http://pubsonline.informs.org/journal/ijoc

INFORMS JOURNAL ON COMPUTING
Vol. 33, No. 1, Winter 2021, pp. 143–162
ISSN 1091-9856 (print), ISSN 1526-5528 (online)

Exploiting the Structure of Two-Stage Robust Optimization Models with Exponential Scenarios

Hossein Hashemi Doulabi,[a,b] **Patrick Jaillet,**[c] **Gilles Pesant,**[b,d] **Louis-Martin Rousseau**[b,e]

[a] Department of Mechanical, Industrial and Aerospace Engineering, Concordia University, Montreal, Quebec H3G 1M8, Canada; [b] Interuniversity Research Center on Enterprise Networks, Logistics and Transportation (CIRRELT), Université de Montréal, Montreal, Quebec H3C 3J7, Canada; [c] Department of Electrical Engineering and Computer Science, Laboratory for Information and Decision Systems, Operations Research Center, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139; [d] Department of Computer and Software Engineering, Polytechnique Montréal, Montreal, Quebec H3T 1J4, Canada; [e] Department of Mathematics and Industrial Engineering, Polytechnique Montréal, Montreal, Quebec H3T 1J4, Canada

Contact: hossein.hashemi@concordia.ca, https://orcid.org/0000-0002-7385-1274 (HHD); jaillet@mit.edu (PJ); gilles.pesant@polymtl.ca (GP); louis-martin.rousseau@polymtl.ca (L-MR)

Received: February 7, 2018
Revised: February 3, 2019; April 20, 2019; July 24, 2019
Accepted: July 31, 2019
Published Online in Articles in Advance: June 15, 2020

https://doi.org/10.1287/ijoc.2019.0928

Abstract. This paper addresses a class of two-stage robust optimization models with an exponential number of scenarios given implicitly. We apply Dantzig–Wolfe decomposition to exploit the structure of these models and show that the original problem reduces to a single-stage robust problem. We propose a Benders algorithm for the reformulated single-stage problem. We also develop a heuristic algorithm that dualizes the linear programming relaxation of the inner maximization problem in the reformulated model and iteratively generates cuts to shape the convex hull of the uncertainty set. We combine this heuristic with the Benders algorithm to create a more effective hybrid Benders algorithm. Because the master problem and subproblem in the Benders algorithm are mixed-integer programs, it is computationally demanding to solve them optimally at each iteration of the algorithm. Therefore, we develop novel stopping conditions for these mixed-integer programs and provide the relevant convergence proofs. Extensive computational experiments on a nurse planning problem and a two-echelon supply chain problem are performed to evaluate the efficiency of the proposed algorithms.

History: Accepted by Andrea Lodi, Area Editor, Design & Analysis of Algorithms—Discrete.
Supplemental Material: Data and the electronic supplement are available at https://doi.org/10.1287/ijoc.2019.0928.

Keywords: integer programming • Dantzig–Wolfe decomposition • two-stage robust optimization

1. Introduction

In the operations research literature, there are many different methodologies to address uncertainty in optimization problems. Stochastic approaches are one of the main classes and are applicable if probability distributions of uncertain parameters are known. However, these approaches are usually criticized for requiring information on the probability distributions and also for computational complexities. Robust optimization, a more recent methodology, generally assumes that uncertain parameters belong to an uncertainty set, and it aims to find a robust solution immunizing the decision maker against the worst-case scenario within this uncertainty set.

Robust optimization was initially proposed for single-stage optimization problems where the decision maker must choose a complete solution before the disclosure of information about the real values of uncertain parameters (Soyster 1973, Ben-Tal and Nemirovski 1999). Then it was extended to multistage problems where the values of uncertain parameters are revealed gradually in several stages (Ben-Tal et al. 2004, Delage and Iancu 2015). In multistage robust problems, the decision maker does not choose a complete solution at the beginning but instead makes partial decisions sequentially after observing the values of uncertain parameters over different stages.

In robust optimization problems, choosing an appropriate uncertainty set is critical and can highly affect the robustness and the optimal objective value of the obtained solution. The decision maker should select a suitable uncertainty set to reasonably represent the randomness of the uncertain parameters while taking into account the computational issues arising in the solution algorithm. From the literature on robust optimization, the most prevalent uncertainty sets are box uncertainty sets (Soyster 1973), ellipsoidal uncertainty sets (El Ghaoui and Lebret 1997, El Ghaoui et al. 1998, Ben-Tal and Nemirovski 1999), polyhedral uncertainty sets, and Γ-cardinality uncertainty sets (Bertsimas and Sim 2004). In box uncertainty sets, uncertain parameters are assumed to take their values from different intervals independently. Box uncertainty sets usually result in overly conservative solutions because all parameters are allowed to take their worst values simultaneously. Ellipsoidal uncertainty sets alleviate this issue by restricting the uncertain parameters to an ellipsoidal space, and this prevents them from taking worst values at the same time. Polyhedral uncertainty sets confine the uncertain parameters to a polyhedral space and can be viewed as a special case of ellipsoidal uncertainty sets (Ben-Tal and Nemirovski 1999). In Γ-cardinality uncertainty sets, for each constraint, the number of uncertain parameters deviating from their nominal values must be less than Γ.

In the literature, convex uncertainty sets are used to model robust problems. The main advantage of these uncertainty sets is that they can be simply formulated by continuous parameters, and the problem remains tractable in many cases such as linear programs. However, it is sometimes unavoidable or desirable to use integer parameters to formulate the uncertainty set, which results in an exponential number of scenarios. Nguyen and Lo (2012) studied a single-stage robust portfolio problem where the weights of portfolios are fixed such that a generic objective function is optimized for the worst possible ranking of portfolios. Thus, in this application it is necessary to use integer parameters to formulate the ranking of portfolios. Feige et al. (2007) and Gupta et al. (2014) also studied several classical covering problems where in their uncertainty sets, integer parameters were used to choose a set of active clients in a graph. Moreover, in some cases, integer parameters are used to approximate nonconvex uncertainty sets. For instance, Siddiq (2013) and Chan et al. (2018) studied a robust facility location problem and discussed how nonconvex uncertainty sets can be approximated by discretization.

In this work, we assume that the uncertainty appears on the right-hand-side values and the corresponding technology matrix of recourse decision variables has a block-diagonal structure. The main contribution of our work is a novel reformulation exploiting the block-diagonal structure and three solution methods for a class of two-stage robust problems with an exponential number of scenarios given implicitly. This decomposition reduces the original two-stage problem to a single-stage problem. We then develop a Benders algorithm for the reformulated problem. We also develop a heuristic algorithm and combine it with the Benders algorithm to create a more effective hybrid Benders algorithm. Because the master problem and subproblem in the Benders algorithm are mixed-integer programs, it is computationally expensive to solve them to optimality. Hence, we propose novel stopping conditions for these mixed-integer programs and prove the convergence of the algorithm. We evaluate the computational performance of the proposed algorithms in a nurse planning and a two-echelon supply chain application.

We organize the remainder of this paper as follows. In Section 2, we provide a literature review on robust optimization with a focus of two-stage problems. In Section 3, we introduce the structure of the two-stage robust optimization problems studied in this paper and apply Dantzig–Wolfe decomposition to reformulate the original two-stage robust problem as a single-stage robust problem. In Section 4, we develop solution methods for the reformulated problem. In Section 5, we propose stopping conditions for the master problem and subproblem of the Benders algorithm. In Section 6, we show how to apply the proposed reformulation on a two-stage nurse planning problem and a two-echelon supply chain problem. We provide extensive computational results on these applications in Section 7. Finally, we give concluding remarks and future research directions in Section 8. Omitted proofs are provided in the electronic supplement.

2. Literature Review

In a single-stage robust optimization problem, constraints must be satisfied for all possible realizations of uncertain parameters. Therefore, by repeating constraints for different values of uncertain parameters, we can view a robust problem as a mathematical program with a large number of constraints. Depending on the structure of the uncertainty set, two techniques are usually applied to solve single-stage robust problems. The first approach is to iteratively generate violated constraints of the mathematical program explained above using a constraint generation algorithm (Fischetti and Monaci 2012, Bertsimas et al. 2016). In the second approach, the problem is reformulated as its deterministic robust counterpart and then solved directly. Soyster (1973) presented such a deterministic counterpart model for robust linear problems with box uncertainty sets. Ben-Tal and Nemirovski (1999) proposed a second-order cone program for uncertain linear programs with ellipsoidal uncertainty sets. They also showed that in the case of polyhedral uncertainty sets, the robust counterpart model is a linear program. Bertsimas and Sim (2004) showed that robust linear programs with Γ-cardinality uncertainty sets can be reformulated as deterministic linear programs.

Multistage robust problems are more complicated than single-stage robust problems and are generally intractable (Ben-Tal et al. 2004). There are two common solution approaches for these problems. Both approaches transform the multistage problem to a single-stage problem and then apply the solution methods of the single-stage robust problem. In the first approach, the recourse decisions are restricted to a function of uncertain parameters resulting in a single-stage robust problem. In this context, affine adaptability, also referred to as linear decision rules, assumes recourse decisions to be affine functions of uncertain parameters. This method is very popular and is applied in various areas such as supply chain management (Ben-Tal et al. 2005), inventory control (Ben-Tal et al. 2009), portfolio management (Fonseca and Rustem 2012), warehouse management (Ang et al. 2012), capacity management (Ouorou 2013), and network design (Poss and Raack 2013). Chen and Zhang (2009) introduced the extended affine adaptability by reparameterizing the primitive parameters and then applying the affine adaptability. Bertsimas et al. (2011) proposed a more accurate approximation of recourse decisions using polynomial adaptability. A drawback of the functional adaptability is its inability to handle problems with integer recourse decisions. Another approach is finite adaptability in which the uncertainty set is split into a number of smaller subsets, each with its own set of recourse decisions. The number of these subsets can be either fixed a priori or decided by the optimization model (Bertsimas and Caramanis 2010, Vayanos et al. 2011, Hanasusanto et al. 2015, Bertsimas and Dunning 2016, Postek and den Hertog 2016). An important advantage of the finite adaptability is that, in contrast to the functional adaptability approach, it easily handles problems with integer recourse variables.

There are many papers in the literature that have proposed Benders algorithms to solve two-stage robust optimization problems (Zheng et al. 2012, Bertsimas et al. 2013, Remli and Rekik 2013, Zhang et al. 2015). In these papers, assuming that the problem is set as a min-max-min problem, the authors have dualized the inner minimization to reformulate the problem to a min-max problem with bilinear terms in the objective function. Then, they have applied a Benders algorithm to solve the first-stage problem together with cuts generated from an outer approximation algorithm, which solves the maximization problem.

Column-and-constraint generation is another exact algorithm to solve a two-stage robust optimization problem (Zhao and Zeng 2012b; Zeng and Zhao 2013; Danandeh et al. 2014; Lee et al. 2014, 2015; Wang et al. 2014; Li et al. 2015, 2017; Chen et al. 2016; Ding et al. 2016; Wang et al. 2016; Neyshabouri and Berg 2017). The underlying idea of this approach is to make copies of recourse decision variables and also second-stage constraints for each possible realization of uncertain parameters which results in a large-scale mixed-integer programming model. As it is impossible to solve this model directly, a column-and-constraint generation algorithm is essential to generate critical uncertain scenarios and their corresponding recourse decision variables and second-stage constraints. The tricky part of this approach is the reformulation of the max-min subproblem to a max problem using Karush-Kuhn-Tucker conditions. The reformulated subproblem includes bilinear terms in constraints, which are linearized later by introducing a set of binary variables and adding big-M constraints.

The reformulation approach proposed in this paper is different from the ones used in the aforementioned Benders and column-and-constraint generation algorithms, as it does not result in any bilinear term in our models. Therefore, our solution methodology does not require an outer approximation algorithm (Bertsimas et al. 2013) or any linearization by introducing extra binary variables and big-M constraints (Zeng and Zhao 2013). Moreover, our modeling approach is capable of handling second-stage integer variables, whereas these algorithms work only on problems with continuous recourse variables. To handle second-stage integer variables, Zhao and Zeng (2012a) extended the original column-and-constraint generation algorithm to a trilevel algorithm. However, the extended algorithm only works on special problems satisfying three restrictive assumptions: (1) including at least one continuous recourse variable, (2) holding the *extended relative complete recourse* property for recourse problem when the second-stage integer variables are ignored, and (3) satisfying the *quasiconvex* property for the inner max-min problem. The first and third conditions are not satisfied in applications studied in this paper.

The reformulation approach that we propose is inspired from the one proposed in Siddiq (2013) that presented a reformulation for a specific facility location problem. The advantage of our reformulation is that it is more general and applicable to any two-stage robust problem with block-diagonal structure in the technology matrix of recourse decision variables. Here, we emphasize that the block-diagonal structure of uncertainty sets addressed in Ben-Tal et al. (2006) and Ben-Tal and Nemirovski (2002) is different from the block-diagonal structure in the technology matrix of recourse decision variables considered in this work. Moreover, the reformulation proposed in this work is completely different from the reformulation proposed by Zhang (2018) that provides an augmented Lagrangian lower and upper bound for a two-stage robust optimization problem with objective uncertainty.

3. Model and Reformulation

We study a class of two-stage robust optimization problems with the following structure:

$$\text{(P1)} \qquad \min_{x\in\mathcal{X}}\left(c_1^\top x + \max_{u\in\mathcal{U}}\left(\min_{y\in\mathcal{Y}(x,u)} c_2^\top y\right)\right). \qquad (1)$$

In the above formulation, x and y are the vectors of decision variables in the first stage and the second stage, respectively; u is the vector of uncertain parameters that are restricted to the uncertainty set $\mathcal{U}$; and c_1 and c_2 are given cost vectors. In the second-stage problem, we have $\mathcal{Y}(x,u)=\{y\in\tilde{\mathcal{Y}}\,|\,Cy\le b-Ax-Bu\}$, where A, B, and C are known matrices with appropriate dimensions, and b denotes the known vector of right-hand-side values. Also $x\in\mathcal{X}$ and $y\in\tilde{\mathcal{Y}}$ represent the integrality and bound constraints that we may have for variables in the first and second stages, respectively. Objective (1) minimizes the sum of the first- and second-stage costs. In this model, $x\in\mathcal{X}$ must be selected such that for all realizations $u\in\mathcal{U}$ there is at least one $y\in\mathcal{Y}(x,u)$. We assume that $\mathcal{U}$ is a finite uncertainty set. This assumption is necessary for the convergence proofs of the proposed Benders algorithms discussed in Section 5. The first- and second-stage variables can be continuous, integer, or mixed, but without loss of generality, all uncertain parameters are supposed to be integer. In fact, our method generalizes to finite set of fractional parameters (as shown in Section EC.14 of the electronic companion), and for the sake of clarity, we present the overall approaches over integer parameters. We also assume that C is a block-diagonal matrix. The main focus of this research is to exploit this block-diagonal structure and develop algorithms to solve the reformulated problem efficiently.

In the remainder of this section, we propose a reformulation of model (P1) and use it to develop solution methods in Section 4. For this, we need the following additional notation, used throughout the rest of the paper:

$\mathcal{K}$: The index set of blocks in matrix C.
C_k: The kth block in matrix C.
Row_k: The number of rows in block C_k.
Col_k: The number of columns in block C_k.
y_k: The subset of variables y involved in block C_k.
$\mathcal{Y}_k$: The set of integrality and bound constraints corresponding to variables y_k.
c_{2k}: The subset of c_2 corresponding to variables y_k.
b_k: The right-hand-side values in front of block C_k.
A_k: The rows in matrix A in front of block C_k.
B_k: The rows in matrix B in front of block C_k.

With respect to the block-diagonal structure of matrix C, we can rewrite constraint $Cy\le b-Ax-Bu$ included within the second-stage feasible space $\mathcal{Y}(x,u)$ as follows:

$$C_k y_k \le b_k - A_k x - B_k u \quad k\in\mathcal{K}. \tag{2}$$

Furthermore, we define some notation related to $B_k u$ as follows:

$\mathcal{S}'_k$: The set of all realizations for $B_k u$; that is, $\mathcal{S}'_k=\{v\in\mathbb{R}^{Row_k}\,|\,v=B_k u, u\in\mathcal{U}\}$.
$\mathcal{S}_k$: The index set of $\mathcal{S}'_k$; that is, $\mathcal{S}_k=\{1,2,\dots,|\mathcal{S}'_k|\}$.
e_{ks}: The sth member of $\mathcal{S}'_k$ (defined for $s\in\mathcal{S}_k$).
w_{ks}: A binary variable that takes 1 if $B_k u$ is equal to e_{ks} and 0 otherwise.

Using all the above notation, we reformulate model (P1) as

$$\text{(P2)}\quad \min_{x\in\mathcal{X}}\left(c_1^\top x+\max_{(u,w)\in(\mathcal{U},\mathcal{W})}\left(\min_{y\in\mathcal{Y}(x,w)}\sum_{k\in\mathcal{K}}c_{2k}^\top y_k\right)\right), \tag{3}$$

$$(\mathcal{U},\mathcal{W})=\Big\{(u,w)\,|\,u\in\mathcal{U}, \tag{4}$$

$$B_k u=\sum_{s\in\mathcal{S}_k}e_{ks}w_{ks},\qquad k\in\mathcal{K}, \tag{5}$$

$$\sum_{s\in S_k}w_{ks}=1,\qquad k\in\mathcal{K}, \tag{6}$$

$$w_{ks}\in\{0,1\},\qquad k\in\mathcal{K}\; s\in\mathcal{S}_k\Big\}, \tag{7}$$

$$\mathcal{Y}(x,w)=\Big\{y\,|\,y_k\in\mathcal{Y}_k,\qquad k\in\mathcal{K}, \tag{8}$$

$$C_k y_k\le b_k-A_k x-\sum_{s\in\mathcal{S}_k}e_{ks}w_{ks},\quad k\in\mathcal{K}\Big\}. \tag{9}$$

In the following, we introduce a new model that is equivalent to model (P2) as we will show later in Theorem 1 and Corollary 1. To introduce model (P3), for each $k\in\mathcal{K}$, we make $|\mathcal{S}_k|$ copies of variables $y_k\in\mathbb{R}^{Col_k}$ and define variables $y'_{ks}\in\mathbb{R}^{Col_k}(s\in\mathcal{S}_k)$. Model (P3) is given by (10)–(12):

$$\text{(P3)}\quad \min_{x\in\mathcal{X}}\left(c_1^\top x+\max_{(u,w)\in(\mathcal{U},\mathcal{W})}\left(\min_{y'\in\mathcal{Y}'(x)}\sum_{k\in\mathcal{K}}\sum_{s\in\mathcal{S}_k}c_{2k}^\top y'_{ks}w_{ks}\right)\right), \tag{10}$$

$$\mathcal{Y}'(x)=\Big\{y'\,|\,y'_{ks}\in\mathcal{Y}_k,\qquad k\in\mathcal{K}\; s\in\mathcal{S}_k, \tag{11}$$

$$C_k y_{ks}\le b_k-A_k x-e_{ks},\qquad k\in\mathcal{K}\; s\in\mathcal{S}_k\Big\}. \tag{12}$$

The structure of model (P3) is such that, if w_{ks} takes 1, y'_{ks} is equal to the optimal solution of y_k in model (P2). Indeed, by introducing Constraint (12), we have made $|\mathcal{S}_k|$ copies of Constraint (9) to compute the values of y'_{ks} independently. Moreover, to ensure that the optimal objective values of models (P2) and (P3) are the same, $c_{2k}^\top y'_{ks}$ in (10) is multiplied by w_{ks}.

Theorem 1. *Suppose that model* (P2) *is feasible. Then,*

a. *$\hat{x}$ is a first-stage feasible solution of model* (P2) *if and only if it is a first-stage feasible solution of model* (P3),

b. *the objective values of models* (P2) *and* (P3) *for the first-stage solution $\hat{x}$ are the same if* $\max_{u,w}$ *and* $\min_{y'}$ *are solved optimally, and*

c. *for this first-stage solution, the optimal values of variables y'_{ks} in model* (P3) *represent the second-stage optimal policies in model* (P2).

The following corollary states the relations between models (P2) and (P3).

Corollary 1. *Models* (P2) *and* (P3) *are equivalent; that is, either*

1. *both models are unbounded, or*
2. *both models are infeasible, or*
3. *both models are feasible and bounded with the same optimal objective value and the same optimal solution for the first-stage variables; in this case, the optimal solution of variables* y'_{ks} *in model* (P3) *represents the optimal policies for variables* y_k *in model* (P2).

Proof. Theorem 1 directly results in cases (1) and (3). To prove case (2), we note that with respect to Theorem 1 for any feasible solution in model (P2), there is an equivalent feasible solution in model (P3). Therefore, model (P3) is infeasible if and only if model (P2) is infeasible.

The next theorem shows that model (P3) can be reduced to a single-stage problem.

Theorem 2. *In model* (P3), *the objective function* $\max_{u,w}\cdot(\min_{y'}(.))$ *can be replaced by* $\min_{y'}(\max_{u,w}(.))$.

Therefore, we can rewrite model (P3) as

$$\text{(P4)}\quad \min_{(x,y')\in(\mathscr{X},\mathscr{Y}')}\left(c_1^{\top}x+\max_{(u,w)\in(\mathscr{U},\mathscr{W})}\left(\sum_{k\in\mathscr{K}}\sum_{s\in\mathscr{S}_k}c_{2k}^{\top}y'_{ks}w_{ks}\right)\right). \tag{13}$$

In fact, the reformulation presented in this section shows that we can transform the two-stage robust problem (P1) to a single-stage robust problem. In the above model, we have $(\mathscr{X},\mathscr{Y}')=\{(x,y')\,|\,x\in\mathscr{X},y'\in\mathscr{Y}'(x)\}$. We use the latter model to present our solution methods in Section 4.

4. Solution Methods

In this section we propose three solution methods for model (P4). We present a Benders algorithm that iterates between a master problem and a subproblem to tighten the optimality gap. We also propose a heuristic that dualizes the linear programming relaxation of the inner max problem in model (P4). Then it iteratively generates cuts to shape the convex hull of the uncertainty set. We also present a hybrid Benders algorithm that applies the heuristic within the framework of the Benders algorithm.

4.1. Benders Algorithm

In our Benders algorithm, valid lower and upper bounds are obtained by solving the master problem and the subproblem, respectively. The algorithm iterates between these problems until the bounds converge. In the following, we present the master problem and subproblem. Then we explain the framework of the Benders algorithm.

Suppose that m scenarios $(\hat{u}^j,\hat{w}^j)\in(\mathscr{U},\mathscr{W}), j=1,2,\ldots,m$, are already generated by solving the subproblem. We define the master problem of the Benders algorithm as

$$\text{(MP)}\qquad \min_{(x,y')\in(\mathscr{X},\mathscr{Y}'),\theta}\ \theta \tag{14}$$

$$\theta\geq c_1^{T}x+\sum_{k\in\mathscr{K}}\sum_{s\in\mathscr{S}_k}c_{2k}^{\top}y'_{ks}\hat{w}_{ks}^{j},\qquad j=1,2,\ldots,m. \tag{15}$$

Theorem 3. *The optimal objective value of model* (MP) *is a valid lower bound for model* (P4).

For a feasible solution $(\hat{x}',\hat{y}')\in(\mathscr{X},\mathscr{Y}')$, a valid upper bound is obtained by solving the inner max problem in model (P4). We refer to the following problem as the subproblem of the Benders algorithm:

$$\text{(SP)}\qquad \max_{(u,w)\in(\mathscr{U},\mathscr{W})}\left(c_1^{\top}\hat{x}+\sum_{k\in\mathscr{K}}\sum_{s\in\mathscr{S}_k}c_{2k}^{\top}\hat{y}'_{ks}w_{ks}\right). \tag{16}$$

Algorithm 1 provides the pseudocode of the Benders algorithm. In this algorithm, *UB* and *LB* respectively denote the best upper and lower bounds found during the algorithm. In line 3, we obtain an initial solution $(\hat{x},\hat{y}')$ by a heuristic algorithm that is explained at the end of Section 4.2. In line 6, the algorithm sets *UB* equal to the optimal objective value of the subproblem if it is less than the current *UB*. The stopping conditions of the Benders algorithm are then checked in line 9, where $\delta_{acc}^{Benders}$ and *AlgTimeLimit* respectively denote the maximum acceptable optimality gap and the available computational time.

Algorithm 1 (Benders Algorithm)
```
1: Input parameters: $\delta_{acc}^{Benders}$ and $AlgTimeLimit$.
2: Set $UB$=$\infty$, $LB$=-$\infty$, and $m = 0$.
3: Find an initial solution $(\hat{x},\hat{y}')$ by a heuristic.
4: repeat
5:     Modify the objective function of subproblem
         (SP) using $(\hat{x},\hat{y}')$.
6:     Solve the subproblem and update $UB$ if it is
         necessary.
7:     Add a new optimality cut (15) to the master
         problem and set $m = m + 1$.
8:     Solve the master problem and update $LB$.
9: until $(100(UB - LB)/LB \leq \delta_{acc}^{Benders}$ or time limit
       $AlgTimeLimit$ is reached)
```

4.2. Heuristic Algorithm

In this section we present a heuristic algorithm. This algorithm dualizes the linear programming relaxation of the inner max problem in model (P4) to transform the min-max problem to a single minimization problem. Let us assume that constraints forming the convex hull of the inner max problem in model (P4) are

$$Du+Ew\leq b_2. \tag{17}$$

In Constraint (17), u and w are the vectors of uncertainty variables, D and E are technology matrices with appropriate dimensions, and b_2 is the known vector of right-hand-side values. Note that D, E, and b_2 are independent from the values of $(\hat{x}, \hat{y}')$ that are fixed in the outer min problem in model (P4). This is because the solution space of uncertainty variables does not depend on the variables in the outer min problem. If we have Constraints (17), we can replace constraints $(u, w) \in (\mathcal{U}, \mathcal{W})$ with them. In this case, the inner max problem is a linear programming model for fixed values of $(\hat{x}, \hat{y}')$ in the outer min problem. Therefore by dualizing the inner max problem, we obtain the following model:

$$\text{(D-P4)} \quad \min_{\hat{x},\hat{y}',\gamma} (c_1^\top x + b_2^\top \gamma) \tag{18}$$

$$(x, y') \in (\mathcal{X}, \mathcal{Y}'), \tag{19}$$

$$E_{ks}^\top \gamma \geq c_{2k}^\top y'_{ks}, \qquad k \in \mathcal{K}, s \in \mathcal{S}_k, \tag{20}$$

$$D^\top \gamma = 0, \tag{21}$$

$$\gamma \geq 0. \tag{22}$$

In model (D-P4), γ is the vector of dual variables for Constraint (17), and E_{ks} is the column in E that includes coefficients of variable w_{ks}. We can observe that the min-max problem in model (P4) reduces to a single min problem and can be solved directly as a mixed-integer programming model. In the literature, the above dualization technique is prevalent to simplify single-stage robust problems where the inner max problem is a linear programming model. However, in our model, the inner max problem is a mixed-integer program and Constraints (17) forming the convex hull of the uncertainty set are unknown. In the following, we present a heuristic algorithm that relaxes the integrality constraints of the variables in the inner max problem of model (P4). Then by iteratively generating cuts, it attempts to shape the solution space of the relaxed inner max problem into its convex hull before the relaxation. To present this heuristic, we first need to define models (P5) and (P6):

$$\text{(P5)} \quad \min_{(x,y')\in(\mathcal{X},\mathcal{Y}')} \left(c_1^\top x + \max_{(u,w)\in(\mathcal{U},\mathcal{W})'} \left(\sum_{k\in\mathcal{K}} \sum_{s\in\mathcal{S}_k} c_{2k}^\top y'_{ks} w_{ks} \right) \right) \tag{23}$$

$$(\mathcal{U}, \mathcal{W})' = \{(u, w) \in (\mathcal{U}, \mathcal{W}) \mid \tag{24}$$

$$Fu + Gw \leq b_3\}. \tag{25}$$

In model (P5), Constraint (25) is the set of valid cuts that the heuristic algorithm generates iteratively. This constraint set is empty at the beginning of the algorithm. In this constraint, F and G are technology matrices with appropriate dimensions and b_3 the known vector of right-hand-side values. We obtain the following model (P6) by relaxing the integrality constraints of variables u and w in model (P5) and then dualizing the inner max problem:

$$\text{(P6)} \quad \min_{x,y',\pi,\lambda,\alpha,\beta} \left(c_1^\top x + b_3^\top \pi + b_4^\top \lambda + \sum_{k\in\mathcal{K}} \alpha_k \right) \tag{26}$$

$$(x, y') \in (\mathcal{X}, \mathcal{Y}'), \tag{27}$$

$$B_k^\top \beta_k + H^\top \lambda + F^\top \pi = 0, \tag{28}$$

$$\alpha_k + e_{ks}^\top \beta_k + G_{ks}^\top \pi \geq c_{2k}^\top y'_{ks}, \quad k \in \mathcal{K}, s \in \mathcal{S}_k. \tag{29}$$

In model (P6), β_k and π are vectors of dual variables for Constraints (5) and (25), respectively; α_k is the dual variable of Constraint (6) defined for each $k \in \mathcal{K}$; and G_{ks} is the column in G that includes coefficients of variable w_{ks}. To write the dual of the inner max problem in model (P5), we have supposed that linear constraints hidden in uncertainty set $\mathcal{U}$ in Constraint (4) are represented by $Hu \leq b_4$. In model (P6), λ denotes the vector of dual variables for $Hu \leq b_4$.

Algorithm 2 provides the pseudocode of our heuristic algorithm. In line 2, we suppose that no instance of Constraint (25) is available at the beginning of the algorithm and that F, G, and b_3 are empty. The term *Ite* is the iteration counter of the loop starting in line 3. In line 5, we solve model (P6) to obtain a feasible solution for $(\hat{x}, \hat{y}')$. For a fixed solution $(\hat{x}, \hat{y}')$ in model (P5), the inner max problem is an integer program, and we denote it by InnerMax$(\hat{x}, \hat{y}')$. In line 7, we call Procedure 1. In each iteration of this procedure, we solve the linear programming relaxation of InnerMax$(\hat{x}, \hat{y}')$ and obtain a new fractional scenario $(\hat{u}, \hat{w})$. Then this procedure generates a number of valid cuts to remove this fractional scenario. This procedure continues until it cannot detect any other violated cut or time limit *AlgTimeLimit* is reached. We use an integer programming solver to perform Procedure 1 and let it generate valid cuts as explained above. In calling the integer programming solver, we limit the maximum number of nodes to be explored in the branch-and-bound tree to one. In line 8 in Algorithm 2, we extract the cuts generated by the integer programming solver and update F, G, and b_3 in models (P5) and (P6). In line 9, the algorithm checks stopping criteria. One of these stopping criteria checks whether the percentage of the objective value improvement obtained in the current iteration is less than or equal to parameter δ^H_{acc}.

Algorithm 2 (Heuristic Algorithm)

```
1: Input parameters: LocalTimeLimit, AlgTimeLimit,
       and $\delta^H_{acc}$.
2: Set $Ite = 0$ and empty $F$, $G$, and $b_3$ in models (P5)
       and (P6).
3: **repeat**
4:     $Ite{+}{+}$.
```

```
5:     Solve model (P6) with time limit $LocalTimeLimit$
           to obtain a feasible solution $(\hat{x}, \hat{y}')$.
6:     Set $Obj_{Ite}$ equal to the objective value of model
           (P6).
7:     Apply Procedure 1 to generate several cuts (25).
8:     Extract the generated cuts and update $F$, $G$, and
           $b_3$ in models (P5) and (P6).
9:  until (No cut is generated in line 7 in this iteration
           or time limit $AlgTimeLimit$ is reached or
           $(100(Obj_{Ite} - Obj_{Ite-1})/Obj_{Ite-1} \leq \delta^H_{acc})$)
```

Procedure 1 (Cut Generation for the Proposed Heuristic Algorithm)

```
1: repeat
2:     Solve the linear programming relaxation of
           InnerMax$(\hat{x}, \hat{y}')$ to obtain $(\hat{u}, \hat{w})$.
3:     Detect some valid cuts to remove the fractional
           solution $(\hat{u}, \hat{w})$.
4:     Update $F$, $G$, and $b_3$ in InnerMax$(\hat{x}, \hat{y}')$.
5: until (No valid cut is generated or time limit
           $AlgTimeLimit$ is reached)
```

The proposed heuristic algorithm does not necessarily find the optimal solution. Appendix EC.4 in the electronic companion presents an example to demonstrate that the heuristic algorithm does not guarantee optimality. In the Benders algorithm presented in Algorithm 1, in line 3 we solve model (P6) with empty F, G, and b_3 to find an initial solution $(\hat{x}, \hat{y}')$.

4.3. Hybrid Benders Algorithm

In this section, we combine the Benders and heuristic algorithms to create a more efficient algorithm. In this hybrid algorithm, the Benders algorithm guarantees the convergence of the algorithm. The proposed heuristic algorithm improves the overall efficiency by generating valid cuts for the inner max problem and also by finding better solutions $(\hat{x}, \hat{y}')$ by solving model (P6).

Algorithm 3 (Hybrid Benders Algorithm)

```
1:  Input parameters: $WarmupTimeLimit$,
        $AlgTimeLimit$, $LocalHeuristicTimeLimit$,
        $EvaTimeLimit$, $\delta^H_{acc}$, and $\delta^{Benders}_{acc}$.
2:  Set $UB = \infty$, $LB = -\infty$, and $m = 0$, and empty $F$, $G$,
        and $b_3$ in models (P5) and (P6).
3:  Find an initial solution $(\hat{x}, \hat{y}')$ by a heuristic.
4:  repeat
5:      Modify the objective function of subproblem
            (SP) using solution $(\hat{x}, \hat{y}')$.
6:      Solve the subproblem and update $UB$ if it is
            necessary.
7:      Add a new optimality cut to the master prob-
            lem and set $m = m + 1$.
8:      Solve the master problem, update $LB$, and save
            $(\hat{x}, \hat{y}')$ in the solution pool.
9:      if ($WarmupTimeLimit$ is reached) then
10:         Set $Ite = 0$.
11:         repeat
12:             $Ite++$.
13:             Apply Procedure 1 using solution $(\hat{x}, \hat{y}')$ to
                    generate several cuts (25).
14:             Extract the generated cuts and update $F$, $G$,
                    and $b_3$ in models (P5) and (P6).
15:             Solve model (P6) to obtain a solution $(\hat{x}, \hat{y}')$
                    and save it in the solution pool.
16:             Set $Obj_{Ite}$ to the objective value of model
                    (P6).
17:         until (No cut is generated in line 14 or time
                limit $LocalHeuristicTimeLimit$ is reached or
                $100(Obj_{Ite} - Obj_{Ite-1})/Obj_{Ite-1} \leq \delta^H_{acc}$)
18:         Choose the best solution $(\hat{x}, \hat{y}')$ from the
                solution pool.
19:     end if
20: until $(100(UB - LB)/LB \leq \delta^{Benders}_{acc}$ or $AlgTimeLimit$
        is reached)
```

Algorithm 3 provides the pseudocode of the hybrid Benders algorithm. In line 3 of this algorithm, we obtain an initial solution $(\hat{x}, \hat{y}')$ by solving model (P6), whereas F, G, and b_3 are ignored. Lines 4–8 together with line 20 are the same as the main loop of the Benders algorithm given in Algorithm 1. The algorithm finds a scenario by solving the subproblem in line 6. We then add a new optimality cut to the master problem and solve it to find a new solution $(\hat{x}, \hat{y}')$. Then if time limit *WarmupTimeLimit* is already reached, the algorithm enters an inner loop starting in line 11. This loop is taken from the heuristic algorithm and improves the current solution $(\hat{x}, \hat{y}')$ by iteratively generating cuts (25) in line 13 and then solving model (P6) in line 15. Then we check the stopping criteria of the heuristic algorithm in line 17. To check whether time limit *LocalHeuristicTimeLimit* is reached, the algorithm tracks the time from the start of the inner loop in line 11.

After leaving the inner loop in line 17 and before starting a new iteration of the algorithm, we have to decide on the new solution $(\hat{x}, \hat{y}')$ to modify the objective function of the subproblem in line 5. Therefore, during the algorithm we save all generated solutions $(\hat{x}, \hat{y}')$ in a solution pool. Then in line 18 among all solutions in the pool, we choose the one with the lowest worst objective value against all generated scenarios as the current solution $(\hat{x}, \hat{y}')$. In line 9 of Algorithm 3, we have a time limit *WarmupTimeLimit* to prevent from entering the inner loop in line 11 before this time limit. This is because, in small instances, the Benders algorithm converges very quickly without any need of the heuristic algorithm.

There are generally two advantages for combining the Benders and heuristic algorithms. First, by generating cuts (25) in the heuristic algorithm and including them in the subproblem, we hope that the algorithm can solve the subproblem faster in next iterations. Also, it is possible to improve the best solution $(\hat{x}, \hat{y}')$ by solving model (P6) in line 17 in Algorithm 3.

5. Stopping Conditions

The main shortcoming of the Benders and hybrid Benders algorithms is that their master problem and subproblem are mixed-integer programs (MIPs), and therefore it would be very time consuming to optimally solve them in all iterations of the algorithms. In the following, we present novel stopping conditions for these MIPs. Before explaining these conditions, we define "ε-dominant incumbents" for the master problem and subproblem as follows: For a constant $\varepsilon > 0$, an ε-dominant incumbent of the master problem is a feasible solution in the master problem with an objective value that is less than the lower bound of the recent subproblem by a margin of ε. Similarly, an ε-dominant incumbent of the subproblem is a feasible solution in the subproblem with an objective value that is at least ε higher than the upper bound of the recent master problem. The stopping conditions are presented as follows:

Stopping Condition for the Master Problem (Subproblem): The mixed-integer program terminates when the optimal solution is found or at least $Time_{MP}$ seconds ($Time_{SP}$ seconds) is passed from the moment that the first ε-dominant incumbent of the master problem (subproblem) is found.

In Table 1, we present a numerical example with $\varepsilon = 5$ to explain this stopping condition for both the master problem and the subproblem. In this table, the results of the master problem and of the subproblem are presented in separate columns. We report the lower and upper bounds of the related mixed-integer programs. Because these MIPs are not optimally solved, there are gaps between the lower and upper bounds. The "Order" columns also give the order in which these MIPs are solved. In this example, in iterations 1–4, the upper bound of the master problem is at least $\varepsilon = 5$ units less than the lower bound of the previous subproblem. Moreover, in iterations 2–4, the lower bound of the subproblem is at least $\varepsilon = 5$ units higher than the upper bound of the master problem in the previous iteration. In iteration 5, when solving the subproblem, we observe that the lower bound does not increase to $\varepsilon = 5$ units higher than the upper bound of the master problem in iteration 4. Therefore, the stopping condition is not met, and the subproblem has to be solved optimally. Similarly, in the same iteration, when we solve the master problem, the upper bound does not decrease to $\varepsilon = 5$ units less than the lower bound of the subproblem in that iteration. Thus, the stopping condition is not satisfied, and the master problem has to be solved to optimality.

Table 1. A Numerical Example to Explain the Stopping Conditions of MIPs in the Benders Algorithm

	Subproblem			Master problem		
Iteration	Order	*LB*	*UB*	Order	*LB*	*UB*
1	1	**120**	470	2	45	**110**
2	3	**340**	650	4	30	**300**
3	5	**320**	360	6	155	**165**
4	7	**170**	185	8	157	**160**
5	9	**160**	160	10	160	**160**

The upper bound in the master problem and the lower bound in the subproblem correspond to feasible solutions of MIPs. Therefore, the stopping condition for the subproblem means that the subproblem terminates before reaching the optimality if we find a critical uncertain scenario. We refer to a scenario as a critical one if by adding its corresponding cut (15) to the master problem, the objective value of solution $(\hat{x}, \hat{y}')$ found in the previous iteration increases by at least ε units.

Lower bounds in the master problem and the upper bounds in the subproblem are valid lower and upper bounds of the original robust problem, respectively. Therefore, we can impose the following constraints based on the best obtained lower and upper bounds as the Benders algorithm proceeds:

$$\theta \geq LB, \tag{30}$$

$$\theta \leq UB. \tag{31}$$

Constraint (31) is valid because the optimal objective value of the subproblem is an upper bound on the optimal objective value of the original robust problem. As will be discussed at the end of the section, Constraints (30) and (31) are vital for proving the convergence of the Benders algorithm with stopping conditions for the master problem and subproblem.

When we apply the stopping conditions, most of the time the subproblem is not solved optimally. Therefore, the best upper bound obtained by Algorithms 1 and 3 is poor if the algorithm times out. In this case, we call Procedure 2 at the end of Algorithms 1 and 3 to improve the quality of the best upper bound. This procedure sorts all solutions $(\hat{x}, \hat{y}')$ found by the master problem based on their upper bounds. The upper bound of each solution $(\hat{x}, \hat{y}')$ is the upper bound of its corresponding subproblem obtained in line 6 of Algorithms 1 and 3. Procedure 2 evaluates these solutions separately by solving the subproblem without any stopping condition. When solving a subproblem, if we obtain a feasible solution with an objective value higher than the best upper bound UB, the subproblem terminates, and Procedure 2 evaluates the next solution $(\hat{x}, \hat{y}')$ in the sorted list. This is because in this case, another solution with a better upper bound is already known. We consider a time limit *EvaTimeLimit* for this procedure.

Procedure 2 (Evaluation of the Generated Solutions $(\hat{x}, \hat{y}')$)

```
1: Input parameters: $EvaTimeLimit$ and $\delta_{acc}^{Benders}$.
2: if $(100(UB - LB)/LB > \delta_{acc}^{Benders})$ then
3:    Sort solutions $(\hat{x}, \hat{y}')$ in the solution pool.
4:    for $(i = 1$ to $NumberSolutions)$ do
5:       Solve the subproblem for $i$th solution $(\hat{x}, \hat{y}')$
            and update $UB$ if necessary.
6:       if $(100(UB - LB)/LB \le \delta_{acc}^{Benders}$ or $EvaTimeLimit$
            is reached) then
7:          break;
8:       end if
9:    end for
10: end if
```

In the following, we discuss the convergence of the Benders algorithm with and without stopping conditions for the subproblem and master problem. We use the following notation to present the next lemmas and theorems:

$\mathcal{W}$: The set of vectors w for which there is $u \in \mathcal{U}$ such that $(u, w) \in (\mathcal{U}, \mathcal{W})$.
n: The number of scenarios in $(\mathcal{U}, \mathcal{W})$.
n': The number of unique vectors w that the algorithm visits in the subproblem before it converges.
n'': The number of times that the algorithm visits an already encountered vector w before it converges.
ε: A positive constant used in stopping conditions of the master problem and subproblem.
Opt: The optimal objective value of the original robust problem.
O_i^{SP}: The optimal objective value of the subproblem in iteration i.
U_i^{MP}: The upper bound of the master problem in iteration i.
$f(j)$: The iteration in which for the jth times the algorithm generates a scenario with a new vector w in the subproblem.
$g(i)$: The iteration in which for the ith times the algorithm revisits any of the generated vectors w in the subproblem.
I_i: An indicator that is equal to 1 if in iteration i the algorithm generates a scenario with a repeated vector w and 0 otherwise.

Observation 1. The Benders algorithm without stopping conditions for the master problem and subproblem converges in at most $|\mathcal{W}| + 1 \le n + 1$ iterations.

In the following, we present Lemmas 1–4 where Lemma 1 is a basis in the proofs of other lemmas and Lemmas 2–4 are used in the proof of Theorem 4.

Lemma 1. *In the Benders algorithm with stopping conditions for the master problem and subproblem, if the algorithm finds a scenario with a repeated vector w in the subproblem of iteration i, then it is the optimal solution of the subproblem, and the optimal objective value of the subproblem is equal to the upper bound of the recent master problem in iteration* $i - 1$ *(i.e.,* $U_{i-1}^{MP} = O_i^{SP}$*).*

Lemma 2. *In the Benders algorithm with the stopping conditions for the master problem and subproblem, if the algorithm finds a scenario with a repeated vector w in the subproblem of iteration i and* $O_i^{SP} - Opt > \varepsilon$ *holds, then in at most* $k = \lfloor (O_i^{SP} - Opt)/\varepsilon \rfloor$ *iterations, either the algorithm finds a scenario with a new vector w or* $O_{i+k}^{SP} - Opt \le \varepsilon$ *holds.*

Lemma 3. *In the Benders algorithm with the stopping conditions for the master problem and subproblem, if the algorithm finds a scenario with a repeated vector w in the subproblem of iteration i and* $O_i^{SP} - Opt \le \varepsilon$ *holds, then in the next iteration, either the Benders algorithm converges or a scenario with a new vector w is found.*

Lemma 4. *In the Benders algorithm with the stopping conditions, relation* $O_{g(i_1)}^{SP} \ge O_{g(i_2)}^{SP}$ *holds for any integer numbers* i_1 *and* i_2 *satisfying* $1 \le i_1 < i_2 \le n''$.

Theorem 4. *The Benders algorithm with the stopping conditions converges in at most* $\sum_{j=1}^{n'} (1 + (\lfloor (O_{f(j)+1}^{SP} - Opt)/\varepsilon \rfloor + 1) I_{f(j)+1})$ *iterations that is bounded above by* $|\mathcal{W}|(\lfloor (O_{g(1)}^{SP} - Opt)/\varepsilon \rfloor + 2)$ *iterations.*

$O_{g(1)}^{SP}$ is bounded as a result of the boundedness of the feasible area in subproblem (SP). Therefore, Theorem 4 proves the convergence of the Benders algorithm in a finite number of iterations.

6. Applications

In this section, we demonstrate how to apply the proposed reformulation on a nurse planning and a two-echelon supply chain problem.

6.1. Two-Stage Nurse Planning Problem

In a two-stage nurse planning problem, we plan wards' nurses of a hospital for a medium term. The daily workloads of nurses depend on the number of patients brought from operating rooms to wards. Patients are already scheduled in operating rooms over the planning horizon. Before transferring patients from operating rooms to wards, they may stay in intensive care units (ICUs) for several days. The lengths of stays in ICUs and wards are uncertain and discrete. For each patient, a number of local scenarios about the lengths of stays in ICUs and wards is available.

In the first stage of this problem, we assign a number of nurses to wards over the planning horizon. In the second stage, if the nurses' workload on a day is more than the service capacity of nurses assigned to that day, some extra nurses are hired. Nurses hired in the second stage are paid more than those hired in the first stage. Nurse staffing based on the workloads of patients transferred from operating rooms to wards is studied in the literature (Beliën and Demeulemeester 2008). The problem is formulated as follows.

Parameters

c_1: The daily cost of a nurse hired in the first stage.
c_2: The daily cost of a nurse hired in the second stage.
M_d: The maximum number of nurses available for hiring on day d in the second stage.
δ: The amount of service time provided by a first- or second-stage nurses per day (in hours).
ρ: The average of required service time for each patient per day (in hours).
l_{tp}^{ICU}: The length of stay in ICUs for patient t in local scenario $p \in \mathcal{P}_t$.
l_{tp}^{Ward}: The length of stay in wards for patient t in local scenario $p \in \mathcal{P}_t$.
d'_t: The surgery day for patient t.

Sets

$\mathcal{D}$: The set of days in the planning horizon.
$\mathcal{T}$: The set of patients already scheduled in operating rooms over the planning horizon.
$\mathcal{T}_d$: The set of patients scheduled on day d.
$\mathcal{P}_t$: The set of local scenarios for patient t. Each local scenario gives information on the lengths of stays in ICUs and wards.
$\mathcal{P}_{td}$: The subset of local scenarios in $\mathcal{P}_t$ where patient t is in wards on day d; that is, $\mathcal{P}_{td} = \{p \in \mathcal{P}_t : d'_t + l_{tp}^{ICU} \le d, d'_t + l_{tp}^{ICU} + l_{tp}^{Ward} > d\}$.

Variables

x_d: The number of nurses assigned to day d in the first stage.
u_{tp}: 1 if patient t follows local scenario p after its surgery and 0 otherwise. (Uncertainty variable)
y_d: The number of nurses hired on day d in the second stage.

$$\min_{x\in\mathcal{X}}\left(\sum_{d\in\mathcal{D}} c_1x_d + \max_{u\in\mathcal{U}}\left(\min_{y\in\mathcal{Y}(x,u)} \sum_{d\in\mathcal{D}}(c_2y_d)\right)\right) \tag{32}$$

$$\mathcal{X} = \{x \mid x_d \ge 0, \text{integer}\}, \tag{33}$$

$$\mathcal{U} = \left\{u \;\middle|\; \sum_{p\in\mathcal{P}_t} u_{tp} = 1, \quad t \in \mathcal{T}, \right. \tag{34}$$

$$\left. u_{tp} \in \{0,1\}, \quad t \in \mathcal{T},\ p \in \mathcal{P}_t \right\}, \tag{35}$$

$$\mathcal{Y}(x,u) = \left\{y \mid \delta y_d \ge \rho \sum_{t\in\mathcal{T}} \sum_{p\in\mathcal{P}_{td}} u_{tp} - \delta x_d \quad d \in \mathcal{D}, \right. \tag{36}$$

$$\left. 0 \le y_d \le M_d, \text{integer}, \quad d \in \mathcal{D} \right\}. \tag{37}$$

Constraints (33) and (37) represent the bounds and integrality constraints for first- and second-stage variables, respectively. Constraints (34) and (35) define the discrete uncertainty set. Constraint (36) is the daily demand constraints over the planning horizon.

In the following, we give the reformulation of the corresponding nurse planning problem in the form of model (P4). The definitions of variables x_d and u_{ip} from the nurse planning problem remain unchanged.

New Set

$\mathcal{S}_d$: The set of all possible realizations for the number of patients in wards on day d.

New Variable

w_{ds}: Equal to 1 if exactly s patients are in wards on day d and 0 otherwise.

$$\min_{(x,y')\in(\mathcal{X},\mathcal{Y}')}\left(\sum_{d\in\mathcal{D}} c_1x_d + \max_{(u,w)\in(\mathcal{U},\mathcal{W})}\left(\sum_{d\in\mathcal{D}}\sum_{s\in\mathcal{S}_d} c_2w_{ds}y'_{ds}\right)\right) \tag{38}$$

$$(\mathcal{U},\mathcal{W}) = \left\{(u,w) \;\middle|\; \sum_{s\in\mathcal{S}_d} w_{ds} = 1, \quad d \in \mathcal{D}, \right. \tag{39}$$

$$\sum_{t\in\mathcal{T}}\sum_{p\in\mathcal{P}_{td}} u_{tp} = \sum_{s\in\mathcal{S}_d} sw_{ds}, \quad d \in \mathcal{D}, \tag{40}$$

$$\sum_{p\in\mathcal{P}_t} u_{tp} = 1, \qquad t \in \mathcal{T}, \tag{41}$$

$$w_{ds} \in \{0,1\}, \qquad d \in \mathcal{D},\ s \in \mathcal{S}_d, \tag{42}$$

$$\left. u_{tp} \in \{0,1\}, \qquad t \in \mathcal{T},\ p \in \mathcal{P}_t \right\}, \tag{43}$$

$$(\mathcal{X},\mathcal{Y}') = \left\{(x,y') \mid \delta x_d + \delta y'_{ds} \ge \rho \times s, \quad d \in \mathcal{D}, s \in \mathcal{S}_d, \right. \tag{44}$$

$$x_d \ge 0, \text{integer}, \qquad d \in \mathcal{D}, \tag{45}$$

$$\left. 0 \le y'_{ds} \le M_d, \text{integer}, \qquad d \in \mathcal{D}, s \in S_d \right\}. \tag{46}$$

6.2. Two-Echelon Supply Chain Problem

We consider a two-echelon supply chain problem where each customer's order requires different numbers of various products. The second-layer facilities make the products and send them to the first-layer facilities that consolidate the products corresponding to each customer before shipping. There are several uncertain local scenarios for the demand of each customer. Similar two-echelon supply chain problems are studied in the literature (Amiri 2006, Gendron and Semet 2009, Sadjady and Davoudpour 2012, Pan and Nagi 2013).

In a two-stage robust optimization setting, the decision maker chooses which facilities to open in both layers in the first stage. Then the worst-case scenario for customers' demands realizes. In the second stage, the decision maker decides on the transportation of products from the second-layer facilities to first-layer facilities and from them to the customers. We formulate the problem as follows.

Sets

- $\mathcal{F}_1$: The set of first-layer facilities.
- $\mathcal{F}_2$: The set of second-layer facilities.
- $\mathcal{I}$: The set of customers.
- $\mathcal{K}$: The set of products.
- $\mathcal{P}_i$: The set of local scenarios for customer i. Each local scenario gives information on the demands of the customer for various products.

Parameters

- c_f: The opening cost of facility f.
- d_{ikp}: The demand of customer i for product k in the local scenario $p \in \mathcal{P}_i$.
- t_{kif}: The per-unit transportation cost of product k from first-layer facility f to customer i.
- $t'_{kff'}$: The per-unit transportation cost of product k from first-layer facility f to second-layer facility f'.
- c_{ifp}: The transportation cost of products from first-layer facility f to customer i if local scenario p happens for the customer. We have $c_{ifp} = \sum_{k\in\mathcal{K}} d_{ikp}t_{kif}$.
- $c_{ikff'p}$: The transportation cost of all products k demanded by customer i, from first-layer facility f to second-layer facility f' if local scenario p happens for the customer. We have $c_{ikff'p} = d_{ikp}t_{kff'}$.
- b_k: The maximum amount of product k that can be demanded by all customers.

Variables

- x_f: Equal to 1 if facility f is opened and 0 otherwise.
- u_{ip}: Equal to 1 if local scenario p realizes for customer i and 0 otherwise. (Uncertainty variable)
- y^1_{if}: Equal to 1 if the first-layer facility f supplies the demand of customer i and 0 otherwise.
- $y^2_{ikff'}$: Equal to 1 if customer i's demand for product k is transported from the second-layer facility f' to the first-layer facility f and 0 otherwise.

$$\min_{x\in\mathcal{X}}\left(\sum_{f\in\mathcal{F}_1\cup\mathcal{F}_2} c_f x_f + \max_{u\in\mathcal{U}}\left(\min_{y\in\mathcal{Y}(x,u)}\sum_{i\in\mathcal{I}}\sum_{f\in\mathcal{F}_1}\sum_{p\in\mathcal{P}_i}(c_{ifp}u_{ip}y^1_{if}) + \right.\right.$$
$$\left.\left. + \sum_{f\in\mathcal{F}_1}\sum_{f'\in\mathcal{F}_2}\sum_{i\in\mathcal{I}}\sum_{k\in\mathcal{K}_i}\sum_{p\in\mathcal{P}_i}(c_{ikff'p}u_{ip}y^2_{ikff'})\right)\right) \tag{47}$$

$$\mathcal{X} = \{x \mid x_f \in \{0,1\}, \quad f \in \mathcal{F}_1\cup\mathcal{F}_2\}, \tag{48}$$

$$\mathcal{U} = \left\{u \,\middle|\, \sum_{p\in\mathcal{P}_i} u_{ip} = 1, \quad i\in\mathcal{I}, \right. \tag{49}$$
$$\sum_{i\in\mathcal{I}}\sum_{p\in\mathcal{P}_i} d_{ikp}u_{ip} \le b_k, \quad k\in\mathcal{K}, \tag{50}$$
$$\left. u_{ip}\in\{0,1\}, \quad i\in\mathcal{I}\ p\in\mathcal{P}_i\right\}, \tag{51}$$

$$\mathcal{Y}(x,u) = \left\{y \,\middle|\, \sum_{f\in\mathcal{F}_1} y^1_{if} = 1, \quad i\in\mathcal{I}, \right. \tag{52}$$
$$y^1_{if} \le x_f, \quad i\in\mathcal{I},\ f\in\mathcal{F}_1, \tag{53}$$
$$\sum_{f'\in\mathcal{F}_2} y^2_{ikff'} = y^1_{if}, \quad i\in\mathcal{I},\ k\in\mathcal{K},\ f\in\mathcal{F}_1, \tag{54}$$
$$y^2_{ikff'} \le x_{f'}, \quad i\in\mathcal{I},\ k\in\mathcal{K},\ f\in\mathcal{F}_1,\ f'\in\mathcal{F}_2, \tag{55}$$
$$y^1_{if}\in\{0,1\}, \quad i\in\mathcal{I},\ f\in\mathcal{F}_1, \tag{56}$$
$$\left. y^2_{ikff'}\in\{0,1\}, \quad i\in\mathcal{I},\ k\in\mathcal{K},\ f\in\mathcal{F}_1,\ f'\in\mathcal{F}_2\right\}. \tag{57}$$

Constraint (49) implies that exactly one of the local scenarios realizes for each customer. Constraint (50) is a budget constraint that makes the uncertainty set more general. Constraint (52) states that each customer receives his or her order exactly from one of the first-layer facilities. First- and second-stage variables are linked by Constraints (53) and (55) that let y^1_{if} and $y^2_{ikff'}$ take 1 only if $x^1_f = 1$ and $x^2_{f'} = 1$ hold, respectively. Constraint (54) links the second-stage variables y^1_{if} and $y^2_{ikff'}$ to each other.

Model (47)–(57) is not in the format of model (P1) because, in objective function (47), the second-stage variables are multiplied by the uncertainty variables u_{ip}. After performing the reformulation explained in Section EC.10 of the electronic companion, we obtain the following model that is in the format of model (P4).

New Variables

- y^1_{ifp}: Equal to 1 if first-layer facility f supplies the demand of customer i assuming that local scenario p has happened for the customer and 0 otherwise.
- $y^2_{ikff'p}$: Equal to 1 if customer i's demand for product k is transported from second-layer facility f' to first-layer facility f assuming that local scenario p has happened for the customer and 0 otherwise.

$$\min_{(x,y)\in(\mathcal{X},\mathcal{Y})}\left(\sum_{f\in\mathcal{F}_1\cup\mathcal{F}_2} c_f x_f + \max_{u\in\mathcal{U}}\left(\sum_{i\in\mathcal{I}}\sum_{f\in\mathcal{F}_1}\sum_{p\in\mathcal{P}_i}(c_{ifp}u_{ip}y^1_{ifp}) + \right.\right.$$
$$\left.\left. + \sum_{f\in\mathcal{F}_1}\sum_{f'\in\mathcal{F}_2}\sum_{i\in\mathcal{I}}\sum_{k\in\mathcal{K}_i}\sum_{p\in\mathcal{P}_i}(c_{ikff'p}u_{ip}y^2_{ikff'p})\right)\right) \tag{58}$$

$$(49)-(51), \tag{59}$$

$$(\mathcal{X},\mathcal{Y}) = \Bigg\{(x,y)\,\Bigg|\, x_f \in \{0,1\}, \quad f \in \mathcal{F}_1 \cup \mathcal{F}_2, \tag{60}$$

$$\sum_{f\in\mathcal{F}_1} y^1_{ifp} = 1, \quad i \in \mathcal{I},\ p \in \mathcal{P}, \tag{61}$$

$$y^1_{ifp} \le x_f, \quad i \in \mathcal{I},\ f \in \mathcal{F}_1,\ p \in \mathcal{P}, \tag{62}$$

$$\sum_{f'\in\mathcal{F}_2} y^2_{ikff'p} = y^1_{ifp}, \quad i \in \mathcal{I},\ k \in \mathcal{K},\ f \in \mathcal{F}_1,\ p \in \mathcal{P}, \tag{63}$$

$$y^2_{ikff'p} \le x_{f'}, \quad i \in \mathcal{I},\ k \in \mathcal{K},\ f \in \mathcal{F}_1,\ f' \in \mathcal{F}_2,\ p \in \mathcal{P}, \tag{64}$$

$$y^1_{ifp} \in \{0,1\}, \quad i \in \mathcal{I},\ f \in \mathcal{F}_1,\ p \in \mathcal{P}, \tag{65}$$

$$y^2_{ikff'p} \in \{0,1\}, \quad i \in \mathcal{I},\ k \in \mathcal{K},\ f \in \mathcal{F}_1,\ f' \in \mathcal{F}_2,\ p \in \mathcal{P}\Bigg\}. \tag{66}$$

7. Computational Results

In this section, we present extensive computational results for the nurse planning and two-echelon supply chain problems introduced in Section 6. We implemented all algorithms in C++ and used IBM ILOG CPLEX 12.6 to solve the mixed-integer programs. We ran experiments on a computer with two 3.07 GHz Intel Xeon X5675 processors and a total of 12 cores. We ran each instance on a single core.

For all computational experiments, we set *AlgTimeLimit* to 2 hours in Algorithms 1 to 3. In the hybrid Benders algorithm, we fix the convergence limits $\delta^{Benders}_{acc}$ and δ^{H}_{acc} at 0.1%. We use the same values for $\delta^{Benders}_{acc}$ and δ^{H}_{acc} for the Benders and heuristic algorithms, respectively. We also consider 5 seconds for $Time_{LB}$ and $Time_{UB}$ in the stopping conditions of the master problem and the subproblem in Algorithms 1 and 3. Furthermore, to run Procedure 2 for Algorithms 1 and 3, we set *EvaTimeLimit* to 2 hours. Therefore, considering parameters *AlgTimeLimit* and *EvaTimeLimit*, we run a problem instance for at most 4 hours by Algorithms 1 and 3 and 2 hours by Algorithm 2. In Algorithms 2 and 3, we fix *LocalHeuristicTimeLimit* at 30 seconds. In the hybrid Benders algorithm, we consider 20 minutes for *WarmupTimeLimit*. The generated data sets for both applications are available as online supplemental material.

7.1. Nurse Planning Instances

We generated 750 instances with different parameter settings. The parameters considered in the generation of the instances include the length of the planning horizon (L), the incentive factor (IF), and the number of operating rooms over the planning horizon (OR). We set the number of weeks in the planning horizon to {2, 3, 4}. We also assume that surgeries are scheduled only on workdays. We define the incentive factor (IF) as the ratio of c_2/c_1, where c_1 and c_2 are the respective daily costs of first-stage and second-stage nurses in objective function (7). A higher value of the incentive factor shows that the hospital pays more to second-stage nurses than to first-stage ones. We set the incentive factor to {1.1, 1.3, 1.5, 1.7, 1.9}. We suppose that first-stage nurses are paid one unit cost per hour, which for eight work hours results in $c_1 = 8$. Furthermore, we also fix the number of operating rooms over the planning horizon at {1, 2, 3, 4, 5}. For each operating room, we generate three, four, or five surgeries randomly with a uniform distribution. Considering a full factorial experiment, 75 combinations of L, IF, and OR are possible, and we generate 10 instances for each problem setting for a total of 750 instances. For each patient, we generate two scenarios for the lengths of stays in ICUs and wards. In each scenario, both lengths of stays are uniformly generated from interval [1 day, 10 days]. The total number of global scenarios that include information for all patients can be computed by $2^{|T|}$, where $|T|$ is the number of patients in the planning horizon. It is worth noting that in our small-sized instances with 39 surgeries, we have $2^{39} \approx 5.4 \times 10^{11}$ scenarios. We also assume that each nurse works for eight hours a day ($\delta = 8$), and the average daily service time for each patient is two hours ($\rho = 2$).

7.2. Results of Nurse Planning Instances

In this section, we present computational results for three sets of experiments performed on nurse planning instances. In the first set of experiments, we aim at evaluating the computational performance of our proposed heuristic, Benders, and hybrid Benders algorithms. In our computational experiments, we also consider a trilevel Benders algorithm that is inspired by Chen (2013). We give the details about the recent algorithm in Section EC.11 of the electronic companion. In Section EC.12, we have provided some computational experiments to tune ε for the stopping conditions of the Benders and hybrid Benders algorithms that resulted in $\varepsilon = 5$.

In Table 2, we report the results for our heuristic, Benders, and hybrid Benders algorithms and the trilevel algorithm inspired from Chen (2013). In this table, each row gives the average results for 50 instances with different values of the incentive factor. "*Sur.*" and "*Ite.*" respectively represent the number of surgeries in the planning horizon and the number of iterations for different algorithms. For the heuristic algorithm, we do not report LB, as this algorithm does not provide any lower bound. To compute the optimality

Table 2. Computational Results for the Proposed Heuristic, Benders, and Hybrid Benders Algorithms and the Trilevel Benders Algorithm from the Literature

Data info.			Proposed algorithms															Trilevel Benders algorithm from the literature				
			Heuristic algorithm				Benders algorithm					Hybrid Benders algorithm										
L	*OR*	*Sur*.	*Time* (sec)	*Ite*.	*UB*	*Gap* (%)	*Time* (sec)	*Ite*.	*LB*	*UB*	*Gap* (%)	*Time* (sec)	*Ite*.	*LB*	*UB*	*Gap* (%)	Δ(*UB*) (%)	*Time* (sec)	*Ite*.	*LB*	*UB*	*Gap* (%)
2	1	39	1	2	350	4.02	2	29	336	336	0.00	2	34	336	336	0.00	0.00	2,271	85	336	336	0.18
	2	79	55	7	682	4.87	12	49	650	650	0.00	39	87	650	650	0.00	0.00	14,400	119	643	660	2.61
	3	119	100	8	1,000	4.36	341	65	958	958	0.00	1,086	165	958	959	0.05	−0.06	14,400	83	944	974	3.15
	4	157	99	8	1,323	5.46	1,567	91	1,254	1,254	0.00	12,027	298	1,252	1,266	1.08	−0.92	14,400	74	1,232	1,277	3.66
	5	202	34	6	1,666	5.61	8,512	113	1,577	1,581	0.24	14,400	311	1,572	1,602	1.92	−1.31	14,400	73	1,553	1,617	4.08
Average			58	6	1,004	4.86	2,087	69	955	956	0.05	5,511	179	954	963	0.61	−0.46	11,974	87	942	973	2.74
3	1	59	182	12	704	4.77	28	44	672	672	0.00	75	75	672	672	0.00	0.00	14,400	32	650	825	26.69
	2	121	335	13	1,402	6.01	7,235	101	1,323	1,326	0.23	13,838	261	1,314	1,343	2.17	−1.29	14,400	27	1,268	1,664	31.31
	3	182	257	11	2,043	6.70	14,400	171	1,913	1,965	2.68	14,400	276	1,894	1,970	3.97	−0.22	14,400	27	1,837	2,299	25.19
	4	240	332	10	2,691	7.47	14,400	258	2,506	2,618	4.46	14,400	266	2,477	2,601	4.95	0.75	14,400	29	2,407	2,855	18.60
	5	300	580	13	3,344	6.93	14,400	370	3,122	3,294	5.48	14,400	267	3,091	3,259	5.38	1.20	14,400	19	—	—	—
Average			337	12	2,037	6.38	10,093	189	1,907	1,975	2.57	11,422	229	1,890	1,969	3.29	0.09	14,400	27	1,540	1,911	25.45
4	1	80	528	15	1,082	4.95	886	84	1,031	1,031	0.00	7,840	204	1,029	1,036	0.73	−0.52	14,400	34	975	1,458	49.49
	2	163	871	15	2,129	6.34	14,400	183	2,001	2,070	3.40	14,400	251	1,978	2,069	4.53	0.13	14,400	32	1,905	2,624	37.87
	3	241	766	15	3,062	7.25	14,400	459	2,853	3,022	5.90	14,400	237	2,807	2,989	6.42	1.18	14,400	30	2,715	3,399	25.21
	4	318	904	15	3,992	7.27	14,400	523	3,717	3,988	7.25	14,400	242	3,659	3,924	7.16	1.74	14,400	19	—	—	—
	5	397	2,370	19	4,926	7.64	14,400	565	4,573	4,985	8.98	14,400	241	4,496	4,879	8.44	2.17	14,400	1	—	—	—
Average			1,088	16	3,039	6.69	11,697	363	2,835	3,019	5.11	13,088	235	2,794	2,979	5.46	0.94	14,400	23	1,865	2,494	37.53

gap values for the heuristic algorithm, we use the lower bound values of the Benders algorithm. Moreover, under the column "Hybrid Benders algorithm," we have reported $\Delta(UB)(\%)$, which presents the gap between the upper bounds of the Benders and the hybrid Benders algorithms. We compute it by $\Delta(UB)(\%) = 100(UB_B - UB_{HB})/UB_B)$, where UB_B and UB_{HB} denote the upper bound values of the Benders and hybrid Benders algorithms, respectively.

In Table 2, we observe that for most instances the heuristic algorithm converges quickly after only a few iterations and the average optimality gaps are worse than those of the Benders and hybrid Benders algorithms. This is because the heuristic algorithm is a heuristic, whereas the two other algorithms are exact algorithm and converge to the optimal solution. We also observe that the averages of optimality gaps for the hybrid Benders algorithm are 0.61%, 3.29%, and 5.46%. These averages are higher than the averages of optimality gaps for the Benders algorithm. However, the averages of $\Delta(UB)$ are −0.46%, 0.09%, and 0.94%. These values show that the hybrid Benders algorithm finds better upper bounds than the Benders algorithm in instances with planning horizons of three and four weeks. Moreover, the average optimality gaps for the trilevel Benders algorithm, proposed in the literature, are 2.74%, 25.45%, and 37.53%, which are significantly higher than those of our Benders and hybrid Benders algorithms. It is also noteworthy that the trilevel Benders algorithm does not find feasible solutions for $L = 3, OR = 5$ and $L = 4, OR = 4, 5$.

In the second set of experiments, we evaluate the computational efficiency of the Benders algorithm for different levels of block-diagonal decomposition in the nurse planning problem. In Table 3, R stands for the percentage of the smallest-size blocks (day blocks) that are merged with other blocks to form larger ones. "$R = 0$" represents an extreme case where the block-diagonal structure of the nurse planning problem is decomposed as much as possible and each block is corresponding to a single day. Higher values of R mean that the Benders algorithm benefits less from the block-diagonal structure of the problem. This table shows that, for the largest set of instances with $L = 4$, the average optimality gap increases from 5.11% to 12.13% as R increases. Figure 1 depicts the bound values and the number of variables in the subproblem versus R. This figure shows that both LB and UB values deteriorates as R increases. This is because higher values of R lead to more complicated subproblems (more variables) as a result of less benefiting from the block-diagonal structure. Figure 2 shows the improvement of the lower bound during the run time for different value of R. We can see that, during the run time, the lower bound is stronger for cases with smaller R.

In the third set of experiments, we intend to evaluate the performance of our stopping conditions proposed in Section 5. In Table 4, we report the results for three implementations of our Benders algorithm. The first implementation is the one with our proposed stopping condition for $\varepsilon = 5$. In the second implementation, we deactivated our proposed stopping conditions and added the conventional stopping condition. This stopping condition terminates the subproblem as soon as it finds a solution that defines an inequality cutting off the master problem's solution. We can view this conventional stopping condition as a special case of our proposed stopping condition with $\varepsilon = 0$ that works only for the subproblem. The third algorithm reported in Table 4 is the branch-and-cut implementation of our Benders algorithm. In this algorithm, the master problem is solved only once, and whenever a new incumbent solution is found within the branch-and-bound tree, the algorithm solves the subproblem and add the optimality cuts to the tree.

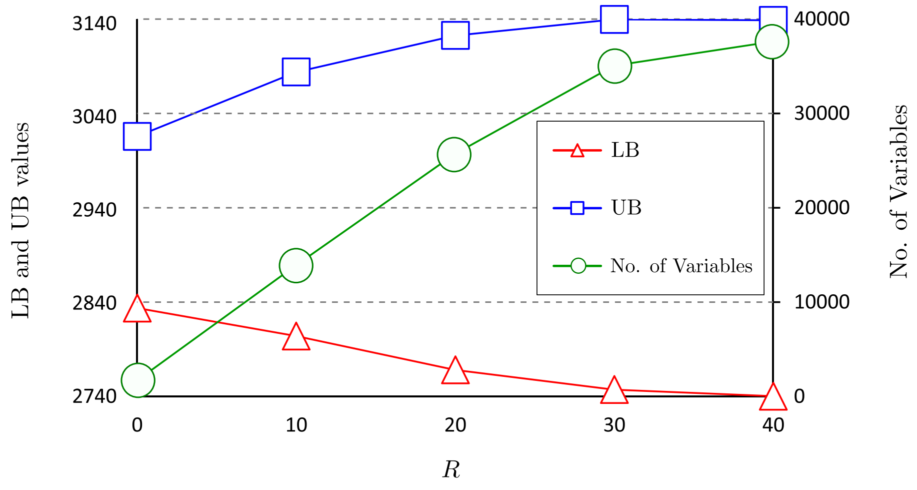
In Table 4, each row gives the average results for 50 instances with different values of the incentive factor. The results of "*Time* (sec)," "*Ite.*," "*LB*," "*UB*," and "*Gap* (%)" for the first implementation are the same as those presented for the Benders algorithm in Table 2. In fact, in Table 4, we report the same results for the first implementation for ease of comparison with the two other algorithms. Moreover, in this table, for the Benders algorithm with our proposed stopping conditions, we have presented additional results: "LB_1" and "UB_1" give the lower and upper bounds in the first iteration of the Benders algorithm, and Gap_1 computes the gap between these bounds. Moreover, "*Imp*" gives the percentage of the upper bound improvement obtained during the Benders algorithm. We compute it by $100(UB_1 - UB)/UB_1$.

We observe that the average optimality gaps for the Benders algorithm when we apply the proposed stopping conditions are 0.05%, 2.57%, and 5.11% for instances with $L = 2$, $L = 3$, and $L = 4$, respectively. However, the average optimality gaps for the Benders algorithm with the conventional stopping condition are 0.29%, 5.59%, and 9.33% and for the branch-and-cut algorithm are 2.32%, 27.06%, and 40.28%. This observation indicates that the proposed stopping conditions are essential for the efficiency of the proposed Benders algorithm. For small instances such as those with $L = 2, OR = 1, 2, 3$, the algorithm with the conventional stopping condition repeats more iterations than the algorithm with the proposed stopping conditions. However, for larger instances such as those with $L = 4, OR = 2, 3, 4, 5$, the former algorithm repeats significantly fewer iterations than the latter algorithm does. There are two reasons for this

Table 3. Computational Results of the Benders Algorithm for Different Level of Decomposing the Block-Diagonal Structure

Data info.		Proposed Benders algorithm																			
		$R = 0$				$R = 10$				$R = 20$				$R = 30$				$R = 40$			
L	*OR*	*Time* (sec)	*LB*	*UB*	*Gap* (%)	*Time* (sec)	*LB*	*UB*	*Gap* (%)	*Time* (sec)	*LB*	*UB*	*Gap* (%)	*Time* (sec)	*LB*	*UB*	*Gap* (%)	*Time* (sec)	*LB*	*UB*	*Gap* (%)
2	1	2	336	336	0.00	6	336	336	0.00	6	336	336	0.00	10	336	336	0.00	10	336	336	0.00
	2	12	650	650	0.00	631	650	650	0.00	825	650	650	0.00	1,247	650	650	0.00	1,938	650	650	0.06
	3	341	958	958	0.00	5,714	958	961	0.33	8,684	958	968	1.03	12,175	958	977	1.97	11,112	958	976	1.92
	4	1,567	1,254	1,254	0.00	14,400	1,253	1,304	4.01	14,400	1,252	1,309	4.59	14,400	1,251	1,312	4.83	14,400	1,252	1,312	4.86
	5	8,512	1,577	1,581	0.24	14,400	1,575	1,653	4.93	14,400	1,573	1,655	5.20	14,400	1,571	1,658	5.52	14,400	1,572	1,659	5.52
Average		2,087	955	956	0.05	7,030	954	981	1.85	7,663	954	983	2.16	8,446	953	986	2.47	8,372	953	987	2.47
3	1	28	672	672	0.00	316	672	672	0.00	665	672	672	0.00	852	672	672	0.00	845	672	672	0.00
	2	7,235	1,323	1,326	0.23	14,140	1,319	1,379	4.56	14,353	1,318	1,393	5.64	14,339	1,317	1,396	6.02	14,400	1,316	1,398	6.13
	3	14,400	1,913	1,965	2.68	14,400	1,905	2,031	6.58	14,400	1,899	2,043	7.57	14,400	1,897	2,044	7.69	14,400	1,898	2,047	7.86
	4	14,400	2,506	2,618	4.46	14,400	2,488	2,684	7.89	14,400	2,472	2,732	10.53	14,400	2,466	2,734	10.92	14,400	2,468	2,729	10.56
	5	14,400	3,122	3,294	5.48	14,400	3,088	3,423	10.84	14,400	3,064	3,471	13.35	14,400	3,049	3,473	13.97	14,400	3,046	3,478	14.26
Average		10,093	1,907	1,975	2.57	11,531	1,895	2,038	5.97	11,644	1,885	2,062	7.42	11,678	1,880	2,064	7.72	11,689	1,880	2,065	7.76
4	1	886	1,031	1,031	0.00	4,433	1,031	1,034	0.32	8,351	1,030	1,040	0.98	10,076	1,030	1,050	1.97	10,938	1,030	1,054	2.27
	2	14,400	2,001	2,070	3.40	14,400	1,994	2,115	6.06	14,400	1,988	2,126	6.92	14,400	1,984	2,132	7.39	14,400	1,983	2,135	7.62
	3	14,400	2,853	3,022	5.90	14,400	2,835	3,065	8.10	14,400	2,810	3,112	10.73	14,400	2,787	3,175	14.00	14,400	2,785	3,164	13.64
	4	14,400	3,717	3,988	7.25	14,400	3,684	4,072	10.52	14,400	3,619	4,188	15.82	14,400	3,573	4,196	17.55	14,400	3,565	4,196	17.81
	5	14,400	4,573	4,985	8.98	14,400	4,481	5,156	15.15	14,400	4,395	5,171	17.75	14,400	4,362	5,171	18.68	14,400	4,337	5,171	19.33
Average		11,697	2,835	3,019	5.11	12,407	2,805	3,088	8.03	13,190	2,768	3,127	10.44	13,535	2,747	3,145	11.92	13,708	2,740	3,144	12.13

Figure 1. (Color online) Lower Bound, Upper Bound, and the Number of Variables in the Subproblem vs. R



behavior: (1) The algorithm with the conventional stopping condition stops the subproblem as soon as it finds a second-stage solution that cuts off the current first-stage solution. As a result, the generated cuts are generally less effective than the cuts that the algorithm with the proposed stopping conditions generates, and therefore the first algorithm requires more iterations for convergence. (2) For small instances, the master problem is simpler, and the algorithm with the conventional stopping condition can optimally solve it fairly quickly. However, for larger instances, proving the optimality of the master problem becomes the bottleneck of the algorithm, whereas the algorithm with the proposed stopping conditions avoids this issue by terminating the master problem when the ε-stopping condition is satisfied.

Furthermore, large optimality gaps of the branch-and-cut algorithm are large because whenever the algorithm finds a first-stage feasible solution, it solves a subproblem that is a mixed-integer programming model. This requires solving a larger number of mixed-integer programs that is computationally expensive. We also observe that the averages of initial gaps in the first iteration of the Benders algorithm with the proposed stopping conditions (Gap_1) are 66.29%, 55.10%, and 47.56%; these are considerably higher than the final optimality gaps. This demonstrates that the Benders algorithm significantly improves the optimality gap. Moreover, the averages of *Imp* are 5.14%, 4.80%, and 3.31%. These averages show that the Benders algorithm improves the upper bound during the algorithm and the improvement of optimality gap is not only because of improving the lower bound. We also observe that the upper bound improvement decreases as the length of the planning horizon increases. This observation confirms that instances with longer planning horizons are more difficult and that the Benders algorithm becomes less effective in solving them. Similarly, instances with more operating rooms are more difficult, and the Benders algorithm performs more iterations before stopping for such instances.

7.3. Supply Chain Instances

We generated 600 instances for the supply chain problem. The parameters that we considered to generate the instances include the number of customers ($|\mathcal{I}|$), the number of first- and second-layer facilities ($|\mathcal{F}_1|$ and $|\mathcal{F}_2|$), customers' demands (d_{ikp}), and transportation costs (t_{kif} and $t_{kff'}$). For each instance, we uniformly generate the coordinates of customers and facilities in a square with a side length of 100 kilometers. Then, we set $t_{kif} = \alpha_k[(x_i - x_f)^2 + (y_i - y_f)^2]^{0.5}$ and $t_{kff'} = \alpha_k[(x_f - x'_f)^2 + (y_f - y'_f)^2]^{0.5}$, where α_k is the per-kilometer transportation cost for product $k \in \mathcal{K} = \{1, 2, 3\}$ and is uniformly chosen from [0.7, 1.3]. Also, we uniformly generate the demands d_{ikp} from [100, 200]. Moreover, we assume $c_f = \lambda\beta_f$, where β_f is uniformly generated from [100, 200] and λ is a parameter to tune the relative magnitude of facilities

Figure 2. (Color online) Lower Bound Trends for Different Values of R During the Run Time

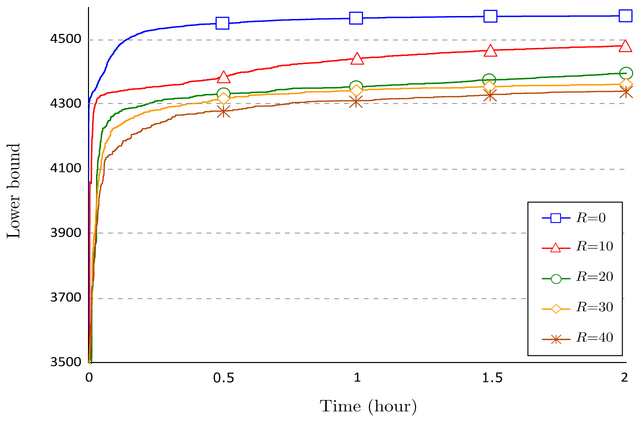


Table 4. Computational Results to Evaluate the Performance of the Proposed Stopping Conditions

Data info.			Benders algorithm with the proposed stopping conditions									Benders algorithm with the conventional stopping condition					Branch-and-cut algorithm			
L	*OR*	*Sur.*	*LB*$_1$	*UB*$_1$	*Gap*$_1$ (%)	*Imp* (%)	*Time* (sec)	*Ite.*	*LB*	*UB*	*Gap* (%)	*Time* (sec)	*Ite.*	*LB*	*UB*	*Gap* (%)	*Time* (sec)	*LB*	*UB*	*Gap* (%)
2	1	39	252	350	41.22	3.96	2	29	336	336	0.00	1	35	336	336	0.00	2	336	336	0.00
	2	79	426	686	70.61	5.16	12	49	650	650	0.00	14	55	650	650	0.00	216	650	650	0.00
	3	119	586	1,008	85.36	4.98	341	65	958	958	0.00	395	89	958	958	0.00	4,333	956	958	0.25
	4	157	796	1,334	77.27	5.91	1,567	91	1,254	1,254	0.00	9,402	96	1,250	1,259	0.64	13,960	1,234	1,270	3.00
	5	202	1,111	1,677	56.97	5.71	8,512	113	1,577	1,581	0.24	14,400	124	1,572	1,586	0.83	14,400	1,516	1,640	8.34
Average			634	1,011	66.29	5.14	2,087	69	955	956	0.05	4,842	80	953	958	0.29	6,582	938	971	2.32
3	1	59	432	713	75.08	5.64	28	44	672	672	0.00	28	46	672	672	0.00	167	672	672	0.00
	2	121	860	1,424	77.92	6.83	7,235	101	1,323	1,326	0.23	12,971	85	1,320	1,329	0.69	14,135	1,282	1,356	5.72
	3	182	1,474	2,076	44.81	5.31	14,400	171	1,913	1,965	2.68	14,400	95	1,876	2,002	6.53	14,400	1,799	2,440	35.44
	4	240	1,945	2,718	42.70	3.58	14,400	258	2,506	2,618	4.46	14,400	97	2,441	2,683	9.48	14,400	2,203	3,323	50.98
	5	300	2,552	3,383	34.53	2.57	14,400	370	3,122	3,294	5.48	14,400	75	3,040	3,385	11.24	14,400	2,815	4,031	43.16
Average			1,453	2,063	55.01	4.79	10,093	189	1,907	1,975	2.57	11,240	80	1,870	2,014	5.59	11,501	1,754	2,365	27.06
4	1	80	687	1,100	69.40	6.25	886	84	1,031	1,031	0.00	938	92	1,031	1,031	0.00	9,148	1,028	1,036	0.73
	2	163	1,421	2,167	61.82	4.43	14,400	183	2,001	2,070	3.40	14,400	68	1,967	2,106	6.93	14,400	1,888	2,723	43.41
	3	241	2,283	3,119	40.94	3.02	14,400	459	2,853	3,022	5.90	14,400	64	2,763	3,064	10.61	14,400	2,582	4,145	60.17
	4	318	3,114	4,057	32.17	1.67	14,400	523	3,717	3,988	7.25	14,400	59	3,603	4,098	13.42	14,400	3,351	5,118	52.56
	5	397	3,833	5,043	33.08	1.14	14,400	565	4,573	4,985	8.98	14,400	51	4,423	5,139	15.69	14,400	4,227	6,116	44.52
Average			2,268	3,097	47.48	3.30	11,697	363	2,835	3,019	5.11	11,708	67	2,757	3,087	9.33	13,350	2,615	3,828	40.28

fixed costs compared with the transportation costs. We set the number of customers ($|\mathcal{I}|$) to $\{50, 60, 70\}$. For the number of facilities, we consider five cases of $(|\mathcal{F}_1|, |\mathcal{F}_2|) \in \{(5,5), (5,10), (10,10), (10,20), (20,20)\}$. Finally, we set the relative cost parameter λ to $\{1, 10, 100, 1{,}000\}$. We generated 600 instances by considering 10 instances for each combination of $|\mathcal{I}|$, $(|\mathcal{F}_1|, |\mathcal{F}_2|)$, and λ. For each test instance, we set $\underline{b}_k$ in Constraint (50) equal to $1.5\sum_{i\in\mathcal{I}} \bar{d}_{ik}/|\mathcal{I}|$, where $\bar{d}_{ik}$ represents the average demand of product k by customer i.

7.4. Results of Supply Chain Instances

As demonstrated in Table 2, the Benders algorithm outperforms the heuristic and hybrid Benders algorithms in terms of the optimality gap. Therefore, in Table 5, we have provided the computational results to compare the proposed Benders algorithm with the trilevel Benders algorithm proposed by Chen (2013). We have explained the different components of the trilevel algorithm for the supply chain problem in Section EC.13 of the electronic companion. In Table 5, columns "*LB*$_1$," "*UB*$_1$," "*Gap*$_1$ (%)," "*Imp* (%)," "*Time* (sec)," "*LB*," "*UB*," and "*Gap* (%)" are the same as those in Tables 2 and 4. Furthermore, under the "Trilevel Benders algorithm from the literature" column, we have reported $\Delta(UB)$ (%), which gives the gap between the upper bounds of the Benders and trilevel algorithms. We compute it by $\Delta(UB)$ (%) $= 100(UB_T - UB_B)/\, UB_T)$, where UB_B and UB_T denote the upper bound values of the Benders and trilevel Benders algorithms, respectively. Table 5 shows that the average optimality gap of the proposed Benders algorithm for instances with $L = 2$, 3, and 4 is 0.78%, 1.25%, and 1.23%, respectively. However, the optimality gaps for the trilevel Benders algorithm are very poor, mainly because of very weak lower bounds. As explained at the end of Section EC.13 of the electronic companion, this is because the structure of the supply chain problem is such that optimality cuts for the outer master problem cannot be enhanced. Also, the average values of $\Delta(UB)$(%) are 30.47%, 28.57%, and 28.90%, implying that the solutions found by the proposed Benders algorithm are significantly superior to those of the trilevel Benders algorithm. There are also two noteworthy points about *Gap*$_1$ (%) and *Imp* (%). Comparison of *Gap*$_1$ (%) and *Gap* (%) for the Benders algorithm shows that the algorithm significantly improves the optimality gap from the first iteration to the last one. Also, the values of *Imp* (%) show that the final upper bound values are about 60% stronger than the initial upper bound values. This demonstrates that the improvement of the optimality gap from the first iteration to the last iteration of the Benders algorithm is not just because of improving the lower bound.

8. Conclusion

We have considered a class of two-stage robust optimization models with an exponential number of scenarios. We exploited the structure of the problem

Table 5. Computational Results to the Proposed Benders Algorithm and the Trilevel Benders Algorithm from the Literature for the Supply Chain Problem

Data info.			Proposed Benders algorithm									Trilevel Benders algorithm from the literature					
$\lvert\mathcal{I}\rvert$	$\mathcal{F}_1$	$\mathcal{F}_2$	LB_1	UB_1	Gap_1 (%)	*Imp* (%)	*Time* (sec)	*Ite.*	*LB*	*UB*	*Gap* (%)	*Time* (sec)	*Ite.*	*LB*	*UB*	*Gap* (%)	$\Delta(UB)$ (%)
50	5	5	1,247,540	2,627,100	114.93	49.14	53	19	1,247,930	1,247,910	0.00	14,205	785	396,992	1,263,130	31,905.10	1.41
	5	10	1,148,410	2,956,400	167.45	59.41	4,700	184	1,148,640	1,149,170	0.05	14,400	844	157,548	1,279,660	57,170.10	13.59
	10	10	972,704	2,845,650	209.88	62.21	10,037	119	977,884	981,521	0.38	14,400	523	123,985	1,265,680	73,034.50	35.37
	10	20	873,758	2,832,400	247.82	67.45	10,446	58	877,424	884,883	0.98	14,400	873	114,936	1,167,640	73,744.80	39.93
	20	20	808,930	2,554,760	255.38	65.17	12,552	24	820,099	837,699	2.52	14,400	830	100,384	1,219,510	87,906.60	62.06
Average			1,010,268	2,763,262	199.09	60.68	7,558	81	1,014,395	1,020,237	0.78	14,361	771	178,769	1,239,124	64,752.22	30.47
60	5	5	1,572,470	3,262,890	111.17	45.56	966	85	1,572,540	1,572,580	0.00	14,216	767	368,722	1,592,880	36,815.90	1.59
	5	10	1,434,340	3,551,960	164.27	57.11	8,502	280	1,436,820	1,441,550	0.29	14,400	789	161,716	1,567,880	68,612.50	10.15
	10	10	1,152,620	3,152,940	190.71	60.44	12,396	119	1,161,530	1,174,380	1.21	14,400	671	131,269	1,517,890	82,908.30	34.61
	10	20	1,050,400	3,242,130	230.28	66.72	11,776	57	1,057,910	1,073,850	1.70	14,400	838	113,491	1,409,470	89,415.20	38.17
	20	20	999,629	3,567,130	292.77	69.03	13,707	25	1,016,500	1,045,190	3.03	14,400	1,006	107,327	1,513,820	101,375.00	58.33
Average			1,241,892	3,355,410	197.84	59.77	9,470	113	1,249,060	1,261,510	1.25	14,363	814	176,505	1,520,388	75,825.38	28.57
70	5	5	1,785,810	4,133,370	135.83	54.59	3,025	175	1,787,480	1,788,830	0.08	14,285	776	413,016	1,798,490	42,785.80	0.68
	5	10	1,513,660	3,593,600	140.13	55.43	11,903	233	1,521,500	1,534,590	0.95	14,400	857	160,591	1,667,430	74,067.80	9.99
	10	10	1,279,060	3,867,250	218.25	63.82	12,974	84	1,286,440	1,307,560	1.73	14,400	525	123,108	1,692,220	101,293.00	34.98
	10	20	1,226,830	3,833,660	233.50	64.16	13,333	58	1,229,710	1,254,860	2.15	1,4400	481	106,025	1,710,550	120,517.00	43.51
	20	20	1,409,450	3,756,658	185.11	59.55	10,141	133	1,414,838	1,429,470	1.23	14,370	691	195,849	1,677,816	82,897.80	55.36
Average			1,442,962	3,836,908	182.56	59.51	10,275	137	1,447,994	1,463,062	1.23	14,371	666	199,718	1,709,301	84,312.28	28.90

using Dantzig–Wolfe decomposition and reduced the original two-stage robust problem to a single-stage robust problem. We then proposed a Benders and a heuristic algorithm for the reformulated problem and combined them to create a more effective hybrid algorithm capable of finding solutions with better objective values. Because the master problem and subproblem of the Benders algorithm are mixed-integer programs, it is computationally demanding to optimally solve them in each iteration of the algorithm. Therefore, we presented novel stopping conditions for them and provided the relevant convergence proofs. We performed extensive computational experiments to evaluate the performance the proposed algorithms in a nurse planning problem and a supply chain problem. For the nurse planning problem, the computational results demonstrated that the Benders and hybrid Benders algorithms find solutions with an average optimality gap of less than 3% over all instances with planning horizons up to four weeks. Moreover, our experiments showed that the proposed Benders algorithm is capable of finding quality solutions with an average optimality gap of less than 1.25% for the supply chain instances with up to 70 customers and 40 facilities. A possible future research direction would be to explore the extension of the proposed algorithms to multistage robust problems with exponential scenarios. Moreover, applying the proposed algorithms to other applications should be of interest.

Acknowledgments

The authors thank the associate editor and two anonymous referees for their constructive comments that significantly improved the quality of this work.

References

Amiri A (2006) Designing a distribution network in a supply chain system: Formulation and efficient solution procedure. *Eur. J. Oper. Res.* 171(2):567–576.

Ang M, Lim YF, Sim M (2012) Robust storage assignment in unit-load warehouses. *Management Sci.* 58(11):2114–2130.

Beliën J, Demeulemeester E (2008) A branch-and-price approach for integrating nurse and surgery scheduling. *Eur. J. Oper. Res.* 189(3):652–668.

Ben-Tal A, Nemirovski A (1999) Robust solutions of uncertain linear programs. *Oper. Res. Lett.* 25(1):1–13.

Ben-Tal A, Nemirovski A (2002) On tractable approximations of uncertain linear matrix inequalities affected by interval uncertainty. *SIAM J. Optim.* 12(3):811–833.

Ben-Tal A, Boaz G, Shimrit S (2009) Robust multi-echelon multi-period inventory control. *Eur. J. Oper. Res.* 199(3):922–935.

Ben-Tal A, Boyd S, Nemirovski A (2006) Extending scope of robust optimization: Comprehensive robust counterparts of uncertain problems. *Math. Programming* 107(1–2):63–89.

Ben-Tal A, Golany B, Nemirovski A, Vial J-P (2005) Retailer-supplier flexible commitments contracts: A robust optimization approach. *Manufacturing Service Oper. Management* 7(3):248–271.

Ben-Tal A, Goryashko A, Guslitzer E, Nemirovski A (2004) Adjustable robust solutions of uncertain linear programs. *Math. Programming* 99(2):351–376.

Bertsimas D, Caramanis C (2010) Finite adaptability in multistage linear optimization. *IEEE Trans. Automatic Control* 55(12):2751–2766.

Bertsimas D, Dunning I (2016) Multistage robust mixed-integer optimization with adaptive partitions. *Oper. Res.* 64(4):980–998.

Bertsimas D, Sim M (2004) The price of robustness. *Oper. Res.* 52(1): 35–53.

Bertsimas D, Dunning I, Lubin M (2016) Reformulations versus cutting-planes for robust optimization. *Comput. Management Sci.* 13(2):195–217.

Bertsimas D, Iancu DA, Parrilo P (2011) A hierarchy of near-optimal policies for multistage adaptive optimization. *IEEE Trans. Automatic Control* 56(12):2809–2824.

Bertsimas D, Litvinov E, Sun XA, Zhao J, Zheng T (2013) Adaptive robust optimization for the security constrained unit commitment problem. *IEEE Trans. Power Systems* 28(1):52–63.

Chan TCY, Shen Z-JM, Siddiq A (2018) Robust defibrillator deployment under cardiac arrest location uncertainty via row-and-column generation. *Oper. Res.* 66(2):358–379.

Chen B (2013) A new trilevel optimization algorithm for the two-stage robust unit commitment problem. Unpublished master's thesis, Iowa State University, Ames.

Chen X, Zhang Y (2009) Uncertain linear programs: Extended affinely adjustable robust counterparts. *Oper. Res.* 57(6): 1469–1482.

Chen X, Wu W, Zhang B (2016) Robust restoration method for active distribution networks. *IEEE Trans. Power Systems* 31(5): 4005–4015.

Danandeh A, Zhao L, Zeng B (2014) Job scheduling with uncertain local generation in smart buildings: Two-stage robust approach. *IEEE Trans. Smart Grid* 5(5):2273–2282.

Delage E, Iancu DA (2015) Robust multistage decision making. Aleman DM, Thiele AC, eds. *The Operations Research Revolution*, TutORials in Operations Research (INFORMS, Catonsville, MD), 20–46.

Ding T, Liu S, Yuan W, Bie Z, Zeng B (2016) A two-stage robust reactive power optimization considering uncertain wind power integration in active distribution networks. sustainable energy. *IEEE Trans. Sustainable Energy* 7(1):301–311.

El Ghaoui L, Lebret H (1997) Robust solutions to least-squares problems with uncertain data. *SIAM J. Matrix Anal. Appl.* 18(4):1035–1064.

El Ghaoui L, Oustry F, Lebret H (1998) Robust solutions to uncertain semidefinite programs. *SIAM J. Optim.* 9(1):33–52.

Feige U, Jain K, Mahdian M, Mirrokni V (2007) Robust combinatorial optimization with exponential scenarios. Fischetti M, Williamson DP, eds. *12th Internat. Integer Programming Combinatorial Optim. Conf. Proc.* (Springer, Berlin), 439–453.

Fischetti M, Monaci M (2012) Cutting plane vs. compact formulations for uncertain (integer) linear programs. *Math. Programming Comput.* 4(3):239–273.

Fonseca RJ, Rustem B (2012) International portfolio management with affine policies. *Eur. J. Oper. Res.* 223(1):177–187.

Gendron B, Semet F (2009) Formulations and relaxations for a multi-echelon capacitated location–distribution problem. *Comput. Oper. Res.* 36(5):1335–1355.

Gupta A, Nagarajan V, Ravi R (2014) Thresholded covering algorithms for robust and max–min optimization. *Math. Programming* 146(1–2):583–615.

Hanasusanto GA, Kuhn D, Wiesemann W (2015) *K*-Adaptability in two-stage robust binary programming. *Oper. Res.* 63(4):877–891.

Lee C, Liu C, Mehrotra S, Bie Z (2015) Robust distribution network reconfiguration. *IEEE Trans. Smart Grid* 6(2):836–842.

Lee C, Liu C, Mehrotra S, Shahidehpour M (2014) Modeling transmission line constraints in two-stage robust unit commitment problem. *IEEE Trans. Power Systems* 29(3):1221–1231.

Li Z, Wu W, Shahidehpour M, Zhang B (2015) Adaptive robust tie-line scheduling considering wind power uncertainty for interconnected power systems. *IEEE Trans. Power Systems* 31(4):2701–2713.

Li Z, Wu W, Zeng B, Shahidehpour M, Zhang B (2017) Decentralized contingency-constrained tie-line scheduling for multi-area power grids. *IEEE Trans. Power Systems* 32(1):354–367.

Neyshabouri S, Berg BP (2017) Two-stage robust optimization approach to elective surgery and downstream capacity planning. *Eur. J. Oper. Res.* 260(1):21–40.

Nguyen T-D, Lo AW (2012) Robust ranking and portfolio optimization. *Eur. J. Oper. Res.* 221(2):407–416.

Ouorou A (2013) Tractable approximations to a robust capacity assignment model in telecommunications under demand uncertainty. *Comput. Oper. Res.* 40(1):318–327.

Pan F, Nagi R (2013) Multi-echelon supply chain network design in agile manufacturing. *Omega* 41(6):969–983.

Poss M, Raack C (2013) Affine recourse for the robust network design problem: Between static and dynamic routing. *Networks* 61(2): 180–198.

Postek K, den Hertog D (2016) Multistage adjustable robust mixed-integer optimization via iterative splitting of the uncertainty set. *INFORMS J. Comput.* 28(3):553–574.

Remli N, Rekik M (2013) A robust winner determination problem for combinatorial transportation auctions under uncertain shipment volumes. *Transportation Res. Part C* 35(October):204–217.

Sadjady H, Davoudpour H (2012) Two-echelon, multi-commodity supply chain network design with mode selection, lead-times and inventory costs. *Comput. Oper. Res.* 39(7):1345–1354.

Siddiq A (2013) Robust facility location under demand location uncertainty. Unpublished master's thesis, University of Toronto, Toronto.

Soyster AL (1973) Convex programming with set-inclusive constraints and applications to inexact linear programming. *Oper. Res.* 21(5):1154–1157.

Vayanos P, Kuhn D, Rustem B (2011) Decision rules for information discovery in multi-stage stochastic programming. *2011 50th IEEE Conf. Decision Control Eur. Control Conf.* (IEEE, Piscataway, NJ), 7368–7373.

Wang Z, Chen B, Wang J, Kim J, Begovic MM (2014) Robust optimization based optimal DG placement in microgrids. *IEEE Trans. Smart Grid* 5(5):2173–2182.

Wang C, Jiao B, Guo L, Tian Z, Niu J, Li S (2016) Robust scheduling of building energy system under uncertainty. *Appl. Energy* 167(April):366–376.

Zeng B, Zhao L (2013) Solving two-stage robust optimization problems using a column-and-constraint generation method. *Oper. Res. Lett.* 41(5):457–461.

Zhang N (2018) Two-stage robust mixed integer programming problem with objective uncertainty. *Optim. Lett.* 12(5): 959–969.

Zhang B, Yao T, Friesz TL, Sun Y (2015) A tractable two-stage robust winner determination model for truckload service procurement via combinatorial auctions. *Transportation Res. Part B* 78(August):16–31.

Zhao L, Zeng B (2012a) An exact algorithm for two-stage robust optimization with mixed integer recourse problems. Working paper, University of South Florida, Tampa.

Zhao L, Zeng B (2012b) Robust unit commitment problem with demand response and wind energy. *Power Energy Soc. General Meeting* (IEEE, Piscataway, NJ), 1–8.

Zheng T, Zhao J, Litvinov E, Zhao F (2012) Robust optimization and its application to power system operation. *Proc. CIGRE SC C2 Session* (CIGRE, Paris).