Noname manuscript No.
(will be inserted by the editor)

SIPLIB 2.0

Stochastic Integer Programming Library version 2.0

Yongkyu Cho · Kibaek Kim · Cong Han Lim · James Luedtke · Jeffrey Linderoth

Received: date / Accepted: date

Abstract We present a collection of stochastic integer programming problem instances.

Keywords Stochastic Integer Programming · Problem Instances

1 Introduction

`SIPLIB` [2] is an abbreviated term of the Stochastic Integer Programming (SIP) LIBrary firstly contructed in 2002 by Shabbir Ahmed and his colleagues. The library has been providing a collection of test instances to facilitate computational and algorithmic research in SIP. Some new test problems with instances have been added to `SIPLIB` gradually and now it contains nine different problems in total. The instances are basically given in the standard `SMPS` format accompanied with additional information including parameter data, size of the instance in terms of the number of rows, columns, and integers, benchmarking information such as best known objective value or bounds, optimality gap, and solution time.

At the time `SIPLIB` appeared, it provided enoughly large-sized instances that is reasonable to argue that the performance of algorithm is remarkable if it solves the instances. The state-of-art in SIP combined with the speedup in computing machinery, however, makes many instances of `SIPLIB` trivial so that we have no more basis to use `SIPLIB` for showing the excellence of

Yongkyu Cho, Kibaek Kim
Mathematics and Computer Science Division, Argonne National Laboratory, Lemont, IL 60439, USA
E-mail: choy@anl.gov; kimk@anl.gov

Cong Han Lim · James Luedtke · Jeffrey Linderoth
Department of Industrial and Systems Engineering, University of Wisconsin-Madison Madison, WI 53706, USA
E-mail: clim9@wisc.edu; jim.luedtke@wisc.edu; linderoth@wisc.edu

new solution methods. At this point, we are motivated to develop the second version of SIPLIB, say SIPLIB 2.0 that provides larger-sized test instances with higher degree of tailorability, e.g., users can expand the size of instance as much as they want in terms of the number of scenarios included.

Stochastic programming (SP) is a framework for modeling optimization problems that involve uncertainty. Whereas optimization problems are typically formulated with known parameters, the problems in real world contain some unknown parameters in many cases. For details on SP, see, e.g., [3,4]. SIP is a branch of SP that indicates any type of SP including at least one integer decision variable. The integers can be placed anywhere in general SIP. However, we restrict our focus on two-stage SIP that contains integer variables (including binary) in its second stage throughout this paper and SIPLIB 2.0. The main reason is that the class of SIP is most widely used to model real world problems. Moreover, two-stage SIP itself has enough difficulties that have not been conquered yet even without any other details like chance-constraints and multi-stages. The main difficulty in solving two-stage SIP is that the second-stage value function is not necessarily convex, but only lower semicontinuous. Thus, the standard decomposition approaches that work nicely for stochastic *linear* programs, break down when the second stage integer variables are present [5]. Hereinafter, we use the term SIP to indicate the two-stage SIP that contains integer variables in its second stage.

We provide the test sets in two formats: SMPS files (*.cor, *.tim, *.stoch) and Julia files (*.jl). SMPS is widely used to describe stochastic linear and quadratic programs. Once having SMPS files of a problem instance, we can directly solve it using various mixed integer linear program (MILP) solvers such as CPLEX, GUROBI, and CBC. One can also use the existing open-source SIP solvers like DSP [6], PySP [7], and SMI [8] given that SMPS files. A drawback of SMPS format is its low readability by human, which we decided to provide Julia files to let users be able to easily read problems and tailor the instances.

Julia is an open source high-level, high-performance dynamic programming language for numerical computing. It is also known as its good performance, approaching that of statically-compiled languages like C [9]. The syntax of Julia is simple and should feel familiar to anyone who has experienced in another high-level languages like MATLAB or Python. A Julia package called JuMP (Julia for Mathematical Programming [10]) provides a domain-specific modeling language for mathematical optimization embedded in Julia. JuMP enables us to easily translate mathematical model to JuMP.Model-type object. Some structured mathematical models like SIP can also be translated to the JuMP.Model-type object combined with the package StructJuMP [11]. Once we have a Julia code for constructing JuMP.Model-type object, it is easy to generate instances whenever we need to modify the original mathematical model. For each problem in SIPLIB 2.0, we provide a Julia script for constructing JuMP.Model-type object. We also provide a Julia script (SmpsWriter.jl) for converting any JuMP.Model-type objects to SMPS files for users' convenience. Those who feel the given instances are not large enough can simply generate more scenario data by just modifying the parameter in `Julia` script corresponding to the number of scenarios.

By `SIPLIB 2.0`, we provide 1) richer collection of test instances for computational and algorithmic research in SIP with benchmarking computational results, 2) not only `SMPS` files but also `Julia` files that are easily readable/tailorable. Hence, the users can obtain as large-sized instances as they need by generating new scenarios and including them into instances. For those who want to utilize the instances in the legacy `SIPLIB` with strong tailorability provided by `SIPLIB 2.0`, we include the original `SIPLIB` instances as well.

2 Stochastic integer programming

In this section, we explain general description of SIP. This includes formal mathematical formulation, existing general solution methods to solve the SIPs, and currently available software libraries.

2.1 Formulation

In this subsection, we introduce the form of SIP of interest. The notations and dimensional information are summarized in Table 1. We are interested in finding solution for two-stage SIP of the form:

$$z := \min_{x \in X} \left\{ c^\top x + \mathcal{Q}(x) : \; Ax \geq b \right\}, \tag{1}$$

where $\mathcal{Q}(x) := \mathbb{E}_{\boldsymbol{\xi}} \left[\phi \left(h(\boldsymbol{\xi}) - T(\boldsymbol{\xi})x \right) \right]$ is the recourse function associated with the random variable (r.v.) $\boldsymbol{\xi}$. We assume that $\boldsymbol{\xi}$ follows a known discrete probability distribution with the finite realizations, called *scenarios*, $\xi_1, \cdots, \xi_r$ and respective nonnegative probabilities $\mathbb{P}(1), \cdots, \mathbb{P}(r)$, i.e., $\mathbb{P}(s) \equiv \mathbb{P}[\boldsymbol{\xi} = \xi_s]$ for $s \in \mathcal{S} := \{1, \ldots, r\}$. When the distribution is continuous, we can approximate it by a suitably discretized distribution. The real-valued map $\phi_{\xi_s} : \mathbb{R}^{m_2} \to \mathbb{R}$ is the optimal value of the second-stage problem defined by

$$\phi_{\xi_s}(t) := \min_{y_s \in Y} \left\{ q(\xi_s)^\top y_s : \; W(\xi_s) y_s \geq t \right\}, \; t \in \mathbb{R}^{m_2}, \tag{2}$$

where ξ_s is an arbitrarily realized scenario. The sets $X \subseteq \mathbb{R}^{n_1}$ and $Y \subseteq \mathbb{R}^{n_2}$ represent integer or binary restrictions on a subset of the decision variables x and y_s, respectively. The first-stage problem data comprise A, b, and c. The second-stage data are given by $T(\xi_s)$, $W(\xi_s)$, $h(\xi_s)$, and $q(\xi_s)$ (for dimensional information refer to Table 1). Hereinafter, we use the simplified notations

(T_s, W_s, h_s, q_s). The SIP (1) can be rewritten in the extensive form

$$z = \min_{x, y_s} c^\top x + \sum_{s=1}^{r} \mathbb{P}(s)(q_j^\top y_s), \tag{3a}$$

$$\text{s.t. } Ax \geq b, \tag{3b}$$

$$T_s x + W_s y_s \geq h_s, \quad \forall s \in \{1, \ldots, r\}, \tag{3c}$$

$$x \in X, \tag{3d}$$

$$y_s \in Y, \quad \forall s \in \{1, \ldots, r\}. \tag{3e}$$

Table 1 Summary of notations in SIP formulation

Sets:	
$X \subseteq \mathbb{R}^{n_1}$	first-stage polyhedral set (continuous, integer, binary)
$Y \subseteq \mathbb{R}^{n_2}$	second-stage polyhedral set (continuous, integer, binary)
$\mathcal{S} = \{1, \ldots, r\}$	index set of realizable scenarios
Scalas:	
$\boldsymbol{\xi}$	r.v. denoting scenario that realizes by one of the set $\{\xi_1, \cdots, \xi_r\}$
$z \in \mathbb{R}$	optimal objective value of the SIP
$r \in \mathbb{N}$	number of scenarios
$s \in \mathcal{S}$	index denoting scenario
$\mathbb{P}(s) \in [0, 1]$	probability that scenario s happens, i.e., $\mathbb{P}(s) \equiv \mathbb{P}[\boldsymbol{\xi} = \xi_s]$
Vectors:	
$x \in \mathbb{R}^{n_1}$	first-stage decision vector
$c \in \mathbb{R}^{n_1}$	first-stage cost vector
$b \in \mathbb{R}^{m_1}$	first-stage RHS vector
$y_s \in \mathbb{R}^{n_2}$	second-stage decision vector under scenario ξ_s
$q_s \equiv q(\xi_s) \in \mathbb{R}^{n_2}$	second-stage cost vector
$h_s \equiv h(\xi_s) \in \mathbb{R}^{m_2}$	second-stage RHS vector
Matrices:	
$A \in \mathbb{R}^{m_1 \times n_1}$	first-stage constraint matrix corresponds to decision vector x
$W_s \equiv W(\xi_s) \in \mathbb{R}^{m_2 \times n_2}$	second-stage constraint matrix corresponds to decision vector y_s
$T_s \equiv T(\xi_s) \in \mathbb{R}^{m_2 \times n_1}$	second-stage constraint matrix corresponds to decision vector x
Functions:	
$\phi_{\xi_s} : \mathbb{R}^{m_2} \to \mathbb{R}$	second stage program optimal value under the realization of scenario ξ_s
$\mathcal{Q} : \mathbb{R}^{n_1} \to \mathbb{R}$	recourse function (the expectation of $\phi\,(h(\boldsymbol{\xi}) - T(\boldsymbol{\xi})x)$ over the r.v. $\boldsymbol{\xi}$)

2.2 Solution methods

2.2.1 Stage-wise decomposition algorithm

2.2.2 Scenario-wise decompostion algorithm

2.3 Software libraries

2.3.1 Modeling languages

2.3.2 Solvers

3 Summary of the test sets

3.1 Type of problems

Table 2 The type of problems in `SIPLIB 2.0`

Problem name	Description	Main reference
`DCAP`	Dynamic capacity planning with stochastic demand	Ahmed and Garcia [5]
`DCLP`	Data center location problem	Kim et al. [20]
`MPTSPs`	Multi-path traveling salesman problem with stochastic travel costs	Tadei et al. [14]
`SIZES`	Optimal product substitution with stochastic demand	Jorjani et al. [16]
`SMKP`	Stochastic multiple knapsack problem	Angulo et al. [18]
`SSLP`	Stochastic server location problem	Ntaimo and Sen [21]
`SUCW`	Wind power stochastic unit commitment	Papavasiliou and Oren [19]

Table 3 The components of problems in `SIPLIB 2.0`

	1st stage		2nd stage	
Problem	Variable	Constraint	Variable	Constraint
`DCAP`	$\mathbb{C}$, $\mathbb{B}$		$\mathbb{B}$	`PAR`, `MO11`
`DCLP`				
`MPTSPs`	$\mathbb{C}$, $\mathbb{B}$	`PAR2`, `GEN1`	$\mathbb{B}$	`GEN2`
`SIZES`	$\mathbb{I}$	`VBD1`, `GEN2`	$\mathbb{B}$, $\mathbb{I}$	`IKN1`
`SMKP`	$\mathbb{B}$	`KNA1`	$\mathbb{B}$	`KNA1`
`SSLP`	$\mathbb{B}$	`IVK1`, `GEN1`	$\mathbb{C}$, $\mathbb{I}$	`GEN2`
`SUCW`	$\emptyset$		$\mathbb{C}$, $\mathbb{B}$	

3.2 Instance catalog

3.2.1 Instance naming rule

Table 4 Problem-specific instance naming rules

Problem	Instance name	Description
DCAP		
DCLP		
MPTSPs	MPTSPs_Dx_Ny_Sz	node distribution strategy Dx, number of nodes y, and number of scenarios z
SIZES	SIZESz	number of scenario is z
SMKP		
SSLP		
SUCW		

4 How to run a test, generate new instance, and convert to SMPS

We explain the structure of SIPLIB 2.0. We explain procedure to generate new instances with user-generated scenario data using Julia scripts. The problem-specific descriptions are given in Section 6. We explain how to convert JuMP.Model-type object to SMPS files.

5 Implementation of SMPS Writer

We describe our Julia implementation, how to model SIP and generate SMPS files..

6 Problem descriptions

In this section, we introduce each problem in SIPLIB 2.0. For each problem, we provide various size of instances as a default. We also explain the scenario data generation procedures. Due to limited access to the original data and inconsistencies present in reference papers, we selectively choose the methods from the references and modify some of them without harming validity. Also, we guess some parameters about scenario generation to make the procedure clear.

6.1 DCAP: Dynamic capacity planning with stochastic demand

DCAP is the problem of determining a capacity expansion schedule for a set of resources, and the assignment of resource capacity to task with stochastic requirement over a multi-period planning horizon. In SIPLIB, 12 instances are available in SMPS format with the largest instance comprises of 500 scenarios correspond to the size of 9012 rows and 18018 columns. We refer to [5] for writing Julia scripts for DCAP described by the following subsections.

6.1.1 Mathematical formulation

$$
\begin{aligned}
(\texttt{DCAP}) \ \min \ & \sum_{t \in T} \sum_{i \in M} (\alpha_{it} x_{it} + \beta_{it} u_{it}) + \sum_{s \in \mathcal{S}} \mathbb{P}(s) \sum_{t \in T} \sum_{i \in M \cup \{0\}} \sum_{j \in N} c^s_{ijt} y^s_{ij} && \text{(4a)} \\
\text{s.t. } & x_{it} \le u_{it}, \quad \forall i \in M, \ \forall t \in T, && \text{(4b)} \\
& \sum_{j \in N} d^s_{jt} y^s_{ijt} \le \sum_{\tau=1}^{t} x_{i\tau}, \quad \forall i \in M, \ \forall t \in T, \ \forall s \in \mathcal{S}, && \text{(4c)} \\
& \sum_{i \in M \cup \{0\}} y^s_{ijt} = 1, \quad \forall j \in N, \ \forall t \in T, \ \forall s \in \mathcal{S}, && \text{(4d)} \\
& x_{it} \ge 0, \quad \forall i \in M \cup \{0\}, \ \forall t \in T, && \text{(4e)} \\
& u_{it} \in \{0,1\}, \quad \forall i \in M \cup \{0\}, \ \forall t \in T, && \text{(4f)} \\
& y^s_{ijt} \in \{0,1\}, \quad \forall i \in M \cup \{0\}, \ \forall j \in N, \ \forall t \in T, \ \forall s \in \mathcal{S}, && \text{(4g)}
\end{aligned}
$$

Table 5 Notations for `DCAP`

Index sets:	
M	index set of resources ($i \in M \cup \{0\}$ where 0 is a dummy resource with infinite capacity)
N	index set of tasks ($j \in N$)
T	index set of time periods ($t \in T$)
$\mathcal{S}$	index set of scenarios ($s \in \mathcal{S}$)
Parameters:	
α_{it}	variable cost for expanding capacity of resource i
β_{it}	fixed cost for expanding capacity of resource i
c^s_{ijt}	cost of processing task j using resource i in period t under scenario s
d^s_{jt}	processing requirement for task j in period t under scenario s
$\mathbb{P}(s)$	the probability of occurence of scenario s
Decision variables:	
x_{it} (1st stage)	capacity acquisition amount of resource i in period t
u_{it} (2nd stage)	1 if capacity of resource i is expanded in period t, 0 otherwise
y^s_{ijt} (2nd stage)	1 if resource i is assigned to task j in period t under scenario s, 0 otherwise

6.1.2 Scenario data generation

6.2 `DCLP`: Data center location problem

6.2.1 Mathematical formulation

6.2.2 Scenario data generation

6.3 `SMKP`: Stochastic multiple knapsack problem

`SMKP` is a class of stochastic multiple binary knapsack problems. Unlike typical knapsack problems where the objective is to maximize total profits under the restriction of the weight capacity of each knapsack, `SMKP` is to minimize total weights while satisfying a certain required profit for each knapsack.

`SIPLIB` provides 30 instances of `SMKP` in total. The first-stage problems contain 240 binary variables and 50 knapsack constraints. The second-stage problems have 120 binary variables and 5 knapsack constraints. Each instance has 20 scenarios. We refer [18] for writing `Julia` script for `SIPLIB 2.0` and explain the model throughout the following subsections.

6.3.1 Mathematical formulation

We have three types of items x, z, and y where the first two types are of the first-stage and the last one is of the second-stage with stochastic scenarios. For each type, we have $|I|$ number of items where I is the index set of the items. Hence, we define the binary variables x_i, z_i, and y_i^s are equal to 1 if the i^{th} item is decided to be included (s denotes scenario so only appears in y-type variables). We consider two types of knapsacks: one associated with x-type and z-type items (say `xz`-knapsack) and the other one with x-type and y-type items (say `xy`-knapsack). `xz`-knapsacks are indexed by $j \in J$ and `xy`-knapsacks are indexed by $k \in K$. Each knapsack has its own minimum level of profit that should be satisfied by the items of the associated types, e.g., the profit of the j^{th} `xz`-type knapsack is calculated based on the inclusion or exclusion of x-type and z-type items and should satisfy a certain requirement b_j. Bear in mind that the inclusion or exclusion of a certain item i affects all the associated knapsacks.

Each parameter c_i, d_i, and q_i^s denotes the gain of weight when including items of type x, z, and y, respectively. Here, c_i and d_i are deterministic and q_i^s is stochastic. Parameters a_{ij}, e_{ij}, t_{ik}, and w_{ik} are all deterministic and denote the profits for including items in the knapsacks. The RHS parameters b_j and h_k are the minimum levels of profit requirements for `xz`-knapsacks and `xy`-knapsacks, respectively.

The extensive form of `SMKP` is as follows and the notations used are summrized in Table 6.

$$
\begin{aligned}
(\texttt{SMKP}) \ \min \ & \sum_{i \in I} (c_i x_i + d_i z_i) + \sum_{s \in \mathcal{S}} \mathbb{P}(s) \sum_{i \in I} q_i^s y_i^s && \text{(5a)} \\
\text{s.t.} \ & \sum_{i \in I} a_{ij} x_i + \sum_{i \in I} e_{ij} z_i \geq b_j, \quad \forall j \in J, && \text{(5b)} \\
& \sum_{i \in I} t_{ik} x_i + \sum_{i \in I} w_{ik} y_i^s \geq h_k, \quad \forall k \in K, \ \forall s \in \mathcal{S}, && \text{(5c)} \\
& x_i \in \{0,1\}, \quad \forall i \in I, && \text{(5d)} \\
& z_i \in \{0,1\}, \quad \forall i \in I, && \text{(5e)} \\
& y_i^s \in \{0,1\}, \quad \forall i \in I, \ \forall s \in \mathcal{S}. && \text{(5f)}
\end{aligned}
$$

The objective (5a) is to minimize the expected value of the total weights. Constraint (5b) ensures the minimum levels of profit requirements for all `xz`-knapsacks are satisfied. Constraint (5c) guarantees the minimum levels of profit requirements are satisfied for all `xy`-knapsacks under every scenario. Constraints (5d)-(5f) are binary restriction of the decision variables.

Table 6 Notations for `SMKP`

Index sets:	
I	index set of items for each type ($i \in I$)
J	index set of `xz`-knapsacks ($j \in J$)
K	index set of `xy`-knapsacks ($k \in K$)
$\mathcal{S}$	index set of scenarios ($s \in \mathcal{S}$)
Parameters:	
c_i	weight of the i^{th} x-type item
d_i	weight of the i^{th} z-type item
q_i^s	weight of the i^{th} y-type item under scenario s
a_{ij}	profit of the j^{th} `xz`-knapsack for including i^{th} x-type item
e_{ij}	profit of the j^{th} `xz`-knapsack for including i^{th} z-type item
t_{ik}	profit of the k^{th} `xy`-knapsack for including i^{th} x-type item
w_{ik}	profit of the k^{th} `xy`-knapsack for including i^{th} y-type item
b_j	minimum required profit for the j^{th} `xz`-knapsack
h_k	minimum required profit for the k^{th} `xy`-knapsack
$\mathbb{P}(s)$	the probability of occurence of scenario s
Decision variables:	
x_j (1st stage)	1 if the j^{th} x-type item is decided to be included, 0 otherwise
z_j (1st stage)	1 if the j^{th} z-type item is decided to be included, 0 otherwise
y_j^s (2nd stage)	1 if the j^{th} y-type item is decided to be included under scenario s, 0 otherwise

6.3.2 Scenario data generation

6.4 `SSLP`: Stochastic server location problem

6.4.1 Mathematical formulation

6.4.2 Scenario data generation

6.5 `MPTSPs`: Mutli-path Traveling Salesman Problem with Stochastic Travel Times

`MPTSPs` is a variant of the travelling salesman problem (TSP) where a set of paths exists between any two nodes and each path is chracterized by a random travel time. In `SIPLIB`, only limited data (e.g., number of nodes, coordinates of nodes, generated travel times) are provided and no `SMPS` file is available. We mainly refer to [13] for deriving the mathematical formulation. Due to some unclear subtour breaking constraints, we refer to another paper [15]. Combining the two references, we construct the forthcoming single commodity flow-based formulation `MPTSPs` that is used for `SIPLIB 2.0`.

6.5.1 Mathematical formulation

We consider a two-stage SIP with recourse. The travel time oscillation e_{ij}^k by using path k between nodes i and j. We present each realization (scenario) of random travel time oscillation by e^s_{ijk} where s indicates the scenario. In `MPTSPs`, at the first stage, the decision-maker does not have any information about the travel time oscillation. The tour paths among the nodes, however, should be determined before the complete information is available. The first stage decision variable y_{ij} is represented by the selection of nodes i and j to be visited in a tour. In the second stage where the random travel time c^s_{ijk} are avilable, the paths k between each couple of nodes i and j under scenario s, x^s_{ijk} can be calculated.

Let N and K_{ij}, respectively, bethe finite set of nodes of the graph and the set of paths between the pair of nodes $i, j \in N$. We denote with $\mathcal{S}$ the set of scenarios with associated equally distributed probability of each scenario $\mathbb{P}(s)$, i.e., $\mathbb{P}(s) \equiv 1/|\mathcal{S}|$. Each path $k \in K_{ij}$ between nodes $i, j \in N$ is chracterized by a non-negative estimation of the mean unit travel time $\bar{c}_{ij}$ and a non-negative unit random travel time c^s_{ijk} under the scenario $s \in S$. Let $e^s_{ijk} \equiv c^s_{ijk} - \bar{c}_{ij}$ be the error on the travel time estimated for the path $k \in K_{ij}$ under time scenario $s \in S$.

The first stage binary variables $y_{ij} = 1$ if node $j \in N$ is visited right after node $i \in N$, 0 otherwise. The second stage binary variables $x^s_{ijk} = 1$ if path $k \in K_{ij}$ between nodes $i, j \in N$ is selected at the second stage, 0 otherwise. We have one more set of first stage variables ϕ_{ij} which is introduced to break the subtours [15]. The non-negative continuous variables ϕ_{ij} describe the flow of a single commodity to node 1 from every other nodes (without loss of generality, 1 is the starting node).

The extensive form of `MPTSPs` is as follows and the notations used are summrized in Table 7.

$$
\begin{aligned}
(\texttt{MPTSPs})\ \min\ & \sum_{i\in N}\sum_{j\in N} \bar{c}_{ij} y_{ij} + \sum_{s\in\mathcal{S}} \mathbb{P}(s) \sum_{i\in N}\sum_{j\in N}\sum_{k\in K_{ij}} e^s_{ijk} x^s_{ijk} && \text{(6a)}\\
\text{s.t.}\ & \sum_{j\in N: j\neq i} y_{ij} = 1, \quad \forall i \in N, && \text{(6b)}\\
& \sum_{i\in N: i\neq j} y_{ij} = 1, \quad \forall j \in N, && \text{(6c)}\\
& \sum_{j\in N} \phi_{lj} - \sum_{i\in N\setminus\{1\}} \phi_{il} = 1, \quad \forall l \in N\setminus\{1\}, && \text{(6d)}\\
& \phi_{ij} \leq (|N|-1)\, y_{ij}, \quad \forall i \in N\setminus\{1\},\ \forall j \in N, && \text{(6e)}\\
& \sum_{k\in K_{ij}} x^s_{ijk} = y_{ij}, \quad \forall i \in N,\ \forall j \in N,\ \forall s \in \mathcal{S}, && \text{(6f)}\\
& x^s_{ijk} \in \{0,1\}, \quad \forall i \in N,\ \forall j \in N,\ \forall k \in K_{ij},\ \forall s \in \mathcal{S}, && \text{(6g)}\\
& y_{ij} \in \{0,1\}, \quad \forall i \in N,\ \forall j \in N, && \text{(6h)}\\
& \phi_{ij} \geq 0, \quad \forall i \in N,\ \forall j \in N. && \text{(6i)}
\end{aligned}
$$

The first sum in the objective function (6a) represents the first stage travel cost, while the second sum represents the recourse action, consisting in choosing the best path $k \in K_{ij}$ under scenario $s \in \mathcal{S}$. The constraints (6b) and

(6c) form the assignment constraints and ensure that each node is visited only once. Given the fixed values of y_{ij}, constraint (6d) and (6e) form a network flow problem, and therefore the ϕ_{ij} values will be integer. In case the solutions of the above formulation contain at least one subtour, the constraints (6d) and (6e) are violated. Moreover, no tour can exist that does not contain node 1 by the two constraints. For more explanation on the subtour breaking mechanism accompanied with rigorous proof, refer to [17]. The constraint (6f) guarantees that path k between nodes i and j can be chosen at stage 2 only if nodes i and j were part of the tour fixed at stage 1. Finally, the constraints (6g)-(6i) restrict the polyhedral sets from which the variables take values.

Table 7 Notations for `MPTSPs`

Index sets:	
N	index set of nodes ($i, j, l \in N$)
K_{ij}	index set of paths between nodes i and j ($k \in K_{ij}$)
$\mathcal{S}$	index set of scenarios ($s \in \mathcal{S}$)
Parameters:	
c^s_{ijk}	unit random travel time of path k between nodes i, j under scenario s
$\bar{c}_{ij}$	estimation of the mean unit travel time (expectation of c^s_{ijk} over all s and k)
e^s_{ijk}	the error on the travel time on estimated for arc (i, j) and path k under scenario s
$\mathbb{P}(s)$	the probability of occurence of scenario s
Decision variables:	
ϕ_{ij} (1st stage)	the nonnegative real-valued flow on arc (i, j)
x^s_{ijk} (1st stage)	1 if node j is visited just after node i, 0 otherwise
y_{ij} (2nd stage)	1 if path k between nodes $i, j \in N$ is selected at the second stage, 0 otherwise

6.5.2 Scenario data generation

We follow the scenario generation methods described through the references [12, 13, 14]. For `MPTSPs`, there are three mainly distinguished characteristics for each instance: the nodes partition strategy ($D \in \{D0, D1, D2, D3\}$, explanation on each strategy is forthcoming), the number of nodes ($|N| \in \{2, 3, \ldots\}$), and the number of scenarios ($|S| \in \{1, 2, \ldots\}$). Another important charicteristic $|K_{ij}| \in \{1, 2, 3, \ldots\}$ is the number of paths for each edge which is fixed by 3 as a default following [14]. Once we decide D, $|N|$, and $|S|$ by $D = D$x, $|N| =$ y, and $|S| =$ z, each instance is named by `MPTSPs_Dx_Ny_Sz`.

The nodes are distributed in a circle with radius equal to r km. We use Cartesian coordinate system where the geometric center of the circle is (r, r). The nodes are distinguished by two subsets: *central* and *suburban*. If the Euclidean distance between a node and the geometric center is less than or equal to the half of the radius ($r/2$), then the node is of *central* type. Otherwise, if the Euclidean distance is greater than the half of the radius, the node is of *suburban* type. Each arc between any two nodes i and j is either *homogeneous* or *heterogeneous*. If the two nodes are of the same type of node, i.e., both are *central* or both are *suburban*, the type of the arc is *homogeneous*. Otherwise, the type of the arc is *heterogeneous*. Later, the travel time of each path between two nodes are affected by the type of arc.

The nodes are generated by one of the following distribution strategies:

- $D0$: All the nodes are *central*.
- $D1$: All the nodes are *suburban*.
- $D2$: 3/4 of the nodes are *central* and the remaining 1/4 are *suburban*.
- $D3$: 1/2 of the nodes are *central* and the remaining 1/2 are *suburban*.

Given $D, |N|$ and $|S|$, the next procedure can be summarized as follows:

1. Generate $|N|$ nodes based on the predetermined strategy D. Then, the nodes are generated by acceptance-rejection procedure with uniform random number generation. Again following [14], we fix $r = 7km$.
2. Calculate Euclidean distances between the nodes (EC_{ij}).
3. We guess and fix the deterministic velocity profile by 40*km/h* for the *central* nodes and 80*km/h* for the *suburban* nodes: $v_{cntr} = 40$ and $v_{sbrb} = 80$.
4. Generate random travel times (c^s_{ijk}) for each scenario s.
   - The velocity for traveling arc (i, j) is affected by its arc type.
   - If the arc is *homogeneous*, the random travel time of all the paths are generated only based on the corresponding velocity profile.
   - If the arc is *heterogeneous*, $\left\lceil \frac{|K_{ij}|}{3} \right\rceil$ paths are generated based on $v_{cntr} = 40$ and the remaining paths are generated based on $v_{sbrb} = 80$.
   - The velocities are distributed by $Unif(\frac{v}{2}, 2v)$ for $v = v_{cntr}, v_{sbrb}$.
   - In summary, if the arc (i, j) is *homogeneous*,

$$c^s_{ijk} \sim \begin{cases} \frac{EC_{ij}}{Unif(\frac{v_{cntr}}{2}, 2v_{cntr})} & \text{if } i, j \text{ are both } \textit{central}, \\ \frac{EC_{ij}}{Unif(\frac{v_{sbrb}}{2}, 2v_{sbrb})} & \text{if } i, j \text{ are both } \textit{suburban}, \end{cases} \quad \forall k \in K_{ij}.$$

   - Otherwise, if (i, j) is *heterogeneous*,

$$c^s_{ijk} \sim \begin{cases} \frac{EC_{ij}}{Unif(\frac{v_{cntr}}{2}, 2v_{cntr})} & \text{for } k \in \left\{1, \ldots, \left\lceil \frac{|K_{ij}|}{3} \right\rceil \right\}, \\ \frac{EC_{ij}}{Unif(\frac{v_{sbrb}}{2}, 2v_{sbrb})} & \text{for } k \in \left\{ \left\lceil \frac{|K_{ij}|}{3} \right\rceil + 1, \ldots, |K_{ij}| \right\}. \end{cases}$$

5. Finally, we multiply 3600 for each component of c^s_{ijk} to convert the unit from *hours* to *seconds*.

6.6 `SIZES`: Selection of an optimal subset of sizes

`SIZES` is a simplified version of the cutting stock problem with multi-period stochastic demand. In this problem, the term *period* can be exchangeably used with *stage*. The first period (first stage) demand is deterministic whereas demands for the other periods (stages) are stochastic with scenarios. We only consider the two-periods (i.e., two stage) model to follow [16] as well as the SIP of interest discussed in Section 2. In `SIPLIB`, only three instances are available in `SMPS` files. We refer to the mathematical formulation in [16] to construct `JuMP.Model`. Due to some unclear explanations (or typo), we slightly modify the formulation and use it for `SIPLIB 2.0`.

6.6.1 Mathematical formulation

Table 8 Notations for `SIZES`

Index sets	
N	index set of items ($i, j \in N$)
T	index set of time periods ($t \in T$)
$\mathcal{L}$	index set of scenarios ($l \in \mathcal{L}$)
Parameters	
d_{it}^l	demand for item i at time t under scenario l
p_i	unit production cost for item i
s	setup cost for producing any item
r	unit cutting cost
c_t^l	production capacity at time t under scenario l
$\mathbb{P}(l)$	the probability of occurence of scenario l
Decision variables	
y_{it} (1st stage)	number of units of size i produced at time t
x_{ijt}^l (2nd stage)	number of units of size i cut to meet demand for smaller size j at time t under scenario l
z_{it}^l (2nd stage)	1 if we produce size i at time t under scenario l, 0 otherwise

$$
\begin{aligned}
(\texttt{SIZES}) \min \quad & \sum_{t\in T}\sum_{i\in N} p_i y_{it} + \sum_{l\in\mathcal{L}} \mathbb{P}(l) \sum_{t\in T} \left(\sum_{i\in N} s z_{it}^l + r \sum_{i\in N\setminus\{1\}} \sum_{j=1}^{i-1} x_{ijt}^l \right) && \text{(7a)}\\
\text{s.t.} \quad & \sum_{i\in N} y_{it} \le c_t^l, \quad \forall t \in T,\ \forall l \in \mathcal{L}, && \text{(7b)}\\
& \sum_{i=j}^{|N|} x_{ijt}^l \ge d_{jt}^l, \quad \forall j \in N,\ \forall t \in T,\ \forall l \in \mathcal{L}, && \text{(7c)}\\
& \sum_{t'=1}^{t}\sum_{j=1}^{i} x_{ijt'}^l \le \sum_{t'=1}^{t} y_{it'}, \quad \forall i \in N,\ \forall t \in T,\ \forall l \in \mathcal{L}, && \text{(7d)}\\
& y_{it} \le c_t^l z_{it}^l, \quad \forall i \in N,\ \forall t \in T,\ \forall l \in \mathcal{L}, && \text{(7e)}\\
& y_{it} \in \mathbb{Z}_+, \quad \forall j \in N,\ \forall t \in T, && \text{(7f)}\\
& x_{ijt}^l \in \mathbb{Z}_+, \quad \forall i \in N,\ \forall j \in N,\ \forall t \in T,\ \forall l \in \mathcal{L}, && \text{(7g)}\\
& z_{it}^l \in \{0,1\}, \quad \forall i \in N,\ \forall t \in T,\ \forall l \in \mathcal{L}. && \text{(7h)}
\end{aligned}
$$

6.6.2 Scenario data generation

6.7 `SUCW`: Stochastic unit commitment problem with wind power

6.7.1 Mathematical formulation

$$\begin{aligned}
\min\ & \sum_{\sigma\in\mathcal{S}} \mathbb{P}(\sigma) \sum_{t\in T}\sum_{g\in G}\left(C_g^{\text{fx}} x_{gt}^\sigma + C_g^{\text{up}} u_{gt}^\sigma + C_g^{\text{dn}} v_{gt}^\sigma + \sum_{k\in K} C_{gk}^{\text{mar}} q_{gkt}^\sigma \right) && \text{(8a)}\\
\text{s.t. } & 1 - x_{g(t-1)}^\sigma \geq u_{gt}^\sigma, \quad \forall \sigma\in\mathcal{S},\ g\in G,\ t\in T, && \text{(8b)}\\
& x_{g(t-1)}^\sigma \geq v_{gt}^\sigma, \quad \forall \sigma\in\mathcal{S},\ g\in G,\ t\in T, && \text{(8c)}\\
& x_{gt}^\sigma - x_{g(t-1)}^\sigma = u_{gt}^\sigma - v_{gt}^\sigma, \quad \forall \sigma\in\mathcal{S},\ g\in G,\ t\in T, && \text{(8d)}\\
& x_{gt}^\sigma \geq \sum_{\tau=\max\{1,t-UT_g+1\}}^{t} u_{g\tau}^\sigma, \quad \forall \sigma\in\mathcal{S},\ g\in G,\ t\in T, && \text{(8e)}\\
& 1 - x_{gt}^\sigma \geq \sum_{\tau=\max\{1,t-DT_g+1\}}^{t} u_{g\tau}^\sigma, \quad \forall \sigma\in\mathcal{S},\ g\in G,\ t\in T, && \text{(8f)}\\
& -RD_g \leq p_{gt}^\sigma - p_{g(t-1)}^\sigma \leq RU_g - s_{gt}^\sigma, \quad \forall \sigma\in\mathcal{S},\ g\in G,\ t\in T, && \text{(8g)}\\
& s_{gt}^\sigma \leq RC_g x_{gt}^\sigma, \quad \forall \sigma\in\mathcal{S},\ g\in G,\ t\in T, && \text{(8h)}\\
& \sum_{g\in G} s_{gt}^\sigma \geq SR_t, \quad \forall \sigma\in\mathcal{S},\ t\in T, && \text{(8i)}\\
& p_{gt}^\sigma = P_g^{\min} x_{gt}^\sigma + \sum_{k\in K} q_{gkt}^\sigma, \quad \forall \sigma\in\mathcal{S},\ g\in G, t\in T, && \text{(8j)}\\
& p_{gt}^\sigma + s_{gt}^\sigma \leq P_g^{\max} x_{gt}^\sigma, \quad \forall \sigma\in\mathcal{S},\ g\in G,\ t\in T, && \text{(8k)}\\
& q_{gkt}^\sigma \leq Q_{gk}^{\max} x_{gt}^\sigma, \quad \forall \sigma\in\mathcal{S},\ g\in G,\ k\in K,\ t\in T, && \text{(8l)}\\
& \sum_{g\in G} p_{gt}^\sigma = \sum_{n\in N} D_{nt}^\sigma - \sum_{w\in W} W_{wt}^\sigma, \quad \forall \sigma\in\mathcal{S},\ t\in T, && \text{(8m)}\\
& -F_l^{\max} \leq \sum_{g\in G} LSF_{lg} p_{gt}^\sigma - \sum_{n\in N} LSF_{ln} D_{nt}^\sigma \\
& \qquad + \sum_{w\in W} LSF_{lw} W_{wt}^\sigma \leq F_l^{\max}, \quad \forall \sigma\in\mathcal{S},\ l\in L,\ t\in T, && \text{(8n)}\\
& x_{gt}^{\sigma_1} = x_{gt}^{\sigma_2},\ u_{gt}^{\sigma_1} = u_{gt}^{\sigma_2},\ v_{gt}^{\sigma_1} = v_{gt}^{\sigma_2}, \quad \forall \sigma_1,\sigma_2\in\mathcal{S},\ g\in G_s,\ t\in T, && \text{(8o)}\\
& x_{g0}^\sigma = X_g^{\text{init}}, \quad \forall \sigma\in\mathcal{S},\ g\in G, && \text{(8p)}\\
& x_{gt}^\sigma = 1, \quad \forall \sigma\in\mathcal{S},\ g\in G,\ t\in\{1,\ldots,UT_g^{\text{init}}\}, && \text{(8q)}\\
& x_{gt}^\sigma = 0, \quad \forall \sigma\in\mathcal{S},\ g\in G,\ t\in\{1,\ldots,DT_g^{\text{init}}\}, && \text{(8r)}\\
& p_{g0}^\sigma = P_g^{\text{init}}, \quad \forall \sigma\in\mathcal{S},\ g\in G, && \text{(8s)}\\
& x_{gt}^\sigma, u_{gt}^\sigma, v_{gt}^\sigma \in \{0,1\}, \quad \forall \sigma\in\mathcal{S},\ g\in G,\ t\in T, && \text{(8t)}\\
& p_{gt}^\sigma, q_{gkt}^\sigma, s_{gt}^\sigma \geq 0, \quad \forall \sigma\in\mathcal{S},\ g\in G,\ t\in T. && \text{(8u)}
\end{aligned}$$

Table 9 Notations for the SUCW

Index sets:	
G	index set of all generators ($g \in G$)
G_s	index set of slow generators ($g \in G_s$)
G_f	index set of fast generators ($g \in G_f$)
K	index set of linear segments of the piece-wise linear power generation cost ($k \in K$)
L	index set of transmission lines ($l \in L$)
N	index set of buses ($n \in N$)
T	index set of time periods ($t \in T$)
W	index set of wind power generators ($w \in W$)
$\mathcal{S}$	index set of scenarios ($\sigma \in \mathcal{S}$)
Parameters:	
C_g^{up}	start-up cost of generator g
C_g^{dn}	shut-down cost of generator g
C_g^{fx}	fixed cost of operating the generator g
C_{gk}^{mar}	k^{th} marginal cost of production of generator g
X_g^{init}	initial on/off status of generator g
UT_g^{init}	initial minimum uptime of generator g
UT_g	minimum uptime of generator g
DT_g^{init}	initial minimum downtime of generator g
DT_g	minimum downtime of generator g
RU_g	ramp-up limit of generator g
RD_g	ramp-down limit of generator g
RC_g	ramping capacity of generator g
P_g^{init}	initial power output of generator g
P_g^{min}	minimum power output of generator g
P_g^{max}	maximum power output of generator g
Q_{gk}^{max}	maximum power output of generator g with the k^{th} marginal cost
SR_t	spinning reserve required at time t
F_l^{max}	maximum power flow of transmission line l
LSF_{ln}	load-shift factor of transmission line l with respect to bus n
$\mathbb{P}(\sigma)$	probability of scenario σ
D_{nt}^{σ}	demand load at bus n at time t in scenario σ
W_{wt}^{σ}	wind power generation from generator w at time t in scenario σ
Decision variables:	
x_{gt}^{σ}	on/off indicator of generator g at time t in scenario σ
u_{gt}^{σ}	start-up indicator of generator g at time t in scenario σ
v_{gt}^{σ}	shut-down indicator of generator g at time t in scenario σ
p_{gt}^{σ}	power output of generator g at time t in scenario σ
q_{gkt}^{σ}	power output of generator g at time t with the k^{th} marginal cost in scenario σ
s_{gt}^{σ}	spinning reserve of generator g at time t in scenario σ

6.7.2 Scenario data generation

7 Solution report

8 Concluding remarks

References

1. Author1 and Author2, paper paper paper paper, Journal Title 68 (2011), 1207–1221.

2. S. Ahmed, R. Garcia, N. Kong, L. Ntaimo, G. Parija, F. Qiu, S. Sen. SIPLIB: A Stochastic Integer Programming Test Problem Library. http://www.isye.gatech.edu/ sahmed/siplib, 2015.
3. SPS: Stochastic Programming Society (https://stoprog.org/what-stochastic-programming).
4. Introduction to Stochastic Programming, J. R. Birge, F. Louveaus.
5. S. Ahmed and R. Garcia. "Dynamic Capacity Acquisition and Assignment under Uncertainty," Annals of Operations Research, vol.124, pp. 267-283, 2003.
6. Kibaek Kim and Victor M. Zavala. "Algorithmic Innovations and Software for the Dual Decomposition Method applied to Stochastic Mixed-Integer Programs" Mathematical Programming Computation, 2015.
7. J.-P. Watson, D. L. Woodruff, and W. E. Hart, PySP: modeling and solving stochastic programs in Python, Mathematical Programming Computation, 2012.
8. SMI - Stochastic Modeling Interface. https://github.com/coin-or/Smi
9. Julia: A Fresh Approach to Numerical Computing. Jeff Bezanson, Alan Edelman, Stefan Karpinski and Viral B. Shah (2017) SIAM Review, 59: 6598.
10. JuMP - Julia for Mathematical Optimization, https://jump.readthedocs.io/en/latest/index.html
11. StructJuMP - Parallel algebraic modeling framework for block structured optimization models in Julia, https://github.com/StructJuMP/StructJuMP.jl
12. F. Maggioni, G. Perboli, and R. Tadei, The multi-path traveling salesman problem with stochastic travel socts: Building realistic instances for city ligistics applications, Transportation Research Procedia, 2014
13. G. Perboli, L. Gobbato, and F. Maggioni, A progressive hedging method for the multi-path travelling salesman problem with stochastic travel times, IMA Journal of Management Mathematics, 2017
14. R. Tadei, G. Perboli, and F. Perfetti, The multi-path traveling salesman problem with stochastic travel costs, EURO Journal on Transportation and Logistics, 2017
15. Andre Langevin, Francois Soumis, and Jacques Desrosiers, Classification of travelling salesman problem formulations, Operations Research Letters, 1990
16. Soheila Jorjani, Carlton H. Scott, and David L. Woodruff, Selection of an optimal subset of sizes, International Journal of Production Research, 1999
17. B. Gavish and S.C. Graves, The travelling salesman problem and related problems, Working paper GR-078-78, Operations Research Center, Massachusetts Institute of Technology, 1978
18. Gustavo Angulo, Shabbir Ahmed, and Santanu S. Dey, Improving the integer L-shaped method, 2014.
19. Anthony Papavasiliou and Shmuel S. Oren, Multiarea stochastic unit commitment for high wind penetration in a transmission constrained network, Operations Research, 2013
20. Kibaek Kim, Fan Yang, Victor M. Zavala, and Andrew A. Chien, Data centers as dispatchable loads to harness stranded power, IEEE Transactions on sustainable energy, 2017
21. Lewis Ntaimo and Suvrajeet Sen, The million-variable "March" for stochastic combinatorial optimization, Journal of Global Optimization, 2005