.05	1.02	1.00	1.00	1.00
	0.05	1.13	1.12	1.13	1.02	1.03	1.01	1.00	1.00	1.00
	0.25	1.13	1.12	1.13	1.02	1.02	1.00	1.00	1.00	1.00
	0.5	1.13	1.12	1.14	1.02	1.02	1.00	1.00	1.01	1.00
	0.999	1.15	1.14	1.15	1.01	1.01	1.00	1.00	1.00	1.00
Montreal_r10.1_02	0.001	1.35	1.32	1.36	1.08	1.10	1.06	1.00	1.00	1.00
	0.05	1.34	1.31	1.36	1.03	1.03	1.02	1.00	1.00	1.00
	0.25	1.33	1.30	1.35	1.02	1.02	1.01	1.00	1.00	1.00
	0.5	1.33	1.30	1.35	1.01	1.02	1.00	1.00	1.00	1.00
	0.999	1.34	1.31	1.36	1.00	1.01	1.00	1.00	1.00	1.00
Montreal_r10.1_03	0.001	1.33	1.30	1.33	1.02	1.02	1.01	1.00	1.00	1.00
	0.05	1.33	1.30	1.33	1.03	1.02	1.03	1.00	1.00	1.00
	0.25	1.32	1.29	1.32	1.03	1.03	1.03	1.00	1.00	1.00
	0.5	1.31	1.29	1.31	1.02	1.03	1.02	1.00	1.00	1.00
	0.999	1.31	1.29	1.31	1.02	1.02	1.01	1.00	1.00	1.00
Nobel-EU_01	0.001	12.50	9.02	14.59	1.21	1.51	1.36	1.02	1.01	1.01
	0.05	11.24	8.58	13.58	1.16	1.26	1.11	1.02	1.01	1.00
	0.25	9.89	7.88	11.04	1.35	3.22	1.00	1.02	1.02	1.00
	0.5	8.58	7.43	10.12	4.28	6.81	3.00	1.03	1.02	1.07
	0.999	7.61	6.81	9.11	3.80	6.23	2.74	1.01	1.00	1.11

Table 8: The remaining ratios corresponding to Figure 5 split by correlation structure

		Comparison A			**Comparison B**			**Comparison C**		
Test Name	$\mathbf{g_k/C_k}$	$\rho=\mathbf{0}$	$\rho>\mathbf{0}$	$\rho\gtrless\mathbf{0}$	$\rho=\mathbf{0}$	$\rho>\mathbf{0}$	$\rho\gtrless\mathbf{0}$	$\rho=\mathbf{0}$	$\rho>\mathbf{0}$	$\rho\gtrless\mathbf{0}$
Nobel-EU_02	0.001	16.17	10.45	16.01	3.70	4.95	4.99	1.02	1.01	1.01
	0.05	14.74	10.03	14.85	3.06	4.20	1.50	1.01	1.00	1.00
	0.25	12.52	9.13	12.87	2.55	2.40	1.39	1.02	1.01	1.00
	0.5	10.83	8.33	11.58	2.24	2.21	1.30	1.00	1.00	1.00
	0.999	9.34	7.51	9.89	1.98	2.03	1.18	1.00	1.00	1.00
Nobel-EU_03	0.001	16.68	10.78	16.95	4.41	5.59	6.19	1.01	1.01	1.01
	0.05	15.68	10.41	15.96	3.19	4.32	1.56	1.00	1.00	1.02
	0.25	13.35	9.52	13.90	2.66	2.46	1.43	1.04	1.02	1.02
	0.5	11.65	8.74	12.28	2.35	2.28	1.31	1.03	1.02	1.01
	0.999	9.82	7.72	10.27	2.04	2.06	1.17	1.00	1.01	1.01
NY_01	0.001	22.52	8.73	23.11	5.17	3.65	5.24	1.03	1.01	1.01
	0.05	20.55	8.45	21.39	4.74	3.54	4.86	1.03	1.01	1.01
	0.25	17.71	7.97	17.91	4.13	3.36	4.13	1.03	1.02	1.03
	0.5	14.97	7.46	16.11	6.02	3.70	7.95	1.00	1.00	1.01
	0.999	12.58	6.77	14.13	5.22	3.41	7.08	1.02	1.01	1.02
NY_02	0.001	25.80	9.14	25.09	4.73	4.46	1.67	1.05	1.02	1.00
	0.05	23.63	8.88	23.11	6.96	4.55	2.98	1.06	1.02	1.01
	0.25	18.79	8.21	20.10	4.81	2.70	3.04	1.03	1.01	1.01
	0.5	16.44	7.66	17.71	2.63	2.66	2.19	1.01	1.01	1.03
	0.999	14.12	7.02	15.10	5.53	3.48	7.47	1.01	1.00	1.02
NY_04	0.001	27.91	9.34	26.74	5.95	3.81	5.66	1.02	1.01	1.03
	0.05	25.38	9.07	24.78	5.43	3.71	5.26	1.01	1.00	1.02
	0.25	20.58	8.38	20.73	4.55	3.50	4.69	1.03	1.01	1.05
	0.5	17.75	7.82	18.24	3.98	3.29	4.18	1.00	1.00	1.03
	0.999	14.73	7.10	15.61	3.55	3.04	3.69	1.00	1.00	1.00
US_01	0.001	1.12	1.06	1.15	1.03	1.00	1.03	1.02	1.03	1.02
	0.05	1.09	1.06	1.13	1.01	1.00	1.02	1.00	1.00	1.00
	0.25	1.08	1.06	1.11	1.00	1.00	1.00	1.00	1.00	1.00
	0.5	1.09	1.07	1.12	1.00	1.00	1.00	1.00	1.00	1.00
	0.999	1.15	1.11	1.16	1.03	1.02	1.07	1.03	1.02	1.07
US_02	0.001	1.11	1.05	1.14	1.02	1.00	1.01	1.01	1.02	1.01
	0.05	1.11	1.07	1.13	1.02	1.01	1.01	1.01	1.03	1.01
	0.25	1.10	1.07	1.13	1.00	1.00	1.00	1.00	1.00	1.00
	0.5	1.11	1.07	1.13	1.00	1.00	1.00	1.00	1.00	1.00
	0.999	1.14	1.10	1.17	1.00	1.00	1.00	1.00	1.00	1.00
US_04	0.001	1.11	1.07	1.13	1.02	1.00	1.01	1.00	1.00	1.00
	0.05	1.08	1.07	1.11	1.00	1.00	1.00	1.00	1.00	1.00
	0.25	1.09	1.07	1.12	1.00	1.00	1.00	1.00	1.00	1.00
	0.5	1.12	1.08	1.15	1.02	1.00	1.04	1.02	1.00	1.02
	0.999	1.16	1.13	1.18	1.00	1.00	1.00	1.00	1.00	1.00