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

Hossein Hashemi Doulabi[1,5], Patrick Jaillet[2], Gilles Pesant[3,5], Louis-Martin Rousseau[4,5]
[1]Department of Mechanical, Industrial, and Aerospace Engineering, Concordia University, Montreal, Canada
[2]Department of Electrical Engineering and Computer Science, Laboratory for Information and Decision Systems, Operations Research Center, MIT, Cambridge, USA
[3]Department of Computer and Software Engineering, Polytechnique Montreal, Montreal, Canada
[4]Department of Mathematics and Industrial Engineering, Polytechnique Montreal, Montreal, Canada
[5]Interuniversity Research Center on Enterprise Networks, Logistics and Transportation (CIRRELT), Montreal, Canada
hossein.hashemi@concordia.ca, jaillet@mit.edu, {louis-martin.rousseau, gilles.pesant}@polymtl.ca

July 31, 2019*

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. Since 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 and a two-echelon supply chain problem are performed to evaluate the efficiency of the proposed algorithms.

Key words: Integer programming, Dantzig-Wolfe decomposition, two-stage robust optimization.
History: Submitte on February 07, 2018; Revised on April 16 and July 23, 2019, Accepted on July 31, 2019

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 aims to find a robust

* Accepted for publication in *INFORMS Journal on Computing*

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 multi-stage problems where the values of uncertain parameters are revealed gradually in several stages (Ben-Tal et al. 2004, Delage and Iancu 2015). In multi-stage 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 (Ben-Tal and Nemirovski 1999, El Ghaoui and Lebret 1997, El Ghaoui et al. 1998), 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 non-convex uncertainty sets. For instance, Siddiq (2013) and Chan et al. (2017) studied a robust facility location problem and discussed how non-convex 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. Since 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. 2015). 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.

Multi-stage 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 multi-stage 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 re-parameterizing 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 (Vayanos et al. 2011, Bertsimas and Caramanis 2010, Hanasusanto et al. 2015, Postek and Den Hertog 2014, Bertsimas and Dunning 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 (Zeng and Zhao 2013, Zhao and Zeng 2012b, Danandeh et al. 2014, Ding et al. 2016, Wang et al. 2014, Lee et al. 2014, 2015, Li et al. 2015, 2017, Chen 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 (KKT) 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 handing second-stage integer variables, while 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 tri-level 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) are 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 (2017) that provides 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:

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

In the above formulation, x and y are the vector of decision variables in the first and the second stage, respectively. u is the vector of uncertain parameters that are restricted to the uncertainty set $\mathcal{U}$. c_1 and c_2 are given cost vectors. In the second stage problem, we have $\mathcal{Y}(x,u) = \{y \in \tilde{\mathcal{Y}} \,|\, Cy \leq b - Ax - Bu\}$ where A, B and C are known matrices with appropriate dimensions and b denotes the known vector of right-hand side values. $x \in \mathcal{X}$ and $y \in \tilde{\mathcal{Y}}$ represent the integrality and bound constraints that we may have for variables in the first and second stages. 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 EC.14) 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 reminder 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 k-th 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 $Cy \leq b - Ax - Bu$ included within the second-stage feasible space $\mathcal{Y}(x,u)$ as follows.

$$C_k y_k \leq b_k - A_k x - B_k u \qquad k \in \mathcal{K} \tag{2}$$

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

$\mathcal{S}'_k$: The set of all realizations for $B_k u$, i.e., $\mathcal{S}'_k = \{v \in \mathbb{R}^{Row_k} \,|\, v = B_k u, u \in \mathcal{U}\}$.

$\mathcal{S}_k$: The index set of $\mathcal{S}'_k$, i.e., $\mathcal{S}_k = \{1, 2, ..., |\mathcal{S}'_k|\}$.

e_{ks} : The s-th 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}, 0 otherwise.

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

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

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

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

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

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

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

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

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).

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

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

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

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, $c_{2k}^{\intercal} y'_{ks}$ 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*

- both models are unbounded, or

- both models are infeasible, or

- 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 $\max_{u,w}(\min_{y'}(.))$ can be replaced by $\min_{y'}(\max_{u,w}(.))$.*

Therefore, we can rewrite model (P3) as:

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

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 $(\mathcal{X},\mathcal{Y}') = \{(x,y') \,|\, x\in\mathcal{X}, y'\in\mathcal{Y}'(x)\}$. 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 $(\hat{u}^j, \hat{w}^j) \in (\mathcal{U}, \mathcal{W}), j = 1, 2, ..., m$ are already generated by solving the subproblem. We define the master problem of the Benders algorithm as:

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

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

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

For a feasible solution $(\hat{x}', \hat{y}') \in (\mathcal{X}, \mathcal{Y}')$ 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.

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

Algorithm 1 provides the pseudo code 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=∞, LB=-∞, 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 as:

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

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. 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.

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

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

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

$$D^\intercal \gamma = 0 \tag{21}$$

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

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).

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

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

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

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.

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

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

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

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

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 pseudo code of our heuristic algorithm. In Line 2, we suppose that no instance of constraint (25) is available at the beginning of the algorithm and F, G and b_3 are empty. "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 Lines 9, the algorithm checks stopping criteria. One of these stopping criteria checks if 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 δ^H_{acc}.
2: Set $Ite = 0$ and empty F, G and b_3 in models (P5) and (P6).
3: **repeat**
4: $\quad Ite++$.
5: $\quad$ Solve model (P6) with time limit $LocalTimeLimit$ to obtain a feasible solution $(\hat{x}, \hat{y}')$.
6: $\quad$ Set Obj_{Ite} equal to the objective value of model (P6).
7: $\quad$ Apply Procedure 1 to generate several cuts (25).
8: $\quad$ 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: $\quad$ Solve the linear programming relaxation of InnerMax$(\hat{x}, \hat{y}')$ to obtain $(\hat{u}, \hat{w})$.
3: $\quad$ Detect some valid cuts to remove the fractional solution $(\hat{u}, \hat{w})$.
4: $\quad$ 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 presents an example to demonstrate that the heuristic algorithm does not guarantee

optimality. In the Benders algorithm presented by 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 Bender algorithm

1: Input parameters: $WarmupTimeLimit$, $AlgTimeLimit$, $LocalHeuristicTimeLimit$, $EvaTimeLimit$, δ^{H}_{acc} and $\delta^{Benders}_{acc}$.
2: Set $UB = \infty$, $LB = -\infty$, $m = 0$ 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 problem 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 pseudo code of the hybrid Benders algorithm. In Line 3 of this algorithm, we obtain an initial solution $(\hat{x}, \hat{y}')$ by solving model (P6) while F, G and b_3 are

ignored. Lines 4 to 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 if 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 fast 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 ($Time_{SP}$

Table 1. A numerical example to explain the stopping conditions of MIPs in the Benders algorithm.

Iteration	Subproblem			Master problem		
	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**

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. Since these MIPs are not optimally solved there are gaps between the lower and upper bounds. Columns "Order" also give the order in which these MIPs are solved. In this example, in iterations 1 to 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 to 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.

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 it will be discussed later, constraints (30)-(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** $(100(UB - LB)/LB \leq \delta_{acc}^{Benders}$ 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 j-th times the algorithm generates a scenario with a new vector w in the subproblem.

$g(i)$: The iteration in which for the i-th times the algorithm re-visits 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, 0 otherwise.

Observation 1. The Benders algorithm without stopping conditions for the master problem and subproblem converges in at most $|\mathcal{W}| + 1 \leq n + 1$ 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 $O_{i+k}^{SP} - Opt \leq \varepsilon$ 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 $O_i^{SP} - Opt \leq \varepsilon$ 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 $O_{g(i_1)}^{SP} \geq O_{g(i_2)}^{SP}$ holds for any integer numbers i_1 and i_2 satisfying $1 \leq i_1 < i_2 \leq n''$.*

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

$O^{SP}_{g(1)}$ 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 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 are 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^{ICU}_{tp} : The length of stay in ICUs for patient t in local scenario $p \in \mathcal{P}_t$.

l^{Ward}_{tp} : 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.

Set:

$\mathcal{D}$: 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, i.e., $\mathcal{P}_{td} = \left\{p \in \mathcal{P}_t : d'_t + l^{ICU}_{tp} \leq d, d'_t + l^{ICU}_{tp} + l^{Ward}_{tp} > d\right\}$

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, 0 otherwise. (uncertainty variable)

y_d : The number of nurses hired on day d in the second stage.

$$\min_{x\in\mathcal{X}} \left(\sum_{d\in\mathcal{D}} c_1 x_d + \max_{u\in\mathcal{U}} \left(\min_{y\in\mathcal{Y}(x,u)} \sum_{d\in\mathcal{D}} (c_2 y_d) \right) \right) \tag{32}$$

$$\mathcal{X} = \left\{ x \,|\, x_d \geq 0, \text{integer} \right\} \tag{33}$$

$$\mathcal{U} = \Big\{ u \,|\, \sum_{p\in\mathcal{P}_t} u_{tp} = 1 \qquad t \in \mathcal{T}, \tag{34}$$

$$u_{tp} \in \{0,1\} \qquad t \in \mathcal{T}\ p \in \mathcal{P}_t \Big\} \tag{35}$$

$$\mathcal{Y}(x,u) = \Big\{ y \,|\, \delta y_d \geq \rho \sum_{t\in\mathcal{T}} \sum_{p\in\mathcal{P}_{td}} u_{tp} - \delta x_d d \in \mathcal{D}, \tag{36}$$

$$0 \leq y_d \leq M_d, \text{integer} \qquad d \in \mathcal{D} \Big\} \tag{37}$$

Constraints (33) and (37) represent the bounds and integrality constraints for first- and second-stage variables, respectively. (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 corresponding nurse planning problem reformulation in the form of model (P4). The definitions of variables x_d and u_{ip} from the nurse planning problem remain unchanged.

New sets:

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

New variables:

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

$$\min_{(x,y')\in(\mathcal{X},\mathcal{Y}')} \left(\sum_{d\in\mathcal{D}} c_1 x_d + \max_{(u,w)\in(\mathcal{U},\mathcal{W})} \left(\sum_{d\in\mathcal{D}} \sum_{s\in\mathcal{S}_d} c_2 w_{ds} y'_{ds} \right) \right) \tag{38}$$

$$(\mathcal{U},\mathcal{W}) = \Big\{ (u,w) \,|\, \sum_{s\in\mathcal{S}_d} w_{ds} = 1 \qquad d \in \mathcal{D}, \tag{39}$$

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

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

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

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

$$(\mathcal{X},\mathcal{Y}') = \Big\{(x,y')\,|\,\delta x_d + \delta y'_{ds} \geq \rho\times s \qquad d\in\mathcal{D}, s\in\mathcal{S}_d, \tag{44}$$

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

$$0\leq y'_{ds}\leq M_d, \text{integer} \qquad d\in\mathcal{D}, s\in S_d\Big\} \tag{46}$$

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 about 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:

Set:

$\mathcal{F}_1$: 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 number of product k that can be demanded by all customers.

Variables:

x_f : 1 if facility f is opened; 0 otherwise.

u_{ip} : 1 if local scenario p realizes for customer i, 0 otherwise. (uncertainty variable)

y^1_{if} : 1 if first-layer facility f supplies the demand of customer i, 0 otherwise.

$y^2_{ikff'}$: 1 if customer i's demand for product k is transported from second-layer facility f' to first-layer facility f, 0 otherwise.

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

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

$$\mathcal{U} = \Big\{u \,|\, \sum_{p\in\mathcal{P}_i} u_{ip} = 1 \qquad i\in\mathcal{I}, \tag{49}$$

$$\sum_{i\in\mathcal{I}}\sum_{p\in\mathcal{P}_i} d_{ikp}u_{ip} \le b_k \qquad k\in\mathcal{K} \tag{50}$$

$$u_{ip}\in\{0,1\} \qquad i\in\mathcal{I}\ p\in\mathcal{P}_i\Big\} \tag{51}$$

$$\mathcal{Y}(x,u) = \Big\{y \,|\, \sum_{f\in\mathcal{F}_1} y^1_{if} = 1 \qquad i\in\mathcal{I} \tag{52}$$

$$y^1_{if} \le x_f \qquad i\in\mathcal{I}\ f\in\mathcal{F}_1 \tag{53}$$

$$\sum_{f'\in\mathcal{F}_2} y^2_{ikff'} = y^1_{if} \qquad i\in\mathcal{I}\ k\in\mathcal{K}\ f\in\mathcal{F}_1 \tag{54}$$

$$y^2_{ikff'} \le x_{f'} \qquad i\in\mathcal{I}\ k\in\mathcal{K}\ f\in\mathcal{F}_1\ f'\in\mathcal{F}_2 \tag{55}$$

$$y^1_{if}\in\{0,1\} \qquad i\in\mathcal{I}\ f\in\mathcal{F}_1 \tag{56}$$

$$y^2_{ikff'}\in\{0,1\} \qquad i\in\mathcal{I}\ k\in\mathcal{K}\ f\in\mathcal{F}_1\ f'\in\mathcal{F}_2\Big\} \tag{57}$$

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/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 Appendix EC.10, we obtain the following model that is in the format of model (P4).

New variables:

y^1_{ifp} : 1 if first-layer facility f supplies the demand of customer i assuming that local scenario p has happened for the customer, 0 otherwise.

$y^2_{ikff'p}$: 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, 0 otherwise.

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

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

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

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

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

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

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

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

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

7. Computational results

In this section, we present extensive computational results for the nurse planning and two-echelon supply chain problem introduced in Section 2. 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 Intel Xeon X5675 processors, 3.07 Ghz, 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 $\delta_{acc}^{Benders}$ and δ_{acc}^{H} at 0.1%. We use the same values $\delta_{acc}^{Benders}$ and $delta_{acc}^{H}$ 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 an online supplement.

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 daily cost 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 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 1 unit cost per hour which for 8 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 3, 4, or 5 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 ICU and wards. In each scenario, both lengths of stays are uniformly generated from interval [1 day, 10 days]. The total number of global scenarios which 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 8 hours a day ($\delta = 8$) and the average daily service time for each patient is 2 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 tri-level Benders algorithm that is inspired from Chen (2013). We give the details about the recent algorithm in Appendix EC.11. In Appendix 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 tri-level algorithm inspired from Chen (2013). In this table, each row gives the average results for 50 instances with different values of the incentive factor. For the heuristic algorithm, we do not report "LB" as this algorithm does not provide any lower bound. To compute the optimality gap values for the heuristic algorithm, we use the lower bound values of the Benders algorithm. Moreover, under Column "*Hybrid Benders algorithm*", we have reported $\Delta(UB)(\%)$ that 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} denotes the upper bound values of the Benders and hybrid Benders algorithms.

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, while 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%, respectively. These values show that the hybrid Benders algorithm finds better upper bounds than the Benders algorithm in instances with planning horizons of 3 and 4 weeks. Moreover, the average optimality gaps for the tri-level Benders algorithm, proposed in the literature, are 2.74%, 25.45%, and 37.53% that are significantly higher than those of our Benders and hybrid Benders algorithms. It is also noteworthy that the tri-level 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 percent 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 means 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

Table 2. Computational results for the proposed heuristic, Benders, and hybrid Benders algorithms and the tri-level Benders algorithm from the literature

Data Info.			*Proposed algorithms*															*Tri-Level 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 (%)*	$\Delta(UB)$ (%)	*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	2271	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	14400	119	643	660	2.61
	3	119	100	8	1000	4.36	341	65	958	958	0.00	1086	165	958	959	0.05	-0.06	14400	83	944	974	3.15
	4	157	99	8	1323	5.46	1567	91	1254	1254	0.00	12027	298	1252	1266	1.08	-0.92	14400	74	1232	1277	3.66
	5	202	34	6	1666	5.61	8512	113	1577	1581	0.24	14400	311	1572	1602	1.92	-1.31	14400	73	1553	1617	4.08
Average			58	6	1004	4.86	2087	69	955	956	0.05	5511	179	954	963	0.61	-0.46	11974	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	14400	32	650	825	26.69
	2	121	335	13	1402	6.01	7235	101	1323	1326	0.23	13838	261	1314	1343	2.17	-1.29	14400	27	1268	1664	31.31
	3	182	257	11	2043	6.70	14400	171	1913	1965	2.68	14400	276	1894	1970	3.97	-0.22	14400	27	1837	2299	25.19
	4	240	332	10	2691	7.47	14400	258	2506	2618	4.46	14400	266	2477	2601	4.95	0.75	14400	29	2407	2855	18.60
	5	300	580	13	3344	6.93	14400	370	3122	3294	5.48	14400	267	3091	3259	5.38	1.20	14400	19	INF	INF	INF
Average			337	12	2037	6.38	10093	189	1907	1975	2.57	11422	229	1890	1969	3.29	0.09	14400	27	1540	1911	25.45
4	1	80	528	15	1082	4.95	886	84	1031	1031	0.00	7840	204	1029	1036	0.73	-0.52	14400	34	975	1458	49.49
	2	163	871	15	2129	6.34	14400	183	2001	2070	3.40	14400	251	1978	2069	4.53	0.13	14400	32	1905	2624	37.87
	3	241	766	15	3062	7.25	14400	459	2853	3022	5.90	14400	237	2807	2989	6.42	1.18	14400	30	2715	3399	25.21
	4	318	904	15	3992	7.27	14400	523	3717	3988	7.25	14400	242	3659	3924	7.16	1.74	14400	19	INF	INF	INF
	5	397	2370	19	4926	7.64	14400	565	4573	4985	8.98	14400	241	4496	4879	8.44	2.17	14400	1	INF	INF	INF
Average			1088	16	3039	6.69	11697	363	2835	3019	5.11	13088	235	2794	2979	5.46	0.94	14400	23	1865	2494	37.53

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) due to less benefiting from the bock-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.

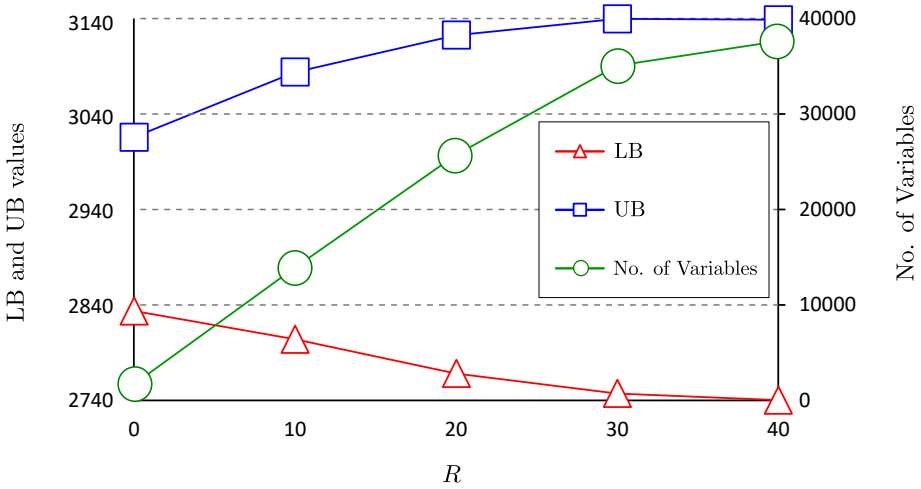

Figure 1 **Lower bound, upper bound, and the number of variables in the subproblem versus R.**

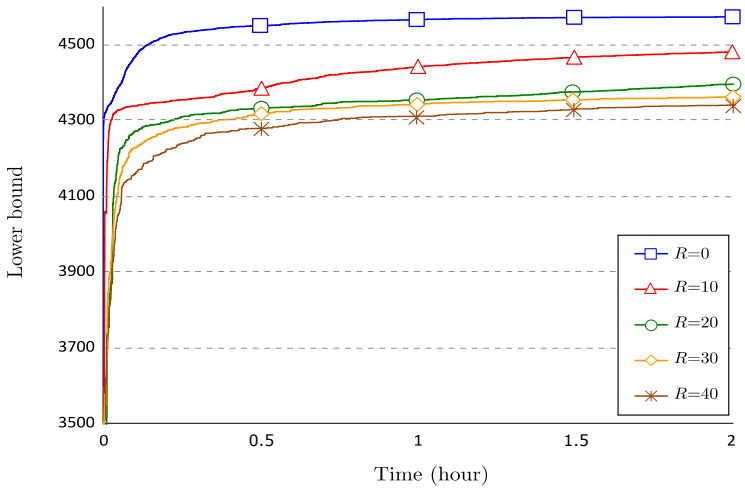

Figure 2 **Lower bound trends for different values of R during the run time.**

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 $L = 4$. However, the average optimality gaps for the Benders algorithm with the conventional stopping condition and the branch-and-cut algorithm are 0.29%, 5.59%, 9.33% and 2.32%, 27.06 %, 40.28%, respectively. 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 behaviour: 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, while 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 due to the fact that, whenever the algorithm finds a first-stage feasible solution, it solves a subproblem that is a

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	1247	650	650	0.00	1938	650	650	0.06
	3	341	958	958	0.00	5714	958	961	0.33	8684	958	968	1.03	12175	958	977	1.97	11112	958	976	1.92
	4	1567	1254	1254	0.00	14400	1253	1304	4.01	14400	1252	1309	4.59	14400	1251	1312	4.83	14400	1252	1312	4.86
	5	8512	1577	1581	0.24	14400	1575	1653	4.93	14400	1573	1655	5.20	14400	1571	1658	5.52	14400	1572	1659	5.52
	Average	2087	955	956	0.05	7030	954	981	1.85	7663	954	983	2.16	8446	953	986	2.47	8372	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	7235	1323	1326	0.23	14140	1319	1379	4.56	14353	1318	1393	5.64	14339	1317	1396	6.02	14400	1316	1398	6.13
	3	14400	1913	1965	2.68	14400	1905	2031	6.58	14400	1899	2043	7.57	14400	1897	2044	7.69	14400	1898	2047	7.86
	4	14400	2506	2618	4.46	14400	2488	2684	7.89	14400	2472	2732	10.53	14400	2466	2734	10.92	14400	2468	2729	10.56
	5	14400	3122	3294	5.48	14400	3088	3423	10.84	14400	3064	3471	13.35	14400	3049	3473	13.97	14400	3046	3478	14.26
	Average	10093	1907	1975	2.57	11531	1895	2038	5.97	11644	1885	2062	7.42	11678	1880	2064	7.72	11689	1880	2065	7.76
4	1	886	1031	1031	0.00	4433	1031	1034	0.32	8351	1030	1040	0.98	10076	1030	1050	1.97	10938	1030	1054	2.27
	2	14400	2001	2070	3.40	14400	1994	2115	6.06	14400	1988	2126	6.92	14400	1984	2132	7.39	14400	1983	2135	7.62
	3	14400	2853	3022	5.90	14400	2835	3065	8.10	14400	2810	3112	10.73	14400	2787	3175	14.00	14400	2785	3164	13.64
	4	14400	3717	3988	7.25	14400	3684	4072	10.52	14400	3619	4188	15.82	14400	3573	4196	17.55	14400	3565	4196	17.81
	5	14400	4573	4985	8.98	14400	4481	5156	15.15	14400	4395	5171	17.75	14400	4362	5171	18.68	14400	4337	5171	19.33
	Average	11697	2835	3019	5.11	12407	2805	3088	8.03	13190	2768	3127	10.44	13535	2747	3145	11.92	13708	2740	3144	12.13

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	1008	85.36	4.98	341	65	958	958	0.00	395	89	958	958	0.00	4333	956	958	0.25
	4	157	796	1334	77.27	5.91	1567	91	1254	1254	0.00	9402	96	1250	1259	0.64	13960	1234	1270	3.00
	5	202	1111	1677	56.97	5.71	8512	113	1577	1581	0.24	14400	124	1572	1586	0.83	14400	1516	1640	8.34
	Average		634	1011	66.29	5.14	2087	69	955	956	0.05	4842	80	953	958	0.29	6582	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	1424	77.92	6.83	7235	101	1323	1326	0.23	12971	85	1320	1329	0.69	14135	1282	1356	5.72
	3	182	1474	2076	44.81	5.31	14400	171	1913	1965	2.68	14400	95	1876	2002	6.53	14400	1799	2440	35.44
	4	240	1945	2718	42.70	3.58	14400	258	2506	2618	4.46	14400	97	2441	2683	9.48	14400	2203	3323	50.98
	5	300	2552	3383	34.53	2.57	14400	370	3122	3294	5.48	14400	75	3040	3385	11.24	14400	2815	4031	43.16
	Average		1453	2063	55.01	4.79	10093	189	1907	1975	2.57	11240	80	1870	2014	5.59	11501	1754	2365	27.06
4	1	80	687	1100	69.40	6.25	886	84	1031	1031	0.00	938	92	1031	1031	0.00	9148	1028	1036	0.73
	2	163	1421	2167	61.82	4.43	14400	183	2001	2070	3.40	14400	68	1967	2106	6.93	14400	1888	2723	43.41
	3	241	2283	3119	40.94	3.02	14400	459	2853	3022	5.90	14400	64	2763	3064	10.61	14400	2582	4145	60.17
	4	318	3114	4057	32.17	1.67	14400	523	3717	3988	7.25	14400	59	3603	4098	13.42	14400	3351	5118	52.56
	5	397	3833	5043	33.08	1.14	14400	565	4573	4985	8.98	14400	51	4423	5139	15.69	14400	4227	6116	44.52
	Average		2268	3097	47.48	3.30	11697	363	2835	3019	5.11	11708	67	2757	3087	9.33	13350	2615	3828	40.28

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% that 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 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 fixed costs compared to 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 $\{1, 10, 100, 1000\}$. 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 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 tri-level Benders algorithm proposed by Chen (2013). We have explained the different components of the tri-level algorithm for the supply chain problem in Appendix EC.13. In Table 5, Columns LB_1, UB_1,

$Gap_1(\%)$, and $Imp(\%)$, $Time(sec)$, LB, UB, and $Gap(\%)$ are the same as those in Tables 2 and 4. Furthermore, under Column "*Tri-Level Benders algorithm*", we have reported $\Delta(UB)(\%)$ that gives the gap between the upper bounds of the Benders and tri-level algorithms. We compute it by $\Delta(UB)(\%) = 100(UB_T - UB_B)/UB_T)$ where UB_B and UB_T respectively denote the upper bound values of the Benders and tri-level Benders algorithms. 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 tri-level Benders algorithm are very poor that is mainly due to very weak lower bounds. As explained at the end of Appendix EC.13, 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 than those of the tri-level 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 $Imp(\%)$ shows that the final upper bound values are around 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 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. Since 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 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 multi-stage robust problems with exponential scenarios. Moreover, applying the proposed algorithms to other applications should be of interest.

Table 5. Computational results to the proposed Benders algorithm and the tri-level Benders algorithm from the literature for the supply chain problem.

Data Info.			*Proposed Benders algorithm*									*Tri-Level Benders algorithm from the literature*					
$\|\mathcal{I}\|$	$\mathcal{F}_1$	$\mathcal{F}_2$	LB_1	UB_1	Gap_1 (%)	*Imp* (%)	*Time (sec)*	*Ite.*	LB	UB	*Gap* (%)	*Time (sec)*	*Ite.*	LB	UB	*Gap* (%)	$\Delta(UB)$ (%)
50	5	5	1247540	2627100	114.93	49.14	53	19	1247930	1247910	0.00	14205	785	396992	1263130	31905.10	1.41
	5	10	1148410	2956400	167.45	59.41	4700	184	1148640	1149170	0.05	14400	844	157548	1279660	57170.10	13.59
	10	10	972704	2845650	209.88	62.21	10037	119	977884	981521	0.38	14400	523	123985	1265680	73034.50	35.37
	10	20	873758	2832400	247.82	67.45	10446	58	877424	884883	0.98	14400	873	114936	1167640	73744.80	39.93
	20	20	808930	2554760	255.38	65.17	12552	24	820099	837699	2.52	14400	830	100384	1219510	87906.60	62.06
Average			1010268	2763262	199.09	60.68	7558	81	1014395	1020237	0.78	14361	771	178769	1239124	64752.22	30.47
60	5	5	1572470	3262890	111.17	45.56	966	85	1572540	1572580	0.00	14216	767	368722	1592880	36815.90	1.59
	5	10	1434340	3551960	164.27	57.11	8502	280	1436820	1441550	0.29	14400	789	161716	1567880	68612.50	10.15
	10	10	1152620	3152940	190.71	60.44	12396	119	1161530	1174380	1.21	14400	671	131269	1517890	82908.30	34.61
	10	20	1050400	3242130	230.28	66.72	11776	57	1057910	1073850	1.70	14400	838	113491	1409470	89415.20	38.17
	20	20	999629	3567130	292.77	69.03	13707	25	1016500	1045190	3.03	14400	1006	107327	1513820	101375.00	58.33
Average			1241892	3355410	197.84	59.77	9470	113	1249060	1261510	1.25	14363	814	176505	1520388	75825.38	28.57
70	5	5	1785810	4133370	135.83	54.59	3025	175	1787480	1788830	0.08	14285	776	413016	1798490	42785.80	0.68
	5	10	1513660	3593600	140.13	55.43	11903	233	1521500	1534590	0.95	14400	857	160591	1667430	74067.80	9.99
	10	10	1279060	3867250	218.25	63.82	12974	84	1286440	1307560	1.73	14400	525	123108	1692220	101293.00	34.98
	10	20	1226830	3833660	233.50	64.16	13333	58	1229710	1254860	2.15	14400	481	106025	1710550	120517.00	43.51
	20	20	1409450	3756658	185.11	59.55	10141	133	1414838	1429470	1.23	14370	691	195849	1677816	82897.80	55.36
Average			1442962	3836908	182.56	59.51	10275	137	1447994	1463062	1.23	14371	666	199718	1709301	84312.28	28.90

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, Ali. 2006. Designing a distribution network in a supply chain system: Formulation and efficient solution procedure. *European Journal of Operational Research* **171**(2) 567–576.

Ang, Marcus, Yun Fong Lim, Melvyn Sim. 2012. Robust storage assignment in unit-load warehouses. *Management Science* **58**(11) 2114–2130.

Beliën, Jeroen, Erik Demeulemeester. 2008. A branch-and-price approach for integrating nurse and surgery scheduling. *European journal of operational research* **189**(3) 652–668.

Ben-Tal, Aharon, Golany Boaz, Shtern Shimrit. 2009. Robust multi-echelon multi-period inventory control. *European Journal of Operational Research* **199**(3) 922–935.

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

Ben-Tal, Aharon, Boaz Golany, Arkadi Nemirovski, Jean-Philippe Vial. 2005. Retailer-supplier flexible commitments contracts: a robust optimization approach. *Manufacturing & Service Operations Management* **7**(3) 248–271.

Ben-Tal, Aharon, Alexander Goryashko, Elana Guslitzer, Arkadi Nemirovski. 2004. Adjustable robust solutions of uncertain linear programs. *Mathematical Programming* **99**(2) 351–376.

Ben-Tal, Aharon, Arkadi Nemirovski. 1999. Robust solutions of uncertain linear programs. *Operations research letters* **25**(1) 1–13.

Ben-Tal, Aharon, Arkadi Nemirovski. 2002. On tractable approximations of uncertain linear matrix inequalities affected by interval uncertainty. *SIAM Journal on Optimization* **12**(3) 811–833.

Bertsimas, Dimitris, Constantine Caramanis. 2010. Finite adaptability in multistage linear optimization. *IEEE Transactions on Automatic Control* **55**(12) 2751–2766.

Bertsimas, Dimitris, Iain Dunning. 2016. Multistage robust mixed-integer optimization with adaptive partitions. *Operations Research* **64**(4) 980–998.

Bertsimas, Dimitris, Iain Dunning, Miles Lubin. 2015. Reformulations versus cutting planes for robust optimization. *Computational Management Science* 1–23.

Bertsimas, Dimitris, Dan Andrei Iancu, Pablo Parrilo, et al. 2011. A hierarchy of near-optimal policies for multistage adaptive optimization. *IEEE Transactions on Automatic Control* **56**(12) 2809–2824.

Bertsimas, Dimitris, Eugene Litvinov, Xu Andy Sun, Jinye Zhao, Tongxin Zheng. 2013. Adaptive robust optimization for the security constrained unit commitment problem. *IEEE Transactions on Power Systems* **28**(1) 52–63.

Bertsimas, Dimitris, Melvyn Sim. 2004. The price of robustness. *Operations research* **52**(1) 35–53.

Chan, Timothy CY, Zuo-Jun Max Shen, Auyon Siddiq. 2017. Robust defibrillator deployment under cardiac arrest location uncertainty via row-and-column generation. *Operations Research* **66**(2) 358–379.

Chen, Bokan. 2013. A new trilevel optimization algorithm for the two-stage robust unit commitment problem. Master's thesis, Iowa State University.

Chen, Xin, Wenchuan Wu, Boming Zhang. 2016. Robust restoration method for active distribution networks. *Accepted for publication in IEEE Transactions Power Systems* .

Chen, Xin, Yuhan Zhang. 2009. Uncertain linear programs: Extended affinely adjustable robust counterparts. *Operations Research* **57**(6) 1469–1482.

Danandeh, Anna, Long Zhao, Bo Zeng. 2014. Job scheduling with uncertain local generation in smart buildings: Two-stage robust approach. *IEEE Transactions on Smart Grid* **5**(5) 2273–2282.

Delage, Erick, Dan A. Iancu. 2015. Robust multistage decision making. *Tutorials in Operations Research* 20–46.

Ding, Tao, Shiyu Liu, Wei Yuan, Zhaohong Bie, Bo Zeng. 2016. A two-stage robust reactive power optimization considering uncertain wind power integration in active distribution networks. sustainable energy. *IEEE Transactions on Sustainable Energy* **7**(1) 301–311.

El Ghaoui, Laurent, Hervé Lebret. 1997. Robust solutions to least-squares problems with uncertain data. *SIAM Journal on Matrix Analysis and Applications* **18**(4) 1035–1064.

El Ghaoui, Laurent, Francois Oustry, Hervé Lebret. 1998. Robust solutions to uncertain semidefinite programs. *SIAM Journal on Optimization* **9**(1) 33–52.

Feige, Uriel, Kamal Jain, Mohammad Mahdian, Vahab Mirrokni. 2007. Robust combinatorial optimization with exponential scenarios. *Integer Programming and Combinatorial Optimization*. Springer, 439–453.

Fischetti, Matteo, Michele Monaci. 2012. Cutting plane versus compact formulations for uncertain (integer) linear programs. *Mathematical Programming Computation* **4**(3) 239–273.

Fonseca, Raquel J, Berç Rustem. 2012. International portfolio management with affine policies. *European Journal of Operational Research* **223**(1) 177–187.

Gendron, Bernard, Frédéric Semet. 2009. Formulations and relaxations for a multi-echelon capacitated location–distribution problem. *Computers & Operations Research* **36**(5) 1335–1355.

Gupta, Anupam, Viswanath Nagarajan, R Ravi. 2014. Thresholded covering algorithms for robust and max–min optimization. *Mathematical Programming* **146**(1-2) 583–615.

Hanasusanto, Grani A, Daniel Kuhn, Wolfram Wiesemann. 2015. K-adaptability in two-stage robust binary programming. *Operations Research* **63**(4) 877–891.

Lee, Changhyeok, Cong Liu, Sanjay Mehrotra, Zhaohong Bie. 2015. Robust distribution network reconfiguration. *IEEE Transactions on Smart Grid* **6**(2) 836–842.

Lee, Changhyeok, Cong Liu, Sanjay Mehrotra, Mohammad Shahidehpour. 2014. Modeling transmission line constraints in two-stage robust unit commitment problem. *IEEE Transactions on Power Systems* **29**(3) 1221–1231.

Li, Zhigang, Wenchuan Wu, Mohammad Shahidehpour, Boming Zhang. 2015. Adaptive robust tie-line scheduling considering wind power uncertainty for interconnected power systems. *IEEE Transactions on Power Systems* **31**(4) 2701–2713.

Li, Zhigang, Wenchuan Wu, Bo Zeng, Mohammad Shahidehpour, Boming Zhang. 2017. Decentralized contingency-constrained tie-line scheduling for multi-area power grids. *IEEE Transactions on Power Systems* **32**(1) 354–367.

Neyshabouri, Saba, Bjorn P Berg. 2017. Two-stage robust optimization approach to elective surgery and downstream capacity planning. *European Journal of Operational Research* **260**(1) 21–40.

Nguyen, Tri-Dung, Andrew W Lo. 2012. Robust ranking and portfolio optimization. *European Journal of Operational Research* **221**(2) 407–416.

Ouorou, Adam. 2013. Tractable approximations to a robust capacity assignment model in telecommunications under demand uncertainty. *Computers & Operations Research* **40**(1) 318–327.

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

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

Postek, Krzysztof, Dick Den Hertog. 2014. Multi-stage adjustable robust mixed-integer optimization via iterative splitting of the uncertainty set. *CentER Discussion Paper Series* .

Remli, Nabila, Monia Rekik. 2013. A robust winner determination problem for combinatorial transportation auctions under uncertain shipment volumes. *Transportation Research Part C: Emerging Technologies* **35** 204–217.

Sadjady, Hannan, Hamid Davoudpour. 2012. Two-echelon, multi-commodity supply chain network design with mode selection, lead-times and inventory costs. *Computers & Operations Research* **39**(7) 1345–1354.

Siddiq, Auyon. 2013. Robust facility location under demand location uncertainty. Master's thesis, University of Toronto.

Soyster, Allen L. 1973. Technical note—convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations research* **21**(5) 1154–1157.

Vayanos, Phebe, Daniel Kuhn, Berç Rustem. 2011. Decision rules for information discovery in multi-stage stochastic programming. *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*. IEEE, 7368–7373.

Wang, Chengshan, Bingqi Jiao, Li Guo, Zhe Tian, Jide Niu, Siwei Li. 2016. Robust scheduling of building energy system under uncertainty. *Applied Energy* **167** 366–376.

Wang, Zhaoyu, Bokan Chen, Jianhui Wang, Jinho Kim, Miroslav M. Begovic. 2014. Robust optimization based optimal dg placement in microgrids. *IEEE Transactions on Smart Grid* **5**(5) 2173–2182.

Zeng, Bo, Long Zhao. 2013. Solving two-stage robust optimization problems using a column-and-constraint generation method. *Operations Research Letters* **41**(5) 457–461.

Zhang, Bo, Tao Yao, Terry L. Friesz, Yuqi Sun. 2015. A tractable two-stage robust winner determination model for truckload service procurement via combinatorial auctions. *Transportation Research Part B: Methodological* **78** 16–31.

Zhang, Ning. 2017. Two-stage robust mixed integer programming problem with objective uncertainty. *Optimization Letters* 1–11.

Zhao, Long, Bo Zeng. 2012a. An exact algorithm for two-stage robust optimization with mixed integer recourse problems. *Optimization-Online.org* .

Zhao, Long, Bo Zeng. 2012b. Robust unit commitment problem with demand response and wind energy. *Power and Energy Society General Meeting, San Diego, CA*. IEEE, 1–8.

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

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

Contents

EC.1. Proof of Theorem 1

We use the following notation.

$(\mathcal{U},\mathcal{W})$: The uncertainty set that is defined by (4)-(7).

$\mathcal{J}$: The index set of $(\mathcal{U},\mathcal{W})$ that is defined as $\mathcal{J}=\{1,2,...,|(\mathcal{U},\mathcal{W})|\}$ where $|(\mathcal{U},\mathcal{W})|$ represents the cardinality of $(\mathcal{U},\mathcal{W})$.

(u^j,w^j) : The j-th member of $(\mathcal{U},\mathcal{W})$.

We also define $f_k(x,w)$ and $g_k(x,e_{ks})$ as follows.

$$f_k(x,w)=\min_{y_k} c_{2k}^{\intercal}y_k \tag{EC.1.1}$$

$$C_k y_k \le b_k - A_k x - \sum_{k\in\mathcal{K}} e_{ks}w_{ks} \qquad k\in\mathcal{K} \tag{EC.1.2}$$

$$y_k \in \mathcal{Y}_k \qquad k\in\mathcal{K} \tag{EC.1.3}$$

$$g_k(x,e_{ks})=\min_{y'_{ks}} c_{2k}^{\intercal}y'_{ks} \tag{EC.1.4}$$

$$C_k y'_{ks} \le b_k - A_k x - e_{ks} \qquad k\in\mathcal{K} \tag{EC.1.5}$$

$$y'_{ks} \in \mathcal{Y}_k \qquad k\in\mathcal{K} \tag{EC.1.6}$$

For each scenario $(y,w)\in(\mathcal{U},\mathcal{W})$ with index $j\in\mathcal{J}$, with respect to constraint (6)-(7), exactly one of the variables w^j_{ks} $s\in\mathcal{S}_k$ is equal to 1 for each $k\in\mathcal{K}$. Let s_j denote the index in $\mathcal{S}_k$ for which $w^j_{ks_j}$ is equal to 1. Therefore we have the following relations.

$$w^j_{ks_j}=1 \qquad j\in\mathcal{J} \tag{EC.1.7}$$

$$w^j_{ks}=0 \qquad j\in\mathcal{J}, s\neq s_j \tag{EC.1.8}$$

In the following we prove the *if*-statement of Theorem 1. The *only if*-statement of this theorem can be proven in a reverse direction. Assume that $\hat{x}$ is a first-stage feasible solution of model (P2). In the following we separately prove that

- $\hat{x}$ is also a first-stage feasible solution of model (P3).
- The objective values of (P2) and (P3) for this fixed first-stage solution are the same if $\max\limits_{u,w}$ and $\min\limits_{y'}$ are solved optimally.

Proof of Part 1: Since model (P2) is feasible, there is at least a feasible second-stage policy $\langle\alpha_{kj}\rangle_{(k\in\mathcal{K})}$ for each $j\in\mathcal{J}$ such that

$$C_k\alpha_{kj} \le b_k - A_k\hat{x} - \sum_{s\in\mathcal{S}_k} e_{ks}w^j_{ks} \qquad k\in\mathcal{K}, j\in\mathcal{J} \tag{EC.1.9}$$

$$\alpha_{kj}\in\mathcal{Y}_k \qquad k\in\mathcal{K}, j\in\mathcal{J} \tag{EC.1.10}$$

Using (EC.1.7) and (EC.1.8), we can rewrite relations (EC.1.9)-(EC.1.10) as follows.

$$C_k\alpha_{kj} \le b_k - A_k\hat{x} - e_{ks_j} \qquad k \in \mathcal{K}, j \in \mathcal{J} \tag{EC.1.11}$$

$$\alpha_{kj} \in \mathcal{Y}_k \qquad k \in \mathcal{K}, j \in \mathcal{J} \tag{EC.1.12}$$

Relations (EC.1.11)-(EC.1.12) demonstrate that for each $k \in \mathcal{K}$ and $s \in \mathcal{S}_k$ there is at least one $j \in \mathcal{J}$ such that for $y'_{ks} = \alpha_{kj}$ constraints $C_k y'_{ks} \le b_k - A_k x - e_{ks}$ and $y'_{ks} \in \mathcal{Y}_k$ are satisfied. Therefore, $\hat{x}$ is also a first-stage feasible solution of model (P3).

Proof of Part 2: To prove that the objective values of (P2) and (P3) for the fixed first-stage solution $\hat{x}$ are the same, it is enough to prove that relation (EC.1.13) or its equivalent, relation (EC.1.14), holds.

$$c_1^\intercal\hat{x} + \max_{(u,w)\in(\mathcal{U},\mathcal{W})}\left(\sum_{k\in\mathcal{K}} f_k(\hat{x},w)\right) = c_1^\intercal\hat{x} + \max_{(u,w)\in(\mathcal{U},\mathcal{W})}\left(\sum_{k\in\mathcal{K}}\sum_{s\in\mathcal{S}_k} g_k(\hat{x},e_{ks})w_{ks}\right) \tag{EC.1.13}$$

$$\max_{(u,w)\in(\mathcal{U},\mathcal{W})}\left(\sum_{k\in\mathcal{K}} f_k(\hat{x},w)\right) = \max_{(u,w)\in(\mathcal{U},\mathcal{W})}\left(\sum_{k\in\mathcal{K}}\sum_{s\in\mathcal{S}_k} g_k(\hat{x},e_{ks})w_{ks}\right) \tag{EC.1.14}$$

Moreover, regarding (EC.1.7) and (EC.1.8), in constraint (EC.1.2) of $f_k(\hat{x},w^j)$ we can substitute $\sum_{s\in\mathcal{S}_k} e_{ks}w^j_{ks}$ by e_{ks_j}. It is then clear that mathematical programs corresponding to $g_k(\hat{x},e_{ks_j})$ and $f_k(\hat{x},w^j)$ have the same structure and following relations hold.

$$g_k(\hat{x},e_{ks_j}) = f_k(\hat{x},w^j) \qquad k \in \mathcal{K}, j \in \mathcal{J} \tag{EC.1.15}$$

$$\arg\min_{y'_{ks}}\left(g_k(\hat{x},e_{ks_j})\right) = \arg\min_{y_k}\left(f_k(\hat{x},w^j)\right) \qquad k \in \mathcal{K}, j \in \mathcal{J} \tag{EC.1.16}$$

The following stream of equalities proves the validity of (EC.1.14). In the following relations the second equality is obtained using (EC.1.15). The third equality is valid because of (EC.1.7)-(EC.1.8).

$$\max_{(u,w)\in(\mathcal{U},\mathcal{W})}\left(\sum_{k\in\mathcal{K}} f_k(\hat{x},w)\right) = \max_{j\in\mathcal{J}}\left(\sum_{k\in\mathcal{K}} f_k(\hat{x},w^j)\right) = \max_{j\in\mathcal{J}}\left(\sum_{k\in\mathcal{K}} g_k(\hat{x},e_{ks_j})\right) =$$
$$\max_{j\in\mathcal{J}}\left(\sum_{k\in\mathcal{K}}\sum_{s\in\mathcal{S}_k} g_k(\hat{x},e_{ks})w^j_{ks}\right) = \max_{(u,w)\in(\mathcal{U},\mathcal{W})}\left(\sum_{k\in\mathcal{K}}\sum_{s\in\mathcal{S}_k} g_k(\hat{x},e_{ks})w_{ks}\right)$$

In addition, (EC.1.16) shows that we can obtain the second-stage optimal policies for variables y_k in model (P2) from the optimal values of variables y'_{ks}.

EC.2. Proof of Theorem 2

As discussed in Appendix EC.1, we can present the inner max problem in model (P3) by

$$\max_{(u,w)\in(\mathcal{U},\mathcal{W})}\left(\sum_{k\in\mathcal{K}}\sum_{s\in\mathcal{S}_k} g_k(\hat{x},e_{ks})w_{ks}\right) \tag{EC.2.1}$$

where $g_k(\hat{x}, e_{ks})$ is defined as:

$$g_k(x, e_{ks}) = \min_{y'_{ks}} c_{2k}^{\intercal} y'_{ks} \tag{EC.2.2}$$

$$C_k y'_{ks} \leq b_k - A_k x - e_{ks} \qquad k \in \mathcal{K} \tag{EC.2.3}$$

$$y'_{ks} \in \mathcal{Y}_k \qquad k \in \mathcal{K} \tag{EC.2.4}$$

It is clear that the optimal values of vectors y'_{ks} for $k \in \mathcal{K}, s \in \mathcal{S}_k$ are independent of $(u, w) \in (\mathcal{U}, \mathcal{W})$ and are defined by

$$y'^{*}_{ks} = \arg\min_{y'_{ks} \in \mathcal{G}_{ks}} (c_{2k}^{\intercal} y'_{ks}) \qquad k \in \mathcal{K}, s \in \mathcal{S}_k \tag{EC.2.5}$$

where $\mathcal{G}_{ks} = \{y'_{ks} \in \mathcal{Y}_k \,|\, C_k y'_{ks} \leq b_k - A_k \hat{x} - e_{ks}\}$. Therefore, because of the independence of $y'_{ks}, k \in \mathcal{K}, s \in \mathcal{S}_k$ from $(u, w) \in (\mathcal{U}, W)$, we can swap $\max\limits_{(u,w)}$ and $\min\limits_{y'}$ in model (P3) and Theorem 2 is proven.

EC.3. Proof of Theorem 3

Consider the following problem.

$$(\text{MP}') \qquad \min_{(x,y') \in (\mathcal{X}, \mathcal{Y}')} \left(c_1^{\intercal} x + \max_{(y,w) \in (\mathcal{U}, \mathcal{W})'} \left(\sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} c_{2k}^{\intercal} y'_{ks} w_{ks} \right) \right) \tag{EC.3.1}$$

where $(\mathcal{U}, \mathcal{W})' = \{(u^j, w^j), j = 1, 2, ..., m\}$. Since $(\mathcal{U}, \mathcal{W})' \subseteq (\mathcal{U}, \mathcal{W})$ the optimal objective value of model (MP$'$) is a valid lower bound for the optimal objective value of the original robust problem (P4). In the following we demonstrate that (MP$'$) is equivalent to (MP). By writing the convex combination of m scenarios (u^j, w^j), model (MP$'$) can be rewritten as:

$$(\text{MP}'') \qquad \min_{(x,y') \in (\mathcal{X}, \mathcal{Y}')} \left(c_1^{\intercal} x + \max_{\lambda} \left(\sum_{j=1}^{m} \lambda_j \left(\sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} c_{2k}^{\intercal} y'_{ks} \hat{w}^j_{ks} \right) \right) \right) \tag{EC.3.2}$$

$$\sum_{j=1}^{m} \lambda_j = 1 \tag{EC.3.3}$$

$$\lambda_j \geq 0 \qquad j = 1, 2, ..., m. \tag{EC.3.4}$$

In model (MP$''$), for a fixed value of (x, y'), the inner max problem is a linear programming model and one of its extreme points will be the optimal solution. Each extreme point of this model corresponds to one of the scenarios (u^j, w^j). Therefore, model (MP$''$) is equivalent to model(MP$'$). By dualizing the inner max problem in model (MP$''$) and assuming θ as the dual variables of constraint (EC.3.3) we obtain model (MP) and Theorem 3 is proven.

EC.4. An example to show the local optimality of the heuristic algorithm

Consider the problem $\min_{(x_1,x_2)\in\mathcal{X}}(2x_1+1.5x_2+\max_{(u_1,u_2)\in\mathcal{U}}(x_1u_1+x_2u_2))$ where

$$\mathcal{U}=\{(u_1,u_2)\in\mathbb{N}^2\,|\,u_1\le 2,\ u_2\ge 1,\ 0.99u_1+2u_2\le 5.98,\ 1.99u_1+u_2\ge 2.99\}$$

and

$$\mathcal{X}=\{(x_1,x_2)\in\mathbb{R}^2\,|\,x_1+x_2=1,\ (x_1,x_2)\in\{0,1\}^2\}.$$

The solution space of the variables (u_1,u_2) are four points A, B, C, and D in Figure EC.1. The optimal solution of this problem is $(x_1,x_2)=(0,1)$. For this solution the objective line in $\max_{(u_1,u_2)\in\mathcal{U}}$ is Line L1. This objective line shows that scenarios A and B in the problem are optimal with a total objective value of 3.5. If we relax the integrality constraints on variables u_1 and u_2 the solution space in the problem extends to polytope E-B-C-D. In this case, for solution $(x_1,x_2)=(0,1)$ the optimal scenario is Point E with an objective value of 4.49. However, for solution $(x_1,x_2)=(1,0)$ the objective line L2 represents the objective function of the inner max problem. This objective line finds points B and C as the optimal scenarios with an objective value of 4. In this example, if we apply the heuristic algorithm to solve this problem the algorithm converges in the first iteration by finding the non-optimal solution $(x_1,x_2)=(1,0)$.

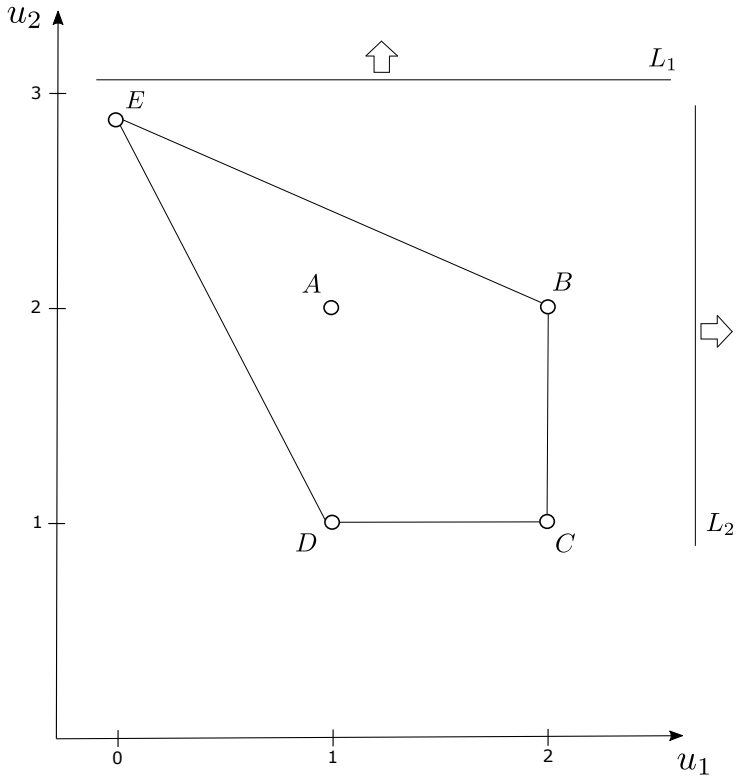

Figure EC.1 **The solution space of the uncertainty variables in the example presented to show the non-optimality of the heuristic algorithm.**

Notation used in EC.5 to E.9

We use the following notation in the proofs of Appendices EC.5 to EC.9.

$\mathcal{W}$: 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 in the subproblem before it converges.

ε : A positive constant used in stopping conditions of the master problem and subproblem.

$MP(i)$: The master problem in iteration i.

$SP(i)$: The subproblem in iteration i.

Opt : The optimal objective value of the original robust problem.

U_i^{MP}: The upper bound of the master problem in iteration i.

O_i^{MP}: The optimal objective value of the master problem in iteration i.

L_i^{MP} : The lower bound of the master problem in iteration i.

U_i^{SP} : The upper bound of the subproblem in iteration i.

O_i^{SP} : The optimal objective value of the subproblem in iteration i.

L_i^{SP} : The lower bound of the subproblem in iteration i.

$f(j)$: The iteration in which for the j-th times the algorithm generates a scenario with a new vector w in the subproblem.

$g(i)$: The iteration in which for the i-th times the algorithm re-visits 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, 0 otherwise.

EC.5. Proof of Lemma 1

Let $(\hat{x}, \hat{y}')$ and $\hat{\theta}$ respectively denote the solution and the objective value of the master problem in iteration $i-1$. Furthermore, let $(\hat{u}, \hat{w})$ denote the scenario with the repeated vector $w = \hat{w}$ found in the subproblem in iteration i. Since vector $w = \hat{w}$ is repeated, we have already included an instance of constraint (15) corresponding to this vector in the master problem in iteration $i-1$ and the following relation holds.

$$\hat{\theta} \geq c_1^\intercal \hat{x} + \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} c_{2k}^\intercal \hat{y}'_{ks} \hat{w}_{ks} \tag{EC.5.1}$$

The Benders algorithm applies solution $(\hat{x}, \hat{y}')$ to modify the objective function of the subproblem in iteration i. If $(\hat{u}, \hat{w})$ is not the optimal solution of subproblem then it means that in the subproblem the following stopping condition is satisfied.

$$c_1^\intercal \hat{x} + \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} c_{2k}^\intercal \hat{y}'_{ks} \hat{w}_{ks} \geq \hat{\theta} + \varepsilon \tag{EC.5.2}$$

Obviously relation (EC.5.2) is in contrast with (EC.5.1) and we conclude that if the algorithm visits a scenario with a repeated vector w in the subproblem, this scenario is the optimal solution of the subproblem. To prove that the optimal objective value of the subproblem is equal to the upper bound of the master problem in iteration $i-1$, we have to show that in the master problem, an instance of constraint (15) corresponding to the repeated vector $\hat{w}$ is binding. If for another scenario with a different repeated vector $w = w'$, constraint (15) is binding, then we must have $c_1^\intercal \hat{x} + \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} c_{2k}^\intercal \hat{y}'_{ks} \hat{w}_{ks} < c_1^\intercal \hat{x} + \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} c_{2k}^\intercal \hat{y}'_{ks} w'_{ks}$ which is a contradiction regarding the optimality of $(\hat{u}, \hat{w})$ in the subproblem in iteration i. Therefore, if the algorithm finds a scenario with a repeated vector $\hat{w}$ in the subproblem, the optimal objective value of the subproblem is equal to the upper bound of the recent master problem.

EC.6. Proof of Lemma 2

Equivalently this lemma states that if in $k = \lfloor (O_i^{SP} - Opt)/\varepsilon \rfloor$ iterations after iteration i the algorithm does not find any scenario with a repeated vector w then $O_{i+k}^{SP} - Opt \leq \varepsilon$ holds. In iteration i, since the algorithm found a scenario with a repeated vector w in subproblem $SP(i)$, regarding Lemma 1 this scenario is the optimal solution of the subproblem and $L_i^{SP} = O_i^{SP}$ holds. Furthermore, in the master problem $MP(i)$ that is solved after subproblem $SP(i)$, two cases are possible.

Case 1) $O_i^{MP} > O_i^{SP} - \varepsilon$ holds. First note that $O_i^{MP} < Opt$ is a valid regarding Theorem 3. $O_i^{MP} > O_i^{SP} - \varepsilon$ together with $O_i^{MP} < Opt$ results in $Opt > O_i^{SP} - \varepsilon$. The latter relation contradicts with the initial assumption $O_i^{SP} - Opt > \varepsilon$. Therefore, this case does not happen.

Case 2) $O_i^{MP} \leq O_i^{SP} - \varepsilon$ holds. This relation is equivalent to $O_i^{MP} \leq L_i^{SP} - \varepsilon$ with respect to relation $L_i^{SP} = O_i^{SP}$. Regarding $O_i^{MP} \leq L_i^{SP} - \varepsilon$, the stopping condition in master problem $MP(i)$ is satisfied and the master problem stops when it finds a feasible solution with an upper bound U_i^{MP} satisfying the following relation.

$$U_i^{MP} \leq L_i^{SP} - \varepsilon = O_i^{SP} - \varepsilon \tag{EC.6.1}$$

We have assumed that no scenario with a new vector w is generated in $k = \lfloor (O_i^{SP} - Opt)/\varepsilon \rfloor$ iterations after iteration i. Therefore, in iteration $i+1$ a scenario with a repeated vector w is generated and regarding Lemma 1 we have $O_{i+1}^{SP} = U_i^{MP}$. The recent relation together with (EC.6.1) results in the following relation.

$$O_{i+1}^{SP} \leq O_i^{SP} - \varepsilon \tag{EC.6.2}$$

Similarly we can show that for $k \leq \lfloor (O_i^{SP} - Opt)/\varepsilon \rfloor$ relation (EC.6.3) holds. This is because it is supposed form iteration i to iteration $i + \lfloor (O_i^{SP} - Opt)/\varepsilon \rfloor$ all visited scenarios have repeated vectors w.

$$O_{i+h}^{SP} \leq O_{i+h-1}^{SP} - \varepsilon \qquad h \in \{1, 2, ..., k\} \tag{EC.6.3}$$

Relation (EC.6.3) is equivalent to (EC.6.4).

$$\frac{O_{i+h}^{SP} - Opt}{\varepsilon} \leq \frac{O_{i+h-1}^{SP} - Opt}{\varepsilon} - 1 \qquad h \in \{1, 2, ..., k\} \tag{EC.6.4}$$

From (EC.6.4) we can simply obtain

$$\frac{O_{i+k}^{SP} - Opt}{\varepsilon} \leq \frac{O_{i+h-1}^{SP} - Opt}{\varepsilon} - k \tag{EC.6.5}$$

For $k = \lfloor (O_i^{SP} - Opt)/\varepsilon \rfloor$ we will have:

$$\frac{O_{(i+k)}^{SP} - Opt}{\varepsilon} \leq \frac{O_{(i+h-1)}^{SP} - Opt}{\varepsilon} - \left\lfloor \frac{O_i^{SP} - Opt}{\varepsilon} \right\rfloor \tag{EC.6.6}$$

which is equivalent to

$$O_{(i+k)}^{SP} - Opt \leq \varepsilon \tag{EC.6.7}$$

Therefore, we proved that if in $k = \lfloor (O_i^{SP} - Opt)/\varepsilon \rfloor$ iterations after iteration i the algorithm does not find any scenario with a repeated vector w then $O_{i+k}^{SP} - Opt \leq \varepsilon$ holds.

EC.7. Proof of Lemma 3

Three cases are possible.

Case 1) $O_i^{SP} - Opt \leq \varepsilon$ and $O_i^{SP} - O_i^{MP} \geq \varepsilon$ hold. We show that in this case in the next iteration the algorithm generates a scenario with a new vector w. Because of $O_i^{SP} - O_i^{MP} \geq \varepsilon$, the stopping condition in the master problem in iteration i is satisfied and the following relation holds.

$$U_i^{MP} \leq O_i^{SP} - \varepsilon \leq Opt \tag{EC.7.1}$$

If the algorithm visits a scenario with a repeated vector w in the subproblem in iteration $i+1$, we must have $U_i^{MP} = O_{i+1}^{SP}$ regarding Lemma 1. Then with respect to (EC.7.1), $O_{i+1}^{SP} < Opt$ holds which is a contradiction because the optimal objective value of the subproblem is an upper bound of the optimal objective of the robust problem. Therefore, in this case in the next iteration a scenario with a new vector w will be generated.

Case 2) $O_i^{SP} - Opt \leq \varepsilon$, $O_i^{SP} - O_i^{MP} \leq \varepsilon$ and $O_i^{MP} < Opt$ hold. We show in the next iteration the algorithm generates a scenario with a new vector w. Because of $O_i^{SP} - O_i^{MP} \leq \varepsilon$, in the master problem in iteration i there is not any scenario satisfying the stopping condition. Thus, the master problem is solved optimally and we will have the following relation.

$$O_i^{MP} = U_i^{MP} \tag{EC.7.2}$$

In the subproblem of next iteration, if the algorithm visits a scenario with a repeated vector w, then regarding Lemma 1 we must have relation (EC.7.3).

$$U_i^{MP} = O_{i+1}^{SP} \tag{EC.7.3}$$

Considering the primary assumption $O_i^{MP} < Opt$ and relations (EC.7.2)- (EC.7.3) we must have $O_{i+1}^{SP} < Opt$ which is a contradiction because the optimal objective value of the subproblem is an upper bound of the optimal objective value of the robust problem. Therefore, in this case in iteration $i+1$ the algorithm generates a scenario with a new vector w.

Case 3) $O_i^{SP} - Opt \leq \varepsilon$, $O_i^{SP} - O_i^{MP} \leq \varepsilon$ and $O_i^{MP} = Opt$ hold. We show that in this case in the next iteration either the Benders algorithm converges or it generates a scenario with a new vector w. Because of $O_i^{SP} - O_i^{MP} \leq \varepsilon$, in the master problem in iteration i there is not any scenario satisfying the stopping condition. Therefore, the master problem is solved optimally and relation (EC.7.4) holds.

$$L_i^{MP} = O_i^{MP} = U_i^{MP} \tag{EC.7.4}$$

In the subproblem of iteration $i+1$, the algorithm generates a scenario with either a new vector w or a repeated vector w. In the later case regarding Lemma 1 we must have relation (EC.7.3). Considering the primary assumption $O_i^{MP} = Opt$ and relations (EC.7.3)-(EC.7.4) we have $L_i^{MP} = Opt = O_{i+1}^{SP}$. This relation demonstrates that the optimal solution of the robust problem is obtained and the Benders algorithm is converged. Therefore, in this case, in the next iteration either the Benders algorithm converges or it generates a scenario with a repeated vector w.

EC.8. Proof of Lemma 4

Regarding constraint (31) since the algorithm visits a scenario with a repeated vector w in the subproblem of iteration $g(i_1)$, in any iteration $j \geq g(i_1)$, the inequality $U_j^{MP} \leq O_{g(i_1)}^{SP}$ holds and by setting $j = g(i_2) - 1 \geq g(i_1)$ we obtain the following relation.

$$U_{g(i_2)-1}^{MP} \leq O_{g(i_1)}^{SP} \tag{EC.8.1}$$

Note that $g(i_2) - 1 \geq g(i_1)$ holds because $i_1 < i_2$. Also regarding Lemma 1, in the subproblem of iteration $g(i_2)$ that the algorithm has visited a scenario with a repeated vector w, we have $U_{g(i_2)-1}^{MP} = O_{g(i_2)}^{SP}$. This relation together with (EC.8.1) demonstrates the validity of $O_{g(i_2)}^{SP} \leq O_{g(i_1)}^{SP}$.

EC.9. Proof of Theorem 4

To prove that the Benders algorithm converges in at most $\sum_{j=1}^{n'}(1 + (\lfloor (O_{f(j)+1}^{SP} - Opt)/\varepsilon \rfloor + 1) I_{f(j)+1})$ iterations it is enough to show it takes at most $1 + (\lfloor (O_{f(j)+1}^{SP} - Opt)/\varepsilon \rfloor + 1) I_{f(j)+1}$ iterations between visiting j-th and $(j+1)$-th new vector w in the subproblem. Let us assume $j < n'$. Two cases are possible.

Case 1) we have $I_{f(j)+1} = 0$ that means in the iteration $f(j)+1$ the algorithm finds a scenario with a new vector w. In this case the number of between visiting j-th and $(j+1)$-th new scenarios is 1.

Case 2) we have $I_{f(j)+1} = 1$ that means in iteration $f(j)+1$ the algorithm visits a scenario with a repeated vector w. In this case, after visiting the a scenario with a repeated vector w in iteration $f(j)+1$, with respect to Lemma 2 it takes at most $k = \lfloor (O_{f(j)+1}^{SP} - Opt)/\varepsilon \rfloor$ to find a scenario with a new vector w or to have $O_{f(j)+1+k}^{SP} - Opt \leq \varepsilon$. In the later case, regarding Lemma 3, we know that in the next iteration $f(j)+k+2$ either the Benders algorithm converges or a scenario with a new vector w is found. Since it is assumed that $j < n'$, the Benders algorithm does not converge before finding the $(j+1)$-th scenario with a new vector w. Thus, we expect that the algorithm generates $(j+1)$-th new vector w by iteration $f(j)+k+2$. In other words, the number of iterations between visiting j-th and $(j+1)$-th new vector w is at

most $\lfloor (O^{SP}_{f(j)+1} - Opt)/\varepsilon \rfloor + 2$. Therefore, for $j < n'$ the number of iterations between visiting j-th and $(j+1)$-th new vector w is computed by relation (EC.9.1).

$$(1 - I_{f(j)+1}) + \left(\left\lfloor \frac{O^{SP}_{f(j)+1} - Opt}{\varepsilon} \right\rfloor + 2 \right) I_{f(j)+1} \tag{EC.9.1}$$

For $j = n'$ we can use a similar reasoning as presented above for $j < n'$. The difference is that only Case 2 is applicable because regarding the definition of n' no new vector w is visited after visiting the n'-th new vector w. Moreover, when we use Lemmas 2 and 3 in Case 2, the generation of a scenario with a new vector w is not an option and we are sure that after finding the n'-th new vector w, the Benders algorithm converges in at most $\left(\lfloor (O^{SP}_{f(j)+1} - Opt)/\varepsilon \rfloor + 2 \right)$ iterations that is the same as (EC.9.1) with respect to $I_{f(j)+1} = 1$ for $j = n'$. Therefore, by summing the number of iterations computed by (EC.9.1) from $j = 1$ to $j = n'$ we obtain the following maximum number of iterations.

$$\begin{aligned}
\sum_{j=1}^{n'} \left(1 + \left(\left\lfloor (O^{SP}_{f(j)+1} - Opt)/\varepsilon \right\rfloor + 1 \right) I_{f(j)+1} \right) &= n' + \sum_{j=1}^{n'} \left(\left(\lfloor (O^{SP}_{f(j)+1} - Opt)/\varepsilon \rfloor + 1 \right) I_{f(j)+1} \right) \\
&\leq n' + \sum_{j=1}^{n'} \left(\lfloor (O^{SP}_{f(j)+1} - Opt)/\varepsilon \rfloor + 1 \right) = n' \left(\lfloor (O^{SP}_{g(1)} - Opt)/\varepsilon \rfloor + 2 \right) \\
&\leq |\mathcal{W}| \left(\lfloor (O^{SP}_{g(1)} - Opt)/\varepsilon \rfloor + 2 \right)
\end{aligned}$$

Proof of the first inequality: We know that in n' iterations the algorithm visits at least one scenario with a repeated vector w. $g(1)$ denotes the iteration in which a repeated vector w is visited for the first time. To prove the first inequality it is enough to show the validity of the following relation (EC.9.2).

$$\left\lfloor \frac{O^{SP}_{g(1)} - Opt}{\varepsilon} \right\rfloor + 1 \geq \left(\left\lfloor \frac{O^{SP}_{f(j)+1} - Opt}{\varepsilon} \right\rfloor + 1 \right) I_{f(j)+1} \qquad j \in \{1, 2, ..., n'\} \tag{EC.9.2}$$

As $O^{SP}_{g(1)} \geq Opt$ is a valid relation, (EC.9.2) holds when $I_{f(j)+1}$ equal 0. In the case that $I_{f(j)+1}$ is equal to 1, regarding the definition of g(1) and $I_{f(j)+1}$ we know that $g(1) \leq f(j) + 1$. Thus, with respect to Lemma 4, we have $O^{SP}_{g(1)} \geq O^{SP}_{f(j)+1}$ that results in $\lfloor (O^{SP}_{g(1)} - Opt)/\varepsilon \rfloor + 1 \geq \lfloor (O^{SP}_{f(j)+1} - Opt)/\varepsilon \rfloor + 1$. Therefore, relation (EC.9.2) is valid.

EC.10. Details of the reformulation for the supply chain problem

First, we have to make model (47)-(57) consistent with model (P1), and then we can apply the proposed reformulation. To this end, we replace $u_{ip} y^1_{if}$ and $u_{ip} y^2_{ikff'}$ by new second-stage variables v^1_{ifp} and $v^2_{ikff'p}$ in the objective function as in (EC.10.1) and add the new constraints (EC.10.2)-(EC.10.3) to $\mathcal{Y}(x, y)$ defined by (52)-(57).

$$\min_{x \in \mathcal{X}} \left(\sum_{f \in \mathcal{F}_1 \cup \mathcal{F}_2} c_f x_f + \max_{u \in \mathcal{U}} \left(\min_{y \in \mathcal{Y}(x,y)} \sum_{i \in \mathcal{I}} \sum_{f \in \mathcal{F}_1} \sum_{p \in \mathcal{P}_i} (c_{ifp} v^1_{ifp}) + \right. \right.$$

$$+\sum_{f\in\mathcal{F}_1}\sum_{f'\in\mathcal{F}_2}\sum_{i\in\mathcal{I}}\sum_{k\in\mathcal{K}_i}\sum_{p\in\mathcal{P}_i}(c_{ikff'p}v^2_{ikff'p})\Bigg)\Bigg) \tag{EC.10.1}$$

$$v^1_{ifp}\geq u_{ip}y^1_{if} \qquad i\in\mathcal{I}\ f\in\mathcal{F}_1\ p\in\mathcal{P}_i \tag{EC.10.2}$$

$$v^2_{ikff'p}\geq u_{ip}y^2_{ikff'} \qquad i\in\mathcal{I}\ k\in\mathcal{K}\ f\in\mathcal{F}_1\ f'\in\mathcal{F}_2\ p\in\mathcal{P}_i \tag{EC.10.3}$$

Then, we linearize (EC.10.2)-(EC.10.3) as (EC.10.4)-(EC.10.5) to make the structure of $\mathcal{Y}(x,y)$ consistent with that of $\mathcal{Y}(x,y)$ in model (P1).

$$v^1_{ifp}\geq u_{ip}+y^1_{if}-1 \qquad i\in\mathcal{I}\ f\in\mathcal{F}_1\ p\in\mathcal{P}_i \tag{EC.10.4}$$

$$v^2_{ikff'p}\geq u_{ip}+y^2_{ikff'}-1 \quad i\in\mathcal{I}\ k\in\mathcal{K}\ f\in\mathcal{F}_1\ f'\in\mathcal{F}_2\ p\in\mathcal{P}_i \tag{EC.10.5}$$

After the above modifications, set $\mathcal{Y}(x,y)$ will be as follows:

$$\mathcal{Y}(x,y)=\Big\{y\,|\,(52)-(57),(EC.10.4)-(EC.10.5)\Big\} \tag{EC.10.6}$$

The identifier $i\in\mathcal{I}$ is common in all relations defining $\mathcal{Y}(x,y)$ in (EC.10.6). Therefore, we can define the block-diagonal structure for $i\in\mathcal{I}$. In this case, the new binary variable w_{ip} for the uncertainty set is defined and linked to u_{ip} as (EC.10.7)-(EC.10.8).

$$[u_{ip}]_{p\in\mathcal{P}_i}=\sum_{p\in\mathcal{P}_i}e_iw_{ip} \qquad i\in\mathcal{I} \tag{EC.10.7}$$

$$\sum_{p\in\mathcal{P}_i}w_{ip}=1 \qquad i\in\mathcal{I} \tag{EC.10.8}$$

In (EC.10.7), e_i is i-th unit vector with 1 as the i-th entry and 0 as other entries. Constraint (EC.10.7) shows that $u_{ip}=w_{ip}$ holds in this problem. Comparison of (EC.10.7) with (5) shows that B_ku in (5) is equivalent to $[u_{ip}]_{p\in\mathcal{P}_i}$ in (EC.10.7). Therefore, to follow the reformulations presented by models (P3) and (P4), we makes copies of the second-stage variables by adding index $p'\in\mathcal{P}_i$ to them and also multiply the second-stage variables in the objective function by their corresponding $w_{ip'}$ (or equivalently $u_{ip'}$). In this case, the revised model will be as:

$$\min_{x\in\mathcal{X}}\Bigg(\sum_{f\in\mathcal{F}_1\cup\mathcal{F}_2}c_fx_f+\max_{u\in\mathcal{U}}\Bigg(\min_{y\in\mathcal{Y}(x,y)}\sum_{i\in\mathcal{I}}\sum_{f\in\mathcal{F}_1}\sum_{p\in\mathcal{P}_i}\sum_{p'\in\mathcal{P}_i}(c_{ifp}u_{ip'}v^1_{ifpp'})+$$

$$+\sum_{f\in\mathcal{F}_1}\sum_{f'\in\mathcal{F}_2}\sum_{i\in\mathcal{I}}\sum_{k\in\mathcal{K}_i}\sum_{p\in\mathcal{P}_i}\sum_{p'\in\mathcal{P}_i}(c_{ikff'p}u_{ip'}v^2_{ikff'pp'})\Bigg)\Bigg) \tag{EC.10.9}$$

$$\mathcal{X}=\Big\{x\,|\,x_f\in\{0,1\} \qquad f\in\mathcal{F}_1\cup\mathcal{F}_2\Big\} \tag{EC.10.10}$$

$$\mathcal{U}=\Big\{u\,|\,\sum_{p\in\mathcal{P}_i}u_{ip}=1 \qquad i\in\mathcal{I}, \tag{EC.10.11}$$

$$\sum_{i\in\mathcal{I}}\sum_{p\in\mathcal{P}_i}d_{ikp}u_{ip}\leq b_k \qquad k\in\mathcal{K} \tag{EC.10.12}$$

$$u_{ip} \in \{0,1\} \qquad i \in \mathcal{I}\ p \in \mathcal{P}_i \Big\} \tag{EC.10.13}$$

$$\mathcal{Y}(x,y) = \Big\{ y \,|\, \sum_{f \in \mathcal{F}_1} y^1_{ifp'} = 1 \qquad i \in \mathcal{I}\ p' \in \mathcal{P} \tag{EC.10.14}$$

$$y^1_{ifp'} \le x_f \qquad i \in \mathcal{I}\ f \in \mathcal{F}_1\ p' \in \mathcal{P} \tag{EC.10.15}$$

$$\sum_{f' \in \mathcal{F}_2} y^2_{ikff'p'} = y^1_{ifp'} \qquad i \in \mathcal{I}\ k \in \mathcal{K}\ f \in \mathcal{F}_1\ p' \in \mathcal{P} \tag{EC.10.16}$$

$$y^2_{ikff'p'} \le x_{f'} \qquad i \in \mathcal{I}\ k \in \mathcal{K}\ f \in \mathcal{F}_1\ f' \in \mathcal{F}_2\ p' \in \mathcal{P} \tag{EC.10.17}$$

$$y^1_{ifp'} \in \{0,1\} \qquad i \in \mathcal{I}\ f \in \mathcal{F}_1\ p' \in \mathcal{P} \tag{EC.10.18}$$

$$y^2_{ikff'p'} \in \{0,1\} \qquad i \in \mathcal{I}\ k \in \mathcal{K}\ f \in \mathcal{F}_1\ f' \in \mathcal{F}_2\ p' \in \mathcal{P} \tag{EC.10.19}$$

$$y^2_{ikff'p'} \in \{0,1\} \qquad i \in \mathcal{I}\ k \in \mathcal{K}\ f \in \mathcal{F}_1\ f' \in \mathcal{F}_2\ p' \in \mathcal{P} \tag{EC.10.20}$$

$$v^1_{ifpp'} \ge e_{ipp'} + y^1_{ifp'} - 1 \qquad i \in \mathcal{I}\ f \in \mathcal{F}_1\ p \in \mathcal{P}_i\ p' \in \mathcal{P} \tag{EC.10.21}$$

$$v^2_{ikff'pp'} \ge e_{ipp'} + y^2_{ikff'p'} - 1 \qquad i \in \mathcal{I}\ k \in \mathcal{K}\ f \in \mathcal{F}_1\ f' \in \mathcal{F}_2\ p \in \mathcal{P}_i\ p' \in \mathcal{P} \Big\} \tag{EC.10.22}$$

Comparison of constraints (EC.10.21)-(EC.10.22) with constraints (EC.10.4)-(EC.10.5) implies that we have replaced u_{ip} with $e_{ipp'}$ that is a parameter and is equal to 1 for $p = p'$ and 0 otherwise. The reason for the replacement of u_{ip} is that when we make copies of the second-stage constraints, we replace the uncertainty variables with their realizations (e.g. compare (9) with (12)). In constraints (EC.10.21)-(EC.10.22), we have $v^1_{ifpp'} = y^1_{ifp'}$ and $v^1_{ifpp'} = y^2_{ikff'p'}$ for $p = p'$. For $p \neq p'$, $v^1_{ifpp'}$ and $v^1_{ifpp'}$ will be equal to 0 because of their non-negative cost in the objective function. This replacement in model (EC.10.9)-(EC.10.22) results in model (EC.10.23)-(EC.10.34).

$$\min_{x \in \mathcal{X}} \Bigg(\sum_{f \in \mathcal{F}_1 \cup \mathcal{F}_2} c_f x_f + \max_{u \in \mathcal{U}} \Bigg(\min_{y \in \mathcal{Y}(x)} \sum_{i \in \mathcal{I}} \sum_{f \in \mathcal{F}_1} \sum_{p' \in \mathcal{P}_i} (c_{ifp} u_{ip'} y^1_{ifp'}) + \sum_{f \in \mathcal{F}_1} \sum_{f' \in \mathcal{F}_2} \sum_{i \in \mathcal{I}} \sum_{k \in \mathcal{K}_i} \sum_{p' \in \mathcal{P}_i} (c_{ikff'p} u_{ip'} y^2_{ikff'p'}) \Bigg) \Bigg) \tag{EC.10.23}$$

$$\mathcal{X} = \Big\{ x \,|\, x_f \in \{0,1\} \qquad f \in \mathcal{F}_1 \cup \mathcal{F}_2 \Big\} \tag{EC.10.24}$$

$$\mathcal{U} = \Big\{ u \,|\, \sum_{p \in \mathcal{P}_i} u_{ip} = 1 \qquad i \in \mathcal{I}, \tag{EC.10.25}$$

$$\sum_{i \in \mathcal{I}} \sum_{p \in \mathcal{P}_i} d_{ikp} u_{ip} \le b_k \qquad k \in \mathcal{K} \tag{EC.10.26}$$

$$u_{ip} \in \{0,1\} \qquad i \in \mathcal{I}\ p \in \mathcal{P}_i \Big\} \tag{EC.10.27}$$

$$\mathcal{Y}(x) = \Big\{ y \,|\, \sum_{f \in \mathcal{F}_1} y^1_{ifp'} = 1 \qquad i \in \mathcal{I}\ p' \in \mathcal{P} \tag{EC.10.28}$$

$$y^1_{ifp'} \leq x_f \qquad i \in \mathcal{I}\ f \in \mathcal{F}_1\ p' \in \mathcal{P} \qquad \text{(EC.10.29)}$$

$$\sum_{f' \in \mathcal{F}_2} y^2_{ikff'p'} = y^1_{ifp'} \qquad i \in \mathcal{I}\ k \in \mathcal{K}\ f \in \mathcal{F}_1\ p' \in \mathcal{P} \qquad \text{(EC.10.30)}$$

$$y^2_{ikff'p'} \leq x_{f'} \qquad i \in \mathcal{I}\ k \in \mathcal{K}\ f \in \mathcal{F}_1\ f' \in \mathcal{F}_2\ p' \in \mathcal{P} \qquad \text{(EC.10.31)}$$

$$y^1_{ifp'} \in \{0,1\} \qquad i \in \mathcal{I}\ f \in \mathcal{F}_1\ p' \in \mathcal{P} \qquad \text{(EC.10.32)}$$

$$y^2_{ikff'p'} \in \{0,1\} \qquad i \in \mathcal{I}\ k \in \mathcal{K}\ f \in \mathcal{F}_1\ f' \in \mathcal{F}_2\ p' \in \mathcal{P} \qquad \text{(EC.10.33)}$$

$$y^2_{ikff'p'} \in \{0,1\} \qquad i \in \mathcal{I}\ k \in \mathcal{K}\ f \in \mathcal{F}_1\ f' \in \mathcal{F}_2\ p' \in \mathcal{P} \Big\} \qquad \text{(EC.10.34)}$$

This model is in the format of model (P3). Therefore, we can pass the inner $\min_{y \in \mathcal{Y}(x)}(.)$ out of $\max_{u \in \mathcal{U}}(.)$ that results in model (58)-(66)

EC.11. Tri-level Benders algorithm for the nurse scheduling problem

The structure of this tri-level Benders algorithm that we present in this section is proposed by Chen (2013). In this Benders algorithm, we have an outer master problem that fixes a first-stage solution and proposes it to the inner loop. In the inner loop, an inner master problem looks for the worst-case scenario for the uncertainty considering the given first-stage solution. After fixing the uncertainty, we solve a subproblem to find a second-stage solution for the given uncertainty and the fixed first-stage solution. We present the pseudo code of the tri-level Benders algorithm by Algorithm EC.1.

In Line 1, we initialize the values of lower and upper bounds. The inner loop of the algorithm starts in Line 2 and finishes in Line 14 when the optimality gap is small enough or the time limit is reached. In Line 3, we solve the outer master problem (EC.11.1)-(EC.11.7) that provides a first-stage solution for the rest of the algorithm. We have provided the details on the outer master problem after this explanation for the pseudo code of Algorithm 4. We update the lower bound in Line 4 by setting it equal the optimal objective value of the recent outer master problem. Then, the inner loop of the algorithm starts in Line 5 and finishes in Line 11. In this loop, we first add an optimality cut (EC.11.10) to the inner master problem (EC.11.9)-(EC.11.15) using the most recent second-stage solution $< y_d >_{d \in \mathcal{D}}$ that is obtained in the subproblem. In the first iteration of the inner loop, we consider a trivial second-stage solution $\hat{y}_d = M\ d \in \mathcal{D}$. This solution means that we hire M nurses on all days over the planning horizon. In Line 7, we solve the inner master problem (EC.11.9)-(EC.11.15) to find a new worst-case scenario for the realization of the uncertainty. In Line 8, we update the upper bound by $UB := max\{UB, UB_{new}\}$ where UB is the current upper bound in the algorithm and UB_{new} is the sum of current first-stage cost obtained in the outer master problem and the worst-case second stage cost that is the optimal objective value of recent inner master problem. In Line 9,

Algorithm EC.1. Tri-level Benders algorithm

1: Initialize UB=∞ and LB=-∞.
2: **repeat**
3: Solve the outer master problem (EC.11.1)-(EC.11.7) to find a first-stage solution.
4: Update the lower bound.
5: **repeat**
6: Add an optimality cut (EC.11.10) to the inner master problem (EC.11.9)-(EC.11.15) for the current second-stage solution.
7: Solve the inner master problem (EC.11.9)-(EC.11.15) to find a new worst-case scenario.
8: Update the upper bound if necessary.
9: Modify the right-hand side of constraint (EC.11.17) in the subproblem (EC.11.16)-(EC.11.18) based on the new worst-case scenario.
10: Solve the subproblem (EC.11.16)-(EC.11.18) to find a new second-stage solution for the given first-stage solution and the given worst-case scenario.
11: **until** (the objective value of the subproblem is the equal to the objective value of the inner master problem or the time limit $AlgTimeLimit$ is reached)
12: Remove all optimality cuts from the inner master problem.
13: Add the optimality cut (EC.11.2) to the outer master problem.
14: **until** ($100(UB-LB)/LB \leq \delta_{acc}$ or time limit $AlgTimeLimit$ is reached)

we update the right-hand side value of Constraint (EC.11.17) in the subproblem using the current worst-case scenario obtained in the inner master problem (EC.11.9)-(EC.11.15). Then we solve the subproblem (EC.11.16)-(EC.11.18) for the current firs-stage solution and the current worst-case scenario. In Line 11, the inner loop stops if the objective value of the subproblem is the equal to the objective value of the inner master problem or if the time limit is reached. In Line 12, we remove all optimality cuts added to the inner master problem because the inner master problem is solved locally for each first-stage solution fixed in the outer master problem. We then add an optimality cut (EC.11.2) to the outer master problem. In Line 14, the algorithm stops if the optimality gap is less than δ_{acc} or if the time limit is reached.

In the following, we present the outer master problem, inner master problem, and the subproblem of the tri-level benders algorithm. We suppose that all variables, sets, and parameters presented in Section 5 are defined in the same way.

Outer Master Problem

We introduce the following notation for the outer master problem.

Sets:

$\mathcal{J}$: Index set for the first-stage solutions that are evaluated by the inner master problem.

$\mathcal{I}_d$: Index set for the number of first-stage nurses that we may hire for day d. We compute it by $\mathcal{I}_d = \{0, 1, ..., S_d^{\max}\}$.

Parameters:

α_j : The second-stage cost for first-stage solution $j \in \mathcal{J}$.

$S_d^{\max}$: The maximum number of patients that can be in the ward on day d.

$\hat{x}_{jd}$: The number of first-stage nurses on day d in solution $j \in \mathcal{J}$.

Variables:

v_{di} : 1 if we hire i nurses in the first stage, 0 otherwise.

θ_{outer} : A lower bound on the second-stage cost in the outer master problem.

The outer master problem of the tri-level Benders algorithm reads as follows.

$$\text{Outer-MP} \quad \min_{x,\theta_{outer}} c_1^\intercal x + \theta_{outer} \tag{EC.11.1}$$

$$\theta_{outer} \geq \alpha_j - c_2 \sum_{d\in\mathcal{D}} \sum_{\substack{i\in\mathcal{I}_d:\\ i>\hat{x}_{jd}}} (i - \hat{x}_{jd}) v_{di} \qquad j \in \mathcal{J} \tag{EC.11.2}$$

$$x_d = \sum_{i\in\mathcal{I}_d} i v_{di} \qquad d \in \mathcal{D} \tag{EC.11.3}$$

$$\sum_{i\in\mathcal{I}_d} v_{di} = 1 \qquad d \in \mathcal{D} \tag{EC.11.4}$$

$$\theta_{outer} \geq 0 \tag{EC.11.5}$$

$$v_{di} \in \{0, 1\} \qquad d \in \mathcal{D}, i \in \mathcal{I}_d \tag{EC.11.6}$$

$$\delta x_d + \delta M \geq \rho S_d^{\max} \qquad d \in \mathcal{D} \tag{EC.11.7}$$

In (EC.11.1)-(EC.11.6), the objective function (EC.11.1) minimizes the sum of the first-stage cost $c_1^\intercal x$ and θ_{outer} that provides a lower bound on the second-stage. Optimality cut (EC.11.2) approximates the second-stage cost and tightens θ_{outer} as the algorithm adds more cuts to the outer master problem. This cut ensures that θ_{outer} will be equal to the second-stage cost α_j if solution $j \in \mathcal{J}$ is selected again, and for all other first-stage solutions that are not visited yet, θ_{outer} is a lower bound on the second-stage cost. In this constraint, for a fixed day $d \in \mathcal{D}$, if the number of first-stage nurses that the model selects is a units less than the number of hired first-stage nurses in a previously visited solution j (i.e., $\sum_{\substack{i\in\mathcal{I}_d:\\ i>\hat{x}_{jd}}} (i - \hat{x}_{jd}) v_{di} = a$), then the second-stage cost for the new solution on this day is most $c_2 \times a$ less than the second-stage cost α_j. This constraint is an improved version of the following big-M constraint.

$$\theta_{outer} \geq \alpha_j - \sum_{d\in\mathcal{D}} M' \left[1 - \sum_{\substack{i\in\mathcal{I}_d:\\ i\leq\hat{x}_{jd}}} v_{di} \right] \qquad j \in \mathcal{J} \tag{EC.11.8}$$

In Constraint (EC.11.8), M' is a big-M value. This constraint becomes inactive when $\sum_{\substack{i\in\mathcal{I}_d:\\ i\leq\hat{x}_{jd}}} v_{di}=0$. Since, constraint (EC.11.2) is stronger than (EC.11.8), we consider the former constraint in the outer master problem. (EC.11.3) links variables v_{di} to variables x_d. (EC.11.4) implies that for each day $d\in\mathcal{D}$, exactly one of v_{di} must be equal to 1. Constraint (EC.11.6) is the set of all feasibility cuts that prevents from infeasibility in the subproblem.

Inner Master Problem

We introduce the following notation for the inner master problem.

Sets:

$\mathcal{J}'$: Index set for the second-stage solutions that are generated by the subproblem.

Parameters:

$\hat{y}_{jd}$: The number of second-stage nurses on day d in the second-stage solution $j\in\mathcal{J}'$.

Variables:

θ_{inner}: A upper bound on the second-stage cost for the current first-stage solution.

$$\text{Inner-MP} \quad \max_{w,u,\theta_{inner}} \theta_{inner} \tag{EC.11.9}$$

$$\theta_{inner}\leq\sum_{d\in\mathcal{D}}\sum_{s\in\mathcal{S}_d} c_2\hat{y}_{jd}w_{ds} \qquad j\in\mathcal{J}' \tag{EC.11.10}$$

$$\sum_{s\in\mathcal{S}_d} w_{ds}=1 \qquad d\in\mathcal{D} \tag{EC.11.11}$$

$$\sum_{t\in\mathcal{T}}\sum_{p\in\mathcal{P}_{td}} u_{tp}=\sum_{s\in\mathcal{S}_d} s w_{ds} \qquad d\in\mathcal{D} \tag{EC.11.12}$$

$$\sum_{p\in\mathcal{P}_t} u_{tp}=1 \qquad t\in\mathcal{T} \tag{EC.11.13}$$

$$u_{tp}\in\{0,1\} \qquad t\in\mathcal{T}, p\in\mathcal{P}_t \tag{EC.11.14}$$

$$w_{ds}\in\{0,1\} \qquad d\in\mathcal{D}, s\in\mathcal{S}_d \tag{EC.11.15}$$

The objective function (EC.11.9) minimize the upper bound on the second-stage cost. Constraint (EC.11.10) is the optimality cut of the inner master problem and approximate the second-stage cost. This approximation becomes more accurate as more optimality cuts are generated and added to the inner master problem. Constraints (EC.11.11)-(EC.11.12) define variables w_{ds} and link them to variables u_{tp}. Constraint (EC.11.13) implies that for each patient, exactly one of the possible scenarios realizes.

Subproblem

We define the subproblem as follows. In this model, We introduce parameters $\hat{u}_{tp}$ and $\hat{x}_d$ as follows.

$\hat{u}_{tp}$: The value of u_{tp} in the recent inner master problem.

$\hat{x}_d$: The value of x_d in the recent outer master problem.

$$\text{SP} \qquad \min_{y} \sum_{d\in\mathcal{D}} c_2 y_d \tag{EC.11.16}$$

$$\delta\hat{x}_d + \delta y_d \geq \rho \sum_{t\in\mathcal{T}} \sum_{p\in\mathcal{P}_{td}} \hat{u}_{tp} \qquad d\in\mathcal{D} \tag{EC.11.17}$$

$$0 \leq y_d \leq M_d, integer \qquad d\in\mathcal{D} \tag{EC.11.18}$$

The objective function (EC.11.16) minimizes the second-stage cost. Constraint (EC.11.17) is the demand constraint for a first-stage solution fixed by the outer master problem and an uncertainty scenario given by the inner master problem.

EC.12. Experiments on the tuning of ε for nurse planning instances

In the first experiment, we investigate the impact of parameter ε in the proposed stopping conditions. In Table EC.1, we report the computational results of the proposed Benders algorithm for $\varepsilon = \{0.5, 5, 50, 500\}$. In Table EC.1, each row gives the average results for 50 instances with different values of the incentive factor. Under "*Data Info.*", "L", "OR", and "*Sur.*" respectively give the number of weeks in the planning horizon, the number of operating room, and the number of surgeries over the planning horizon. "*Time (sec)*" gives the total computational time of algorithms in seconds. Also "*Ite.*" gives the number of iterations that algorithms repeat their main loops. Furthermore, "LB" and "UB" indicate the best lower and upper bounds in the last iteration of the algorithms and "*Gap*" computes the gap between these bounds.

In Table EC.1, we observe that the algorithm obtains the best average optimality gaps for $\varepsilon = 5$. The average optimality gaps deteriorate for very large or small values of ε. For large values of $\varepsilon = 50$ and 500, the stopping conditions terminate the master problem and the subproblem more rarely. In this case, the algorithm spends a lot of computational time to prove the optimality of the problems that is futile especially in initial iterations. As a result, as we can see in Table EC.1, the algorithm repeats fewer iterations without enough interactions between the master problem and the subproblem. In the other extreme case, when the ε is very small ($\varepsilon = 0.5$), the algorithm spends less time to improve the quality of lower and upper bounds in the master problem and the subproblem and terminates them more frequently. Therefore, the algorithm repeats more iterations compared to the case of $\varepsilon = 5$ and results in slightly higher optimality gaps.

Table EC.1. Computational results of the Benders algorithm for different values of ε in the proposed stopping conditions.

Data Info.			$\varepsilon = 0.5$					$\varepsilon = 5$					$\varepsilon = 50$					$\varepsilon = 500$				
L	*OR*	*Sur.*	*Time (sec)*	*Ite.*	*LB*	*UB*	*Gap (%)*	*Time (sec)*	*Ite.*	*LB*	*UB*	*Gap (%)*	*Time (sec)*	*Ite.*	*LB*	*UB*	*Gap (%)*	*Time (sec)*	*Ite.*	*LB*	*UB*	*Gap (%)*
2	1	39	1	29	336	336	0.00	2	29	336	336	0.00	1	29	336	336	0.00	1	29	336	336	0.00
	2	79	12	49	650	650	0.00	12	49	650	650	0.00	12	49	650	650	0.00	12	49	650	650	0.00
	3	119	340	65	958	958	0.00	341	65	958	958	0.00	343	65	958	958	0.00	349	65	958	958	0.00
	4	157	1583	90	1254	1254	0.00	1567	91	1254	1254	0.00	2377	87	1254	1255	0.02	2542	87	1254	1255	0.04
	5	202	8415	114	1577	1581	0.25	8512	113	1577	1581	0.24	10905	104	1576	1583	0.43	11211	101	1576	1588	0.71
Average			2070	69	955	956	0.05	2087	69	955	956	0.05	2728	67	955	956	0.09	2823	66	955	957	0.15
3	1	59	28	44	672	672	0.00	28	44	672	672	0.00	27	44	672	672	0.00	27	44	672	672	0.00
	2	121	7882	102	1323	1326	0.23	7235	101	1323	1326	0.23	8407	73	1322	1327	0.36	8588	72	1322	1327	0.37
	3	182	14400	171	1914	1968	2.80	14400	171	1913	1965	2.68	14400	98	1903	1973	3.65	14400	70	1876	2048	9.22
	4	240	14400	279	2505	2619	4.52	14400	258	2506	2618	4.46	14400	143	2494	2627	5.29	14400	84	2443	2717	11.23
	5	300	14400	412	3122	3297	5.56	14400	370	3122	3294	5.48	14400	187	3112	3308	6.26	14400	96	3046	3383	11.07
Average			10222	202	1907	1976	2.62	10093	189	1907	1975	2.57	10327	109	1901	1981	3.11	10363	73	1872	2029	6.38
4	1	80	877	83	1031	1031	0.00	886	84	1031	1031	0.00	1125	59	1031	1031	0.00	1015	60	1031	1031	0.00
	2	163	14400	197	2002	2070	3.38	14400	183	2001	2070	3.40	14400	87	1991	2069	3.90	14400	61	1956	2120	8.27
	3	241	14400	491	2852	3027	6.09	14400	459	2853	3022	5.90	14400	212	2842	3034	6.72	14400	81	2767	3119	12.73
	4	318	14400	561	3716	3989	7.30	14400	523	3717	3988	7.25	14400	300	3708	3999	7.81	14400	97	3623	4057	11.98
	5	397	14400	576	4551	4978	9.39	14400	565	4573	4985	8.98	14400	317	4547	4994	9.82	14400	97	4455	5025	12.80
Average			11695	382	2830	3019	5.23	11697	363	2835	3019	5.11	11745	195	2824	3025	5.65	11723	79	2766	3070	9.16

EC.13. Tri-level algorithm for the supply chain problem

The structure of the tri-level algorithm for the supply chain problem is the same as the one already explained for the nurse planning problem in Appendix EC.11. In this appendix, we present the models for the outer master problem, the inner master problem, and the subproblem of the tri-level algorithm for this application. The pseudo code for the algorithm will be the same as the one in Algorithm in Appendix EC.11.

Outer Master Problem

We introduce the following notation for the outer master problem.

Sets:

$\mathcal{J}$: Index set for the first-stage solutions that are evaluated by the inner master problem.

Parameters:

α_j : The second-stage cost for first-stage solution $j \in \mathcal{J}$.

$\hat{x}_{jf}$: 1 if facility $f \in \mathcal{F}_1 \cup \mathcal{F}_2$ is open in solution $j \in \mathcal{J}$, 0 otherwise.

Variables:

x_f : 1 if we decide to open facility $f \in \mathcal{F}_1 \cup \mathcal{F}_2$, 0 otherwise.

θ_{outer} : A lower bound on the second-stage cost in the outer master problem.

The outer master problem of the tri-level Benders algorithm reads as:

$$\text{Outer-MP} \qquad \min_{x,\theta_{outer}} \sum_{f\in\mathcal{F}_1\cup\mathcal{F}_2} c_f x_f + \theta_{outer} \tag{EC.13.1}$$

$$\theta_{outer} \geq \alpha_j - M\left[\sum_{\substack{f\in\mathcal{F}_1\cup\mathcal{F}_2:\\ \hat{x}_{jf}=0}} x_f\right] \qquad j \in \mathcal{J} \tag{EC.13.2}$$

$$x_f \in \{0,1\} \qquad f \in \mathcal{F}_1 \cup \mathcal{F}_2 \tag{EC.13.3}$$

In (EC.13.1)-(EC.13.3), the objective function (EC.11.1) minimizes the sum of the first-stage opening cost and θ_{outer} that provides a lower bound on the second-stage. Optimality cut (EC.13.2) approximates the second-stage cost and tightens θ_{outer} as the algorithm adds more cuts to the outer master problem. This cut implies that θ_{outer} will be equal to the second-stage cost α_j if solution $j \in \mathcal{J}$ is selected again. Otherwise, θ_{outer} provides a lower bound on the second-stage cost. In this constraint, M represents a very large number that we can set to the trivial value α_j.

Inner Master Problem

We introduce the following notation for the inner master problem.

Sets:

$\mathcal{J}'$: Index set for the second-stage solutions that are generated by the subproblem.

Parameters:

$\hat{y}^1_{jif}$: 1 if, in the second-stage solution $j \in \mathcal{J}'$, first-layer facility f supplies the demand of customer i.

$\hat{y}^2_{jikff'}$: 1 if, in the second-stage solution $j \in \mathcal{J}'$, customer i's demand for product k is transported from second-layer facility f' to first-layer facility f; 0 otherwise.

Variables:

θ_{inner}: A upper bound on the second-stage cost for the current first-stage solution.

u_{ip} : 1 if local scenario p realizes for customer i, 0 otherwise.

$$\text{Inner-MP} \qquad \max_{u,\theta_{inner}} \theta_{inner} \tag{EC.13.4}$$

$$\theta_{inner} \le \left(\sum_{i\in\mathcal{I}} \sum_{f\in\mathcal{F}_1} \sum_{p\in\mathcal{P}_i} (c_{ifp}\hat{y}^1_{jif} u_{ip}) + \right.$$
$$\left. + \sum_{f\in\mathcal{F}_1} \sum_{f'\in\mathcal{F}_2} \sum_{i\in\mathcal{I}} \sum_{k\in\mathcal{K}_i} \sum_{p\in\mathcal{P}_i} (c_{ikff'p}\hat{y}^2_{jikff'} u_{ip}) \right) \quad j \in \mathcal{J}' \tag{EC.13.5}$$

$$\sum_{p\in\mathcal{P}_i} u_{ip} = 1 \qquad i \in \mathcal{I} \tag{EC.13.6}$$

$$\sum_{i\in\mathcal{I}} \sum_{p\in\mathcal{P}_i} d_{ikp} u_{ip} \le b_k \qquad k \in \mathcal{K} \tag{EC.13.7}$$

$$u_{ip} \in \{0,1\} \qquad i \in \mathcal{I}\ p \in \mathcal{P}_i \tag{EC.13.8}$$

The objective function (EC.13.4) minimize the upper bound on the second-stage cost. Constraint (EC.13.5) is the optimality cut of the inner master problem and approximate the second-stage cost. This approximation becomes more accurate as more optimality cuts are generated and added to the inner master problem. Constraints (EC.13.6)-(EC.13.8) define the uncertainty set.

Subproblem

We introduce parameters $\hat{u}_{tp}$ and $\hat{x}_d$ as follows.

$\hat{u}_{ip}$: The value of u_{ip} in the recent inner master problem.

$\hat{x}_f$: The value of x_f in the recent outer master problem.

$$\text{SP} \quad \min_{y} \left(\sum_{i\in\mathcal{I}} \sum_{f\in\mathcal{F}_1} \sum_{p\in\mathcal{P}_i} (c_{ifp}\hat{u}_{ip} y^1_{if}) \right.$$
$$\left. + \sum_{f\in\mathcal{F}_1} \sum_{f'\in\mathcal{F}_2} \sum_{i\in\mathcal{I}} \sum_{k\in\mathcal{K}_i} \sum_{p\in\mathcal{P}_i} (c_{ikff'p}\hat{u}_{ip} y^2_{ikff'}) \right) \tag{EC.13.9}$$

$$\sum_{f\in\mathcal{F}_1} y^1_{if} = 1 \qquad i\in\mathcal{I} \tag{EC.13.10}$$

$$y^1_{if} \le \hat{x}_f \qquad i\in\mathcal{I}\ f\in\mathcal{F}_1 \tag{EC.13.11}$$

$$\sum_{f'\in\mathcal{F}_2} y^2_{ikff'} = y^1_{if} \qquad i\in\mathcal{I}\ k\in\mathcal{K}\ f\in\mathcal{F}_1 \tag{EC.13.12}$$

$$y^2_{ikff'} \le \hat{x}_{f'} \qquad i\in\mathcal{I}\ k\in\mathcal{K}\ f\in\mathcal{F}_1\ f'\in\mathcal{F}_2 \tag{EC.13.13}$$

$$y^1_{if} \in \{0,1\} \qquad i\in\mathcal{I}\ f\in\mathcal{F}_1 \tag{EC.13.14}$$

$$y^2_{ikff'} \in \{0,1\} \qquad i\in\mathcal{I}\ k\in\mathcal{K}\ f\in\mathcal{F}_1\ f'\in\mathcal{F}_2 \tag{EC.13.15}$$

The objective function (EC.13.9) minimizes the second-stage cost. Constraints (EC.13.10)-(EC.13.15) are the second-stage constraint (52)-(57) for a first-stage solution fixed by the outer master problem and an uncertainty scenario given by the inner master problem.

An noticeable point about the given tri-level formulation is that the optimality cut (EC.13.2) is similar to the optimality cut (EC.11.8) for the nurse planning problem, because both constraint will be redundant when the the value multiplied by M is equal or larger than 1. In the nurse planning problem, we could consider constraint (EC.11.2) as an enhanced version (EC.11.8) that could provide a valid lower bound approximation on the second-stage cost of neighbourhood solutions. However, in the supply chain problem, the structure of the problem is such that we cannot improve the original optimality cut (EC.13.2). This is why the tri-level algorithm provides poor optimality gaps in Table 5.

EC.14. Generalization to finite sets of fractional parameters

The main reason that we consider integrality constraints on $\mathcal{U}$ variables is that it makes the representation and the reformulation of the problem much easier. Here we argue that, even when the uncertainty set $\mathcal{U}$ includes a finite number of fractional points, we can use integer variables to represent it. Let us suppose that $\mathcal{U}$ includes m fractional points $\hat{u}_1, ..., \hat{u}_m$. To represent $\mathcal{U}$ using integer variables we introduce binary variables $u'_1, ..., u'_m$ where u'_i is 1 if the realized uncertainty is point u_i, 0 otherwise. Set $\mathcal{U}$ can be described as:

$$\mathcal{U} = \Big\{ u \,|\, u = \sum_{i=1}^{m} \hat{u}_i u'_i, \tag{EC.14.1}$$

$$\sum_{i=1}^{m} u'_i = 1, \tag{EC.14.2}$$

$$u'_i \in \{0,1\} \quad i\in\{1,...,m\} \Big\} \tag{EC.14.3}$$

Then in the second-stage constraints of the two-stage robust model, we can substitute variables u using $u = \sum_{i=1}^{m} \hat{u}_i u'_i$ and define the new uncertainty set $\mathcal{U}'$ as:

$$\mathcal{U}' = \Big\{ u' \,|\, \sum_{i=1}^{m} u'_i = 1, \tag{EC.14.4}$$

$$u'_i \in \{0,1\} \quad i \in \{1,...,m\} \Big\} \tag{EC.14.5}$$

In the new uncertainty set all variables are integer. One may criticize this uncertainty set by saying that it is trivial as we have one binary variable for each fractional point. Indeed, in the case of an exponential number of points, $\mathcal{U}'$ would require the same exponential number of binary variables. The answer is that it is possible to reduce the number of such variables significantly using additional information about the relation between the fractional points. For example, in the supply chain problem provided in Section 6.2, the uncertainty set includes $|P_1| \times ... \times |P_{|\mathcal{I}|}|$ fractional demand points. As the demand realizations for customers are independent (local scenarios) then they can be represented by $|P_1| + ... + |P_{|\mathcal{I}|}|$ binary variables, and that is much less than $|P_1| \times ... \times |P_{|\mathcal{I}|}|$.