Submitted to *INFORMS Journal on Computing*
manuscript (Please, provide the manuscript number!)

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

Arthur Mahéo
Monash University, Melbourne, Australia, arthur.maheo@monash.edu
Simon Belieres, Yossiri Adulyasak, Jean-François Cordeau
HEC Montréal, Montréal, Canada, first.last@hec.ca

Two-stage stochastic programs are a class of stochastic problems where data uncertainty is often 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 must be continuous. We propose a novel Benders decomposition-based framework that accommodates mixed-integer variables in both stages as well as uncertainty in all 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. We also study how enhanced Benders decomposition strategies such as the partial decomposition technique can be used to improve the algorithm's convergence. Through an extensive series of experiments, we demonstrate that the proposed framework performs better than state-of-the-art methods. It is able to solve some problem instances with more than one million variables in reasonable time.

Key words: 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* 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 $\mathbb{E}[\cdot]$ be the usual mathematical expectation operator with respect to $\widetilde{\omega}$. A standard formulation for two-stage SMIPs is:

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

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

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

The sets $\mathbb{X} \subseteq \mathbb{R}^{n_1}_+$ and $\mathbb{Y} \subseteq \mathbb{R}^{n_2}_+$ define the domains of the first-stage variables, x, and the second-stage variables, y_ω, respectively. Input parameters $c \in \mathbb{R}^{n_1}, A \in \mathbb{R}^{m_1 \times n_1}, b \in \mathbb{R}^{m_1}$ are known in advance, while $q_\omega \in \mathbb{R}^{n_2}, T_\omega \in \mathbb{R}^{m_2 \times n_1}, W_\omega \in \mathbb{R}^{m_2 \times n_2}, h_\omega \in \mathbb{R}^{m_2}$ 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:

$$\min \quad c^T \cdot x + \sum_{\omega \in \Omega} p_\omega \cdot q_\omega^T \cdot y_\omega \tag{DEF}$$

$$s.t. \quad Ax \geq b \tag{3a}$$

$$T_\omega \cdot x + W_\omega \cdot y_\omega \geq h_\omega \qquad \forall \omega \in \Omega \tag{3b}$$

$$x \in \mathbb{X},\ y_\omega \in \mathbb{Y}, \forall \omega \in \Omega,$$

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, $\mathbb{E}[h(x, \widetilde{\omega})]$ satisfies convexity, and standard decomposition-based approaches, such as Benders decomposition (Benders 1962), can be applied. On the other hand, $\mathbb{E}[h(x, \widetilde{\omega})]$ 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). To the best of our knowledge, one of the most efficient algorithms for generic two-stage SMIPs in the literature is that proposed by Qi and Sen (2017).

In this paper, we introduce the Unified Branch-and-Benders-Cut (UB&BC), a new Benders decomposition-based approach (Benders 1962) for solving two-stage SMIPs with general mixed-integer decisions in both stages and in all the recourse parameters. Our solution strategy can be embedded in any MIP solver with callbacks and can be used in conjunction with a heuristic applied to the recourse subproblem, which corresponds to a set of deterministic MIPs for the different scenarios. This is particularly useful when the (separated) subproblem has a combinatorial structure and one can leverage efficient heuristics originally developed for the deterministic version of that problem. While the method we propose is relatively simple to implement, it is competitive against the state-of-the-art approaches.

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.

As explained earlier, the standard Benders algorithm does not directly apply to two-stage SMIPs with discrete variables in the second stage. 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 $lb(\hat{x})$ 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 $ub(\hat{x})$. In Figure 1, we show the progress of the lower bound, $lb(\hat{x})$, and the upper bound, $ub(\hat{x})$, 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 $lb(\hat{x})$ 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).

Our contribution is threefold. First, we introduce a new exact algorithmic strategy for solving two-stage stochastic programs with general mixed-integer variables in both stages. Second, we assess its performance on instances of the stochastic server location problem (SSLP, Ntaimo and Sen 2005), which are frequently used to benchmark algorithms for

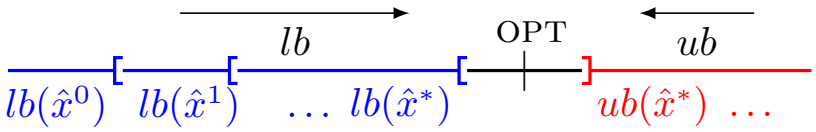

Figure 1 **UB&BC makes use of a heuristic to obtain the upper bound during the branch-and-bound process of the master problem. There could exist an integrality gap which must be closed through a post-processing procedure applied to the open solutions at the end of the branch-and-bound process.**

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). We computationally demonstrate that our approach is competitive against the state-of-the-art approaches and allows to solve open instances. Third, we present a series of experiments on instances of the two-stage stochastic traveling salesman problem with outsourcing (2TSP) to thoroughly analyze how the different components of UB&BC play a role in its performance. We also enhance our solution framework by adopting the partial Benders reformulation (Crainic et al. 2014, 2016) to improve the convergence of our algorithm.

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 our UB&BC while 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. We also summarize two versions of our method that differ according to the type of Benders decomposition solved. UB&BC Base uses a standard Benders decomposition, while UB&BC Enhanced uses a partial Benders decomposition (Crainic et al. 2014, 2016). The latter is expected to improve the convergence of UB&BC, but it only is applicable when parameters q_ω (cost parameters of the subproblem) and T_ω (nonzero coefficients of the first-stage decision variables) are not subject to uncertainty, which is the case of the majority of the two-stage stochastic

problems in the literature. More details on partial Benders reformulations are provided in Section 3.2.

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 Benders cuts.

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).

3. Unified Branch-and-Benders-Cut

We propose a Benders decomposition-based strategy for solving two-stage stochastic mixed-integer programs with general mixed-integer variables in both stages. The main advantages of the approach are its simple implementation and the fact that it requires no problem-specific components. We first present the Unified Branch-and-Benders-Cut (UB&BC) in the context of solving a standard Benders decomposition. The strategy operates a modified branch-and-cut to identify a set of open solutions and it performs a post-processing phase to determine the global optimum. Second, we discuss how using an enhanced Benders decomposition strategy rather than a standard Benders decomposition can improve the convergence of UB&BC. Specifically, we describe the partial Benders reformulation proposed in Crainic et al. (2014, 2016), which can be applied when parameters q_ω and T_ω are not subject to uncertainty.

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

	1st Stage			2nd Stage			Stochastic parameters			
	$\mathbb{R}$	$\mathbb{B}$	$\mathbb{Z}$	$\mathbb{R}$	$\mathbb{B}$	$\mathbb{Z}$	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)	⋆	⋆	⋆	⋆	⋆	⋆	⋆		⋆	⋆
Qi and Sen (2017)	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆
Atakan and Sen (2018)		⋆		⋆	⋆	⋆	⋆	⋆	⋆	⋆
Current study										
UB&BC Base	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆
UB&BC Enhanced	⋆	⋆	⋆	⋆	⋆	⋆		⋆	⋆	

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

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, respectively. The standard master problem is formulated as follows:

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

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

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

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

$$x \in \mathbb{X}.$$

The 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:

$$\begin{aligned} \min \quad & q_\omega^T \cdot y_\omega && \text{(Standard scenario subproblem)} \\ s.t. \quad & W_\omega \cdot y_\omega \geq h_\omega - T_\omega \cdot \hat{x} && \text{(5a)} \\ & y_\omega \in \mathbb{Y}. \end{aligned}$$

The 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 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 problem, and (ii) solves the scenario subproblems heuristically to determine a feasible second-stage solution when possible. Let $opt(\hat{x})$ be the objective function value obtained

Algorithm 1: Unified Branch-and-Benders-Cut

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

$Z = \emptyset, ub^* = \infty$

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* $\geq ub^*$ **then**
  - Fathom branch
- **else if** *An integer solution* $\hat{x}$ *of the* MP *is found* **then**
  - **foreach** *Scenario* $\omega \in \Omega$ **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 lower bound $lb(\hat{x})$ and add $(\hat{x}, lb(\hat{x}))$ to Z
    - **foreach** *Scenario* $\omega \in \Omega$ **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 $ub(\hat{x})$
      - $ub^* = \min(ub^*, ub(\hat{x}))$
- **else**
  - Choose a variable to branch on

Rank elements of Z by ascending order according to their lower bound values

while $Z \neq \varnothing$ **do**
- Select $(\hat{x}, lb(\hat{x}))$ from Z
- **if** $lb(\hat{x}) \leq ub^*$ **then**
  - **foreach** *Scenario* $\omega \in \Omega$ **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

when solving to integer optimality the scenario subproblems associated with $\hat{x}$. 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 $opt(\hat{x})$ 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. By combining $\hat{x}$ with the optimal solutions to the subproblem LP relaxations, we obtain a lower bound on $opt(\hat{x})$, $lb(\hat{x})$. We then solve the scenario subproblems heuristically. If a heuristic solution has been found for all scenario subproblems, we compute an upper bound on $opt(\hat{x})$, $ub(\hat{x})$, by combining $\hat{x}$ and the heuristic solutions. This heuristic upper bound can be used to prune nodes in the branch-and-bound (B&B) tree. Specifically, if $ub(\hat{x})$ 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 $ub(\hat{x})$ 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 when there is no active node remaining. At the end of the second phase, 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 $lb(\hat{x})$, 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 second phase, the incumbent value is used to discard saved master solutions $\hat{x}$ such that $lb(\hat{x})$ 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 A.

3.2. 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: $h_{\omega'} = \sum_{\omega\in\Omega} \alpha_\omega^{\omega'} h_\omega$ and $W_{\omega'} = \sum_{\omega\in\Omega} \alpha_\omega^{\omega'} W$. 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:

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

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

$$T \cdot x + W_{\omega'} \cdot y_{\omega'} \geq h_{\omega'} \tag{6b}$$

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

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

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

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

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.

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* as well as its standard Benders reformulation. We also describe the partial Benders reformulation, as both problems involve fixed recourse matrices and fixed recourse costs, and the heuristic used to solve the Benders subproblem. Finally, we describe the characteristics of the test instances.

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 C.

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:

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

$$s.t. \quad \sum_{j\in J} x_j \leq V \qquad \text{(7a)}$$

$$\sum_{j\in J} x_j \geq w_z \qquad \forall z \in Z \qquad \text{(7b)}$$

$$\sum_{i\in I} d_{ij} \cdot y_{ij}^\omega - y_{j0}^\omega \leq D \cdot x_j \qquad \forall j \in J, \forall \omega \in \Omega \qquad \text{(7c)}$$

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

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

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.

Let I^ω be the set of clients that are present in scenario $\omega \in \Omega$. To generate cuts we use the dual relaxed subproblem associated with the projected y variables. 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:

$$\max \quad \sum_{i \in I^\omega} v^\omega_i - \sum_{j \in J} D \cdot \hat{x}_j \cdot u^\omega_j \qquad \text{(Sub}[\omega]\text{)}$$

$$s.t. \quad -d_{ij} \cdot u^\omega_j + v^\omega_i \leq -q_{ij} \qquad \forall i \in I^\omega, j \in J \qquad (8a)$$

$$u^\omega_j \leq q_{j0} \qquad \forall j \in J \qquad (8b)$$

$$u^\omega_j \geq 0, v^\omega_i \in \mathbb{R}.$$

Then, the master problem is formulated as follows:

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

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

$$\sum_{j \in J} u^\omega_j \cdot D \cdot x_j - \sum_{i \in I^\omega} v^\omega_i \leq z_\omega \qquad (u^\omega, v^\omega) \in \mathcal{O}^\omega, \forall \omega \in \Omega \qquad (9a)$$

$$x_j \in \mathbb{B}, \quad z_\omega \geq 0.$$

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. 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:

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

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

$$\sum_{i\in I} d_{ij}\cdot y_{ij}^{\omega'} - y_{j0}^{\omega'} \le D\cdot x_j \qquad \forall j\in J \qquad (10a)$$

$$\sum_{j\in J} y_{ij}^{\omega'} = h_i^{\omega'} \qquad \forall i\in I \qquad (10b)$$

$$\sum_{i\in I}\sum_{j\in J} q_{ij}\cdot y_{ij}^{\omega'} - \sum_{j\in J} q_{j0}\cdot y_{j0}^{\omega'} = \sum_{\omega\in\Omega} p_\omega \cdot z_\omega \qquad (10c)$$

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

The standard master problem is enhanced with constraints (10a) to (10c). Constraints (10a) and (10b) ensure that all master problem solutions are feasible for the artificial scenario, while constraint (10c) enforces the estimate of the recourse costs to reflect the expected revenue associated with the artificial scenario.

4.1.4. 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 B.1 (see Algorithm 2).

4.1.5. 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 B.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 8.

4.2. Traveling salesman with outsourcing

To demonstrate the efficiency of UB&BC, we also introduce the two-stage stochastic traveling salesman with outsourcing (2TSP). This novel problem is a 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 whereas the customers that will not be served by the vehicle will be outsourced (with an extra fixed cost per customer). 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 a service. 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* (11b) 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;
- d_{ij}, the distance between two nodes; and

- $E(Y) = \{ (i,j) \mid i < j, (i,j) \in Y^2 \}$, 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 binary variables h_i^ω to represent whether customer $i \in N$ has a request. The penalty for not visiting a customer $i \in N$ is b_i. 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.

This leads to the following formulation:

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

$$s.t. \quad \sum_{i \in N} x_i \geq C \tag{11a}$$

$$\sum_{j \in N} y_{ij}^\omega = 2 \cdot x_i \cdot h_i^\omega \qquad \forall i \in N \tag{11b}$$

$$\sum_{i,j \in E(S^\omega)} y_{ij}^\omega \leq |S^\omega| - 1 \qquad S^\omega \subseteq N^\omega, |S^\omega| \geq 2, \forall \omega \in \Omega \tag{11c}$$

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

The constraint (11a) ensures that at least C customers are visited. Constraints (11b) are the degree constraints. Constraints (11c) are the sub-tour elimination constraints.

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 and v^ω as the dual variables associated with constraints (11b) and (11c), 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:

$$\max \quad \sum_{i \in N} 2 \cdot \hat{x}_i \cdot h_i^\omega + \sum_{\substack{S^\omega \subset N^\omega \\ |S^\omega| \geq 2}} (1 - |S^\omega|) v_s^\omega \qquad \text{(Sub}[\omega]\text{)}$$

$$s.t. \quad u_i^\omega - \sum_{\{\, s \in S^\omega | i,j \in s \,\}} v_s^\omega \leq c_{ij} \qquad \forall i,j \in E(N) \qquad \text{(12a)}$$

$$u_i^\omega \in \mathbb{R}, v_s^\omega \geq 0.$$

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:

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

$$s.t. \quad \text{(11a)}$$

$$\sum_{i \in N} 2 \cdot u_i^\omega \cdot x_i \cdot h_i^\omega + \sum_{\substack{S^\omega \subset N^\omega \\ |S^\omega| \geq 2}} v_s^\omega (|S^\omega| - 1) \leq z_\omega \qquad \forall (u^\omega, v^\omega) \in \mathcal{O}^\omega, \forall \omega \in \Omega \qquad \text{(13a)}$$

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

The objective function aims to minimize the penalties for not including customers in the tour. Constraints (13a) 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. 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:

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

$$s.t. \quad (11a)-(13a)$$

$$\sum_{j\in N} y_{ij}^{\omega'} = 2\cdot x_i \cdot h_i^{\omega'} \qquad \forall i\in N \qquad (14a)$$

$$\sum_{i,j\in E} c_{ij}\cdot y_{ij}^{\omega'} = \sum_{\omega\in\Omega} p_\omega \cdot z_\omega \qquad (14b)$$

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

The standard master problem is enhanced with constraints (14a) ensuring that all master problem solutions are feasible for the artificial scenario and constraints (14b) enforcing the estimate of the recourse costs to reflect the cost of the tour computed for the artificial scenario.

4.2.4. Subproblem heuristics. Considering how hard optimal solutions are to obtain, we can make use of well-known TSP heuristic 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 D.2.1. This heuristic is considered the state of the art and has a complexity of $\mathcal{O}(n^{2.2})$.

4.2.5. 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 25 and 500, with an increment of 50.
- *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 first compare the two versions of our method, UB&BC Base and UB&BC Enhanced (i.e., based on the standard and the partial Benders decomposition, respectively), to the state-of-the-art approaches for solving two-stage SMIPs with discrete recourse. Performance indicators are obtained by solving different instances of the stochastic server location problem (SSLP), which is acknowledged as the current benchmark to compare 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 next 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 Calcul Québec computing cluster.[1] The experiments are conducted on an Intel Xeon X5650 processor with a 2.67GHz CPU on a single thread and with 12 GB of RAM. Linear and integer programs are solved using CPLEX v12.7.

[1] Complete specifications available on calculquebec.ca.

5.1. Comparison with state-of-the-art approaches for two-stage stochastic integer programs

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		
	$\mathbb{R}$	$\mathbb{B}$	$\mathbb{Z}$	$\mathbb{R}$	$\mathbb{B}$	$\mathbb{Z}$
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 C.3.

We compare UB&BC with five state-of-the-art solution approaches:

- the disjunctive decomposition algorithm (D^2) proposed by Sen and Higle (2005);
- the integrated progressive hedging dual decomposition algorithm (PH-DD) proposed by Guo et al. (2015);
- the progressive hedging branch-and-bound (PH-B&B) proposed by Atakan and Sen (2018);
- 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 three approaches are benchmarked on the 1st SSLP variant. The fourth and the fifth approach are benchmarked on the 2nd and the 3rd SSLP variants, respectively.

To provide a fair comparison between our results and those presented in the different articles, we scale computation times according to the type of processor employed. To do so, we use the PassMark single thread rating[2] as a performance indicator of the processors and we determine scaling coefficient values accordingly. In Table 3, for each article we report the type of processor employed, the associated single thread rating, and the scaling coefficient. Note that we have not been able to provide scaling coefficients for the two first benchmark approaches. Indeed, the single thread rating of the processor used by Sen and Higle (2005) is not available and Guo et al. (2015) did not specify the model of their processor. Also, 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.

Method	SSLP variant	Processor	Single thread rating	Scaling coefficient
D^2	1	UltraSPARC-III+ (900MHz)	-	-
PH-DD	1	Unknown (3.1GHz)	-	-
PH-B&B	1	Intel i7-3770S (3.1GHz)	2,006	1.52
Gomory	2	Intel CoreQuad (2.66GHz)	1,079	0.82
ABC	3	Intel i7-3770K (3.5GHz)	2,066	1.57
UB&BC	1,2,3	**Intel Xeon X5650 (2.67GHz)**	**1,318**	**1.00**

Table 3 CPU characteristics

5.1.1. 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 4, 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.

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

	UB&BC					
	Base	Enhanced	D^2	PH-DD	PH-B&B	
Instance	Time (s)	Time (s)	Time_o (s)	Time_o (s)	Time_o (s)	Time_s (s)
SSLP_5_25_50	0.72	0.54	0.53	-	0.50	0.76
SSLP_5_25_100	1.38	1.06	1.06	-	1.10	1.67
SSLP_10_50_50	32.01	28.14	239.95	74.00	8.70	13.24
SSLP_10_50_100	61.88	54.10	480.46	175.00	18.60	28.31
SSLP_10_50_500	254.41	205.62	1,902.20	1,033.00	80.80	122.98
SSLP_10_50_1000	589.62	437.15	5,410.10	-	163.60	249.00
SSLP_10_50_2000	1161.98	818.72	9,055.29	-	309.60	471.21
SSLP_15_45_5	5.38	1.81	110.34	-	1.40	2.13
SSLP_15_45_10	34.19	23.01	1,494.89	45.00	2.40	3.65
SSLP_15_45_15	156.07	149.74	7,210.63	123.00	3.20	4.87

Table 4 Performance on the 1st SSLP variant

As expected, using the partial Benders reformulation is beneficial as, overall, UB&BC Enhanced 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. Because the speed (900MHz) reported in Sen and Higle (2005) is largely lower than ours (2.67GHz), it is impossible to infer if UB&BC outperforms D^2. On the other hand, due to the speed (3.1GHz) reported in Guo et al. (2015), we assume that their processor is competitive with ours, and thus that both versions of UB&BC actually outperform PH-DD.

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 Enhanced 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 all the 3rd SSLP variant.

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

Again, UB&BC Enhanced outperforms UB&BC Base as it converges 1.38 times faster. Both versions of UB&BC are less efficient than Gomory on the instances that involve 5 servers. Nevertheless, these instances are relatively simple as they are all solved within 2

	UB&BC			
	Base	Enhanced	Gomory	
Instance	Time (s)	Time (s)	Time_o (s)	Time_s (s)
SSLP_5_25_50	1.41	0.54	0.18	**0.15**
SSLP_5_25_100	0.72	1.07	0.22	**0.18**
SSLP_5_50_50	29.59	**25.98**	109.20	89.48
SSLP_5_50_100	51.76	**44.61**	218.42	178.98
SSLP_5_50_500	260.14	**195.55**	740.38	606.68
SSLP_5_50_1000	540.47	**359.57**	1,615.42	1,323.71
SSLP_5_50_2000	1100.17	**759.74**	2,729.61	2,236.71

Table 5 Performance on the 2nd SSLP variant

seconds by the different methods. On the other hand, both versions of UB&BC terminate faster than Gomory on all the instances with 10 servers. For these instances, UB&BC Enhanced converges 3.37 times faster than Gomory overall, while UB&BC Base converges 2.66 times faster than Gomory overall. Again, it should be noted that Gomory is not as generic as our approach as it does not accommodate general integer variables in the first stage.

5.1.3. Comparison with ABC (3rd SSLP variant). We solve instances of the 3rd SSLP variant and we compare two versions of UB&BC to 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'.

	UB&BC		ABC			UB&BC		ABC	
	Base	Enhanced				Base	Enhanced		
Instance SSLS	Time (s)	Time (s)	Time_o (s)	Time_s (s)	Instance SSLS	Time (s)	Time (s)	Time_o (s)	Time_s (s)
(2×5)_(5×5)_50	0.22	**0.15**	0.30	0.47	(3×5)_(10×5)_500	**6.69**	7.26	27.15	42.56
(2×5)_(5×5)_100	0.31	**0.29**	0.38	0.60	(3×5)_(15×5)_50	**0.98**	1.16	191.92	300.84
(2×5)_(5×5)_500	**2.12**	3.69	3.58	5.61	(3×5)_(15×5)_100	**1.74**	4.02	19.56	30.66
(2×5)_(10×5)_50	**0.39**	0.52	0.52	0.82	(3×5)_(15×5)_500	**7.39**	12.55	1069.92	1,677.13
(2×5)_(10×5)_100	**1.10**	1.23	4.84	7.59	(4×5)_(5×5)_50	**0.40**	0.54	1.59	2.49
(2×5)_(10×5)_500	**2.95**	4.94	4.43	6.94	(4×5)_(5×5)_100	**0.55**	1.23	3.36	5.27
(2×5)_(15×5)_50	**0.89**	0.99	4.26	6.68	(4×5)_(5×5)_500	**3.99**	5.84	20.99	32.90
(2×5)_(15×5)_100	**0.70**	1.00	1.64	2.57	(4×5)_(10×5)_50	**1.11**	2.11	3.60	5.64
(2×5)_(15×5)_500	**6.35**	8.13	51.28	80.38	(4×5)_(10×5)_100	**2.77**	4.76	261.89	410.52
(3×5)_(5×5)_50	0.22	**0.18**	0.59	0.92	(4×5)_(10×5)_500	**13.10**	31.52	745.61	1,168.76
(3×5)_(5×5)_100	**0.54**	0.58	1.23	1.93	(4×5)_(15×5)_50	**3.00**	5.70	1653.67	2,592.17
(3×5)_(5×5)_500	**3.45**	4.56	7.09	11.11	(4×5)_(15×5)_100	**3.14**	5.40	t/o	5,643.10
(3×5)_(10×5)_50	**3.00**	3.70	4.68	7.34	(4×5)_(15×5)_500	**84.09**	100.64	t/o	5,643.10
(3×5)_(10×5)_100	**2.19**	2.42	158.61	248.63					

Table 6 Performance on the 3rd SSLP variant

We observe that using the partial Benders reformulation is not beneficial in that case. Indeed, UB&BC Base outperforms UB&BC Enhanced on 24 out of the 27 instances, and

it converges 1.44 faster overall. 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. 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. Nevertheless, we observe that both versions of UB&BC are strictly more efficient than the ABC algorithm. Both versions of UB&BC close the two open instances and they always converge faster than ABC. The total computation time required by UB&BC Base to solve all the instances is of 153.38 seconds. On the other hand, 8 instances cannot be solved by ABC within this amount of time. Overall, both versions of UB&BC are more than 80 times faster than ABC.

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) described in Section 4.2.5. 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 Enhanced, namely: merging procedure (Appendix D.2.2) and subproblem warm-up (Appendix D.2.3).

The results in Figure 2 and Table 7 clearly show the superiority of the UB&BC over CPLEX. When solving the DEF, CPLEX only manages to return results for small instances, and these results are all timeouts (cluster near the 21,000s mark).

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 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 D.1 and D.2, respectively.

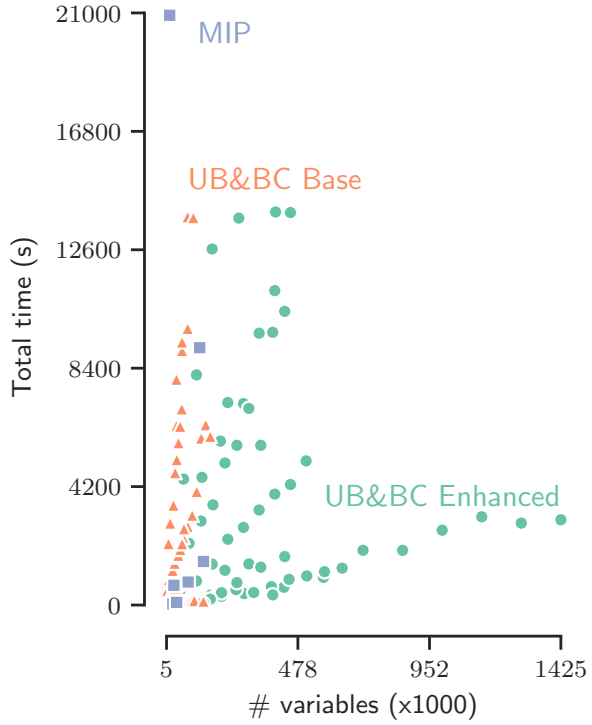

Figure 2 **Solving time per (estimated) size of 2TSP instances using a generic MIP solver, UB&BC Base or UB&BC Enhanced.**

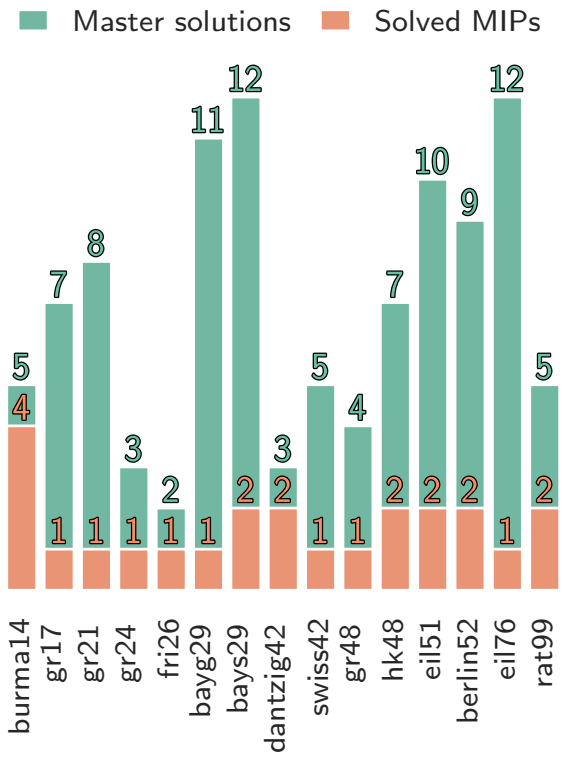

Figure 3 **Number of master solutions explored and and the number of MIPs solved in the post-processing phase when using 100 scenarios.**

		UB&BC				UB&BC	
Instance (/10)	MIP	Base	Enhanced	Instance (/10)	MIP	Base	Enhanced
burma14	6	10	10	swiss42	0	0	10
ulysses16	7	10	10	hk48	0	0	8
gr17	10	10	10	att48	0	0	0
gr21	3	10	10	gr48	0	0	9
ulysses22	7	0	0	eil51	0	0	10
gr24	6	10	10	berlin52	0	0	5
fri26	1	10	10	brazil58	0	0	0
bays29	0	4	10	st70	0	0	0
bayg29	0	5	10	eil76	0	0	10
dantzig42	0	0	10	pr76	0	0	0

Table 7 **Number of instances of 2TSP solved using a MIP or the UB&BC approach.**

6. Conclusions

In this paper, we have 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. In the case of two-stage stochastic programs with fixed recourse matrix and fixed recourse cost, it is possible to strengthen the master problem with a partial Benders reformulation and improve the convergence of our approach.

Through an extensive series of experiments carried out on instances of the stochastic server location problem (SSLP), we have computationally demonstrated that both versions of UB&BC, with and without partial Benders reformulation, are competitive against the state-of-the-art solution algorithms for two-stage SMIPs with discrete recourse. In addition, we have 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 the 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 also have highlighted how using an efficient bounding heuristic can significantly improve the algorithm's speed of convergence.

Finally, the advantages of our framework are simplicity and flexibility. We believe that this straightforward approach is a great candidate for building more advanced algorithms and tackle larger, more difficult problems. There are computational enhancements which 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.

Acknowledgments

Computations were made on the supercomputer "Briarée" from Université de Montréal, managed by Calcul Québec 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.

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.

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.

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 réseaux d'entreprise, la logistique et le transport, Université de Montréal, Montréal 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 réseaux d'entreprise, la logistique et le transport, Université de Montréal, Montréal 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`.

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.

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

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

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.

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.

Appendix A: 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:

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

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:

$$\begin{aligned} \min \quad & 6x_1 + 10x_2 + q && && \text{(Toy Master)} \\ s.t. \quad & -\lambda_1(15x_1 + 22x_2) + 1.5\lambda_2 - 2(\lambda_3 + \lambda_4) \leq q && \forall \lambda_i \in \mathcal{O} && \text{(17a)} \\ & -\lambda_1(15x_1 + 22x_2) + 1.5\lambda_2 - 2(\lambda_3 + \lambda_4) \leq 0 && \forall \lambda_i \in \mathcal{F} && \text{(17b)} \\ & x \in \mathbb{B}. \end{aligned}$$

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

A.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:

$$-15x_1 - 22x_2 + 7.5 \leq 0 \tag{18}$$

2. Augmented by the new constraint (18), 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):

$$1.5 \leq q \tag{19}$$

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.

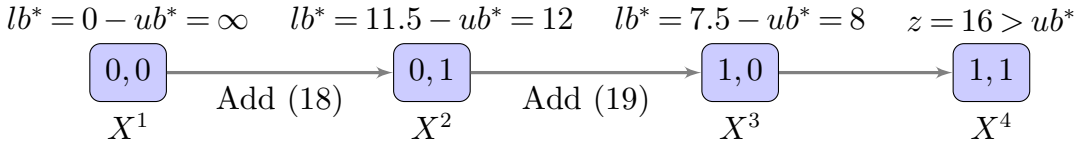

Figure 4 **Example search when using UB&BC with problem** (Toy).

A.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.

Appendix B: Supplement to the experimental design

B.1. SSLP subproblem heuristic

Algorithm 2: Allocation heuristic for the SSLP.

Data: The set of nodes N

Data: The set of opened servers J

$S_j = 0, \forall j \in J$ **foreach** $i \in N$ **do**
- $D_i = \{ d_{ij} \mid \forall j \in J \}$
- Sort each D_i in ascending order
- $\delta_i = D_i^0 - D_i^1$

$N' =$ sort N by decreasing opportunity δ

foreach $i \in N'$ **do**
- $k = \arg\min_{k \in J} S_k + D_i^k \leq D$
- $S_k = S_k + D_i^k$

Result: An allocation of nodes i to servers j

B.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	{50, 100}
2	10	50	{50, 100, 500, 1000, 2000}
3	15	45	{5, 10, 15}

Table 8 **Instance class characteristics**

Appendix C: SSLP variants

Algorithm 3: Allocation heuristic for the SSLS.

Data: The vector of active clients $N \in \mathbb{N}^n$

Data: The vector of opened servers $J \in \mathbb{N}^m$

$S_j = 0$, $\forall j \in J$ **foreach** $n_i \in N$ **do**
  $D_i = \{ d_{ij} \mid \forall m_j \in J \}$
  Sort each D_i in ascending order
  $\delta_i = D_i^0 - D_i^1$
$N' =$ sort N by decreasing opportunity δ
foreach $n_i \in N'$ **do**
  for $n = 0$; $n < n_i$; n+=1 **do**
    $k = \arg\min_{j \in J} S_k + D_i^k \leq D * j$
    $S_k = S_k + D_i^k$

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.

C.1. 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: $y_{j0} \in \mathbb{N}$

C.2. 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 problem 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.

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

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	{50, 100, 500}
2	2	5	10	5	{50, 100, 500}
3	2	5	15	5	{50, 100, 500}
4	3	5	5	5	{50, 100, 500}
5	3	5	10	5	{50, 100, 500}
6	3	5	15	5	{50, 100, 500}
7	4	5	5	5	{50, 100, 500}
8	4	5	10	5	{50, 100, 500}
9	4	5	15	5	{50, 100, 500}

Table 9 Instance class characteristics

C.3. 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 3.

Appendix D: Analyzing the performance of UB&BC for 2TSP

D.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 4.1.3). In Figure 5 we present the results of using the enhanced master formulation.

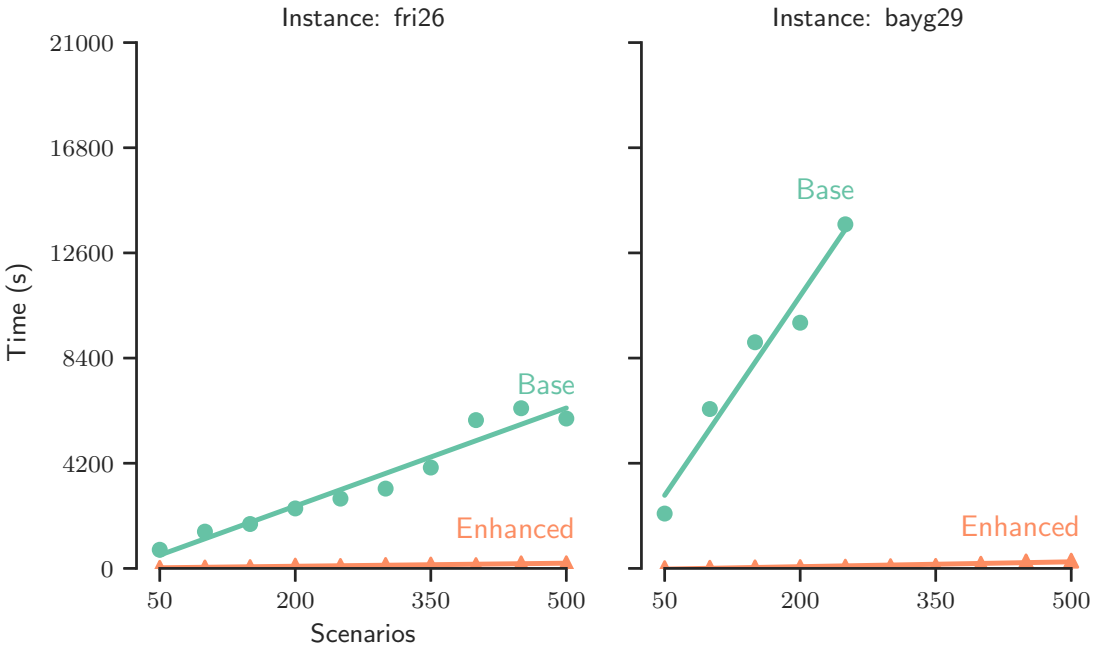

Figure 5 **Solving time of the Base formulation (Auto heuristic and regular MP) and the Enhanced formulation (warmed LKH and extended MP).**

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 Enhanced formulation 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.

D.2. Impact of the subproblem heuristic

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

Instance	Base UB&BC	Enhanced 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.

D.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.

D.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:

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.

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`.

D.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:

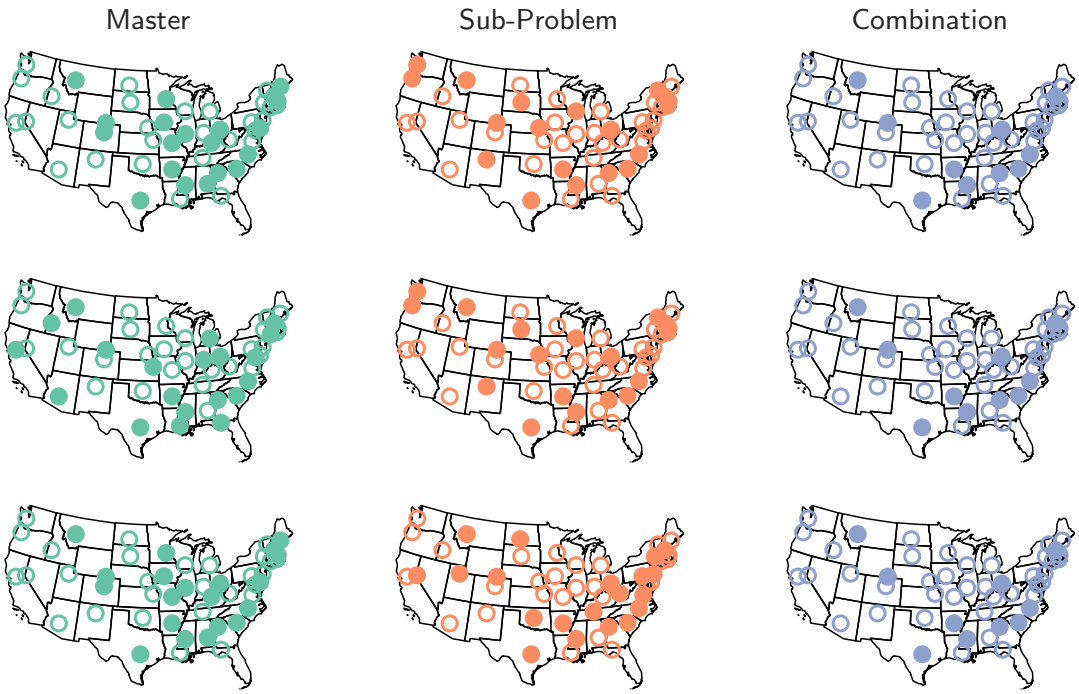

Figure 6 **The first column contains the master configurations, the second the subproblem realizations, and the last is the resulting TSP.**

- 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.

D.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.

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.

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,

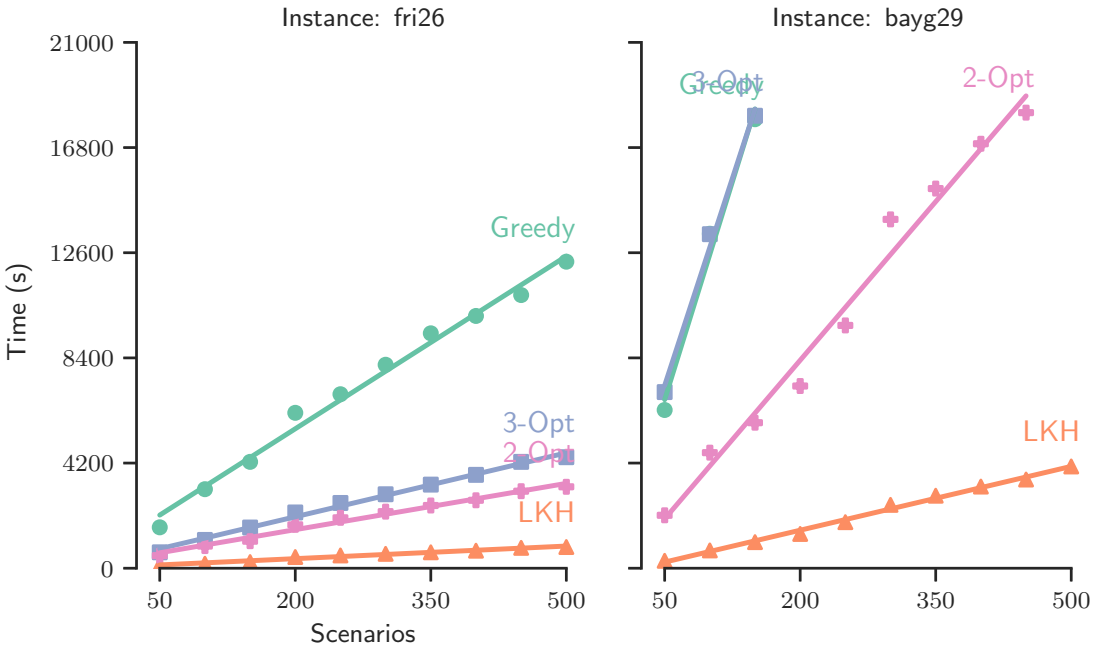

Figure 7 **Comparison of different heuristics as upper-bounding procedures.**

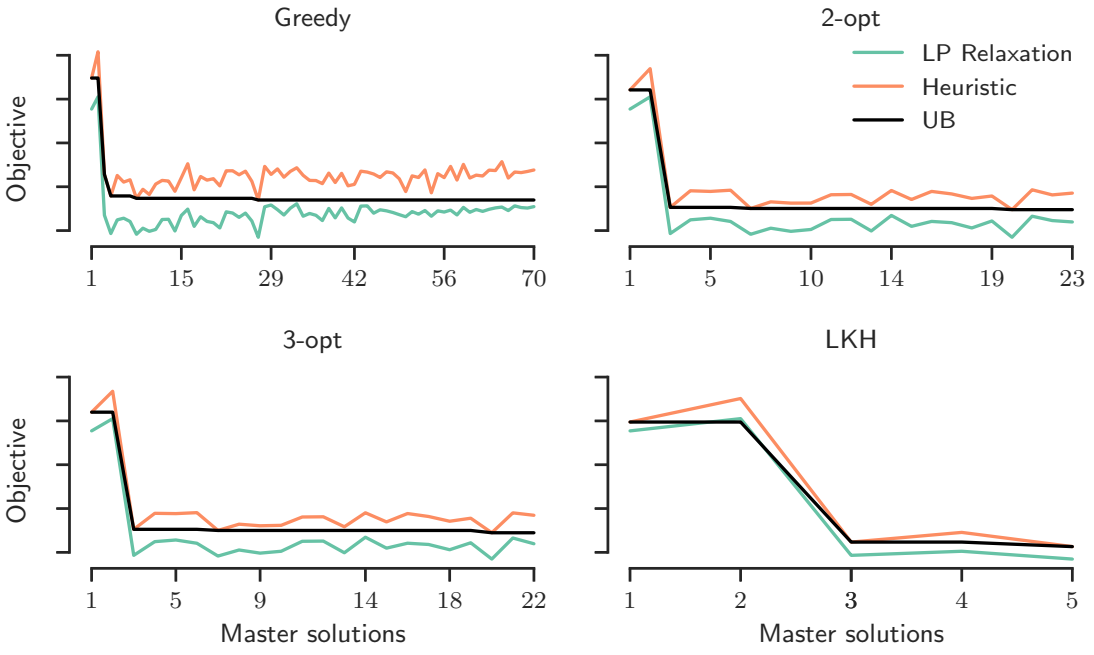

Figure 8 **Upper, lower, and best upper bound values per iteration on `fri26` with 100 scenarios.**

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.

D.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.

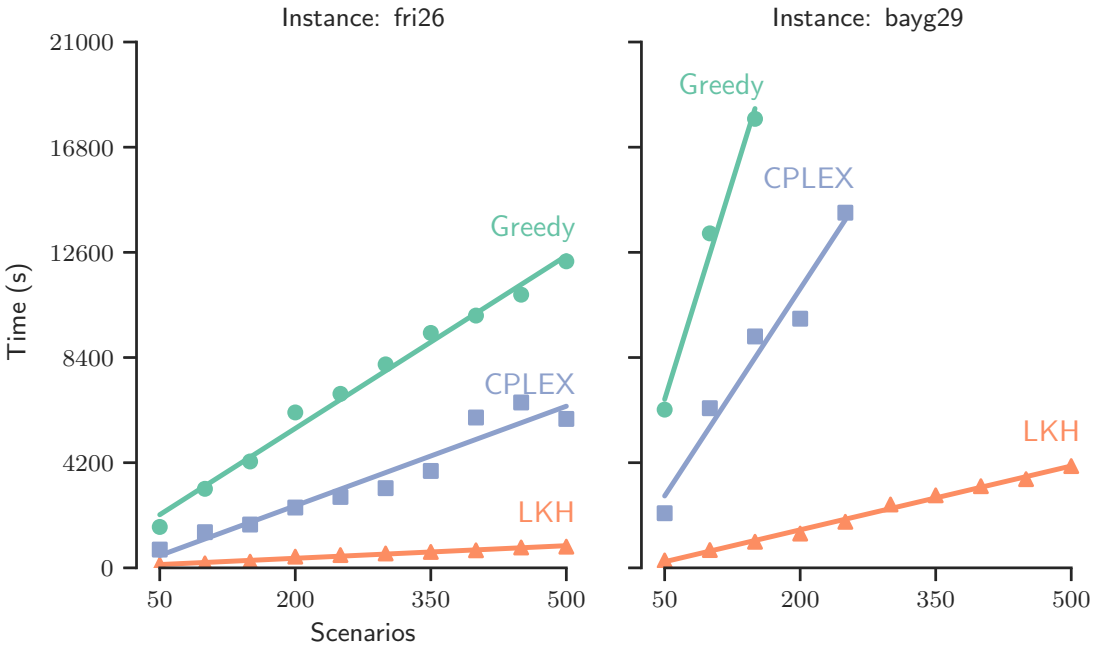

Figure 9 **Comparison of the automatic heuristic vs. best performing heuristic, LKH, and worst performing, Greedy.**

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.