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

The `SIPLIB` [1] 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. State-of-the-art in SIP combined with the speedup in computing machinery, however, makes many instances in `SIPLIB` so easy that we have not enough basis to use them 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

newly suggested 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 easily generate instances of test problems as largely as they want.

(as we include DCLP (data center location problem), this paragraph must be modified) 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., [2,3]. 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 [4]. 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 [5], PySP [6], and SMI [7] 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 [8]. 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 [9]) 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 [10]. 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.

(Contents of the paper, not finished yet) In this paper, we provide a detailed description of the test problems in `SIPLIB 2.0`, including information on the instances, their origin, formal mathematical models with explanation on notations, and scenario data generation procedure. `SIPLIB 2.0` classifies problems based on the stages, variables, and constraints. We borrow some classification criteria from the most frequently utilized mixed integer programming (MIP) library, `MIPLIB 2010` [21], whenever they are applicable to SIP 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

In this section, we explain information about the test problems and corresponding instances in summarized manner. This includes type, components, and instances for each problem. More detailed problem-specific explanation is available in Section 6.

3.1 Type of problems

Table 2 Types of the problems in `SIPLIB 2.0`

Problem name	Description	Main reference
`DCAP`	Dynamic capacity planning with stochastic demand	Ahmed and Garcia [4]
`DCLP`	Data center location problem	Kim et al. [19]
`MPTSPs`	Multi-path traveling salesman problem with stochastic travel costs	Tadei et al. [13]
`SIZES`	Optimal product substitution with stochastic demand	Jorjani et al. [15]
`SMKP`	Stochastic multiple knapsack problem	Angulo et al. [17]
`SSLP`	Stochastic server location problem	Ntaimo and Sen [20]
`SUCW`	Wind power stochastic unit commitment	Papavasiliou and Oren [18]

Table 3 Constraint type legend [21]

Type	Description	Constraint form
AGG	Aggregation	$a_i x_i + a_k x_k = b,\ x_i, x_k$ int. or cont., $a_i, a_k, b \in \mathbb{R}$
VBD	Variable bound	$x_i \leq a_k x_k + b$ or $x_i \geq a_k x_k + b$, x_i, x_k int. or cont., $a_k, b \in \mathbb{R}$
PAR	Set partition	$\sum x_i = 1$, x_i binary
PAC	Set packing	$\sum x_i \leq 1$, x_i binary
COV	Set cover	$\sum x_i \geq 1$, x_i binary
CAR	Cardinality	$\sum x_i = b$, x_i binary, $b \in \mathbb{N}$
EQK	Equality knapsack	$\sum a_i x_i = b$, x_i binary, $a_i, b \in \mathbb{N}$
BIN	Bin packing	$\sum a_i x_i + a_k x_k \leq a_k$, x_i binary, $a_i, a_k \in \mathbb{N}$
IVK	Invariant knapsack	$\sum x_i \leq b$, x_i binary, $b \in \mathbb{N}$
KNA	Knapsack	$a_i x_i \leq b$, x_i binary, $a_i, b \in \mathbb{N}$
IKN	Integer knapsack	$a_i x_i \leq b$, $x_i \geq 0$ integer, $a_i, b \in \mathbb{N}$
MO1	Mixed binary	$\sum a_i x_i + \sum p_j s_j \leq$ or $= b$, x_i binary, s_j cont., $a_i, p_j \in \mathbb{R}$
GEN	General	All other constraint types

Table 4 Components of the problems in SIPLIB 2.0 ($\mathbb{C}, \mathbb{B}, \mathbb{I}$: continuous, binary, integer variables)

	1st stage		2nd stage	
Problem	Variable	Constraint	Variable	Constraint
DCAP	$\mathbb{C}$, $\mathbb{B}$	VBB	$\mathbb{B}$	PAR, MO1
DCLP			$\mathbb{C}$	
MPTSPs	$\mathbb{C}$, $\mathbb{B}$	PAR, GEN	$\mathbb{B}$	GEN
SIZES	$\mathbb{I}$	VBD, GEN	$\mathbb{B}$, $\mathbb{I}$	IKN
SMKP	$\mathbb{B}$	KNA	$\mathbb{B}$	KNA
SSLP	$\mathbb{B}$	IVK, GEN	$\mathbb{C}$, $\mathbb{I}$	GEN
SUCW	$\emptyset$		$\mathbb{C}$, $\mathbb{B}$	

3.2 Instance catalog

Table 5 Problem-specific instance naming rules

Problem	Instance name	Description
DCAP	DCAP_x_y_z_w	number of resources x, number of tasks y, number of time periods z, number of scenarios w
DCLP		
MPTSPs	MPTSPs_Dx_Ny_Sz	node distribution strategy Dx, number of nodes y, and number of scenarios z
SIZES	SIZES_z	number of scenario is z
SMKP	SMKP_x_y	number of types for item x, number of scenarios y
SSLP	SSLP_x_y_z	number of clients x, number of servers y, number of scenarios z
SUCW	SUCW_x_y	day type x, number of scenarios y

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 details for each problem in `SIPLIB 2.0`. We also explain the scenario data generation procedures. Due to limited access to the original data in reference papers, we selectively choose the methods from several available 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 9,012 rows and 18,018 columns. We refer to [4] for writing `Julia` scripts.

6.1.1 `DCAP`: Mathematical formulation

We consider the problem of deciding the capacity expansion schedule for $|R|$ resources over $|T|$ time periods to satisfy the processing requirements of $|N|$ tasks where R, T, and N denote set of resources, set of time periods, and set of tasks, respectively. We define decision variables: the first-stage continuous variable x_{it} for the capacity acquisition of resource i in period t and the second-stage binary variable y^s_{ijt} to indicate whether resource i is assigned to task j in period t under scenario s. Additional first-stage binary variable u_{it} is for logical constraint whether or not we decided to acquire more capacity of resource i in period t. Hence, for all resource $i \in M$ and time $t \in T$, $u_{it} = 1$ if $x_{it} > 0$, $u_{it} = 0$ otherwise.

Under the definition of the decision variables, the extensive form of DCAP is written below and the summarized notation is available in Table 6.

$$\text{(DCAP)}\ \min \sum_{t\in T}\sum_{i\in R}(\alpha_{it}x_{it}+\beta_{it}u_{it})+\sum_{s\in\mathcal{S}}\mathbb{P}(s)\sum_{t\in T}\sum_{i\in M\cup\{0\}}\sum_{j\in N}c^s_{ijt}y^s_{ij} \quad (4a)$$

$$\text{s.t. } x_{it}\le Mu_{it}, \quad \forall i\in R,\ \forall t\in T, \quad (4b)$$

$$\sum_{j\in N}d^s_{jt}y^s_{ijt}\le\sum_{\tau=1}^{t}x_{i\tau}, \quad \forall i\in R,\ \forall t\in T,\ \forall s\in\mathcal{S}, \quad (4c)$$

$$\sum_{i\in R\cup\{0\}}y^s_{ijt}=1, \quad \forall j\in N,\ \forall t\in T,\ \forall s\in\mathcal{S}, \quad (4d)$$

$$x_{it}\ge 0, \quad \forall i\in R\cup\{0\},\ \forall t\in T, \quad (4e)$$

$$u_{it}\in\{0,1\}, \quad \forall i\in R\cup\{0\},\ \forall t\in T, \quad (4f)$$

$$y^s_{ijt}\in\{0,1\}, \quad \forall i\in R\cup\{0\},\ \forall j\in N,\ \forall t\in T,\ \forall s\in\mathcal{S}, \quad (4g)$$

The objective function (4a) is to minimize total expected cost for the capacity expansion schedule. The first double summation denotes the expansion cost for resource i in period t where α_{it} and β_{it} are the variable and fixed cost, respectively. The second term in the objective function represents the expected assignment cost in period t over all scenario $s\in\mathcal{S}$. Note that a dummy resource $i=0$ is included with infinite capacity. The cost c^s_{0jt} denotes the penalty of failing to assign a resource to task j. The dummy resource enforces the *complete recourse property*, which ensures that there is a feasible second-stage assignment in all periods and all scenarios for any capacity acquisition schedule [4]. Constraint (4b) is the logical constraint containing a suitably large value M to define the cost for capacity expansion. Constraint (4c) reflects that the processing requirement of all tasks assigned to a resource in any period cannot exceed the installed capacity in that period under all scenarios. Constraint (4d) guarantees that each task needs to be assigned to exactly one resource in each period under all scenarios. Finally, constraints (4e)-(4g) restrict the space from which the variables take values.

Table 6 Notations for `DCAP`

Index sets:	
R	index set of resources ($i \in R \cup \{0\}$ 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} (1^{st} stage)	capacity acquisition amount of resource i in period t
u_{it} (1^{st} stage)	1 if capacity of resource i is expanded in period t, 0 otherwise
y^s_{ijt} (2^{nd} stage)	1 if resource i is assigned to task j in period t under scenario s, 0 otherwise

6.1.2 `DCAP`: Data generation

There are four factors that define the instance of `DCAP`: $|R|$, $|N|$, $|T|$, and $|\mathcal{S}|$. Once we decide the factors by, $n_R = |R|$, $n_N = |N|$, $n_T = |T|$, and $n_{\mathcal{S}} = |\mathcal{S}|$, each instance is named by `DCAP`$\_n_R\_n_N\_n_T\_n_{\mathcal{S}}$. Since no original parameter is available, we randomly generate the parameters as long as they are valid. Let U be a continuous uniform random variable: $U \sim Unif(0, 1)$. Then, the parameters are generated as follows:

$$
\begin{aligned}
\alpha_{it} &= 5U + 5, \quad \forall i \in R,\ \forall t \in T,\\
\beta_{it} &= 40U + 10, \quad \forall i \in R,\ \forall t \in T,\\
c^s_{ijt} &= 5U + 5, \quad \forall i \in R,\ \forall j \in N,\ \forall t \in T,\ \forall s \in \mathcal{S},\\
c^s_{0jt} &= 500U + 500, \quad \forall j \in N,\ \forall t \in T,\ \forall s \in \mathcal{S},\\
d^s_{jt} &= U + 0.5, \quad \forall j \in N,\ \forall t \in T,\ \forall s \in \mathcal{S}.
\end{aligned}
$$

For the first-stage parameters α_{it} and β_{it}, we fix the random seed to make it deterministic.

6.2 `DCLP`: Data center location problem

6.2.1 `DCLP`: Mathematical formulation

6.2.2 `DCLP`: 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 [17] for writing Julia script for SIPLIB 2.0 and explain the model throughout the following subsections.

6.3.1 SMKP: 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 which 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_{ji}, e_{ji}, t_{ki}, and w_{ki} 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 7.

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

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 7 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_{ji}	profit of the j^{th} xz-knapsack for including i^{th} x-type item
e_{ji}	profit of the j^{th} xz-knapsack for including i^{th} z-type item
t_{ki}	profit of the k^{th} xy-knapsack for including i^{th} x-type item
w_{ki}	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_i (1^{st} stage)	1 if the i^{th} x-type item is decided to be included, 0 otherwise
z_i (1^{st} stage)	1 if the i^{th} z-type item is decided to be included, 0 otherwise
y_i^s (2^{nd} stage)	1 if the i^{th} y-type item is decided to be included under scenario s, 0 otherwise

6.3.2 SMKP: Data generation

There are two factors that define the instance of SMKP: $|I|$ and $|\mathcal{S}|$. The sizes for another sets are fixed by $|J| = 50$ and $|K| = 5$ following [17]. Once we decide the factors by $n_I = |I|$, $n_S = |\mathcal{S}|$, $n_T = |T|$, and $n_{\mathcal{S}} = |\mathcal{S}|$, each instance is named by SMKP_n_I_$n_{\mathcal{S}}$. Again directly following [17], we randomly generate the parameters. Let U be a discrete uniform random variable: $U \sim Unif[1, 100]$. Then, the parameters are generated as follows:

$$
\begin{aligned}
c_i &= U, \quad \forall i \in I, \\
d_i &= U, \quad \forall i \in I, \\
q_i^s &= U, \quad \forall i \in I,\ \forall s \in \mathcal{S}, \\
a_{ji} &= U, \quad \forall j \in J,\ \forall i \in I, \\
e_{ji} &= U, \quad \forall j \in J,\ \forall i \in I, \\
t_{ki} &= U, \quad \forall k \in K,\ \forall i \in I, \\
w_{ki} &= U, \quad \forall k \in K,\ \forall i \in I, \\
b_j &= \frac{3}{4} \sum_{i \in I} (a_{ji} + e_{ji}), \quad \forall j \in J, \\
h_k &= \frac{3}{4} \sum_{i \in I} (t_{ji} + w_{ji}), \quad \forall k \in K.
\end{aligned}
$$

For the first-stage parameters, we fix the random seed to make it deterministic.

6.4 SSLP: Stochastic server location problem

SSLP is a class of problem that finds the optimal location of servers and the optimal allocation of clients to servers which maximizes the expected net income under uncertain presents of clients. SSLP finds applications in a variety of domains such as network design for electric power, internet services, telecommunications, and water distribution. SIPLIB provides 12 instances with varying number of clients, server locations, and scenarios in SMPS format. The largest instance includes 10 server locations, 50 clients, and 2,000 scenarios which corresponds to 120,001 constraints, 1,000,010 binary variables, and 20,000 continuous variables.

We refer to [20] for mathematical formulation and data generation forthcoming through the following subsections.

6.4.1 SSLP: Mathematical formulation

Let I, J, Z, and $\mathcal{S}$ be index sets for the clients, servers, zones, and scenarios. For $i \in I$, $j \in J$, $z \in Z$, and $s \in \mathcal{S}$, we define the notations in Table 8.

Suppose that we place a server at location j. Then, the allocation costs c_j and the server will provide capacity to serve up to u amount of resource to clients. The revenue earned by serving client i from location j is denoted by q_{ij}. We have also a shortage cost (penalty) q_{0j} for each unit of demand that remains unserved among the clients assigned to server j. If client i is served by a server at location j, it uses d_{ij} units of resource from the server. We allow only one server to be installed at each location and each client can only be served by one server. There is a requirement that a minimum number of servers to be located in a zone z, and is denoted by w_z.

The first-stage binary variables x_j decide whether or not a server is located at location j. The second-stage binary variables y_{ij}^s are referred to as recourse decision under scenario s and associated with the decision on serving client i by server j. The variables y_{ij}^s will be implemented in the future, when scenario s is finally observed.

Based on the above, the extensive form of SSLP can be stated as follows:

$$
\begin{aligned}
(\texttt{SSLP})\ \min\ & \sum_{j\in J} c_j x_j - \sum_{s\in\mathcal{S}} \mathbb{P}(s)\left(\sum_{i\in I}\sum_{j\in J} q^s_{ij} y^s_{ij} - \sum_{j\in J} q^s_{0j} y^s_{0j}\right) && \text{(6a)}\\
\text{s.t.}\ & \sum_{j\in J} x_j \le v, && \text{(6b)}\\
& \sum_{j\in J_z} x_j \ge w_z, \quad \forall z\in Z, && \text{(6c)}\\
& \sum_{i\in I} d_{ij} y^s_{ij} - y^s_{0j} \le u x_j, \quad \forall j\in J,\ \cancel{\forall i\in I,}\ \forall s\in\mathcal{S}, && \text{(6d)}\\
& \sum_{j\in J} y^s_{ij} = h^s_i, \quad \forall i\in I,\ \forall s\in\mathcal{S}, && \text{(6e)}\\
& x_j\in\{0,1\}, \quad \forall j\in J, && \text{(6f)}\\
& y^s_{ij}\in\{0,1\}, \quad \forall i\in I,\ j\in J,\ s\in\mathcal{S}, && \text{(6g)}\\
& y^s_{0j}\ge 0, \quad \forall j\in J,\ \forall s\in\mathcal{S}. && \text{(6h)}
\end{aligned}
$$

The objective function (6a) is to maximize total expected revenue of locating servers and serving customers by the servers. Constraint (6b) satisfies the requirement that only up to a total of v available servers can be installed. The zonal requirements that specify how many servers are needed in each zone are given by constraint (6c). Constraint (6d) ensures that a server located at site j can serve only up to its capacity u. The variable y^s_{0j} is introduced in the constraint (6d) to accomodate any overflows that are not served due to limitations in server capacity. These overflows result in a loss of revenue at a rate of q^s_{0j}. The inclusion of an artificial variable may allow a client to be assigned to servers that are not located. However, penalty costs associated with such an assignment may result in such high costs as to preclude it in an optimal solution, unless server capacity is so limited that some clients have to be turned away [20]. Constraint (6e) guarantees that each client is served by only one server. Constraint (6f) and (6g) are binary restrictions on the decision variables. Finally, constraint (6h) is the non-negativity requirement on the overflow variables.

Table 8 Notations for SSLP

Index sets:	
I	index set of clients ($i \in I$)
J	index set of server locations ($j \in J$)
Z	index set of zones ($z \in Z$)
$\mathcal{S}$	index set of scenarios ($s \in \mathcal{S}$)
Parameters:	
c_j	cost of locating a server at location j
q_{ij}^s	revenue from client i being served by server at location j under scenario s
q_{0j}^s	rate of revenue loss for overflows that are not served due to limited server capacity under scenario s
d_{ij}	resource demand of client i from server at location j
u	server capacity
v	upper bound on the total number of servers that can be located
w_z	minimum number of servers to be located in zone z
J_z	subset of server locations that belong to zone z
h_i^s	1 if client i is present under scenario s, 0 otherwise
$\mathbb{P}(s)$	probability of occurence for scenario s
Decision variables:	
x_j (1^{st} stage)	1 if a server is located at site j, 0 otherwise
y_{ij}^s (2^{nd} stage)	1 if client i is served by a server at location j under scenario s, 0 otherwise
y_{0j}^s (2^{nd} stage)	non-negative amount of overflows that are not served due to limitations in server j's capacity

6.4.2 SSLP: Data generation

For each instance of SSLP, we determine $n_I = |I|$, $n_J = |J|$, and $n_{\mathcal{S}} = |\mathcal{S}|$. Then, the instance is named by SSLP_n_I_n_J_$n_{\mathcal{S}}$. The client-server revenue are set to be 1 per unit of client demand. Some of deterministic parameters are randomly generated from the discrete uniform distribution while scenario data are generated from the Bernoulli distribution. In summary, the parameters are generated as follows:

$$
\begin{aligned}
c_j &= Unif[40, 80], \quad \forall j \in J, \\
q_{ij}^s &= d_{ij}, \quad \forall i \in I,\ \forall j \in J,\ \forall s \in \mathcal{S}, \\
q_{0j}^s &= 1000, \quad \forall j \in J,\ \forall s \in \mathcal{S}, \\
d_{ij} &= Unif[0, 25], \quad \forall i \in I,\ \forall j \in J, \\
h_i^s &= Bernoulli(0.5), \quad \forall i \in I,\ \forall s \in \mathcal{S}, \\
v &= |J| \\
u &= \\
w_z &= \\
J_z &=
\end{aligned}
$$

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 [12] for deriving the mathematical formulation. Due to the malfunction of subtour breaking constraints in the reference model, we refer to another paper [14] to break subtours. Combining the two references, we construct the forthcoming single commodity flow-based formulation `MPTSPs` that is used for `SIPLIB 2.0`.

6.5.1 `MPTSPs`: 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_{ijk}^s 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_{ijk}^s are avilable, the paths k between each couple of nodes i and j under scenario s, x_{ijk}^s 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_{ijk}^s under the scenario $s \in S$. Let $e_{ijk}^s \equiv c_{ijk}^s - \bar{c}_{ij}$ 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 $x_{ijk}^s = 1$ 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 [14]. 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 9.

$$\text{(MPTSPs)} \min \sum_{i\in N}\sum_{j\in N}\bar{c}_{ij}y_{ij} + \sum_{s\in\mathcal{S}}\mathbb{P}(s)\sum_{i\in N}\sum_{j\in N}\sum_{k\in K_{ij}} e^s_{ijk}x^s_{ijk} \tag{7a}$$

$$\text{s.t.} \sum_{j\in N:j\neq i} y_{ij} = 1, \quad \forall i \in N, \tag{7b}$$

$$\sum_{i\in N:i\neq j} y_{ij} = 1, \quad \forall j \in N, \tag{7c}$$

$$\sum_{j\in N}\phi_{lj} - \sum_{i\in N\setminus\{1\}}\phi_{il} = 1, \quad \forall l \in N\setminus\{1\}, \tag{7d}$$

$$\phi_{ij} \leq (|N|-1)\, y_{ij}, \quad \forall i \in N\setminus\{1\},\ \forall j \in N, \tag{7e}$$

$$\sum_{k\in K_{ij}} x^s_{ijk} = y_{ij}, \quad \forall i \in N,\ \forall j \in N,\ \forall s \in \mathcal{S}, \tag{7f}$$

$$x^s_{ijk} \in \{0,1\}, \quad \forall i \in N,\ \forall j \in N,\ \forall k \in K_{ij},\ \forall s \in \mathcal{S}, \tag{7g}$$

$$y_{ij} \in \{0,1\}, \quad \forall i \in N,\ \forall j \in N, \tag{7h}$$

$$\phi_{ij} \geq 0, \quad \forall i \in N,\ \forall j \in N. \tag{7i}$$

The first sum in the objective function (7a) 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 (7b) and (7c) form the assignment constraints and ensure that each node is visited only once. Given the fixed values of y_{ij}, constraint (7d) and (7e) 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 (7d) and (7e) 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 [16]. The constraint (7f) 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 (7g)-(7i) restrict the space from which the variables take values.

Table 9 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} (1^{st} stage)	the nonnegative real-valued flow on arc (i, j)
y_{ij} (1^{st} stage)	1 if path k between nodes $i, j \in N$ is selected at the second stage, 0 otherwise
x^s_{ijk} (2^{nd} stage)	1 if node j is visited just after node i, 0 otherwise

6.5.2 `MPTSPs`: Data generation

We follow the scenario generation methods described through the references [11,12,13]. 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 [13]. 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 [13], we fix $r = 7km$.
2. Calculate Euclidean distances between the nodes (EC_{ij}).
3. We guess and fix the deterministic velocity profile by $40km/h$ for the *central* nodes and $80km/h$ 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 [15] 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 [15] 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 `SIZES`: Mathematical formulation

Suppose a product is available in a finite number $|N|$ of sizes where 1 is the index of the smallest size and $|N|$ is the index of the largest size. Further, suppose size i is substitutable for size j if $i > j$, i.e., larger-sized items may fulfill demand for smaller sizes. Unlike typical cutting-stock problem, an item cannot be substituted into several pieces.

Let p_i be the unit production cost for size i. Generally $p_i > p_j$ for $i > j$. Let s be the setup cost for producing units of any size and r be the unit penalty cost of meeting demand for size j with a larger size i. Let d^l_{jt} be the stochastic demand for size j at time t under scenario l. Let c^l_t be the stochastic production capacity at time t under scenario l. $\mathbb{P}(l)$ is the equiprobable probability of occurence for scenario l.

We introduce three decision variables. The first-stage integer variable y_{it} is the number of units of sizes i produced at time t. The second-stage integer variable x^l_{ijt} denotes the number of units of size i cut to meet demand for smaller size j at time t under scenario l. The second-stage binary variable z_i denotes whether or not we produce size i item at time t under scenario l.

Based on the above definitions, `SIZES` can be fomulated by the following extensive form.

$$\text{(SIZES)}\ \min \sum_{t\in T}\sum_{i\in N} p_i y_{it} + \sum_{l\in\mathcal{L}}\mathbb{P}(l)\sum_{t\in T}\left(\sum_{i\in N} s z^l_{it} + r\sum_{i\in N\setminus\{1\}}\sum_{j=1}^{i-1} x^l_{ijt}\right) \quad (8a)$$

$$\text{s.t.}\ \sum_{i\in N} y_{it} \le c^l_t, \quad \forall t\in T,\ \forall l\in\mathcal{L}, \quad (8b)$$

$$\sum_{i=j}^{|N|} x^l_{ijt} \ge d^l_{jt}, \quad \forall j\in N,\ \forall t\in T,\ \forall l\in\mathcal{L}, \quad (8c)$$

$$\sum_{t'=1}^{t}\sum_{j=1}^{i} x^l_{ijt'} \le \sum_{t'=1}^{t} y_{it'}, \quad \forall i\in N,\ \forall t\in T,\ \forall l\in\mathcal{L}, \quad (8d)$$

$$y_{it} \le c^l_t z^l_{it}, \quad \forall i\in N,\ \forall t\in T,\ \forall l\in\mathcal{L}, \quad (8e)$$

$$y_{it} \in \mathbb{Z}_+, \quad \forall j\in N,\ \forall t\in T, \quad (8f)$$

$$x^l_{ijt} \in \mathbb{Z}_+, \quad \forall i\in N,\ \forall j\in N,\ \forall t\in T,\ \forall l\in\mathcal{L}, \quad (8g)$$

$$z^l_{it} \in \{0,1\}, \quad \forall i\in N,\ \forall t\in T,\ \forall l\in\mathcal{L}. \quad (8h)$$

The first sum of the objective function (8a) is the unit costs for producing items for all time periods. The second term corresponds to the expectation of the setup costs and penalty costs for substituting items. Constraint (8b) ensures the production for each period cannot exceed the capacity under all scenarios. Constraint (8c) and (8d) guarantee the demand for each item can be met for all time periods and for all scenarios. In particular, constraint (8d) means the demand can be met by the items that are produced in the previous periods. Constraint (8e) enforces the production limits. Constraints (8f)-(8h) are binary or integer restrictions of decision variables.

Table 10 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^l_{it}	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^l_t	production capacity at time t under scenario l
$\mathbb{P}(l)$	the probability of occurence of scenario l
Decision variables	
y_{it} (1^{st} stage)	number of units of size i produced at time t
x^l_{ijt} (2^{nd} stage)	number of units of size i cut to meet demand for smaller size j at time t under scenario l
z^l_{it} (2^{nd} stage)	1 if we produce size i at time t under scenario l, 0 otherwise

Table 11 Base data for `SIZES` scenarios [15]

i	sleeve length	unit production cost (p_i)	demand (d_i)
1	25	0.748	2500
2	30	0.7584	7500
3	35	0.7688	12500
4	40	0.7792	10000
5	45	0.7896	35000
6	50	0.8	25000
7	55	0.8014	15000
8	60	0.8208	12500
9	65	0.8312	12500
10	70	0.8416	5000
unit cutting cost (u): \$0.008			
setup cost (s): \$453			
production capacity (c_t): 200,000			

6.6.2 SIZES: Data generation

Instances of `SIZES` are generated based on the one-period data given in Table 11. Note that although the table includes sleeve length data, we do not use this information for `SIZES`. Following [15], we set the stochstic parameter $c_t^l = 200,000$ to be deterministic for all $t \in T$ and $l \in \mathcal{L}$, hence only the demand parameter (d_i^l) is stochastic throughout the scenarios. The stochastic demand data in generated based on Table 11. First, we decide the number of scenarios to be generated by $n_{\mathcal{L}} = |\mathcal{L}|$. Then, the demand data is specified by a vector of multipliers: one multiplier for each scenario that is multiplied times the demand vector from Table 11. For example, if $n_{\mathcal{L}} = 3$, the instance is defined by $(0.7, 1, 1.3)$. Or if $n_{\mathcal{L}} = 5$, the instance is defined by $(0.6, 0.8, 1, 1.2, 1.4)$. Since `SIPLIB` provides instances with $n_{\mathcal{L}} \leq 20$, we recommend the users to use `SIPLIB 2.0` to only generate instances with $n_{\mathcal{L}} \geq 20$. In `SIPLIB 2.0`, the multiplier vector is defined by the equally split set of subintervals between $[0.5, 1.5]$, e.g., when $n_{\mathcal{L}} = 20$, the multiplier vector is $(0.5, 0.55, 0.6, \ldots, 1.4, 1.45, 1.5)$. With larger value of $n_{\mathcal{L}}$, we will have vector with finer granularity. After that, the instance is named by `SIZES_`$n_{\mathcal{L}}$.

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

The unit commitment (UC) problem is a production cost model (PCM) that plans power system operations over an exteded time horizon. `SUCW` is a stochastic version of UC for studying the impact of incorporating highly uncertain power generation of large-scale wind turbines with transmission constraints and system component failures. We refer to [18] and [5] for mathematical models.

6.7.1 SUCW: Mathematical formulation

A two-stage stochastic unit commitment model SUCW is presented, where we make decision on slow generators in the first stage and make the commitment decisions for fast generators and the power dispatch decision in the second stage. In SUCW, we also incorporate ramping constraints, reserve constraints and transmission line capacity constraints. We assume the piecewise linear

convex cost function for the power generation.

$$\min \sum_{\sigma\in\mathcal{S}} \mathbb{P}(\sigma) \sum_{t\in T} \sum_{g\in G} \left(C_g^{\mathrm{fx}} x_{gt}^{\sigma} + C_g^{\mathrm{up}} u_{gt}^{\sigma} + C_g^{\mathrm{dn}} v_{gt}^{\sigma} + \sum_{k\in K} C_{gk}^{\mathrm{mar}} q_{gkt}^{\sigma} \right) \tag{9a}$$

$$\text{s.t. } 1 - x_{g(t-1)}^{\sigma} \geq u_{gt}^{\sigma}, \quad \forall \sigma \in \mathcal{S},\ g \in G,\ t \in T, \tag{9b}$$

$$x_{g(t-1)}^{\sigma} \geq v_{gt}^{\sigma}, \quad \forall \sigma \in \mathcal{S},\ g \in G,\ t \in T, \tag{9c}$$

$$x_{gt}^{\sigma} - x_{g(t-1)}^{\sigma} = u_{gt}^{\sigma} - v_{gt}^{\sigma}, \quad \forall \sigma \in \mathcal{S},\ g \in G,\ t \in T, \tag{9d}$$

$$x_{gt}^{\sigma} \geq \sum_{\tau=\max\{1,t-UT_g+1\}}^{t} u_{g\tau}^{\sigma}, \quad \forall \sigma \in \mathcal{S},\ g \in G,\ t \in T, \tag{9e}$$

$$1 - x_{gt}^{\sigma} \geq \sum_{\tau=\max\{1,t-DT_g+1\}}^{t} u_{g\tau}^{\sigma}, \quad \forall \sigma \in \mathcal{S},\ g \in G,\ t \in T, \tag{9f}$$

$$-RD_g \leq p_{gt}^{\sigma} - p_{g(t-1)}^{\sigma} \leq RU_g - s_{gt}^{\sigma}, \quad \forall \sigma \in \mathcal{S},\ g \in G,\ t \in T, \tag{9g}$$

$$s_{gt}^{\sigma} \leq RC_g x_{gt}^{\sigma}, \quad \forall \sigma \in \mathcal{S},\ g \in G,\ t \in T, \tag{9h}$$

$$\sum_{g\in G} s_{gt}^{\sigma} \geq SR_t, \quad \forall \sigma \in \mathcal{S},\ t \in T, \tag{9i}$$

$$p_{gt}^{\sigma} = P_g^{\min} x_{gt}^{\sigma} + \sum_{k\in K} q_{gkt}^{\sigma}, \quad \forall \sigma \in \mathcal{S},\ g \in G, t \in T, \tag{9j}$$

$$p_{gt}^{\sigma} + s_{gt}^{\sigma} \leq P_g^{\max} x_{gt}^{\sigma}, \quad \forall \sigma \in \mathcal{S},\ g \in G,\ t \in T, \tag{9k}$$

$$q_{gkt}^{\sigma} \leq Q_{gk}^{\max} x_{gt}^{\sigma}, \quad \forall \sigma \in \mathcal{S},\ g \in G,\ k \in K,\ t \in T, \tag{9l}$$

$$\sum_{g\in G} p_{gt}^{\sigma} = \sum_{n\in N} D_{nt}^{\sigma} - \sum_{w\in W} W_{wt}^{\sigma}, \quad \forall \sigma \in \mathcal{S},\ t \in T, \tag{9m}$$

$$-F_l^{\max} \leq \sum_{g\in G} LSF_{lg} p_{gt}^{\sigma} - \sum_{n\in N} LSF_{ln} D_{nt}^{\sigma} + \sum_{w\in W} LSF_{lw} W_{wt}^{\sigma} \leq F_l^{\max}, \quad \forall \sigma \in \mathcal{S},\ l \in L,\ t \in T, \tag{9n}$$

$$x_{gt}^{\sigma_1} = x_{gt}^{\sigma_2},\ u_{gt}^{\sigma_1} = u_{gt}^{\sigma_2},\ v_{gt}^{\sigma_1} = v_{gt}^{\sigma_2}, \quad \forall \sigma_1, \sigma_2 \in \mathcal{S},\ g \in G_s,\ t \in T, \tag{9o}$$

$$x_{g0}^{\sigma} = X_g^{\mathrm{init}}, \quad \forall \sigma \in \mathcal{S},\ g \in G, \tag{9p}$$

$$x_{gt}^{\sigma} = 1, \quad \forall \sigma \in \mathcal{S},\ g \in G,\ t \in \{1, \ldots, UT_g^{\mathrm{init}}\}, \tag{9q}$$

$$x_{gt}^{\sigma} = 0, \quad \forall \sigma \in \mathcal{S},\ g \in G,\ t \in \{1, \ldots, DT_g^{\mathrm{init}}\}, \tag{9r}$$

$$p_{g0}^{\sigma} = P_g^{\mathrm{init}}, \quad \forall \sigma \in \mathcal{S},\ g \in G, \tag{9s}$$

$$x_{gt}^{\sigma}, u_{gt}^{\sigma}, v_{gt}^{\sigma} \in \{0, 1\}, \quad \forall \sigma \in \mathcal{S},\ g \in G,\ t \in T, \tag{9t}$$

$$p_{gt}^{\sigma}, q_{gkt}^{\sigma}, s_{gt}^{\sigma} \geq 0, \quad \forall \sigma \in \mathcal{S},\ g \in G,\ t \in T. \tag{9u}$$

The objective function (9a) is to minimize the expected value of the sum of operating, start-up, shut-down, and production cost. Constraints (9b)-(9d) are for logical relationships amongst the commitment, start-up, and shut-down

decisions. Constraints (9e) and (9f) represent the minimum downtime and uptime of generators in each time period. Constraints (9g) and (9h) are ramping restrictions, and (9i) is a spinning reserve constraint. Constraints (9j) and (9k) ensure the amount of power generation is restricted by its minimum and maximum limits. Constraint (9l) represents the piecewise linearized power generation cost. Constraint (9m) guarantees the balanced flow. Constraint (9n) is for the transmission line flow. Constraint (9o) is a *nonanticipativity* constraint that enforces the decisions do not change over scenarios. Constraints (9p)-(9s) represent the initial conditions of generators and production level. Finally, constraints (9t) and (9u) restrict the space from which the decision variables can take values by binary or non-negative continuous.

Table 12 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^{fix}	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 SUCW: Data generation

For SUCW instances, thermal power generators are scheduled over a day. The schedules are subjected to uncertainty in wind power. We use a modified IEEE 188-bus system with 54 generators, 118 buses, and 186 transmission lines provided in [22]. 17 of the 54 generators are assumed to be allowed to start on demand (second-stage) while the other generators should be scheduled in advance (first-stage). We consider 3 identical wind farms. Each of them consts

of 120 wind turbines. We used real wind speed data predicted from the observations of 31 weather stations in Illinois.

Amongst the parameters, only the wind power generation data W^{σ}_{wt} is stochastic subjected to be generated randomly.

7 Solution report

8 Concluding remarks

(not finished yet) Any further contribution or suggestions for `SIPLIB 2.0` are always welcomed. Better solutions than discovered so far, more functions, more problems with `Julia` scripts for instance generation, more effective classification rules, etc.

References

1. 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.
2. SPS: Stochastic Programming Society (https://stoprog.org/what-stochastic-programming).
3. Introduction to Stochastic Programming, J. R. Birge, F. Louveaus.
4. S. Ahmed and R. Garcia. "Dynamic Capacity Acquisition and Assignment under Uncertainty," Annals of Operations Research, vol.124, pp. 267-283, 2003.
5. 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.
6. J.-P. Watson, D. L. Woodruff, and W. E. Hart, PySP: modeling and solving stochastic programs in Python, Mathematical Programming Computation, 2012.
7. SMI - Stochastic Modeling Interface. https://github.com/coin-or/Smi
8. Julia: A Fresh Approach to Numerical Computing. Jeff Bezanson, Alan Edelman, Stefan Karpinski and Viral B. Shah (2017) SIAM Review, 59: 6598.
9. JuMP - Julia for Mathematical Optimization, https://jump.readthedocs.io/en/latest/index.html
10. StructJuMP - Parallel algebraic modeling framework for block structured optimization models in Julia, https://github.com/StructJuMP/StructJuMP.jl
11. 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
12. 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
13. R. Tadei, G. Perboli, and F. Perfetti, The multi-path traveling salesman problem with stochastic travel costs, EURO Journal on Transportation and Logistics, 2017
14. Andre Langevin, Francois Soumis, and Jacques Desrosiers, Classification of travelling salesman problem formulations, Operations Research Letters, 1990
15. Soheila Jorjani, Carlton H. Scott, and David L. Woodruff, Selection of an optimal subset of sizes, International Journal of Production Research, 1999
16. 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
17. Gustavo Angulo, Shabbir Ahmed, and Santanu S. Dey, Improving the integer L-shaped method, 2014.

18. Anthony Papavasiliou and Shmuel S. Oren, Multiarea stochastic unit commitment for high wind penetration in a transmission constrained network, Operations Research, 2013
19. 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
20. Lewis Ntaimo and Suvrajeet Sen, The million-variable "March" for stochastic combinatorial optimization, Journal of Global Optimization, 2005
21. Koch et al., MIPLIB 2010, Mathmatical Programming Computation, 2011
22. Lee, C., Liu, C., Mehrotra, S., Shahidehpour, M, MOdeling transmission line constraints in two-stage robust unit commitment problem, IEEE Transactions on Power Systems, 2014