Unified Branch-and-Benders-Cut for Two-Stage Stochastic Mixed-Integer Programs

Arthur Mahéo[a], Simon Belieres [1b], Yossiri Adulyasak[c], Jean-François Cordeau[c]

[a] *Amazon Research, Luxembourg*
[b] *TBS Business School, 20 Bd Lascrosses, 31000 Toulouse*
[c] *HEC Montréal, 3000 Chem. de la Côte-Sainte-Catherine, Montréal, QC H3T 2A7, Canada*

Abstract

Two-stage stochastic programs are a class of stochastic problems where uncertainty is discretized into scenarios, making them amenable to solution approaches such as Benders decomposition. However, classic Benders decomposition is not applicable to general two-stage stochastic mixed-integer programs due to the restriction that the second-stage variables should be continuous. We propose a novel Benders decomposition-based framework that accommodates mixed-integer variables in both stages as well as uncertainty in all of the recourse parameters. The proposed approach is a unified branch-and-Benders algorithm, where we use a heuristic to maintain a global upper bound and a post-processing phase to determine an optimal solution. This new approach is flexible, allowing practitioners to integrate acceleration techniques such as partial decomposition or convexification schemes. We demonstrate the efficiency of our approach versus classic ones on the stochastic server location problem; and, its generality on a new, complex stochastic problem where the second stage is a traveling salesman problem.

Keywords: Two-stage stochastic mixed-integer programs, Benders decomposition, Branch-and-Benders-Cut

1. Introduction

Stochastic mixed-integer programs (SMIPs) form a class of optimization problems that combine discrete and non-convex aspects of mixed-integer programming (Wolsey 1998) with uncertainty in the data parameters, as in stochastic programming (Birge and Louveaux 1997). In such problems, the decision variables are defined in multiple stages that characterize the moments when part of the stochastic parameter values become known.

In this study, we propose a novel Benders decomposition strategy for solving two-stage scenario-based SMIP models (Küçükyavuz and Sen 2017), where decision variables decompose into a set of *first-stage*

[1] Corresponding author: s.belieres@tbs-education.fr

decisions to be made before the realization of the random events, and a set of *second-stage* decisions (also referred to as *recourse* decisions) to be made after this information is revealed.

Let $\widetilde{\omega}$ be a random vector drawn from a discrete finite probability space $(\Omega, \mathcal{F}, \mathcal{P})$ where the *sample space* Ω defines the set of all possible outcomes, the *event space* $\mathcal{F}$ defines the set of events, an event being a set of outcomes in the sample space, and function $\mathcal{P}$ assigns each event a probability between 0 and 1. Thus, the realization of a particular scenario ω of $\widetilde{\omega}$ has a non-zero probability p_ω to occur and defines a possible outcome for the stochastic parameters. Let $\mathrm{E}[\cdot]$ be the usual mathematical expectation operator with respect to $\widetilde{\omega}$. A standard formulation for two-stage SMIPs is:

$$\begin{aligned} \min \quad & c^T x + E[h(x, \widetilde{\omega})] && (1^{\text{st}} \text{ stage}) \\ s.t. \quad & Ax \geq b && (1a) \\ & x \in X, \end{aligned}$$

where, for a given scenario ω of $\widetilde{\omega}$, $h(x, \omega)$ is defined as:

$$\begin{aligned} \min \ h(x,\omega) = \quad & q_\omega^T y_\omega && (2^{\text{nd}} \text{ stage}) \\ s.t. \quad & T_\omega x + W_\omega y_\omega \geq h_\omega && (2a) \\ & y_\omega \in Y. \end{aligned}$$

The sets $X \subseteq R_+^{n_1}$ and $Y \subseteq R_+^{n_2}$ define the domains of the first-stage variables, x, and the second-stage variables, y_ω, respectively. Input parameters $c \in R^{n_1}, A \in R^{m_1 \times n_1}, b \in R^{m_1}$ are known in advance, while $q_\omega \in R^{n_2}, T_\omega \in R^{m_2 \times n_1}, W_\omega \in R^{m_2 \times n_2}, h_\omega \in R^{m_2}$ are scenario-dependent. The objective function aims to minimize the cost of the first-stage decisions and the expected value of the second-stage costs. Prior to the realization of the random vector $\widetilde{\omega}$, the decision maker determines values for the first-stage variables that satisfy constraints (1a). The realization of a particular scenario ω of $\widetilde{\omega}$ sets the values for the stochastic parameters – i. e., the *recourse cost* q_ω, the *technology matrix* T_ω, the *recourse matrix* W_ω and the *right-hand side* h_ω. Based on this information, the decision maker formulates the recourse problem and determines values for the second-stage variables that satisfy constraints (2a).

By duplicating second-stage variables according to the scenarios, one can formulate two-stage SMIPs in an extensive form. The so-called *deterministic equivalent formulation* (DEF) is:

$$\begin{aligned} \min \quad & c^T x + \sum_{\omega \in \Omega} p_\omega q_\omega^T y_\omega && && (\text{DEF}) \\ s.t. \quad & Ax \geq b && && (3a) \\ & T_\omega x + W_\omega y_\omega \geq h_\omega && \forall \omega \in \Omega && (3b) \\ & x \in X, \ y_\omega \in Y, \forall \omega \in \Omega, \end{aligned}$$

where variables y_ω model the second-stage decisions associated with scenario ω. While this formulation can be solved by a general-purpose mixed-integer programming solver, it is unlikely to be solved in reasonable time if the number of scenarios is large. However, the DEF exhibits what is called a *block-angular structure*, which can be leveraged by decomposition-based algorithms. Indeed, once the first-stage variables are fixed, it decomposes into $|\Omega|$ independent problems.

Two-stage SMIPs can be classified according to the type of the variables involved in the second stage. When second-stage variables are continuous, $h(x, \omega)$ is a convex piece-wise linear function. As a result, $E[h(x, \widetilde{\omega})]$ satisfies convexity, and standard decomposition-based approaches, such as Benders decomposition (Benders 1962), can be applied. On the other hand, $E[h(x, \widetilde{\omega})]$ is no longer convex when the second-stage involves discrete variables, breaking down the standard decomposition-based approaches. Consequently, few methods in the literature are designed to solve two-stage SMIPs with discrete recourse, and most of them necessitate the first-stage variables to be binary (e. g., Sen and Higle 2005, Sen and Sherali 2006, Ntaimo 2010, Gade et al. 2014, Atakan and Sen 2018). There also exist algorithmic strategies that accommodate mixed-integer decisions in both stages. Unfortunately, it is often difficult to assess their scalability, whether because the corresponding articles do not provide a computational study (e. g., Carøe and Schultz 1999, Ralphs and Hassanzadeh 2014) or because the method is tested only on small instances (e. g., Ahmed et al. 2004, Guo et al. 2015).

In this paper, we introduce the Unified Branch-and-Benders-Cut (UB&BC), a new general and exact Benders decomposition-based approach (Benders 1962) for solving two-stage SMIPs. We say UB&BC is exact and general because it accommodates uncertainty in all the recourse parameters and allows for any type of variables in both stages. To maximize performance, note that our approach should primarily be used for tackling two-stage SMIPs where an efficient heuristic procedure can be implemented for the scenario subproblems. The main features of the proposed framework are its ease of implementation, as it only requires a commercial MIP solver that includes callback features, and its flexibility, as it can be used in conjunction with a wide range of acceleration techniques.

Benders decomposition, also referred to as the L-shaped method (Van Slyke and Wets 1969) in the context of stochastic programming, is a well-established algorithm that solves large-scale MIPs by dividing the computational burden into smaller parts. Specifically, the MIP is decomposed into a *master* problem and one or several *subproblems*. The master problem is a relaxation of the original problem that determines values for a subset of the decision variables and an estimate of the optimal objective function value of the subproblems. The solution obtained by solving the master problem is used to formulate the subproblems, which aim to determine values for the remaining variables. The classic Benders algorithm proceeds as follows: (i) it solves the master problem to optimality, (ii) it uses the solution found to formulate the subproblems, (iii) it solves the subproblems to determine a feasible solution to the original MIP, (iv) using LP duality, it derives

the so-called Benders cuts to add to the master problem; finally, (v) it repeats from point (i) until a provably optimal solution to the original MIP is found. For a recent survey on Benders decomposition, we refer the interested reader to Rahmaniani et al. (2017). Note that the Benders algorithm can also be integrated inside a branch-and-cut (B&C) scheme, where the master problem is solved only once. Subproblems act as separation problems to generate Benders cuts and are solved at each branching node or only when an integer master problem solution is found in the tree. The resulting branch-and-Benders-cut (B&BC, Fortz and Poss 2009, De Camargo et al. 2011, Gendron et al. 2016) has become the standard implementation and is the basis of the UB&BC.

The standard Benders algorithm does not directly apply to two-stage SMIPs with discrete variables in the second stage. As explained earlier, this is due to the fact that the algorithm's convergence relies on Benders cuts obtained by applying standard linear programming duality theory to the subproblems. Obviously, we cannot rely on the same mechanism to solve two-stage SMIPs with discrete recourse. In the context of two-stage stochastic programming, the challenge of extending the Benders decomposition to accommodate discrete variables in the second stage has led to multiple studies. A common strategy is to solve the subproblems to integer optimality and develop valid cuts without using the dual information (Laporte and Louveaux 1993). Another approach consists in solving the LP relaxation of the scenario subproblems to generate standard Benders cuts and using cutting-plane procedures to characterize convexifications of the second-stage problems (e. g., Sherali and Fraticelli 2002, Sen and Higle 2005).

To accommodate general mixed-integer variables in the second stage, we present an intermediate approach that uses linear programming duality theory to generate standard Benders cuts and solves to integer optimality subproblems that correspond to promising master solutions. We propose a new Benders decomposition-based algorithm where a modified B&C is used to solve the master problem. Whenever an integer solution $\hat{x}$ is found in a branch-and-bound-tree of the master, (i) we solve the scenario subproblem LP relaxations to compute a lower bound $lb(opt(\hat{x}))$ associated with this integer master problem solution and generate standard Benders cuts, and (ii) we solve the scenario subproblems heuristically to determine a valid upper bound $ub(opt(\hat{x}))$. In Figure 1, we show the progress of the lower bound, $lb(opt(\hat{x}))$, and the upper bound, $ub(opt(\hat{x}))$, with successive candidate solutions. Considering that the B&C finishes at iteration i, there exists a gap between the best lower bound, lb^*, and the best upper bound, ub^*. Therefore, we retain a set of a set of *open solutions*: solutions whose LP relaxation lower bound $lb(opt(\hat{x}))$ is lower than the best upper bound. This process, which takes advantage of a single branch-and-bound tree in solving the master problem, successively refines global upper and lower bounds and guarantees that the global optimal solution is among the open solutions remaining at the end of the branch-and-bound process of the master problem. Finally, we then need to solve these open solutions to integer optimality to determine the global optimum (OPT).

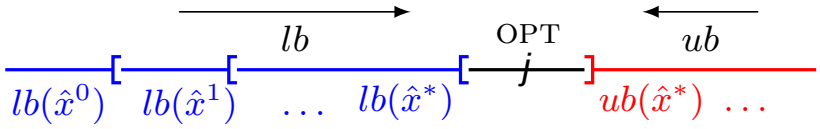

Figure 1: UB&BC makes use of a heuristic to obtain the upper bound during the branch-and-bound process of the master problem. There could exist an integrality gap which must be closed through a post-processing procedure applied to the open solutions at the end of the branch-and-bound process.

Our contribution is threefold. First, we introduce a general exact algorithmic strategy for solving two-stage stochastic programs with general mixed-integer variables in both stages. Second, we demonstrate the flexibility of our solution framework by incorporating acceleration techniques such as partial Benders decomposition (Crainic et al. 2014, 2016) or multi-term disjunctive cuts (Chen et al. 2011, 2012). Third, we provide an extended series of experiments to assess the performance of our framework and gain insight into how its components impact the speed of convergence. Two problems are under study: the stochastic server location problem (SSLP, Ntaimo and Sen 2005), which is frequently used to benchmark algorithms for two-stage stochastic programs with discrete recourse (e. g., Ntaimo and Sen 2005, Guo et al. 2015, Gade et al. 2014, Atakan and Sen 2018, Qi and Sen 2017), and the two-stage stochastic traveling salesman problem with outsourcing (2TSP). To the best of our knowledge, this is the first study to provide an exact and effective method for solving a two-stage SMIP where the scenario subproblem is a TSP, which is a difficult combinatorial problem in itself.

The remainder of the paper is organized as follows. In Section 2, we review the existing solution algorithms for two-stage SMIP models with discrete second-stage variables. Section 3 is dedicated to the description of UB&BC. It also provides a discussion on the scope of problems that can be efficiently solved by UB&BC, as well as a description of the two acceleration techniques stated above. Section 4 presents the experimental setup. We provide an extensive computational study in Section 5. In Section 6, we conclude the paper and discuss future work.

2. Literature review

This section aims to review the existing exact algorithmic strategies for two-stage SMIP models with discrete recourse. The reviewed methods are summarized in Table 1 in a manner similar to Trapp et al. (2013) and Ralphs and Hassanzadeh (2014). For each method, we indicate the type of variables it accommodates in both stages, as well as the assumptions made regarding the potential stochastic parameters.

Laporte and Louveaux (1993) extend the L-shaped method to accommodate general mixed-integer variables in the second stage. They propose a branch-and-Benders-cut method where valid Benders optimality cuts are derived from the objective function values of the scenario subproblems. The resulting algorithm

converges in a finite number of iterations, but suffers from two shortcomings: it is only applicable in the case of pure binary first-stage variables, and, it requires the scenario subproblems to be solved to integer optimality to compute the L-shaped cuts ensuring convergence.

Angulo et al. (2016) propose an extension of the algorithm where, for each master problem solution, the linear relaxation of the scenario subproblems are solved, allowing to derive continuous L-shaped cuts. If these continuous L-shaped cuts are binding, they are added to the master and the branch-and-bound process proceeds. Otherwise, the scenario subproblems are solved to integer optimality, and integer L-shaped cuts are added to the master. Computational experiments demonstrate that this alternating cut strategy significantly increases performance. Nevertheless, this technique does not allow to bypass the requirements for the application of the integer L-shaped method, i. e., pure binary first-stage variables.

Carøe and Tind (1998) propose a generalization of the L-shaped method and use the general duality theory to develop valid Benders optimality cuts. A cutting plane algorithm is used to solve the scenario subproblems to integer optimality. Based on the obtained solutions, "nonlinear dual variables" that take the form of Chvátal functions are used to generate valid Benders optimality cuts. The generalized L-shaped method handles all models that do not involve continuous variables in the second stage. However, the article does not report numerical results.

To avoid solving scenario subproblems to integer optimality, multiple decomposition-based algorithms proposed in the literature embed a cutting-plane procedure to progressively characterize the convex hulls of the subproblem LP relaxations. Sherali and Fraticelli (2002) present an L-shaped method where the scenario subproblems are approximated using the Reformulation-Linearization Technique (RLT, Sherali and Adams 1999) and lift-and-project cutting planes. Cuts are expressed as functions of the first-stage variables and thus valid for all scenario subproblems. This approach tackles programs with binary variables in both stages and continuous variables in the second stage. Sherali and Zhu (2006) present a method that also accommodates continuous variables in the first stage. They propose a decomposition-based branch-and-bound algorithm that follows a hyperrectangular partitioning process and uses a RLT cutting-plane algorithm to convexify the scenario subproblems.

Sen and Higle (2005) present the C^3 theorem and demonstrate that the valid inequalities associated with a given scenario subproblem can be used to derive valid inequalities for any other scenario subproblem. Based on the C^3 theorem, the authors propose the D^2 algorithm, where the master and the subproblems are obtained from the convexification of two disjunctive programs. This approach tackles problems with binary variables in both stages and continuous variables in the second stage. Ntaimo and Sen (2005) present an extension of the D^2 algorithm that accommodates different stochastic parameters. Another extension to the D^2 algorithm is proposed by Sen and Sherali (2006) and allows general mixed-integer variables in the second stage. In this approach, scenario subproblems are solved with a partial branch-and-bound, and dual

coefficients derived from the trees are used to develop valid Benders cuts.

Gade et al. (2014) integrate a convexification procedure based on Gomory cuts into the L-shaped method and solve programs with binary variables in the first stage and general integer variables in the second stage. Qi and Sen (2017) develop new convexification schemes based on multi-term disjunctive cuts and propose the ancestral Benders Cuts (ABC), which allow for general mixed-integer variables in both stages. Other studies on two-stage SMIP with discrete recourse involve solution methods based on value function reformulation (Ahmed et al. 2004, Kong et al. 2006, Trapp et al. 2013, Ralphs and Hassanzadeh 2014), dual decomposition (Carøe and Schultz 1999), progressive hedging (Guo et al. 2015, Atakan and Sen 2018), or Gröbner basis (Schultz et al. 1998). Recent work by Larsen et al. (2022) leverages the progress in machine learning (ML) and integrates generic approximators into the L-shaped framework of Angulo et al. (2016) to estimate the scenario subproblem optimal values. The proposed heuristic is promising and computational experiments indicate that hybridizing ML and OR techniques is a clear direction for efficiently solving two-stage SMIPs.

3. Unified Branch-and-Benders-Cut

We now describe our Benders decomposition-based strategy for solving two-stage stochastic mixed-integer programs with general mixed-integer variables in both stages. We first present the Unified Branch-and-Benders-Cut (UB&BC) which operates a modified branch-and-cut to identify a set of open solutions and performs a post-processing phase to determine the global optimum. Second, we discuss the main features of the proposed method, namely, its generality and ease of implementation, its range of applicability, and its flexibility to be combined and enhanced with other problem-solving components. We then present two such components. We discuss how using an enhanced Benders decomposition strategy Crainic et al. (2014, 2016) rather than a standard Benders decomposition can improve the convergence of UB&BC. Note that this technique relies on specific structural assumptions as it solely applies in the case of a fixed recourse matrix W_ω and a fixed recourse cost q_ω, which is the case for many two-stage stochastic problems found in the literature (Ntaimo and Sen 2005, Zheng et al. 2013, Elçi and Hooker 2022). We also evaluate how convexification procedures can be incorporated to construct polyhedral approximations of the subproblems and further enhance our approach.

3.1. Description of the algorithm

In the context of two-stage stochastic programming, a standard Benders decomposition consists of a master problem that prescribes decisions for the first-stage variables, and a set of scenario subproblems that compute optimal values for the second-stage variables given fixed values for x. Let $\mathcal{F}^\omega$ and $\mathcal{O}^\omega$ denote the sets of extreme rays and extreme points of the dual subproblem polyhedron associated with scenario ω,

	1st Stage			2nd Stage			Stochastic parameters			
	R	B	Z	R	B	Z	T_ω	W_ω	h_ω	q_ω
Laporte and Louveaux (1993)		⋆		⋆	⋆	⋆		⋆	⋆	⋆
Carøe and Tind (1998)	⋆	⋆	⋆		⋆	⋆	⋆		⋆	
Schultz et al. (1998)	⋆				⋆	⋆			⋆	
Carøe and Schultz (1999)	⋆	⋆	⋆	⋆	⋆	⋆	⋆		⋆	⋆
Sherali and Fraticelli (2002)		⋆		⋆	⋆		⋆	⋆	⋆	⋆
Ahmed et al. (2004)	⋆	⋆	⋆		⋆	⋆		⋆	⋆	⋆
Sen and Higle (2005)		⋆		⋆	⋆		⋆		⋆	
Kong et al. (2006)		⋆	⋆		⋆	⋆			⋆	
Sen and Sherali (2006)		⋆		⋆	⋆	⋆	⋆		⋆	
Sherali and Zhu (2006)		⋆		⋆	⋆		⋆	⋆	⋆	
Ntaimo (2010)		⋆		⋆	⋆			⋆		⋆
Trapp et al. (2013)		⋆	⋆		⋆	⋆			⋆	
Gade et al. (2014)		⋆			⋆	⋆	⋆	⋆	⋆	⋆
Ralphs and Hassanzadeh (2014)	⋆	⋆	⋆	⋆	⋆	⋆	⋆		⋆	
Guo et al. (2015)	⋆	⋆	⋆	⋆	⋆	⋆	⋆		⋆	⋆
Angulo et al. (2016)		⋆		⋆	⋆	⋆		⋆	⋆	⋆
Qi and Sen (2017)	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆
Atakan and Sen (2018)		⋆		⋆	⋆	⋆	⋆	⋆	⋆	⋆
Current study										
UB&BC	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆

Table 1: Characteristics of existing exact methods for two-stage SMIP models with discrete recourse

respectively. The standard master problem is formulated as follows:

$$\min \quad c^T x + \sum_{\omega \in \Omega} p_\omega z_\omega \qquad \text{(Standard master problem)}$$

$$s.t. \quad Ax \geq b \tag{4a}$$

$$f^T(h_\omega - T_\omega x) \leq 0 \qquad \forall f \in \mathcal{F}^\omega, \forall \omega \in \Omega \tag{4b}$$

$$o^T(h_\omega - T_\omega x) \leq z_\omega \qquad \forall o \in \mathcal{O}^\omega, \forall \omega \in \Omega \tag{4c}$$

$$x \in X.$$

The master's objective function computes the cost of the first-stage solution and an expected recourse cost, with variable z_ω providing the expected recourse cost for scenario $\omega \in \Omega$. Constraints (4a) characterize the feasible region for the first-stage variables. Constraints (4b) and (4c) are the standard feasibility and optimality cuts added dynamically after solving the scenario subproblems. Given a first-stage solution $\hat{x}$ computed by the master problem, the subproblem associated with scenario ω is formulated as follows:

$$\min \quad q_\omega^T y_\omega \qquad \text{(Standard scenario subproblem)}$$

$$s.t. \quad W_\omega y_\omega \geq h_\omega - T_\omega \hat{x} \tag{5a}$$

$$y_\omega \in Y.$$

The scenario objective function reflects the cost associated with the second-stage solution while constraints (5a) characterize the feasible region for the second-stage variables.

As explained earlier, one cannot employ the classical Benders algorithm when subproblems contain discrete variables, since these discrete variables prevent us from applying standard linear programming duality and generating Benders cuts. In practice, there are two main strategies for allowing the Benders algorithm to deal with discrete scenario subproblems. The first strategy consists in relaxing the integrality constraints on the second-stage variables to generate standard Benders cuts, and integrate a convexification procedure to progressively determine the convex hull of the LP relaxations of the scenario subproblems. The second strategy consists in solving the scenario subproblems to integer optimality and using a procedure to derive valid cuts. Our approach is intermediate, in the sense that we relax the integrality constraints on the second-stage variables to generate standard Benders cuts and we delay solving the subproblems to integer optimality to a post-processing phase, at the end of which we obtain the global optimum. The UB&BC is described in Algorithm 1.

The UB&BC operates in two phases. In the first phase, it solves the master problem using a modified branch-and-Benders-cut (B&BC). Whenever an integer solution $\hat{x}$ with estimated recourse costs z_ω and an objective function value lower than the best-known bound is found at a branch-and-bound node, it (i) solves the LP relaxations of the scenario subproblems and adds the resulting Benders cuts to the master

Algorithm 1: Unified Branch-and-Benders-Cut

Data: An original problem, P, a subproblem heuristic, H, and an optimality tolerance, ϵ

$Z = ;, ub^* = 1$

Define the master problem, MP, and the scenario subproblems, SP_ω, from P

begin Solve MP by a B&B and apply the following steps at each node in the search tree:

- **if** *LP relaxation* ub^* *OR node is infeasible* **then**
  - Fathom branch
- **else if** *An integer solution* $\hat{x}$ *of the* MP *is found* **then**
  - **foreach** *Scenario* ω *2* Ω **do**
    - Solve the LP relaxation of subproblem $SP_\omega(\hat{x})$
    - Add the corresponding Benders cut to MP
  - **if** *All scenario subproblem LP relaxations are feasible* **then**
    - Compute the improved lower bound $lb(opt(\hat{x}))$
    - **if** $lb(opt(\hat{x})) > ub^*$ **then**
      - Prune the current node
    - **else**
      - Add $(\hat{x}, lb(opt(\hat{x})))$ to Z
      - **foreach** *Scenario* ω *2* Ω **do**
        - Solve subproblem $SP_\omega(\hat{x})$ with the heuristic H
      - **if** *A heuristic solution was found for all the scenario subproblems* **then**
        - Compute the heuristic upper bound $ub(opt(\hat{x}))$
        - $ub^* = \min(ub^*, ub(opt(\hat{x})))$
- **else**
  - Choose a variable to branch on

if *The B&B terminated with a gap greater than* ϵ **then**

- Rank elements of Z by ascending order according to their lower bound values
- **while** Z 6= **do**
  - Select $(\hat{x}, lb(opt(\hat{x})))$ from Z
  - **if** $lb(opt(\hat{x}))$ ub^* **then**
    - **foreach** *Scenario* ω *2* Ω **do**
      - Solve $SP_\omega(\hat{x})$ as a MIP
    - **if** *All scenario subproblems are optimal* **then**
      - Compute $opt(\hat{x})$
    - $ub^* = \min(ub^*, opt(\hat{x}))$

Result: ub^*, the optimal solution of P

problem, and (ii) solves the scenario subproblems heuristically to determine a feasible second-stage solution when possible. For each scenario $\omega \in \Omega$, let z_ω^{MIP}, z_ω^{LP}, and z_ω^H denote the optimal value of the MIP subproblem, the optimal value of the subproblems LP relaxation, and the objective function value of the heuristic solution, respectively. Let $opt(\hat{x}) = c^T\hat{x} + \sum_{\omega\in\Omega} p_\omega z_\omega^{MIP}$ be the optimal objective function value associated with master problem solution $\hat{x}$. By solving the LP relaxations of the scenario subproblems and using the subproblem heuristic, we can potentially derive two bounds on $opt(\hat{x})$.

For a given master problem integer solution $\hat{x}$, if there is at least one subproblem LP relaxation that is not feasible, there exists no feasible solution to the original problem based on $\hat{x}$. On the other hand, if all scenario subproblem LP relaxations are feasible, one can hope to determine a feasible solution to the original problem by solving the subproblems to integer optimality. This process to solve all the MIP subproblems to integer optimality, however, can be very time-consuming and should only be performed when necessary. Thus, we further examine $\hat{x}$ if it should be included in the set of open solutions. Recalling that for each scenario $\omega \in \Omega$, both z_ω^{LP} and z_ω reflect lower bounds on the optimal value of the MIP subproblem, one can determine a lower bound on $opt(\hat{x})$, i.e., $lb(opt(\hat{x})) = c^T\hat{x} + \sum_{\omega\in\Omega} p_\omega . \max\left\{z_\omega, z_\omega^{LP}\right\}$. If the lower bound $lb(opt(\hat{x}))$ is worse than the incumbent, the candidate master solution $\hat{x}$ is discarded, the current node is pruned and the master problem branch-and-bound process is resumed. Otherwise, $\hat{x}$ is retained as an open solution and the scenario subproblems are solved heuristically.

If a heuristic solution is found for all scenario subproblems, we compute an upper bound on $opt(\hat{x})$, i.e., $ub(opt(\hat{x})) = c^T\hat{x} + \sum_{\omega\in\Omega} p_\omega z_\omega^H$. This heuristic upper bound can be used to prune nodes in the branch-and-bound (B&B) tree. Specifically, if $ub(opt(\hat{x}))$ is better than the incumbent ub^*, it replaces it, and all fractional/integer master solutions explored with a superior objective function value are eliminated. Note that the only way to update the incumbent is to discover a heuristic bound $ub(opt(\hat{x}))$ with a lower cost. This management of the incumbent ensures not pruning master solutions that yield the global optimum – i.e, $\hat{x}$ such that $opt(\hat{x})$ is optimal for P. The B&B terminates if there is no active node remaining or if the gap between the best known bounds satisfies a given tolerance.

If the gap at the end of the first phase satisfies the optimality tolerance, the algorithm terminates. Otherwise, we have a set of open master solutions which contains the global optimum, i. e., Z contains a master problem solution $\hat{x}$ such that $opt(\hat{x})$ is the optimal solution of P. The second step is a *post-processing* phase where we rank the open solutions according to their lower bound $lb(opt(\hat{x}))$, and we solve to integer optimality the associated subproblems to compute $opt(\hat{x})$. If $opt(\hat{x})$ provides a better upper bound than the incumbent, it becomes the new incumbent. As in the first phase, the incumbent value is used to discard saved master solutions $\hat{x}$ such that $lb(opt(\hat{x}))$ is worse than the incumbent. At the end of the post-processing, the solution with the best combined objective value is the global optimum.

We illustrate the execution of UB&BC on a small numerical example in Appendix 7.

3.2. Discussion

Our approach is an exact algorithm for solving general two-stage SMIPs as it accommodates uncertainty in all the recourse parameters and it allows for any type of first-stage and second-stage variables. The main features of UB&BC are its generality and its ease of implementation, as the framework only requires a commercial MIP solver with callback features. Another strength of our approach is its flexibility and its ability to easily incorporate acceleration techniques such as advanced Benders decomposition or convexification procedures. In the following subsection, we describe two solution techniques that can be used in conjunction with UB&BC.

It is also important to note that UB&BC, in its most elementary form, can efficiently solve multiple optimization problems. In Section 5, we computationally demonstrate this assertion by solving instances of the stochastic server location problem (SSLP) and the stochastic traveling salesman problem with outsourcing (2TSP).

The performance of UB&BC is dependent on the quality of the heuristics used for solving the scenario subproblems. Indeed, since these heuristics are used to determine valid upper bounds in the first phase of UB&BC, they influence both the amount of computational effort required to terminate the branch-and-bound process as well as the number of open solutions to be post-processed in the second phase. For these reasons, it appears evident that our framework should primarily be used for solving two-stage SMIPs where the subproblems, which comprises a set of deterministic problems, one for each scenario, can be solved by effective heuristics procedures. Meanwhile, there exists a wide range of classical problems for which an effective solution heuristic exists for the deterministic subproblems: the traveling salesman problem (Lin and Kernighan 1973), the vehicle routing problem (Laporte et al. 2000), the knapsack problem (Wilbaut et al. 2008), the job shop scheduling problem (Gere Jr 1966), the parallel machine scheduling problem (Mokotoff 2001), and the unit commitment problem (Najafi et al. 2012). Thus, we can leverage efficient heuristics originally developed for a wide range of combinatorial problems in this framework. Many two-stage SMIPs have a recourse problem which is an extension of one of these classical problems (e. g., Sen and Higle 2005, Zheng et al. 2013, Angulo et al. 2016). If one manages to extend existing heuristics to accommodate the specific features of the considered recourse problem, then one can use our solution framework and expect to achieve good performance. In the case where no efficient heuristic procedure is available for the recourse problem, one can heuristically solve the subproblems by truncating the solution process of a general-purpose MIP solver, e. g.stop when the first feasible solution is found. While this configuration does not maximize the performance of UB&BC, we computationally demonstrate in Section 12.2.5 that our framework can achieve satisfying results without using a tailored subproblem heuristic. We also demonstrate the uses of simple heuristics for different SSLP variants and various TSP heuristics for the 2TSP in this work.

3.3. Accelerating UB&BC

Because of its flexibility, our framework can be combined with various acceleration techniques. We next describe two problem-solving components that can be used in conjunction with UB&BC and further improve its speed of convergence, namely, partial Benders decomposition and multi-term disjunctive cuts.

3.3.1. Partial Benders reformulation.

It is recognized that standard Benders decomposition often yields a weak computational performance (Rahmaniani et al. 2017). Indeed, as the subproblems are projected out from the master problem and replaced with Benders cuts, the algorithm would need to generate a significant number of Benders cuts to capture the elements present in the subproblems. Recently, multiple studies have focused on the development of enhanced Benders decomposition strategies to circumvent this effect and reduce the number of iterations until convergence.

Crainic et al. (2014, 2016) recently proposed the *Partial Benders Decomposition* (PBD) for solving two-stage stochastic programs with continuous second-stage variables, fixed recourse matrix, and fixed recourse cost. As a result, in this section we omit the scenario index from parameters q_{omega} and T_{omega}. The idea is to strengthen the master problem with information relative to the scenario subproblems, which can be done by adding variables and constraints associated with an artificial scenario ω' derived from the original scenarios. In partial Benders reformulation, a valid artificial scenario ω' must define a convex combination of the original scenarios, where the weight of each original scenario $\omega \in \Omega$ is characterized as $\alpha_{\omega}^{\omega'} \geq 0$, $\sum_{\omega \in \Omega} \alpha_{\omega}^{\omega'} = 1$. Specifically, the artificial scenario yields the strongest possible bound when $\alpha_{\omega}^{\omega'} = p_{\omega}$, $\forall \omega \in \Omega$ (Crainic et al. 2016). The stochastic parameters associated with the artificial scenario ω' are: $T_{\omega'} = \sum_{\omega \in \Omega} \alpha_{\omega}^{\omega'} T_{\omega}$ and $h_{\omega'} = \sum_{\omega \in \Omega} \alpha_{\omega}^{\omega'} h_{\omega}$. By including the second-stage requirement associated with the artificial scenario into the standard master problem, one can formulate the enhanced master problem as follows:

$$\min \quad c^T x + \sum_{\omega \in \Omega} p_{\omega} z_{\omega} \qquad \text{(Enhanced master problem)}$$

$$s.t. \quad Ax \geq b \tag{6a}$$

$$T_{\omega'} x + y_{\omega'} \geq h_{\omega'} \tag{6b}$$

$$q^T y_{\omega'} = \sum_{\omega \in \Omega} p_{\omega} z_{\omega} \tag{6c}$$

$$f^T (h_{\omega} - Tx) \leq 0 \qquad \forall f \in \mathcal{F}^{\omega}, \forall \omega \in \Omega \tag{6d}$$

$$o^T (h_{\omega} - Tx) \leq z_{\omega} \qquad \forall o \in \mathcal{O}^{\omega}, \forall \omega \in \Omega \tag{6e}$$

$$x \in X, \quad y_{\omega'} \geq 0.$$

Continuous variables $y_{\omega'}$ model the second-stage decisions associated with the artificial scenario. Constraints (6b) ensure that all master problem solutions are feasible for the artificial scenario, while constraints (6c) enforce the estimate of the recourse costs to reflect the cost of second-stage decisions taken for the artificial scenario.

Crainic et al. (2016) demonstrate that the enhanced master problem is a relaxation of the original problem (DEF) such that, when second-stage variables are continuous, the classical Benders algorithm based on the enhanced master problem converges to an optimal solution. Consequently, when solving two-stage SMIPs with discrete recourse, UB&BC based on a partial Benders reformulation also converges to an optimal solution. Thus, we can employ two master problem formulations for UB&BC: the standard master problem and the enhanced master problem. The latter is preferred as it yields stronger bounds, but it can only be applied for two-stage SMIPs with fixed recourse matrix and fixed recourse cost.

3.3.2. Convexification procedure.

As discussed in Section 2, many solution algorithms for two-stage SMIPs with discrete recourse are based on polyhedral approximations. Such approaches (Sen and Higle 2005, Ntaimo 2010, Gade et al. 2014, Qi and Sen 2017) embed a cutting-plane procedure that progressively characterizes the convex hulls of each second-stage mixed-integer program, such that integral second-stage solutions can be obtained while solving the LP relaxation of the scenario subproblems. The integration of such a convexification procedure appears to be suited when tackling two-stage SMIPs where the second-stage problems have weak LP relaxations. Recall that in Phase I of UB&BC, for each integral master solution $\hat{x}$ we compute a lower bound $lb(opt(\hat{x}))$ by combining first-stage costs with the maximum value between estimated recourse costs and optimal solutions to the subproblems LP relaxation. This lower bound is used to discard master solutions as well as to prune nodes in the branch-and-bound tree, thus reducing the computational burden of both Phases I and II. Therefore, the quality of this lower bound is crucial and the use of a convexification procedure for tightening the LP relaxation of the scenario subproblems can be beneficial.

The CPT algorithm (Chen et al. 2011, 2012) is a finite disjunctive programming (Balas 1979) procedure that characterizes the convex hull of general mixed-integer linear programs with bounded integer variables. Thus, embedding this procedure in UB&BC does not reduce our scope as the CPT algorithm does not require structural assumptions on the subproblems. As a pure cutting-plane procedure, the CPT algorithm iteratively: (i) solves the LP relaxation of the considered problem, (ii) identifies a fractional variable in the solution, (iii) determines a disjunction on that variable to construct a cut-generating linear program (CGLP), (iv) solves the CGLP, and (v) adds the resulting cut to the LP relaxation of the problem. These steps are iterated until an optimal solution satisfying integrality requirements is obtained. Specifically, the CPT algorithm keeps track of a branching tree that defines a hierarchy of multi-term disjunctions, which are used to construct an enhanced CGLP. For more details on the CPT algorithm, we refer the interested

reader to Chen et al. (2011, 2012).

In the context of a Benders decomposition-based algorithm, as the CPT algorithm is operated on the subproblems, it should be noted that the formulation of the CGLP is based on the decisions taken for the master problem (Qi and Sen 2017). As a consequence, the cutting planes produced by the CGLPs, as well as the Benders cuts obtained from the convexified subproblem, are not global. They are only valid in the subtree rooted at the node where they were generated in the first-stage branch-and-bound. Specifically, let $\hat{x}$ and $\hat{\hat{x}}$ be distinct candidate master solutions and let ω be a scenario. The cutting planes obtained by applying the CPT algorithm to subproblem $SP_\omega(\hat{x})$ are valid for subproblem $SP_\omega(\hat{\hat{x}})$ iff $\hat{\hat{x}}$ belongs to the subtree of $\hat{x}$ in the first-stage branch-and-bound. Thus, to gradually define the convex hull of the scenario subproblems while ensuring the convergence of the Benders algorithm, scenario subproblems associated with a given master solution $\hat{x}$ should only be reinforced with cutting planes inherited from ancestor nodes in the master branch-and-bound tree. In addition, as scenario subproblems $SP_\omega(\hat{x})$ are convexified according to the branching decisions that led to master solution $\hat{x}$, the resulting Benders cuts are local and valid solely for nodes in the subtree of $\hat{x}$ in the first-stage branch-and-bound. We refer the interested reader to Qi and Sen (2017) for more details on the integration of the CPT algorithm within a Benders decomposition-based algorithm. Phase I of UB&BC combined with the CPT algorithm is described in Appendix 8.

4. Benchmark problems

We consider two different stochastic optimization problems as benchmarks for the UB&BC: the stochastic server location problem (SSLP) and the stochastic traveling salesman problem with outsourcing (2TSP). For each problem, we present its *deterministic equivalent formulation*, its standard Benders reformulation, the heuristic used to solve the Benders subproblem, and the characteristics of the test instances. Note that both problems involve fixed recourse matrices and fixed recourse costs, which enables using partial Benders reformulation. These reformulations are presented in Appendix 9.

4.1. Stochastic server location problem

The server location problem (Berman and Mandowsky 1986) is a variant of the facility location problem with a focus on congestion. It aims to locate a number of servers (facilities) with fixed capacity so as to maximize service quality. Service quality is determined as a measure that each client gives to every server. Unlike in the facility location problem, in the server location problem one can decide to pay a fee for unmet demand instead of having to open new servers.

The stochastic variant of the server location problem captures uncertainty regarding customer demands. In this section, we describe the SSLP studied by Ntaimo and Sen (2005), where first-stage decisions are binary and second-stage decisions can be binary and continuous. Note that two SSLP variants that allow

general integer variables in both stages are also used as benchmarks for the UB&BC. These variants are described in Appendix 11.

4.1.1. Stochastic mixed-integer program.

Let I and J denote the sets for the clients and the potential server locations, respectively. Installing a server at location j incurs a cost c_j. Only one server can be installed per location, and no more than V servers can be installed in total. Let Z denote a given set of zones and let J_z be the subset of server locations that belong to zone $z \in Z$. There is a requirement that at least w_z servers be located in a zone $z \in Z$.

All servers have the same resource capacity of D units. For each client $i \in I$ and each server $j \in J$, there is a resource demand of d_{ij} units. As a client $i \in I$ is assigned to a server $j \in J$, d_{ij} units are used to served the demand and doing so generates q_{ij} units of revenue. Each client must be served by exactly one server. If the total demand assigned to a server $j \in J$ exceeds its capacity, an overflow is necessary, incurring a penalty cost of q_{j0} per unit. Note that revenues q_{ij} and q_{j0} are described as scenario-dependent parameters in the model proposed by Ntaimo and Sen (2005), but they do not vary from one scenario to another in the instances they propose. We thus define revenues as scenario-independent parameters for the sake of simplicity.

Each scenario $\omega \in \Omega$ has a probability p_ω to occur. The stochastic aspects of the problem are represented by binary parameters h_i^ω that indicate whether or not client i is present in scenario ω. The decision variables are the following:

- binary variables x_j take value 1 if and only if a server is located at site j
- binary variables y_{ij}^ω take value 1 if and only if client i is served by server j in scenario ω
- continuous variables y_{j0}^ω represent the overflow associated with server j in scenario ω

The stochastic server location problem is formulated as follows:

$$\min \quad \sum_{j\in J} c_j x_j - \sum_{\omega\in\Omega} p_\omega \left(\sum_{i\in I}\sum_{j\in J} q_{ij} y_{ij}^\omega - \sum_{j\in J} q_{j0} y_{j0}^\omega \right) \tag{SSLP}$$

$$s.t. \quad \sum_{j\in J} x_j \le V \tag{7a}$$

$$\sum_{j\in J_z} x_j \ge w_z \qquad \forall z \in Z \tag{7b}$$

$$\sum_{i\in I} d_{ij} y_{ij}^\omega - y_{j0}^\omega \le D x_j \qquad \forall j \in J, \forall \omega \in \Omega \tag{7c}$$

$$\sum_{j\in J} y_{ij}^\omega = h_i^\omega \qquad \forall i \in I, \forall \omega \in \Omega \tag{7d}$$

$$x_j \in B, \quad y_{ij}^\omega \in B, \quad y_{j0} \ge 0.$$

The objective function aims to minimize the total cost, i. e., the difference between the total installation cost and the total expected revenue. The constraint (7a) ensures that no more than V servers are installed. Constraints (7b) ensure that the required number of servers are installed in the different zones. For each server and each scenario, constraints (7c) ensure that resource capacities are not exceeded and regulate overflows accordingly. The requirement that each client present in a scenario is served by exactly one server is enforced by constraints (7d).

4.1.2. Benders decomposition.

The *deterministic equivalent formulation* presented above can be decomposed into a two-stage stochastic mixed-integer program where the first stage consists in locating the servers and the second stage consists in assigning clients to the servers. This yields a valid Benders decomposition where, once the first-stage variables are fixed, the subproblem decomposes into $|\Omega|$ parts, yielding one subproblem per scenario.

To generate cuts we use the dual relaxed subproblem associated with the projected y variables. Let I^ω be the set of clients that are present in scenario $\omega \in \Omega$. For each scenario $\omega \in \Omega$, we define the dual variables u^ω and v^ω associated with constraints (7c) and (7d), respectively. Let the dual constraints (8a) and (8b) correspond to the variables of the form y^ω_{ij} and y^ω_{j0}, respectively. Given a first-stage solution $\hat{x}$ computed by the master problem, the cut-generating subproblem associated with scenario $\omega \in \Omega$ is formulated as follows:

$$\begin{aligned}
\max \quad & \sum_{i\in I} h^\omega_i v^\omega_i - \sum_{j\in J} D\hat{x}_j u^\omega_j && && \text{(Sub}[\omega]\text{)}\\
s.t. \quad & -d_{ij}u^\omega_j + v^\omega_i \le -q_{ij} && \forall i \in I, j \in J && \text{(8a)}\\
& u^\omega_j \le q_{j0} && \forall j \in J && \text{(8b)}\\
& u^\omega_j \ge 0, v^\omega_i \in R.
\end{aligned}$$

Then, the master problem is formulated as follows:

$$\begin{aligned}
\min \quad & \sum_{j\in J} c_j x_j - \sum_{\omega\in\Omega} p_\omega z_\omega && && \text{(Standard master)}\\
s.t. \quad & \text{(7a)} - \text{(7b)}\\
& \sum_{i\in I} h^\omega_i v^\omega_i - \sum_{j\in J} D x_j u^\omega_j \le z_\omega && (u^\omega, v^\omega) \in \mathcal{O}^\omega, \forall \omega \in \Omega && \text{(9a)}\\
& x_j \in B, \quad z_\omega \in R.
\end{aligned}$$

The objective function aims to minimize the total installation cost. Constraints (9a) are the standard Benders optimality cuts added dynamically after solving the scenario subproblems, with $\mathcal{O}^\omega$ representing the extreme points of the dual subproblem polyhedron associated with scenario ω. Note that Benders feasibility

cuts are not considered. Indeed, as infinite overflows are allowed, the subproblem is feasible regardless of the first-stage solution prescribed by the master problem.

4.1.3. Subproblem heuristic.

In Ntaimo and Sen (2005), the authors propose an algorithm to solve the SSLP as a whole. However, we are only interested in a heuristic to solve a deterministic scenario – once servers have been located by the master problem. Berman and Drezner (2006) propose a heuristic for the case where demand points can also be servers. The resulting heuristic is described in Appendix 10.1 (see Algorithm 3).

4.1.4. Instances.

We use the instances introduced by Ntaimo and Sen (2005) which are available on the SIP test problem library: `https://www2.isye.gatech.edu/~sahmed/siplib/sslp/sslp.html`. (See Appendix 10.2 for a summary.)

The instances, which are described by the name "SSLP_m_n_S," vary according to three parameters: the number of potential server locations (m), the number of client locations (n), and the number of scenarios (S). Ntaimo and Sen (2005) introduce three instance classes that vary according to the number of servers and clients considered. These instance classes are summarized in Table 7.

4.2. Stochastic traveling salesman with routing recourse decisions

We now introduce the two-stage stochastic traveling salesman with outsourcing (2TSP) a novel variant of the Profitable Tour Problem (PTP, Dell'Amico et al. 1995) with stochastic customers (Zhang et al. 2017). In itself, the PTP is a variant of the TSP with profits; for more details on these problems, we refer the interested reader to Feillet et al. (2005).

The 2TSP aims to construct a vehicle route that minimizes the delivery cost from a depot to a set of customers. Conversely to the PTP, the aim is not to maximize the profits collected during the tour but rather to balance travel cost and outsourcing fees. The first stage determines which customers will be served by the vehicle if they happen to make a request in the second-stage. Customers that are not selected in the second stage will potentially need to be served by the third-party, which incurs a first-stage booking fixed cost. In the second stage, some customers request a service and the recourse decisions consist in determining a route visiting all the selected customers assigned to the vehicle in the first stage who have requested service. On the other hand, customers with requests that have not been assigned to the vehicle in the first stage are outsourced to the third-party, which incurs a service cost. The application of this problem arises in the context of repair and maintenance services where the provider/technician can choose a set of customers to serve using their vehicle and outsource the service for the remaining customers to an external provider. The challenge of solving this problem lies in the fact that the routing decisions are scenario-dependent and are

determined in the second stage. Thus, the Benders subproblem corresponds to a traveling salesman problem associated with a given set of selected customers that made a request in each scenario.

4.2.1. Stochastic mixed-integer program.

We base our 2TSP formulation on the classic Dantzig-Fulkerson-Johnson (DFJ) model (Dantzig et al. 1954). This formulation has an exponential number of constraints and the model is solved using a B&C. At the start, the model only contains the *degree constraints* (10b) and thus allows *sub-tours*. At each integer solution, a procedure checks if the solution contains a sub-tour. If so, a constraint preventing this sub-tour is added. The first solution which does not contain a sub-tour is the optimal one.

We recall below the model for the symmetric TSP with SECs. We define:

- N, the set of all nodes;
- c_{ij}, the cost (distance) between two nodes; and
- $E(Y) = \left\{ (i,j) \;\middle|\; i < j, (i,j) \in Y^2 \right\}$, the set of all edges forming a complete graph given a set of nodes Y.

We define the TSP with outsourcing as the problem of determining (i) the set of customers to be served by the vehicle in the first stage, and (ii) the route to serve the selected customers who made a request in the second stage so as to minimize the total expected routing and outsourcing cost to serve (stochastic) customer requests. We consider a set of scenarios $\omega \in \Omega$, each with a probability p_ω of occurring. In each scenario, we use parameters h_i^ω to represent whether customer $i \in N$ has a request. Regarding costs, we refer to b_i as the cost paid to book third-party capacity in the first-stage for customer $i \in N$, and we refer to d_i as the third-party second-stage service cost for customer $i \in N$. Finally, c_{ij} reflects the travel cost from customer $i \in N$ to customer $j \in N$. In addition, we assume, without loss of generality, that at least C customers must be served by the vehicle.

In the *deterministic equivalent formulation* below (2TSP), we use the following decision variables:

- x_i, binary variable taking value 1 iff customer i is visited by the vehicle;
- y_{ij}^ω, binary variable taking value 1 iff edge (i,j) is used in scenario ω.
- β_i^ω, binary variable taking value 1 iff customer i is not selected in the first stage and happens to have a request in scenario ω.

This leads to the following formulation:

$$\min \quad \sum_{i\in N} b_i(1-x_i) + \sum_{\omega\in\Omega} p_\omega \left(\sum_{(i,j)\in E(N)} c_{ij} y_{ij}^\omega + \sum_{i\in N} d_i \beta_i^\omega \right) \tag{2TSP}$$

$$s.t. \quad \sum_{i\in N} x_i \geq C \tag{10a}$$

$$\sum_{j\in N} y_{ij}^\omega = 2x_i h_i^\omega \qquad \forall i \in N, \forall \omega \in \Omega \tag{10b}$$

$$\sum_{(i,j)\in E(S)} y_{ij}^\omega \leq |S| - 1 \qquad S \subseteq N, |S| \geq 2, \forall \omega \in \Omega \tag{10c}$$

$$\beta_i^\omega = (1-x_i) h_i^\omega \qquad \forall i \in N, \forall \omega \in \Omega \tag{10d}$$

$$x_i \in B, \quad y_{ij}^\omega \in B, \quad \beta_i^\omega \in B.$$

The objective functions minimizes the sum of the first-stage third-party booking costs and the second-stage travel costs and third-party service costs. The constraint (10a) ensures that at least C customers are visited. Constraints (10b) are the degree constraints. Constraints (10c) are the sub-tour elimination constraints. Constraints (10d) update the value of the auxiliary variables β_i^ω.

4.2.2. Benders decomposition.

The *deterministic equivalent formulation* presented above can be decomposed into a first stage, which consists in selecting the customers to include in the tour while paying outsourcing fees for required customers that are not selected, and a second stage which aims to construct tours between the selected customers that have a request. This yields a valid Benders decomposition where each scenario forms an independent subproblem. Hence, each subproblem is an instance of a TSP based on the nodes which have a request in this scenario.

To generate cuts, we use the dual subproblem based on said TSP, where for each scenario $\omega \in \Omega$, we define u^ω, v^ω and w^ω as the dual variables associated with constraints (10b) to (10d), respectively. Given a first-stage solution $\hat{x}$ computed by the master problem, the subproblem associated with scenario $\omega \in \Omega$ is formulated as follows:

$$\max \quad \sum_{i\in N} (2\hat{x}_i h_i^\omega) u_i^\omega + \sum_{\substack{S\subset N \\ |S|\geq 2}} (|S|-1) v_S^\omega + \sum_{i\in N} ((1-\hat{x}_i) h_i^\omega) w_i^\omega \tag{Sub[ω]}$$

$$s.t. \quad u_i^\omega - \sum_{\{\, S\subset N | (i,j)\in S \,\}} v_S^\omega \leq c_{ij} \qquad \forall (i,j) \in E(N) \tag{11a}$$

$$w_i^\omega \leq d_i \qquad \forall i \in N \tag{11b}$$

$$u_i^\omega \in R, v_S^\omega \geq 0.$$

We solve the subproblem in its primal form, which is an LP relaxation of a TSP, using the standard cutting-plane technique for the TSP (Dantzig et al. 1954).

The master problem is formulated as follows:

$$\min \quad \sum_{i \in N}(1-x_i)b_i + \sum_{\omega \in \Omega} p_\omega z_\omega \qquad \text{(Standard master)}$$

$$s.t. \qquad (10a)$$

$$\sum_{i \in N} 2x_i h_i^\omega u_i^\omega + \sum_{\substack{S \subset N \\ |S| \geq 2}} (|S|-1)v_S^\omega + \sum_{i \in N}((1-x_i)h_i^\omega)w_i^\omega \leq z_\omega$$

$$\forall (u^\omega, v^\omega) \in \mathcal{O}^\omega, \forall \omega \in \Omega \qquad (12a)$$

$$x_i \in B, \quad z_\omega \geq 0.$$

The objective function aims to minimize the penalties for not including customers in the tour. Constraints (12a) are the standard Benders optimality cuts added dynamically after solving the scenario subproblems, with $\mathcal{O}^\omega$ representing the extreme points of the dual subproblem polyhedron associated with scenario ω. Again, Benders feasibility cuts are not considered as the subproblem is feasible regardless of the first-stage solutions prescribed by the master problem.

4.2.3. Subproblem heuristics.

Considering how hard optimal solutions are to obtain, we can make use of well-known TSP heuristics in the literature to quickly determine solutions of good quality. We decided on four heuristics to get upper bounds, all of them are *local search* heuristics, which means they improve a given solution until no improvement can be found. Obviously, more sophisticated TSP heuristics could be used in this stage as well. However, we opted to use these heuristics as they are simple to implement and demonstrate the efficiency and flexibility of our UB&BC approach.

- *Nearest neighbor* starts at a random customer and, at every step, chooses to visit the closest unvisited customer. This heuristic gives poor results in the general case as it does not consider the *layout* of the tour at all.

- *2-opt* looks for two edges which, if swapped, would yield an improvement in the tour and repeats this process until no further improvement can be found. This heuristic is one of the most commonly used as it has a simple implementation and fairly low complexity of $\mathcal{O}(n^2)$.

- *3-opt* is similar to 2-opt, but it tries to find three edges leading to an improvement instead of two. In this case, the search becomes quite expensive with a complexity of $\mathcal{O}(n^3)$.

- *LKH* is our implementation of the Lin-Kernighan heuristic (Lin and Kernighan 1973), using enhancements proposed in Helsgaun (2000) – implementation details can be found in Appendix 12.2.1. This heuristic is considered the state of the art and has a complexity of $\mathcal{O}(n^{2.2})$.

4.2.4. Instances.

We define our instances using the classic TSPLib (Reinelt 1991) instances which can be found at : `http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html`.The scenarios are generated using the following parameters:

- *Number of scenarios* Between 50 and 500, with an increment of 50, which result in 10 instances of 2STP per each original TSP instance.
- *Customer requests* Customers have a random uniform chance (80%) of appearing in each scenario. We also make sure not to have duplicate scenarios.
- *Unserved requests costs* We define the costs incurred by not visiting a customer with a request as the distance between the customer and the depot: $b_i = d_{0,i}, \forall i \in N$.

5. Computational study

In this section, we evaluate the performance of three versions of our UB&BC. The first version, UB&BC Base, refers to our framework without acceleration techniques. The second version, UB&BC CPT, refers to our framework with the convexification procedure. The third version, UB&BC Partial, refers to our framework with partial Benders reformulation. Note that we also implemented a version that integrates both acceleration techniques, but we do not report the corresponding results as it turned out that the performance of this version was far worse. This is because this approach suffers from numerical issues, which is also an issue mentioned in Qi and Sen (2017). More specifically, the extra constraints considered in the improved master problem yield a CGLP formulation with poor numerical stability. Performance indicators are obtained by solving different instances of the stochastic server location problem (SSLP). As these SSLP instances are generally used to benchmark algorithms for two-stage SMIPs with discrete recourse (e. g., Ntaimo and Sen 2005, Guo et al. 2015, Gade et al. 2014, Atakan and Sen 2018, Qi and Sen 2017), we report the results found in the corresponding articles. We also perform an extensive series of experiments on instances of the stochastic traveling salesman problem with outsourcing (2TSP) to gain insights into how the different components of UB&BC play a role in its performance.

Our algorithm is coded in Python 3.5 and its implementation is available in the GitLab project: `https://gitlab.com/Soha/brandec` and it is executed on the Compute Canada HPC cluster.[2] The experiments

[2]Complete specifications available on calculquebec.ca.

are conducted on an Intel E5-2683 processor with a 2.1GHz CPU on a single thread and with 12 GB of RAM. Linear and integer programs are solved using CPLEX v12.7.

5.1. Performance of UB&BC on the SSLP

Algorithms for two-stage SMIPs with discrete recourse are often benchmarked on instances of the SSLP, especially because these instances involve a significant number of discrete variables (Ntaimo and Sen 2005). Three SSLP variants are studied in the literature, with each variant being different from another regarding the type of variables it involves in the first stage and the second stage. We sort these SSLP variants by ascending difficulty. In the 1st variant, the first stage involves binary variables while the second stage involves binary/continuous variables. In the 2nd variant, the first stage involves binary variables while the second stage involves binary/integer variables. In the 3rd variant, also referred to as the stochastic server location and sizing (SSLS), both stages solely involve general integer variables. These SSLP variants are summarized in Table 2.

	1st Stage			2nd Stage		
	R	B	Z	R	B	Z
1st variant		$\star$		$\star$	$\star$	
2nd variant		$\star$			$\star$	$\star$
3rd variant			$\star$			$\star$

Table 2: Type of variables involved in the different SSLP variants

Note that the 3rd variant requires a modified heuristic for solving the subproblem. We present our modified allocation heuristic in Appendix 11.4. As there already exist several efficient algorithms for solving two-stage stochastic mixed-integer programs with binary first and second stages, we focus on the last two variants. These variants correspond to a difficult class of problems that remains to be addressed in the literature. Therefore, we compare UB&BC with the two state-of-the-art solution approaches:

- the decomposition algorithm with parametric Gomory cuts (Gomory) proposed by Gade et al. (2014);
- the ancestral Benders' cutting plane algorithm (ABC) proposed by Qi and Sen (2017).

The first approach is benchmarked on the 2nd SSLP variant, while the second approach is benchmarked on the 3rd SSLP variant.

Providing a fair comparison between our approach and those of Gade et al. (2014) and Qi and Sen (2017) would require running experiments in the same computing environment. However, because the codes for these approaches are not available, we report scaled computation times according to the type of processor employed in Gade et al. (2014) and Qi and Sen (2017). In a similar way as Glize et al. (2020), we use

the PassMark single thread rating[3] as a performance indicator of the processors and we determine scaling coefficient values accordingly, i. e., for processors A and B we use a relationship of the form $R_A T_A = R_B T_B$, where R is the single thread rating and T is the computation time. In Table 3, for each article we report the type of processor employed, the associated single thread rating, and the scaling coefficient. Note that, as Qi and Sen (2017) do not specify the exact model of their Intel CoreQuad processor, we retained the lowest single thread rating among all Intel CoreQuad processors, i. e., the most powerful candidate processor.

Even though this method allows us to draw comparisons with other approaches, we want to remark that this scaled computing time is a rough approximation which can be inaccurate and the information is provided supplementarily. We thus refrain from drawing strong conclusions with respect to the superiority of our approaches versus the approaches in the literature and we focus our analyses on computational insights of the proposed algorithms.

Method	SSLP variant	Processor	Single thread rating	Scaling coefficient
Gomory	2	Intel CoreQuad (2.66GHz)	1,079	0.64
ABC	3	Intel i7-3770K (3.5GHz)	2,066	1.23
UB&BC	2,3	**Intel E5-2683 (2.1GHz)**	**1,677**	**1.00**

Table 3: CPU characteristics

5.1.1. Results on the 2nd SSLP variant.

We solve instances of the 2nd SSLP variant and we compare the two versions of UB&BC as well as scaled times for the decomposition algorithm with parametric Gomory cuts (Gomory). For each instance, the best computation time at termination is indicated in bold.

We first observe that using the convexification procedure is not worth it in that case as the extra computational effort spent does not allow to reduce computation times. In particular, UB&BC Base outperforms UB&BC CPT on all the instances and converges 3.65 times faster overall. On the other hand, using the partial decomposition is beneficial as UB&BC Partial converges 1.60 times faster than UB&BC Base overall, and outperforms it on all instances but SSLP_10_50_50.

5.1.2. Results on the 3rd SSLP variant.

We solve instances of the 3rd SSLP variant and we compare two versions of UB&BC as well as scaled times for the ancestral Benders' cutting plane algorithm (ABC). For each instance, the best computation time at termination is indicated in bold. We indicate timeouts (longer than 3,600s) with 't/o'.

[3]Data available at `https://www.cpubenchmark.net/singleThread.html`

	UB&BC				
	Base	CPT	Partial	Gomory	
Instance	Time (s)	Time (s)	Time (s)	Time_o (s)	Time_s (s)
SSLP_5_25_50	0.88	2.74	0.52	0.18	**0.12**
SSLP_5_25_100	1.73	7.86	1.14	0.22	**0.14**
SSLP_5_50_50	0.80	5.38	0.61	0.27	**0.17**
SSLP_5_50_100	1.81	9.50	0.97	0.48	**0.31**
SSLP_10_50_50	**16.67**	65.28	16.82	109.20	69.89
SSLP_10_50_100	42.73	135.87	**30.38**	218.42	139.79
SSLP_10_50_500	416.22	1013.99	**282.38**	740.38	473.84
SSLP_10_50_1000	1,135.81	2,488.74	**731.04**	1,615.42	1,033.87
SSLP_10_50_2000	4,742.90	6996.37	1,783.62	2,729.61	**1,746.95**

Table 4: Performance on the 2nd SSLP variant

	UB&BC						UB&BC				
	Base	CPT	Partial	ABC			Base	CPT	Partial	ABC	
Instance SSLS	Time (s)	Time (s)	Time (s)	Time_o (s)	Time_s (s)	Instance SSLS	Time (s)	Time (s)	Time (s)	Time_o (s)	Time_s (s)
(2 5)_(5 5)_50	**0,25**	0,36	0,48	0,30	0,37	(3 5)_(10 5)_500	**6,94**	12,83	7,50	27,15	33,39
(2 5)_(5 5)_100	**0,33**	0,55	0,71	0,38	0,47	(3 5)_(15 5)_50	**0,98**	1,88	1,18	181,92	223,76
(2 5)_(5 5)_500	**2,20**	3,34	3,77	3,58	4,40	(3 5)_(15 5)_100	**1,51**	4,39	4,05	19,56	24,06
(2 5)_(10 5)_50	**0,31**	1,31	0,46	0,52	0,64	(3 5)_(15 5)_500	**6,65**	60,37	11,75	1069,92	1316,00
(2 5)_(10 5)_100	**0,93**	1,02	1,10	4,84	5,95	(4 5)_(5 5)_50	0,40	1,49	**0,38**	1,59	1,96
(2 5)_(10 5)_500	**3,01**	8,93	3,95	4,43	5,45	(4 5)_(5 5)_100	**0,58**	1,91	1,29	3,36	4,13
(2 5)_(15 5)_50	0,97	**0,76**	1,22	4,26	5,24	(4 5)_(5 5)_500	**3,92**	9,47	5,00	20,99	25,82
(2 5)_(15 5)_100	**0,73**	0,80	0,99	1,64	2,02	(4 5)_(10 5)_50	**1,28**	7,73	2,39	3,60	4,43
(2 5)_(15 5)_500	**6,33**	7,40	8,86	51,28	63,07	(4 5)_(10 5)_100	**3,45**	20,48	5,50	261,89	322,12
(3 5)_(5 5)_50	**0,17**	0,54	0,29	0,59	0,73	(4 5)_(10 5)_500	**13,31**	29,67	31,74	745,61	917,10
(3 5)_(5 5)_100	**0,52**	0,73	0,59	1,23	1,51	(4 5)_(15 5)_50	**4,23**	6,21	7,14	1653,67	2034,01
(3 5)_(5 5)_500	**3,36**	7,45	4,83	7,09	8,72	(4 5)_(15 5)_100	**3,59**	6,39	5,46	t/o	t/o
(3 5)_(10 5)_50	**2,92**	3,01	3,86	4,68	5,76	(4 5)_(15 5)_500	83,73	**34,68**	106,42	t/o	t/o
(3 5)_(10 5)_100	**2,25**	4,14	2,87	158,61	195,09						

Table 5: Performance on the 3rd SSLP variant

Overall, we observe that the acceleration techniques are not beneficial in that case. Indeed, UB&BC Base outperforms UB&BC CPT on 25 out of the 27 instances, and it outperforms UB&BC Partial on 26 out of the 27 instances. When analyzing the results, we observe that the first phase of UB&BC (i. e., the B&B) is twice longer as the partial Benders reformulation is applied. This likely comes from the increase in size of the master problem: the original problem is quite small – four master problem variables for the largest instances. When using a partial formulation, we add up to 60 additional variables – one per client, per server. Although we observe that the partial reformulation does find a better lower bound faster, this does not translate to faster convergence.

In a similar manner, using a CPT to raise the subproblem's bounds does not offset the extra computational effort. Indeed, in the Base configuration, solving the subproblems does not exceed 50% of the total time – see Section 13. When we use a subproblem with a CPT, solving the subproblem represents up to 90% of the total time. We conclude that strengthening the bounds produced by the master problem is at the cost of an extra computational effort that is too significant to improve convergence speed in this case. In terms of scaled times, all versions of UB&BC appear more effective than the ABC algorithm.

5.2. Performance of UB&BC on the 2TSP

We perform an extensive series of experiments on the instances of the stochastic traveling salesman problem with outsourcing (2TSP). These instances are more combinatorially challenging and allow us to assess the performance of our framework on a difficult problem. We focus on two versions of our framework, namely UB&BC Base and UB&BC Partial, as incorporating the convexification procedure does not increase computational performance. Instances are described in Section 4.2.4. Figure 2 shows the performance of the two versions of UB&BC on different instances of the TSPLib, based on their size. The size of the instance is defined as the number of variables in the model, used as a proxy for difficulty. We plot the results of using the UB&BC with its best performing heuristic (LKH, cf. Figure 7) against the DEF MIP formulation solved with CPLEX implementation of branch-and-cut. Note that we use further algorithmic refinements in UB&BC Partial, namely: merging procedure (Appendix 12.2.2) and subproblem warm-up (Appendix 12.2.3).

The results in Figure 2 and Table 6 clearly show the superiority of the UB&BC over CPLEX. Specifically, Figure 2 demonstrate that, when solving the DEF, the computation time required by CPLEX increases significantly with the number of variables considered. On the other hand, the results of Table 6 indicate that 154 instances were solved to optimality by UB&BC Partial whereas CPLEX could solve the DEF to optimality for only 12 instances .

One of the main goals of the UB&BC is to reduce the number of open solutions we have to solve to integer optimality. Figure 3 shows the number of solutions explored during the master B&C and the number of MIPs solved in the post-processing phase. Our strategy proves to always be beneficial: there is not a

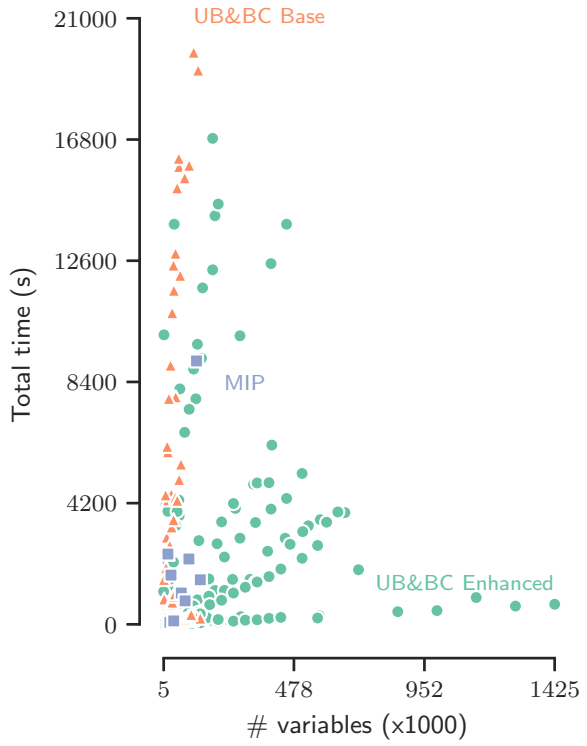

Figure 2: Solving time per (estimated) size of 2TSP instances using a generic MIP solver, UB&BC Base or UB&BC Partial.

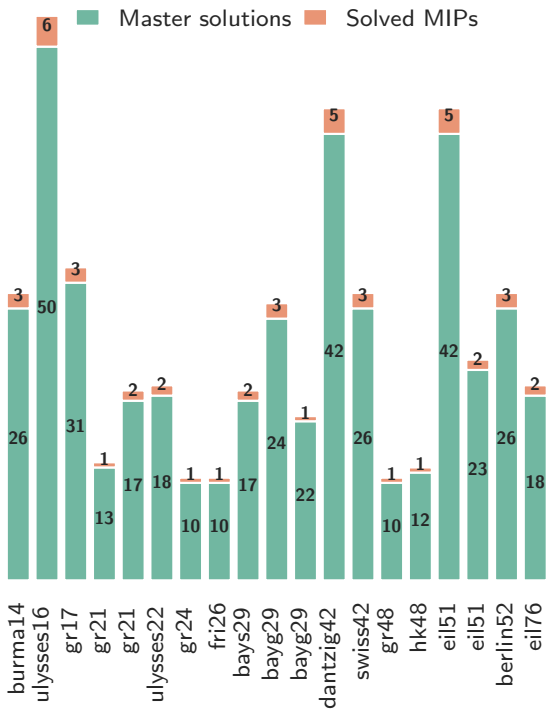

Figure 3: Number of master solutions explored and and the number of MIPs solved in the post-processing phase when using 100 scenarios.

		UB&BC				UB&BC	
Instance (/10)	MIP	Base	Partial	Instance (/10)	MIP	Base	Partial
att48	0	0	0	gr17	0	10	10
bayg29	1	3	**10**	gr21	3	10	10
bays29	0	1	**10**	gr24	8	10	10
berlin52	0	0	**10**	gr48	0	0	**10**
brazil58	0	0	0	hk48	0	0	**10**
burma14	0	10	10	pr76	0	0	0
dantzig42	0	0	**10**	st70	0	0	0
eil51	0	0	**10**	swiss42	0	0	**10**
eil76	0	0	**10**	ulysses16	0	10	10
fri26	0	8	**10**	ulysses22	0	0	**4**

Table 6: Number of instances of 2TSP solved using a MIP or the UB&BC approach.

single instance with as many MIPs solved as solutions explored. We also have a number of instances where the first solution solved in the post-processing phase allows us to prune all the others.

To further analyze the performance of UB&BC, we study the impact of using a partial Benders decomposition as well as the impact of the subproblem heuristic in Appendices 12.1 and 12.2, respectively.

6. Conclusions

In this paper, we presented a new framework for solving two-stage stochastic mixed-integer programs. Our Unified Branch-and-Benders-Cut (UB&BC) tackles two-stage stochastic mixed-integer programs with uncertainty in all the recourse parameters, and it accommodates general mixed-integer variables in both the first and the second stage. The UB&BC relies on both linear programming duality and on a heuristic global bounding procedure to determine a set of open master solutions. In a post-processing phase, the scenario subproblems associated with these open solutions are solved to integer optimality, enabling us to determine the global optimum. We also demonstrated the flexibility of our approach which can incorporate acceleration techniques such as partial Benders decompositions or convexification schemes.

Through an extensive series of experiments carried out on instances of the stochastic server location problem (SSLP), we have computationally demonstrated that the basic version of our framework, as well as that with partial Benders reformulation, can be quite effective. On the other hand, the version integrating the convexification scheme appeared to be computationally less effective for the considered problems. We have also performed a computational study on the two-stage stochastic traveling salesman with outsourcing (2TSP) to assess the efficiency of each component of UB&BC. In particular, we have shown that partial decomposition creates a virtuous circle of improvement. With it, the algorithm explores fewer integer master solutions during the branch-and-Benders-cut and thus reduces the computational burden of the post-processing phase. We have also highlighted how using an efficient bounding heuristic can significantly improve the algorithm's speed of convergence.

Finally, two advantages of our framework are simplicity and flexibility. We believe that this straightforward approach is a great candidate for building more advanced algorithms and tackling larger, more difficult problems. There are computational enhancements that we have not explored. For example, we only consider single-threaded execution; modern computing relies on multicore infrastructures and we could solve either the B&C or the post-processing phase in parallel. Another area of interest is to exploit the structure of scenarios, not only from a computational point of view but also to inform the master problem. Other possible avenues include advanced branching schemes, improved cut generation, or constraint propagation.

Computations were made on the supercomputer "Graham" managed by and Compute Canada. The operation of this supercomputer is funded by the Canada Foundation for Innovation (CFI), the Ministère de l'économie, de la science et de l'innovation du Québec (MESI) and the Fonds de recherche du Québec –

Nature et technologies (FRQ-NT).

References

Ahmed S, Tawarmalani M, Sahinidis NV (2004) A finite branch-and-bound algorithm for two-stage stochastic integer programs. *Mathematical Programming* 100(2):355–377.

Angulo G, Ahmed S, Dey SS (2016) Improving the integer L-shaped method. *INFORMS Journal on Computing* 28(3):483–499.

Atakan S, Sen S (2018) A progressive hedging based branch-and-bound algorithm for mixed-integer stochastic programs. *Computational Management Science* 15(3-4):501–540.

Balas E (1979) Disjunctive programming. *Annals of discrete mathematics* 5:3–51.

Benders JF (1962) Partitioning procedures for solving mixed variables programming problems. *Numerische Mathematik* 4:238–252.

Berman O, Drezner Z (2006) Location of congested capacitated facilities with distance-sensitive demand. *IIE Transactions* 38(3):213–221.

Berman O, Mandowsky RR (1986) Location-allocation on congested networks. *European Journal of Operational Research* 26(2):238–250.

Birge JR, Louveaux FV (1997) *Introduction to stochastic programming* (Springer Science & Business Media).

Carøe CC, Schultz R (1999) Dual decomposition in stochastic integer programming. *Operations Research Letters* 24(1-2):37–45.

Carøe CC, Tind J (1998) L-shaped decomposition of two-stage stochastic programs with integer recourse. *Mathematical Programming* 83(1-3):451–464.

Chen B, Küçükyavuz S, Sen S (2011) Finite disjunctive programming characterizations for general mixed-integer linear programs. *Operations Research* 59(1):202–210.

Chen B, Küçükyavuz S, Sen S (2012) A computational study of the cutting plane tree algorithm for general mixed-integer linear programs. *Operations research letters* 40(1):15–19.

Crainic TG, Hewitt M, Rei W (2014) Partial decomposition strategies for two-stage stochastic integer programs. *Publication CIRRELT-2014-13, Centre interuniversitaire de recherche sur les reseaux d'entreprise, la logistique et le transport, Universite de Montreal, Montreal QC, Canada* .

Crainic TG, Rei W, Hewitt M, Maggioni F (2016) Partial Benders decomposition strategies for two-stage stochastic integer programs. *Publication CIRRELT-2016-37, Centre interuniversitaire de recherche sur les reseaux d'entreprise, la logistique et le transport, Universite de Montreal, Montreal QC, Canada* .

Dantzig G, Fulkerson R, Johnson S (1954) Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America* 2(4):393–410.

De Camargo RS, de Miranda Jr G, Ferreira RP (2011) A hybrid outer-approximation/Benders decomposition algorithm for the single allocation hub location problem under congestion. *Operations Research Letters* 39(5):329–337.

Dell'Amico M, Maffioli F, Varbrand P (1995) On prize-collecting tours and the asymmetric travelling salesman problem. *International Transactions in Operational Research* 2(3):297–308, URL `http://doi.wiley.com/10.1111/j.1475-3995.1995.tb00023.x`.

Elçi Ö, Hooker J (2022) Stochastic planning and scheduling with logic-based benders decomposition. *INFORMS Journal on Computing* .

Farkas J (1902) Theorie der einfachen Ungleichungen. *Journal für die reine und angewandte Mathematik* 124:1–27.

Feillet D, Dejax P, Gendreau M (2005) Traveling salesman problems with profits. *Transportation Science* 39(2):188–205.

Fortz B, Poss M (2009) An improved Benders decomposition applied to a multi-layer network design problem. *Operations Research Letters* 37(5):359–364.

Gade D, Küçükyavuz S, Sen S (2014) Decomposition algorithms with parametric Gomory cuts for two-stage stochastic integer programs. *Mathematical Programming* 144(1-2):39–64.

Gendron B, Scutellà MG, Garroppo RG, Nencioni G, Tavanti L (2016) A branch-and-Benders-cut method for nonlinear power design in green wireless local area networks. *European Journal of Operational Research* 255(1):151–162.

Gere Jr WS (1966) Heuristics in job shop scheduling. *Management Science* 13(3):167–190.

Glize E, Roberti R, Jozefowiez N, Ngueveu SU (2020) Exact methods for mono-objective and bi-objective multi-vehicle covering tour problems. *European Journal of Operational Research* 283(3):812–824.

Guo G, Hackebeil G, Ryan SM, Watson JP, Woodruff DL (2015) Integration of progressive hedging and dual decomposition in stochastic integer programs. *Operations Research Letters* 43(3):311–316.

Helsgaun K (2000) An effective implementation of the Lin–Kernighan traveling salesman heuristic. *European Journal of Operational Research* 126(1):106–130.

Kong N, Schaefer AJ, Hunsaker B (2006) Two-stage integer programs with stochastic right-hand sides: A superadditive dual approach. *Mathematical Programming* 108(2-3):275–296.

Küçükyavuz S, Sen S (2017) An introduction to two-stage stochastic mixed-integer programming. *Leading Developments from INFORMS Communities*, 1–27 (INFORMS).

Laporte G, Gendreau M, Potvin JY, Semet F (2000) Classical and modern heuristics for the vehicle routing problem. *International transactions in operational research* 7(4-5):285–300.

Laporte G, Louveaux FV (1993) The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters* 13(3):133–142.

Larsen E, Frejinger E, Gendron B, Lodi A (2022) Fast continuous and integer L-shaped heuristics through supervised learning. *arXiv preprint arXiv:2205.00897* .

Lin S, Kernighan BW (1973) An effective heuristic algorithm for the traveling-salesman problem. *Operations Research* 21(2):498–516.

Mokotoff E (2001) Parallel machine scheduling problems: A survey. *Asia-Paci c Journal of Operational Research* 18(2):193.

Najafi S, et al. (2012) A new heuristic algorithm for unit commitment problem. *Energy Procedia* 14:2005–2011.

Ntaimo L (2010) Disjunctive decomposition for two-stage stochastic mixed-binary programs with random recourse. *Operations Research* 58(1):229–243.

Ntaimo L, Sen S (2005) The million-variable "march" for stochastic combinatorial optimization. *Journal of Global Optimization* 32(3):385–400.

Qi Y, Sen S (2017) The ancestral Benders' cutting plane algorithm with multi-term disjunctions for mixed-integer recourse decisions in stochastic programming. *Mathematical Programming* 161(1-2):193–235.

Rahmaniani R, Crainic TG, Gendreau M, Rei W (2017) The Benders decomposition algorithm: A literature review. *European Journal of Operational Research* 259(3):801–817.

Ralphs TK, Hassanzadeh A (2014) A generalization of Benders' algorithm for two-stage stochastic optimization problems with mixed integer recourse. *Technical Report 14T-005, Department of Industrial and Systems Engineering, Lehigh University* .

Reinelt G (1991) Tsplib—A traveling salesman problem library. *ORSA Journal on Computing* 3(4):376–384.

Schultz R, Stougie L, Van Der Vlerk MH (1998) Solving stochastic programs with integer recourse by enumeration: A framework using Gröbner basis. *Mathematical Programming* 83(1-3):229–252.

Sen S, Higle JL (2005) The C 3 theorem and a D 2 algorithm for large scale stochastic mixed-integer programming: Set convexification. *Mathematical Programming* 104(1):1–20.

Sen S, Sherali HD (2006) Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming. *Mathematical Programming* 106(2):203–223.

Sherali HD, Adams WP (1999) A Reformulation-Linearization technique for solving discrete and continuous nnconvex problems. *Kluwer Academic Publishers* .

Sherali HD, Fraticelli BM (2002) A modification of Benders' decomposition algorithm for discrete subproblems: An approach for stochastic programs with integer recourse. *Journal of Global Optimization* 22(1-4):319–342.

Sherali HD, Zhu X (2006) On solving discrete two-stage stochastic programs having mixed-integer first-and second-stage variables. *Mathematical Programming* 108(2-3):597–616.

Trapp AC, Prokopyev OA, Schaefer AJ (2013) On a level-set characterization of the value function of an integer program and its application to stochastic programming. *Operations Research* 61(2):498–511.

Van Slyke RM, Wets R (1969) L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics* 17(4):638–663.

Wilbaut C, Hanafi S, Salhi S (2008) A survey of effective heuristics and their application to a variety of knapsack problems. *IMA Journal of Management Mathematics* 19(3):227–244.

Wolsey LA (1998) *Integer Programming*, volume 52 (John Wiley & Sons).

Zhang M, Wang J, Liu H (2017) The probabilistic profitable tour problem. *International Journal of Enterprise Information Systems* 13(3):51–64.

Zheng QP, Wang J, Pardalos PM, Guan Y (2013) A decomposition approach to the two-stage stochastic unit commitment problem. *Annals of Operations Research* 210(1):387–410.

7. Toy problem

We use a toy problem with four variables to illustrate how the UB&BC proceeds. For the sake of simplicity, we consider a problem with a single subproblem. Consider the following integer program:

$$\begin{aligned} \min \quad & 6x_1 + 10x_2 + y_1 + 2y_2 && \text{(Toy)} \\ s.t. \quad & -15x_1 - 22x_2 + 5y_1 + 8y_2 \leq 0 && \text{(13a)} \\ & y_1 + y_2 \geq 1.5 && \text{(13b)} \\ & x \in B, y \in \{0,1,2\}. \end{aligned}$$

If we relax integrality and project out variables y, we obtain the following LP relaxation for the subproblem:

$$\begin{aligned} q(\hat{x}) = \min \quad & y_1 + 2y_2 && \text{(Toy Sub)} \\ s.t. \quad & 5y_1 + 8y_2 \leq 15\hat{x}_1 + 22\hat{x}_2 && (\lambda_1) \\ & y_1 + y_2 \geq 1.5 && (\lambda_2) \\ & y_1 \leq 2 && (\lambda_3) \\ & y_2 \leq 2 && (\lambda_4) \\ & y \geq 0. \end{aligned}$$

We denote by λ_i the dual variables associated with the constraints of the model above. Let $\mathcal{O}$ be the set of extreme points and $\mathcal{F}$ the set of extreme rays associated with the dual of (Toy Sub). If we denote by q the variable representing the lower estimator of the subproblem, we obtain the following master problem:

$$\begin{aligned} \min \quad & 6x_1 + 10x_2 + q && && \text{(Toy Master)} \\ s.t. \quad & -\lambda_1(15x_1 + 22x_2) + 1.5\lambda_2 - 2(\lambda_3 + \lambda_4) \leq q && \forall \lambda \in \mathcal{O} && \text{(15a)} \\ & -\lambda_1(15x_1 + 22x_2) + 1.5\lambda_2 - 2(\lambda_3 + \lambda_4) \leq 0 && \forall \lambda \in \mathcal{F} && \text{(15b)} \\ & x \in B. \end{aligned}$$

As a heuristic for the subproblem, we will round the value of the variables in a solution to their next integer: $h(y) = \lceil y_1 \rceil + 2\lceil y_2 \rceil$.

7.1. Master Branch-and-Cut

At the start, we have $ub^* = \infty$ and $lb^* = 0$, and all x variables relaxed to their continuous domain.

1. The first integer master solution we find when solving (Toy Master) without any constraints is $x^1 = (0,0)$ with value 0. Setting x^1 in (Toy Sub) results in an infeasible problem. Using a Farkas certificate

(Farkas 1902), we find the extreme ray $(1, 5, 0, 0)$ and thus add the following feasibility cut to the master problem:

$$-15x_1 - 22x_2 + 7.5 \leq 0 \tag{16}$$

2. Augmented by the new constraint (16), the next master solution becomes $x^2 = (0, 1)$ with value 10. We pass the new solution to the subproblem, and this results in a feasible solution $Y = \{1.5, 0\}$. Thus we can update the lower bound to $lb^* = 11.5$ and the upper bound to $ub^* = 12$.
   We add the following optimality cut from the dual values of (Toy Sub):

$$1.5 \leq q \tag{17}$$

3. Now having two Benders cuts, the search of the master's solution space proceeds to $x^3 = (1, 0)$ with value 6. Again, we find $Y = \{1.5, 0\}$ as solution to the subproblem. We can add the same optimality cut again, or just skip it. However, we can update the bounds to $lb^* = 7.5$ and $ub^* = 8$.
4. The search continues until the next potential master solution $x^4 = (1, 1)$ with value 16. At this node, we find that the master's solution value already exceeds our upper bound. We can thus prune the tree rooted at this node.

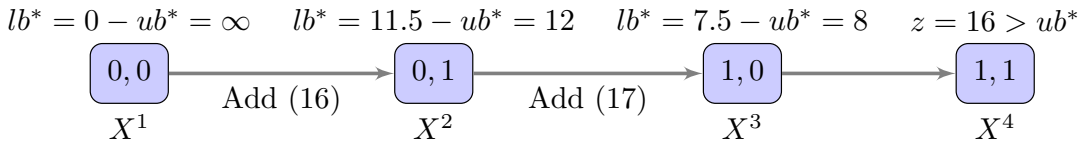

Figure 4: Example search when using UB&BC with problem (Toy).

7.2. Post-processing

After the B&B of the master problem finishes, we have a set of explored solutions with their upper and lower bounds saved. We reintroduce the integrality constraints and solve the resulting MIPs to obtain the optimal integer value.

1. We start with the solution found at node 3 as it has the lowest upper bound. Solving the associated MIP gives us the solution $Y^1 = \{2, 0\}$ with an objective value of $6 + 2 = 8$. We can either use this value as upper bound or keep the previous value of ub^*.
2. The lower bound of the solution associated with node 2 is already higher than our current best upper bound, we can thus discard the solution.

8. Hybridizing UB&BC and the CPT algorithm

For each master solution $\hat{x}$, we first solve the LP relaxation of the scenario subproblems without cutting planes inherited from ancestor nodes in the master branch-and-bound tree. This way, we develop global Benders cuts for the master problem, i. e.cuts that are valid for all nodes in the master branch-and-bound tree, similar to what is done in Phase I of Algorithm 1. We also use the obtained solutions to determine a first lower bound $lb(opt(\hat{x}))$ on $opt(\hat{x})$. If that lower bound $lb(opt(\hat{x}))$ is worse than the incumbent, the master solution is discarded and the current master node is pruned. If not, we improve the polyhedral approximation of the subproblems to reinforce the lower bound.

We then enhance the scenario subproblems with cutting planes inherited by the current node of the first-stage branch-and-bound, and we apply D iterations of the CPT algorithm to further strengthen the subproblem formulations. This way, we develop stronger but local Benders cuts for the master problem. By combining first-stage costs with the objective function value of the improved subproblem solutions, we obtain a stronger lower bound $lb^{+}(\hat{x})$ that is compared again to the incumbent. If the improved lower bound is still worse than the incumbent, $\hat{x}$ is retained as an open solution and the scenario subproblems are solved heuristically. Note that we initialize $D \leftarrow 2$ and increase it by increments of 2 at each iteration where the convexification procedure is used. This implementation is similar to that of Qi and Sen, as "seeking very accurate objective function estimates requires much more computational resources than what can be justified in early iterations" (Qi and Sen 2017).

While this extension of UB&BC and the algorithm of Qi and Sen use the same toolbox, namely, Benders decomposition and the CPT algorithm, it should be noted that both approaches proceed significantly differently. First, while Qi and Sen use the CPT algorithm at each iteration, we apply this extra effort solely if our first lower bound is not strong enough to discard the current master solution. Second, solving the subproblems without cutting planes in a first step allows us to generate global Benders cuts, in addition to the local Benders cuts proposed by Qi and Sen (2017), i.e Benders cuts that are not global but valid for a specific region of the branch-and-bound on the first-stage decisions. Third, in the approach proposed by Qi and Sen, obtaining a feasible solution to the original problem is only achieved when the convexification procedure has converged, i. e., for each scenario subproblem an integer solution is found while solving the tightened LP relaxation.

Algorithm 2: Phase I of UB&BC with CPT algorithm

Data: An original problem, P, and a subproblem heuristic, H

$Z = ;, ub^* = 1, D = 2$

Define the master problem, MP, and the scenario subproblems, SP_ω, from P

begin Solve MP by a B&B and apply the following steps at each node in the search tree:

- **if** *LP relaxation* ub^* *OR node is infeasible* **then**
  - Fathom branch
- **else if** *An integer solution* $\hat{x}$ *of the* MP *is found* **then**
  - **foreach** *Scenario* ω *2* Ω **do**
    - Solve the LP relaxation of subproblem $SP_\omega(\hat{x})$
    - Add the corresponding global Benders cut to MP
  - **if** *All scenario subproblem LP relaxations are feasible* **then**
    - Compute the improved lower bound $lb(opt(\hat{x}))$
    - **if** $lb(opt(\hat{x})) > ub^*$ **then**
      - Prune the current node
    - **else**
      - **foreach** *Scenario* ω *2* Ω **do**
        - Reinforce $SP_\omega(\hat{x})$ with inherited cutting planes
        - Apply D iterations of the CPT algorithm to $SP_\omega(\hat{x})$
        - Add the corresponding local Benders cut to MP
        - Save the newly generated cutting planes
      - Compute new lower bound $lb^+(opt(\hat{x}))$ using solutions of the reinforced subproblems
      - D += 2
      - **if** $lb^+(\hat{x}) > ub^*$ **then**
        - Prune the current node
      - **else**
        - Add $(\hat{x}, lb^+(\hat{x}))$ to Z
        - **foreach** *Scenario* ω *2* Ω **do**
          - Solve subproblem $SP_\omega(\hat{x})$ with the heuristic H
        - **if** *A heuristic solution was found for all the scenario subproblems* **then**
          - Compute the heuristic upper bound $ub(opt(\hat{x}))$
          - $ub^* = \min(ub^*, ub(opt(\hat{x})))$
- **else**
  - Choose a variable to branch on

Result: A set of candidate solutions Z

9. Partial Benders reformulations

9.1. SSLP partial Benders reformulation

To improve the solutions produced by the master problem, we include an artificial scenario ω' defined as a mean of all the original scenarios. In the SSLP, scenario-dependent parameters are characterized by the binary parameters h_i^ω that indicate whether or not client i is present in scenario ω. Therefore, for each client i, we define $h_i^{\omega'}$ as $\sum_{\omega\in\Omega} p_\omega h_i^\omega$. Given continuous variables $y_{ij}^{\omega'}$ and $y_{j0}^{\omega'}$ associated with the artificial scenario ω', the enhanced master problem is formulated as follows:

$$\min \quad \sum_{j\in J} c_j x_j - \sum_{\omega\in\Omega} p_\omega z_\omega \qquad \text{(Enhanced master)}$$

$$s.t. \quad (7a)-(7b), (9a) \text{ and}$$

$$\sum_{i\in I} d_{ij} y_{ij}^{\omega'} - y_{j0}^{\omega'} \le D x_j \qquad \forall j \in J \qquad (18a)$$

$$\sum_{j\in J} y_{ij}^{\omega'} = h_i^{\omega'} \qquad \forall i \in I \qquad (18b)$$

$$\sum_{i\in I}\sum_{j\in J} q_{ij} y_{ij}^{\omega'} - \sum_{j\in J} q_{j0} y_{j0}^{\omega'} = \sum_{\omega\in\Omega} p_\omega z_\omega \qquad (18c)$$

$$x_j \in B, \quad z_\omega \ge 0, \quad 0 \le y_{ij}^{\omega'} \le 1, \quad y_{j0}^{\omega'} \ge 0.$$

The standard master problem is enhanced with constraints (18a) to (18c). Constraints (18a) and (18b) ensure that all master problem solutions are feasible for the artificial scenario, while constraint (18c) enforces the estimate of the recourse costs to reflect the expected revenue associated with the artificial scenario.

9.2. 2TSP partial Benders reformulation

We improve the solutions produced by the master problem by including an artificial scenario ω' defined as the mean of all the original scenarios. Scenario-dependent parameters are characterized by the binary parameters h_i^ω that indicate whether or not customer i has a request in scenario ω. Therefore, for each customer i, we define $h_i^{\omega'}$ as $\sum_{\omega\in\Omega} p_\omega h_i^\omega$. Given continuous variables $y_{ij}^{\omega'}$ associated with the artificial

scenario ω', the enhanced master problem is formulated as follows:

$$\min \quad \sum_{i \in N} (1 - x_i) b_i + \sum_{\omega \in \Omega} p_\omega z_\omega \qquad \text{(Enhanced master)}$$

$$s.t. \quad \text{(10a)} - \text{(12a)}$$

$$\sum_{j \in N} y_{ij}^{\omega'} = 2 x_i h_i^{\omega'} \qquad \forall i \in N \qquad \text{(19a)}$$

$$z_i^{\omega'} = (1 - x_i) h_i^{\omega'} \qquad \forall i \in N \qquad \text{(19b)}$$

$$\sum_{(i,j) \in E(N)} c_{ij} y_{ij}^{\omega'} = \sum_{\omega \in \Omega} p_\omega z_\omega \qquad \text{(19c)}$$

$$x_i \in B, \quad 0 \leq y_{ij}^{\omega'} \leq 1, \quad z_\omega \in R.$$

The standard master problem is enhanced with constraints (19a) and constraints (19b) ensuring that all master problem solutions are feasible for the artificial scenario. Last constraints (19c) enforce the estimate of the recourse costs to reflect the cost of the tour computed for the artificial scenario.

10. Supplement to the experimental design

10.1. SSLP subproblem heuristic

Algorithm 3: Allocation heuristic for the SSLP.

Data: The set of nodes N

Data: The set of opened servers J

$S_j = 0$, $8j \ 2 \ J$ **foreach** $i \ 2 \ N$ **do**

  $D_i = \mathsf{f} \, d_{ij} \ \mathsf{j} \ 8j \ 2 \ J \, \mathsf{g}$

  Sort each D_i in ascending order

  $\delta_i = D_i^0 \quad D_i^1$

$N' =$ sort N by decreasing opportunity δ

foreach $i \ 2 \ N'$ **do**

  $k =_{k \in J} S_k + D_i^k \quad D$

  $S_k = S_k + D_i^k$

Result: An allocation of nodes i to servers j

10.2. SSLP instances

Instances in the SIPLib (Ntaimo and Sen 2005) are generated according to the following rules.

- Problem data are generated from uniform distributions:
  - server location cost in $[40, 80]$;
  - client demands in $[0, 25]$;
  - client-server revenue equal to the demand;
  - overflow cost $q_{j0} = 1000, \forall j \in J$;
  - one server location per node.
- The scenario data are generated from a Bernoulli distribution:
  - a client is present in a scenario with probability $p = 0.5$;
  - we check that there are no duplicate scenarios.
- The difficulty of an instance is controlled by a ratio (r) of the total server capacity to the maximum possible demand – the lower the r, the harder the instance as servers cannot fulfill the demand.

Class	m	n	S
1	5	25	$f50, 100g$
2	10	50	$f50, 100, 500, 1000, 2000g$
3	15	45	$f5, 10, 15g$

Table 7: Instance class characteristics

11. SSLP variants

Algorithm 4: Allocation heuristic for the SSLS.

Data: The vector of active clients $N \ 2 \ N^n$

Data: The vector of opened servers $J \ 2 \ N^m$

$S_j = 0, \ 8j \ 2 \ J$ **foreach** $n_i \ 2 \ N$ **do**

  $D_i = f \, d_{ij} \, j \, 8m_j \ 2 \ J \, g$

  Sort each D_i in ascending order

  $\delta_i = D_i^0 \quad D_i^1$

$N' =$ sort N by decreasing opportunity δ

foreach $n_i \ 2 \ N'$ **do**

  for $n = 0; \ n < n_i; \ n + = 1$ **do**

    $k =_{j \in J} S_k + D_i^k \quad D \quad j$

    $S_k = S_k + D_i^k$

Result: An allocation of nodes i to servers j

Two variants of the SSLP are also used to benchmark our solution approach. These problems are more challenging than the original SSLP in the sense that they allow general integer variables in both stages. In this section, we describe the *deterministic equivalent formulation* associated with these SSLP variants.

11.1. Comparison with D2, PH-DD, and PH-B&B (1st SSLP variant).

We compare the two versions of UB&BC to the disjunctive decomposition algorithm (D^2), the integrated progressive hedging dual decomposition algorithm (PH-DD) and the progressive hedging branch-and-bound (PH-B&B), which are all benchmarked on the 1st SSLP variant. In Table 8, we present the computation time required by each version of UB&BC to reach termination. For the state-of-the-art approaches, we report computation times from the original article in column *Time_o*. When the single thread performance is available, the scaled computation times, i. e., computation times divided by the corresponding scaling coefficient values, are reported in column *Time_s*. Note that a dash '-' indicates that the considered instance was not tested in the corresponding article.

	UB&BC						
	Base	Partial	CPT	D^2	PH-DD	PH-B&B	
Instance	Time (s)	Time (s)	Time (s)	Time_o (s)	Time_o (s)	Time_o (s)	Time_s (s)
SSLP_5_25_50	0.64	0.45	1.75	0.53	-	0.50	0.76
SSLP_5_25_100	1.43	0.91	3.45	1.06	-	1.10	1.67
SSLP_10_50_50	16.90	15.63	52.83	239.95	74.00	8.70	13.24
SSLP_10_50_100	37.25	23.34	123.22	480.46	175.00	18.60	28.31
SSLP_10_50_500	428.53	270.85	880.26	1,902.20	1,033.00	80.80	122.98
SSLP_10_50_1000	1,193.19	626.87	2,227.68	5,410.10	-	163.60	249.00
SSLP_10_50_2000	5,050.26	1,911.34	6,925.33	9,055.29	-	309.60	471.21
SSLP_15_45_5	0.52	0.27	4,503.85	110.34	-	1.40	2.13
SSLP_15_45_10	12.65	16.98	21.55	1,494.89	45.00	2.40	3.65
SSLP_15_45_15	41.50	64.53	40.90	7,210.63	123.00	3.20	4.87

Table 8: Performance on the 1st SSLP variant

As expected, using the partial Benders reformulation is beneficial as, overall, UB&BC Partial converges 1.44 times faster than UB&BC Base. For most instances, the computation times obtained with the two versions of UB&BC are lower than those reported for the D^2 and the PH-DD algorithms. However, as single thread performances are not available for these state-of-the-art approaches, we cannot provide a fair comparison.

Both versions of UB&BC are competitive with PH-B&B on the instances that involve 5 servers. This is not the case as the number of servers increases. Overall, PH-B&B is 1.84 times faster than UB&BC Partial on the instances that involves 10 servers, and 12.63 times faster on the instances that involves 15 servers. Nevertheless, it should be noted that the PH-B&B algorithm only accommodates binary variables in the first stage and cannot tackle the 3rd SSLP variant.

11.2. Stochastic server location problem with pure integer second stage

The first SSLP variant is introduced by Gade et al. (2014) and involves pure discrete variables in both the first and the second stage. Specifically, the SSLP with pure integer second stage is obtained by changing the declaration of the y_{j0} variables in (SSLP) to: $y_{j0} \in N$

11.3. Stochastic server location problem and sizing problem

The second SSLP variant is introduced by Qi and Sen (2017) and involves pure general integer variables in both the first stage and the second stage. In the stochastic server location and sizing problem (SSLS), the number of servers that can be installed at a potential server location is limited to u units, and each client

location may consist of up to v clients. Similarly to the first version, the declaration of the y_{j0} variables is changed from continuous to integer. Also, the declaration of the x_j and y_{ij}^{ω} variables is changed from binary to integer, and bounded by u and v, respectively. Note that, in this second SSLP variant, the stochastic parameters h_i^{ω} are integer and indicate the number of clients at location i in scenario ω.

$$x_j \in \{0, 1, \ldots, u\}, \quad y_{ij}^{\omega} \in \{0, 1, \ldots, v\}, \quad y_{j0} \in N.$$

Qi and Sen (2017) introduce SSLS instances that vary according to the following parameters: the number of potential server locations (m), the maximum number of servers allowed for each location (u), the number of client locations (n), the maximum number of potential clients for each locations (v), and the number of scenarios (S). Consequently, these instances are described by the name "SSLS-(m_u)-(n_v)-S." The authors describe 9 instance classes that are summarized in Table 9.

Class	m	u	n	v	S
1	2	5	5	5	f50, 100, 500g
2	2	5	10	5	f50, 100, 500g
3	2	5	15	5	f50, 100, 500g
4	3	5	5	5	f50, 100, 500g
5	3	5	10	5	f50, 100, 500g
6	3	5	15	5	f50, 100, 500g
7	4	5	5	5	f50, 100, 500g
8	4	5	10	5	f50, 100, 500g
9	4	5	15	5	f50, 100, 500g

Table 9: Instance class characteristics

11.4. Allocation heuristic for the SSLS

The SSLS can be seen as a generalization of the SSLP. We need to adapt our allocation heuristic to take into account multiple clients and multiple servers per location. The resulting heuristic is shown in Algorithm 4.

12. Analyzing the performance of UB&BC for 2TSP

12.1. Impact of the partial Benders decomposition

We now study the impact of using a partial Benders decomposition. Using the 2TSP as example, we add an artificial scenario ω' to the master problem (Section 9.2). In Figure 5 we present the results of using the enhanced master formulation.

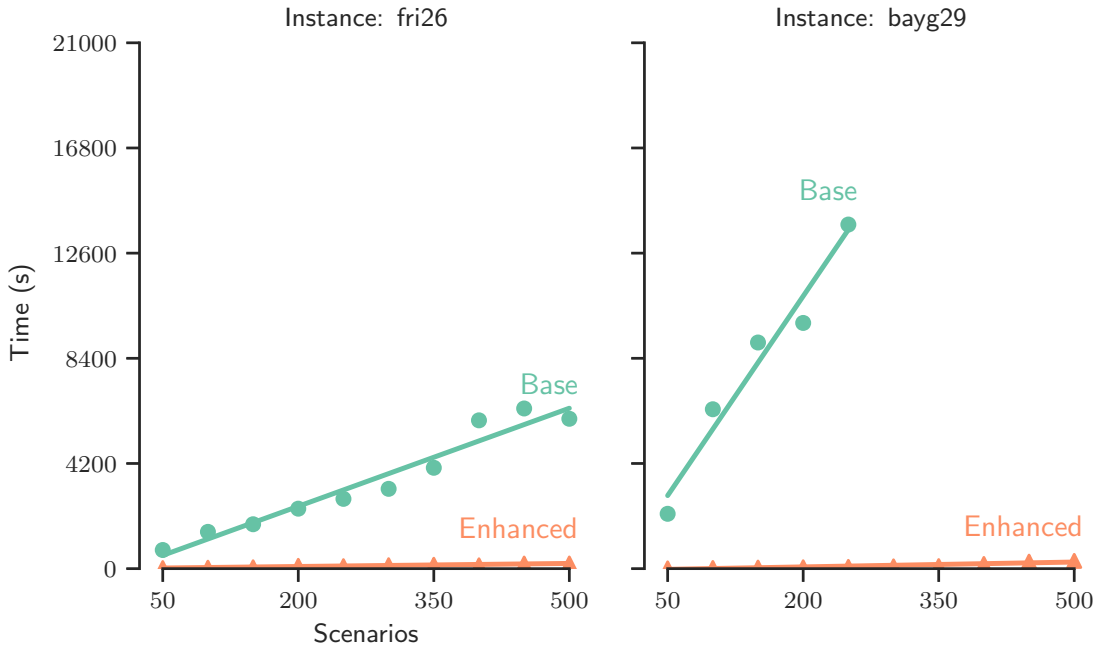

Figure 5: Solving time of the Base formulation (Auto heuristic and regular MP) and the partial formulation (warmed LKH and extended MP).

At virtually no cost, extending the master formulation with subproblem variables provides the best improvement. We can see that both scatter plots follow a linear progression in the number of scenarios, but the extended formulation has a much better scaling. This has been the most efficient optimization we have found to improve the performance of our framework from a modelling point of view.

This is further exemplified in Table 10 where we report the number of solutions explored during the master's B&C. Overall, the number of integer solutions explored by the partial reformulation is more than 7 times smaller than the number of integer solutions explored by the Base formulation. This number goes down to about 1%, effectively eliminating most of the search. By having fewer master solutions explored, we have less work remaining in the post-processing phase.

12.2. Impact of the subproblem heuristic

We now analyze the performance of the heuristics used in the 2TSP.

Instance	Base UB&BC	Partial UB&BC	Ratio (%)
burma14	57.60	9.70	16.84
ulysses16	156.70	17.80	11.36
gr17	113.70	4.80	4.22
gr21	12.70	3.00	23.62
gr24	7.00	3.00	42.86
fri26	33.50	3.00	8.96
bays29	425.50	4.00	0.94
bayg29	355.40	3.30	0.93

Table 10: Average number of master solutions explored during the B&C. We only report instances where the Base configuration managed to finish.

12.2.1. LKH heuristic implementation details.

The Lin-Kernighan and Helsgaun heuristic tries to build a tour by identifying *promising moves.* It starts with a random tour, identifies one edge to remove and one to add which improve the tour length. Instead of stopping at this point, like in 2-opt, it tries to find other edges with the same property. It then restarts from the new, improved tour. The strength of this heuristic comes from combining simple local search operators with intelligent rules. For example, when searching for a pair of edges, the resulting configuration must form a tour.

We did not implement all improvements proposed by Helsgaun. Our current implementation uses:

- *Solution removal*: stop the search if we find a previous solution.
- *Allow disjoint tours*: early in the search, allow the improving configuration to be a disjoint tour.
- *Order neighbors*: order the neighbors from closest to furthest for each node.

12.2.2. Merging solutions.

We represent the master solution and the scenario realization as binary strings: a '1' indicates that the node is selected, a '0' that it is not. As the master problem uses every node available and a scenario is a realization on this set of nodes, we can extract a *merged configuration* from the master solution and the scenario realization by performing a binary `and` between the two.

Such a configuration can occur given different master solution and/or scenario realization:

By keeping track of explored configurations, we can reduce the computational effort by simply recalling previous results. Figure 6 is a graphic representation of the merging procedure applied to the 2TSP in the contiguous U.S. instance, `att48`.

Master		Scenario		Configuration
$[0, 1, 1, 0]$	&	$[1, 0, 1, 0]$	=	$[0, 0, 1, 0]$
$[0, 1, 1, 1]$	&	$[1, 0, 1, 0]$	=	$[0, 0, 1, 0]$
$[0, 1, 1, 0]$	&	$[1, 0, 1, 1]$	=	$[0, 0, 1, 0]$

Table 11: Merging procedure: different master/scenario combinations can lead to the same configuration.

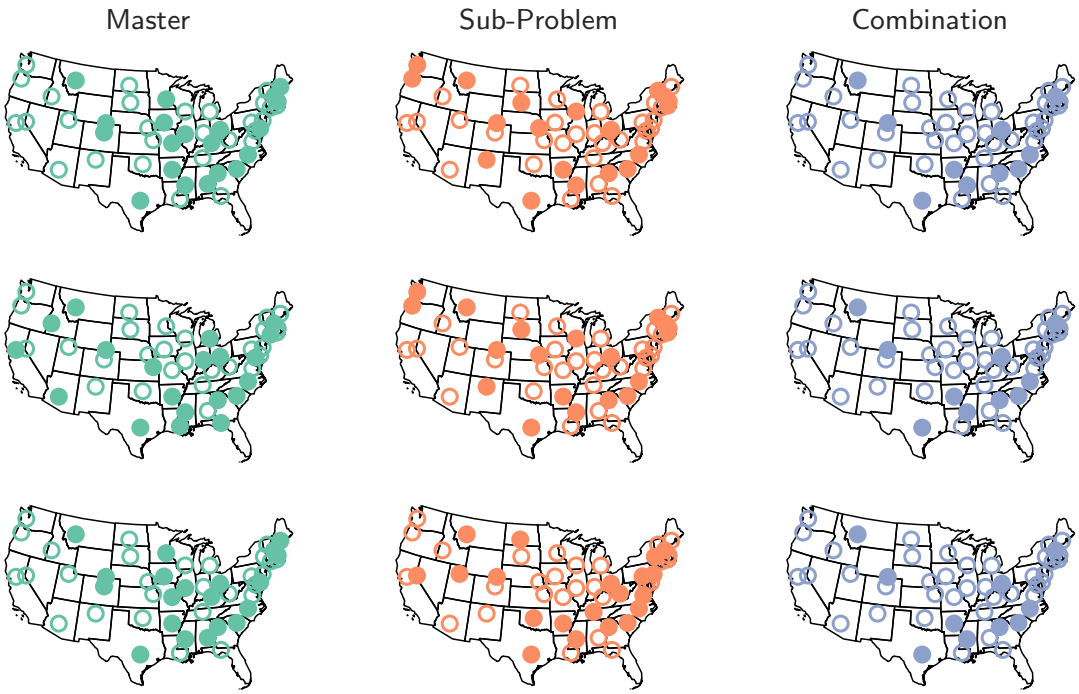

Figure 6: The first column contains the master configurations, the second the subproblem realizations, and the last is the resulting TSP.

12.2.3. Warm-up procedure.

One weakness of our UB&BC algorithm is the loss of information with regards to integer solutions of the subproblem. As a way to retain some of this information we can exploit exact solutions to the TSP. Because our goal is to avoid solving large MIPs repeatedly, we use the following warm-up procedure:

- Before starting the master B&B, solve every scenario as a MIP to optimality, we thus obtain *optimal tours* for each realization.
- Use these tours as *starting solutions* for the heuristics. Indeed, our heuristics are *local search heuristics* which means that they try to improve a starting solution. The quality of the initial tour may thus have a large influence on the final solution.
- Finally, we can also use the optimal tours as starting basis for the MIPs in the post-processing phase. Providing a MIP solver with an initial solution is a well-known strategy for speeding-up the process as it allows the solver to derive strong bounds early on.

12.2.4. Problem-specific heuristics.

Figure 7 presents the results of using the four local search heuristics on `fri26` and `bayg29` using 25 to 500 scenarios. The results show a clear difference based on the *quality* of the heuristic: the Greedy heuristic performs the worst, reaching the time limit before reaching 200 scenarios.

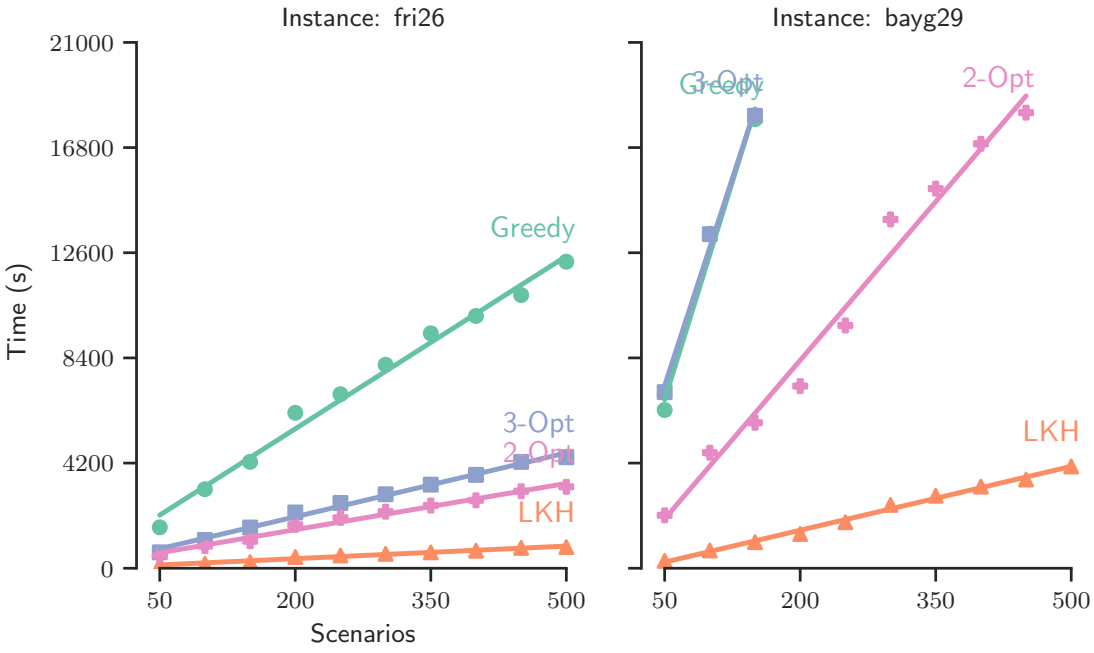

Figure 7: Comparison of different heuristics as upper-bounding procedures.

The difference in performance between 2- and 3-opt shows clearly with a larger number of scenarios. This is because the master B&B for 3-opt takes longer than for 2-opt, but this translates into a shorter post-processing phase. On a larger number of scenarios 3-opt is therefore a better choice.

Figure 8 shows a detailed execution on `fri26` with 100 scenarios. We report the evolution of the LP relaxation objective function value and the heuristic value per iteration – each integer solution explored in the master B&B. The black line shows the best upper bound.

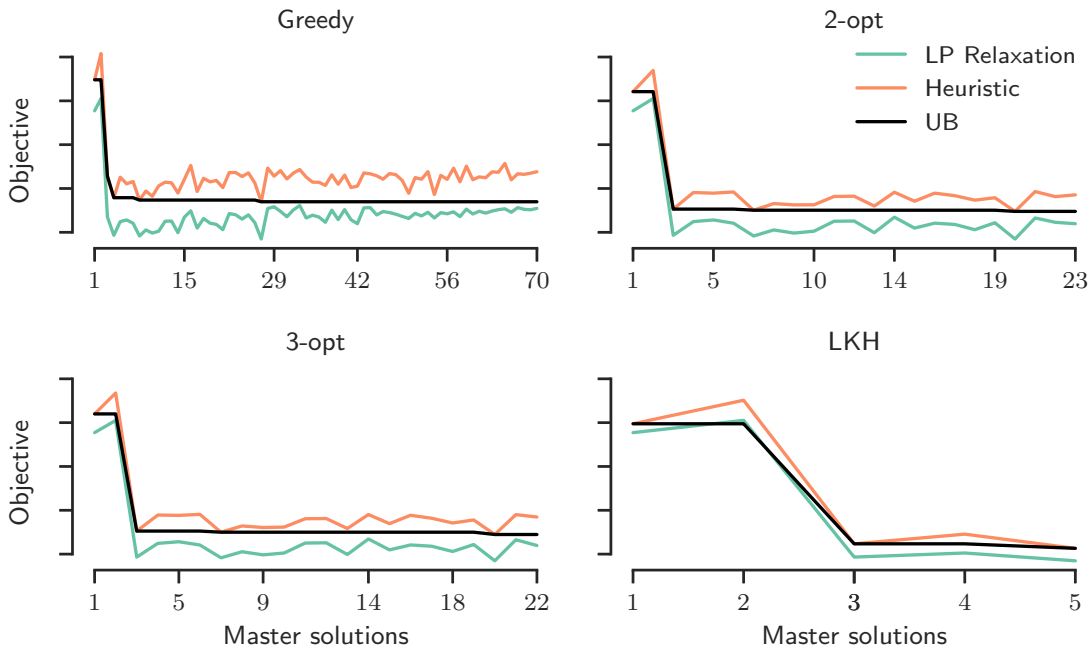

Figure 8: Upper, lower, and best upper bound values per iteration on `fri26` with 100 scenarios.

The main result is that the *better* the heuristic, the fewer iterations because the gap is closed much earlier. This example displays why better heuristics achieve better performances in the post-processing phase. Indeed, by exploring fewer solutions the algorithm has to solve fewer MIPs after the B&B finishes. We have two extreme cases between Greedy and LKH: the latter explores ten times fewer solutions by virtue of providing a solution very close to the optimal.

Overall, LKH dominates the results. It does more work at each integer solution of the master problem but reduces the number of solution explored to such an extent that it results in a much faster post-processing and overall solving time.

12.2.5. Zero-knowledge heuristics.

Using problem-specific heuristics still requires the user to develop, or at least implement, an efficient algorithm. We claim that our framework only requires knowledge of the model. We now present results

using an approach that does not require the use of a tailored algorithm: using the first feasible solution given by the MIP formulation of the subproblem. This heuristic was implemented using the same solver as the framework: CPLEX v12.7. Figure 9 provides the results compared to the best (LKH) and worst (Greedy) performing heuristics.

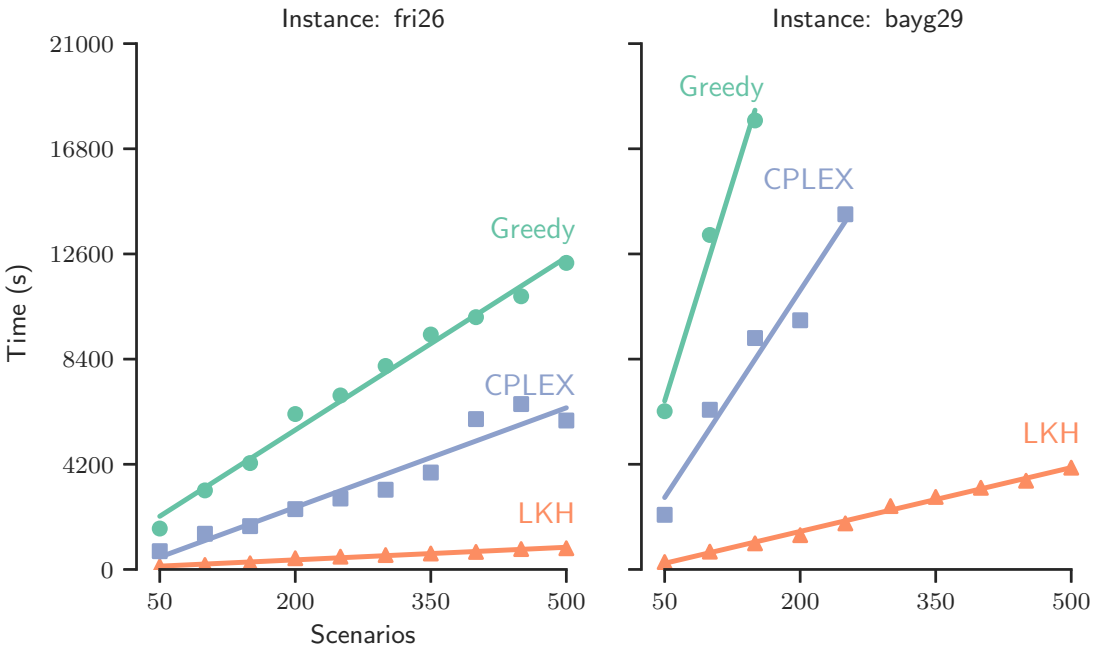

Figure 9: Comparison of the automatic heuristic vs. best performing heuristic, LKH, and worst performing, Greedy.

What is interesting is that although the automatic heuristic performs way worse than the best TSP heuristic, it is still better than Greedy. This shows that one needs to be careful when designing a heuristic, but if need be using a commercial solver can be useful. Using an exact heuristic has no benefit as the extra computational load far outweighs the chance of an early stop – which did not occur during our testing.

In conclusion, using a *strong* heuristic is critical in order to increase the convergence of the algorithm. A good heuristic reduces the number of solutions the B&B tree has to explore by providing a tight bound on the integer value. Also, we have demonstrated that solving the subproblem to optimality at each master solution is actually a computational burden that can be lightened by using a post-processing phase.

13. Detailed results SSLP

				Time (s)			
Servers	Customers	Scenarios	Config	Post-proc.	SP	Heur.	B&B
5	25	50	Base	0.077943	0.531158	0.116505	0.799818
			CPT	0.077585	1.956821	0.109472	2.206591
			Partial	0.076836	0.261214	0.056886	0.441129
		100	Base	0.156101	1.054196	0.230608	1.561920
			CPT	0.156787	7.159993	0.220886	7.700159
			Partial	0.153678	0.608542	0.141440	0.951720
	50	50	Base	0.084756	0.511853	0.083647	0.698363
			CPT	0.086029	4.985654	0.084826	5.179629
			Partial	0.086004	0.261080	0.043448	0.518671
		100	Base	0.170217	1.203006	0.202197	1.629878
			CPT	0.172479	8.910363	0.192744	9.328162
			Partial	0.170916	0.476216	0.076142	0.794671
10	50	50	Base	2.021315	8.005147	1.422345	12.550967
			CPT	0.000000	48.976449	1.549209	53.992955
			Partial	1.958222	8.546058	1.401625	13.420269
		100	Base	3.859151	22.521213	4.138341	38.874840
			CPT	0.000000	110.817125	3.315072	123.219645
			Partial	3.873439	14.808524	2.690785	24.276417
		500	Base	17.484479	126.401408	23.047190	374.476796
			CPT	0.000000	664.036933	18.835176	880.262527
			Partial	19.973128	92.364577	16.653616	256.777123
		1000	Base	34.283313	200.584712	34.465520	1101.529486
			CPT	0.000000	1287.790039	35.398000	2318.291747
			Partial	37.224851	175.986187	27.577001	658.798723
		2000	Base	0.000000	490.562348	89.146689	4605.410268
			CPT	0.000000	2533.492528	67.716871	6930.645608
			Partial	0.000000	331.985994	56.937112	1783.622511

Table 12: Time spent in the different parts of UB&BC runs, using different configurations

				Num. nodes	
Servers	Customers	Scenarios	Config	Solutions	Post-proc.
5	25	50	Base	18	1
			CPT	18	1
			Partial	9	1
		100	Base	18	1
			CPT	18	1
			Partial	11	1
	50	50	Base	14	1
			CPT	14	1
			Partial	7	1
		100	Base	16	1
			CPT	16	1
			Partial	6	1
10	50	50	Base	174	3
			CPT	174	0
			Partial	148	3
		100	Base	181	2
			CPT	181	0
			Partial	116	2
		500	Base	202	3
			CPT	202	0
			Partial	143	3
		1000	Base	191	3
			CPT	191	0
			Partial	137	3
		2000	Base	186	0
			CPT	166	0
			Partial	123	0

Table 13: Time spent in the different parts of UB&BC runs, using different configurations