SIPLIB 2.0

Stochastic Integer Programming Library version 2

Yongkyu Cho[1,2] Kibaek Kim[2]
James Luedtke[3] Jeffrey Linderoth[3]

[1]Department of Industrial and Management Engineering
Pohang University of Science and Technology

[2]Mathematics and Computer Science Division
Argonne National Laboratory

[3]Department of Industrial and Systems Engineering
University of Wisconsin-Madison

INFORMS Annual Meeting 2018

Contents

1. Overview
2. Stochastic Integer Programming
3. Main tools: SMPS and StructJuMP
4. Implementation of a Julia Package: Siplib.jl
5. Computational Experiments (in progress)

Contents

1 Overview

2 Stochastic Integer Programming

3 Main tools: SMPS and StructJuMP

4 Implementation of a Julia Package: Siplib.jl

5 Computational Experiments (in progress)

What is SIPLIB?

[1]SIPLIB: A Stochastic Integer Programming Test Problem Library

- SIPLIB is a collection of test problems to facilitate computational and algorithmic research in stochastic integer programming (SIP).
- The test problem data is provided in the standard format (SMPS) unless otherwise mentioned.
- Where available, information on the underlying problem formulation and known solution is also included.

[1] Available at: https://www2.isye.gatech.edu/ sahmed/siplib/

Limitation of the former SIPLIB

Limitation of the former SIPLIB

1. Number of the instances
   - Researchers in this field need more test set.

Limitation of the former SIPLIB

1. Number of the instances
   - Researchers in this field need more test set.
2. Variation of problem types
   - Provides only 5 different variations in terms of three variable types: continuous, binary, integer.

Limitation of the former SIPLIB

1. Number of the instances
   - Researchers in this field need more test set.
2. Variation of problem types
   - Provides only 5 different variations in terms of three variable types: continuous, binary, integer.
3. Contribution rule
   - Different problem provides different information.
   - For some problems, only limited information is available.

Contribution of SIPLIB 2.0

SIPLIB 2.0 provides

Contribution of SIPLIB 2.0

SIPLIB 2.0 provides

- more instances accompanied with analytic & computational information

Contribution of SIPLIB 2.0

SIPLIB 2.0 provides

- more instances accompanied with analytic & computational information
- a `Julia` package to support
  - generating new instances in SMPS format
  - analyzing the instances (size, sparsity)
  - solving the instances to get some known bounds

Contribution of SIPLIB 2.0

SIPLIB 2.0 provides

- more instances accompanied with analytic & computational information
- a `Julia` package to support
  - generating new instances in SMPS format
  - analyzing the instances (size, sparsity)
  - solving the instances to get some known bounds
- all the details about the problems
  - formulation, deterministic & random parameters
  - parameterized modeling scripts written in `Julia` language

Contents

1. Overview
2. Stochastic Integer Programming
3. Main tools: SMPS and StructJuMP
4. Implementation of a Julia Package: Siplib.jl
5. Computational Experiments (in progress)

Problem of interest

Two-stage Stochastic Integer Programming (TSSIP)

Problem of interest

Two-stage Stochastic Integer Programming (TSSIP)

- Considers only two stages.
  - Present (first-stage) and Future (second-stage)

Problem of interest

Two-stage Stochastic Integer Programming (TSSIP)

- Considers only two stages.
  - Present (first-stage) and Future (second-stage)
- First-stage decision must be made for now.
- Second-stage decision can be made after the future is realized (called *recourse* action).

Problem of interest

Two-stage Stochastic Integer Programming (TSSIP)

- Considers only two stages.
  - Present (first-stage) and Future (second-stage)
- First-stage decision must be made for now.
- Second-stage decision can be made after the future is realized (called *recourse* action).
- First-stage decision can affect the second-stage decision.

Mathematical formulation

TSSIP:

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

- $X \subseteq \mathbb{R}^{n_1 - k_1} \times \mathbb{Z}^{k_1}$
- $c \in \mathbb{R}^{n_1}$
- $A \in \mathbb{R}^{m_1 \times n_1}$
- $b \in \mathbb{R}^{m_1}$
- $\mathcal{Q}(x) := \mathbb{E}_{\boldsymbol{\xi}}[Q(x, \boldsymbol{\xi})]$ (expected recourse function)
- $\boldsymbol{\xi}$ is a random element defined on a probability triple $(\Xi, \mathcal{F}, \mathbb{P})$

Recourse function $Q(\cdot, \cdot)$:

$$Q(x, \xi_s) := \min_{y \in Y} \left\{ q(\xi_s)^\top y : W(\xi_s) y \geq h(\xi_s) - T(\xi_s) x \right\} \tag{2}$$

- ξ_s is a realized random element (called *scenario*)
- $Y \subseteq \mathbb{R}^{n_2 - k_2} \times \mathbb{Z}^{k_2}$
- $q(\xi_s) \in \mathbb{R}^{n_2}$
- $W(\xi_s) \in \mathbb{R}^{m_2 \times n_2}$
- $h(\xi_s) \in \mathbb{R}^{m_2}$
- $T(\xi_s) \in \mathbb{R}^{m_2 \times n_1}$

Mathematical formulation

Extensive Form (EF):

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

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

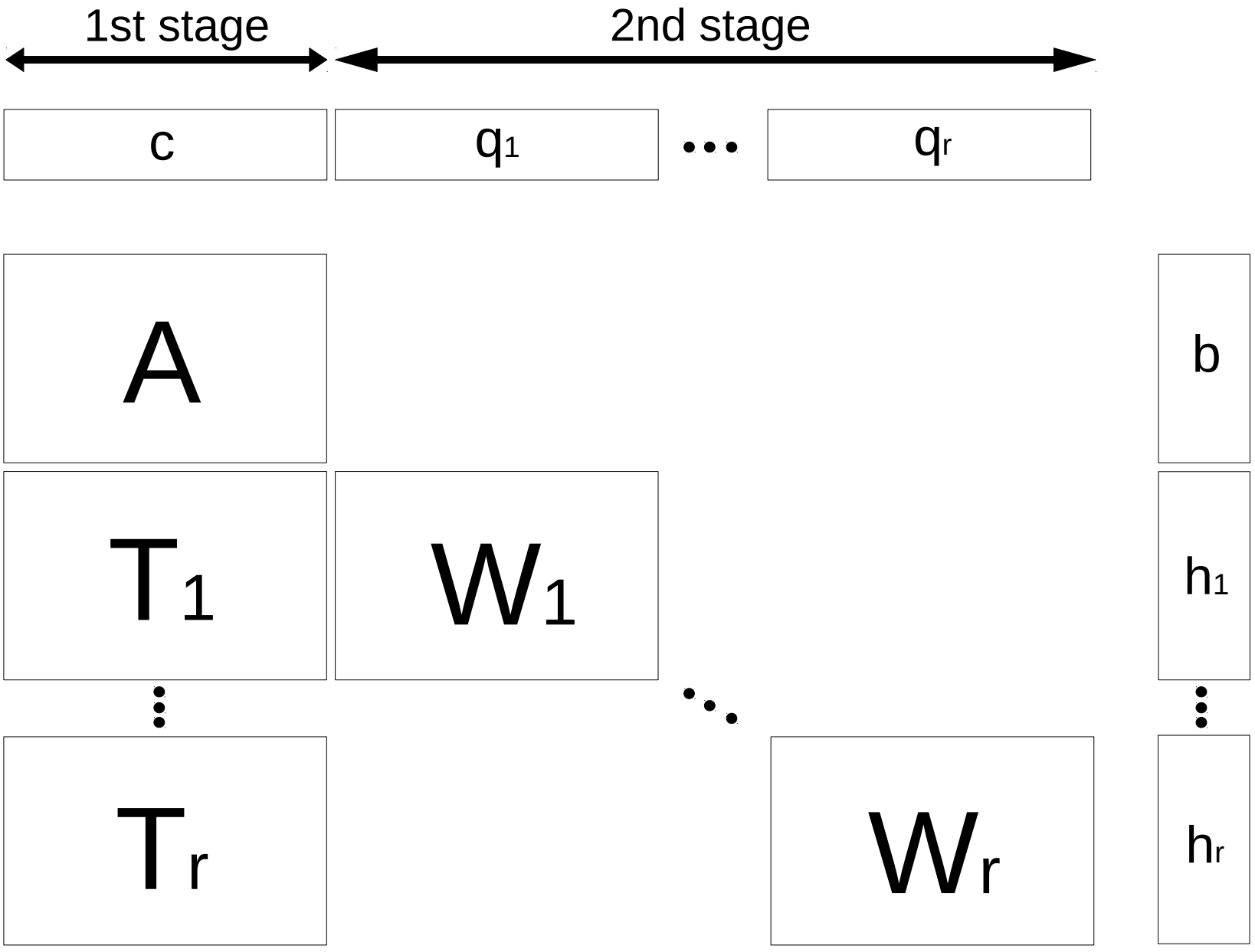
$$T_s x + W_s y_s \geq h_s, \quad \forall s \in \{1, \ldots, r\}, \tag{3c}$$

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

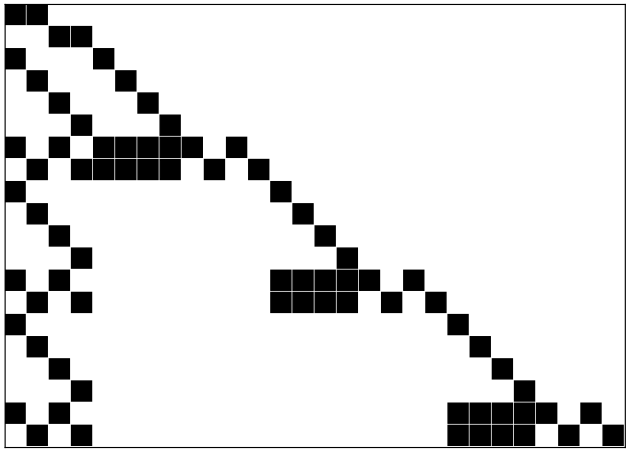
$$y_s \in Y, \quad \forall s \in \{1, \ldots, r\}. \tag{3e}$$

- $y := \{y_1, y_2, \ldots, y_r\}$
- $q_s := q(\xi_s)$
- $W_s := W(\xi_s)$
- $h_s := h(\xi_s)$
- $T_s := T(\xi_s)$

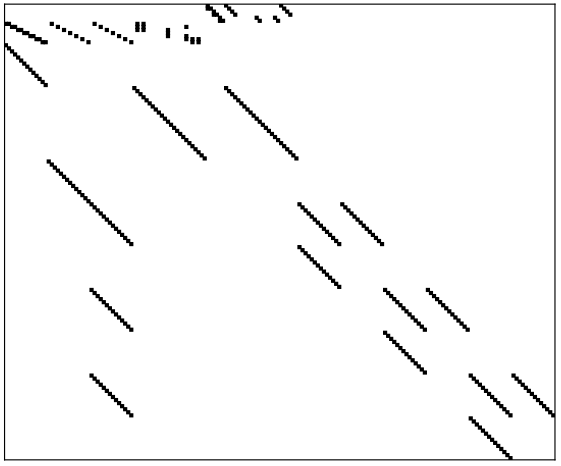
EF: Block-diagonal structure



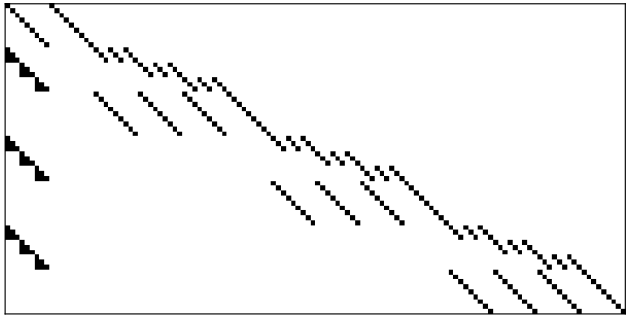
EF: Block-diagonal structure



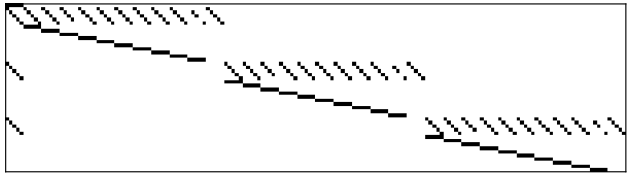
(a) AIRLIFT_3



(b) CHEM_3



(c) DCAP_3_3_3_3



(d) SSLP_5_10_3

Figure: Example: Repeated sparsity patterns in EF

Contents

1. Overview
2. Stochastic Integer Programming
3. Main tools: SMPS and StructJuMP
4. Implementation of a Julia Package: Siplib.jl
5. Computational Experiments (in progress)

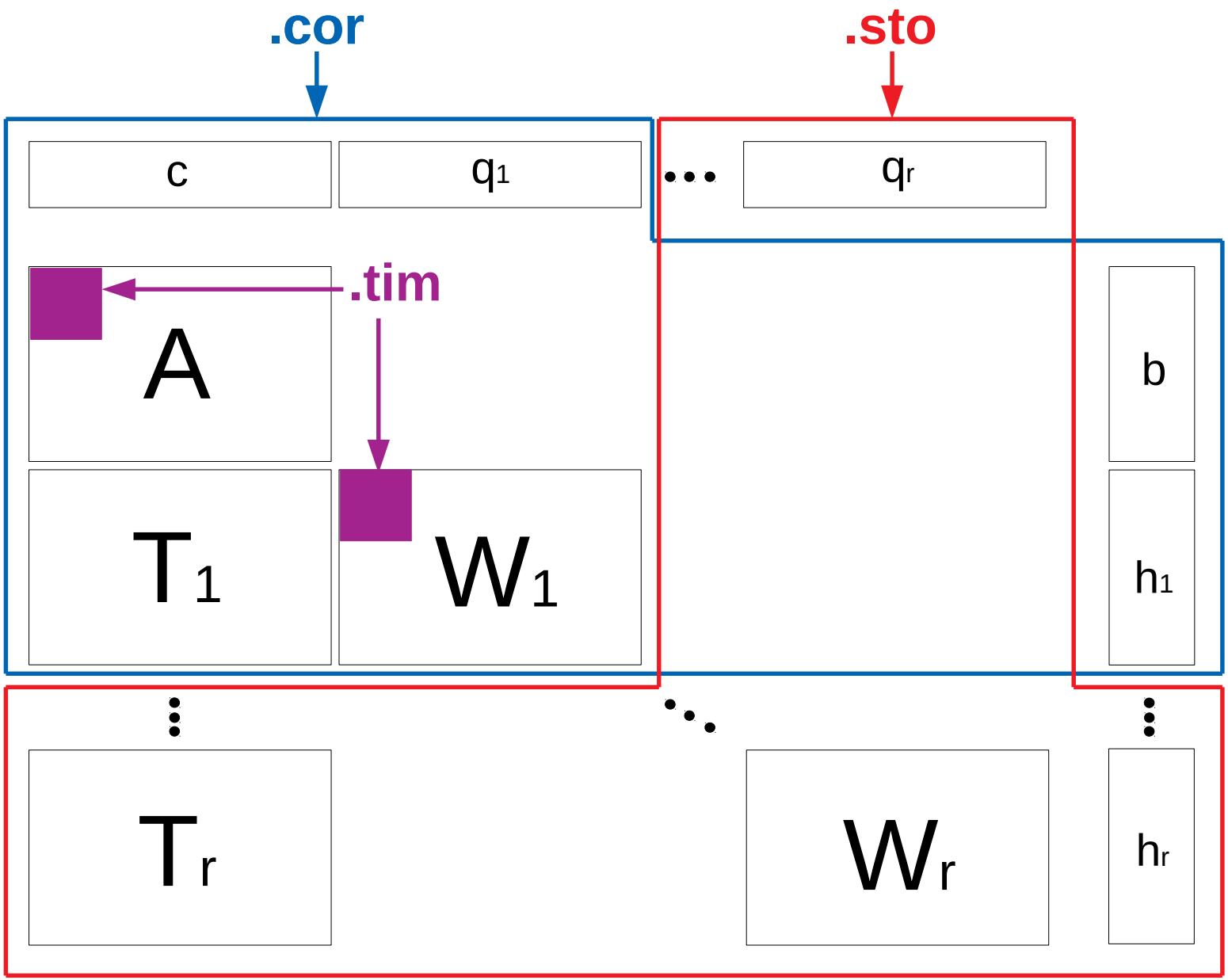
SMPS format

[2]SMPS is a standard column-oriented data format for stochastic programs.

- **(example)** Let DCAP_3_3_3_10 be the instance name. Then, SMPS format comprises
  - DCAP_3_3_3_10.cor
  - DCAP_3_3_3_10.tim
  - DCAP_3_3_3_10.sto
- The role of each file
  - **`.cor`**: Core file written in MPS format. This describes the fundamental problem structure.
  - **`.tim`**: Time file which specifies the location where each stage begins.
  - **`.sto`**: Stoch file which contains scenario data.

[2] For details: http://plato.asu.edu/ftp/smps.pdf

SMPS format: Graphical description



StructJuMP package

[3]**StructJuMP is a `Julia` package for modeling structured optimization models.**

- **StructJuMP** is an extension of **JuMP** package, a domain-specific modeling language for mathematical optimization embedded in `Julia`.
  - **JuMP** is generic, fast, and straightforward modeler.
- **StructJuMP** constructs a **`JuMP.Model`**-type object that contains every information of an SIP instance.

[3] Available at: https://github.com/StructJuMP/StructJuMP.jl

Example: Modeling DCAP with StructJuMP

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

$$\text{s.t. } x_{it}\le \mathrm{M}u_{it},\quad \forall i\in R,\ \forall t\in T,$$

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

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

$$x_{it}\ge 0,\quad \forall i\in R,\ \forall t\in T,$$

$$u_{it}\in\{0,1\},\quad \forall i\in R,\ \forall t\in T,$$

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

Formulation: DCAP (extensive form)

```
using StructJuMP

# construct JuMP.Model
model = StructuredModel(num_scenarios = nS)

## 1st stage
@variable(model, x[i=R,t=T] >= 0)
@variable(model, u[i=R,t=T], Bin)
@objective(model, Min, sum(a[i,t]*x[i,t] + b[i,t]*u[i,t] for i in R for t in T))
@constraint(model, [i=R,t=T], x[i,t] - u[i,t] <= 0)

## 2nd stage
for s in S
    sb = StructuredModel(parent=model, id = s, prob = Pr[s])
    @variable(sb, y[i=R, j=N, t=T], Bin)
    @variable(sb, z[j=N,t=T], Bin)  # modify as SIPLIB 1.0
    @objective(sb, Min, sum(c[i,j,t,s]*y[i,j,t] for i in R for j in N for t in T)
                        + sum(c0[j,t,s]*z[j,t] for j in N for t in T))
    @constraint(sb, [i=R, t=T], -sum(x[i,tau] for tau in 1:t)
                                + sum(d[j,t,s]*y[i,j,t] for j in N) <= 0)
    @constraint(sb, [j=N, t=T], sum(y[i,j,t] for i in R) + z[j,t] == 1)
end
```

Figure: DCAP (modeling script using StructJuMP)

Contents

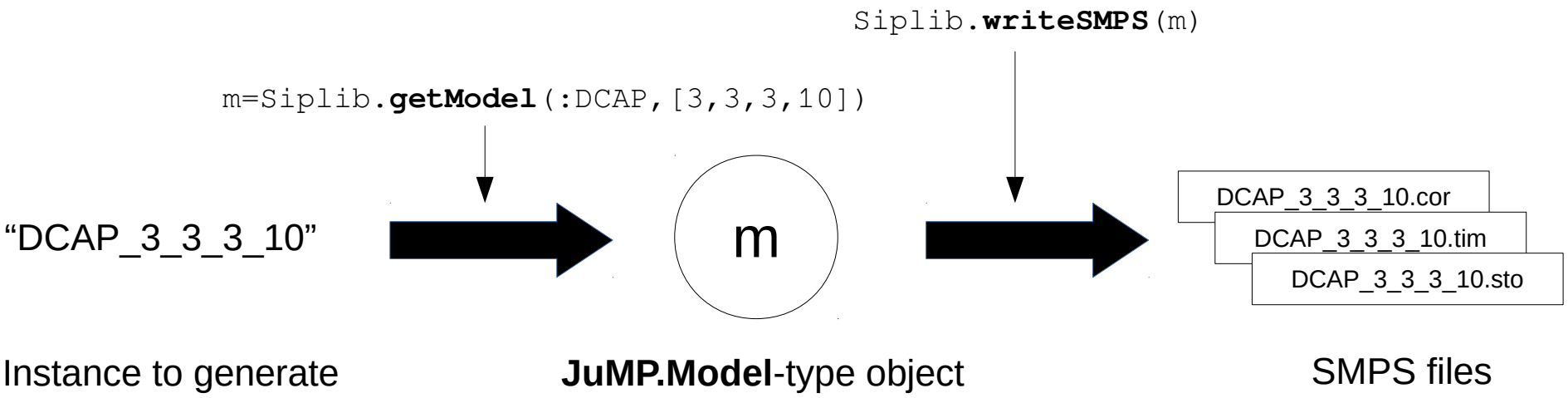
1. Overview
2. Stochastic Integer Programming
3. Main tools: SMPS and StructJuMP
4. Implementation of a Julia Package: Siplib.jl
5. Computational Experiments (in progress)

Core functionality of `Siplib.jl`

Generating SMPS files of SIP instance:

Core functionality of `Siplib.jl`

Generating SMPS files of SIP instance:



Problems available in SIPLIB 2.0

We implement `Julia` scripts for modeling 11 different problems from various sources and embed them into `Siplib.jl`.

Problem	Description	Main reference
AIRLIFT	Airlift operations scheduling	Midler and Wollmer (1969)
CARGO	Cargo network scheduling	Mulvey and Ruszczyński (1995)
CHEM	Design of batch chemical plants	Subrahmanyam et al. (1994)
DCAP	Dynamic capacity planning with stochastic demand	Ahmed and Garcia (2003)
MPTSPs	Multi-path traveling salesman problem with stochastic travel costs	Tadei et al. (2017)
PHONE	Telecommunication network planning	Sen et al. (2004)
SDCP	Stochastic data center placement	Kim et al. (2017)
SIZES	Optimal product substitution with stochastic demand	Jorjani et al. (1999)
SMKP	Stochastic multiple knapsack problem	Angulo et al. (2016)
SSLP	Stochastic server location problem	Ntaimo and Sen (2005)
SUC	Stochastic unit commitment problem	Papavasiliou and Oren (2013)

Problems available in SIPLIB 2.0

We parameterize the problems to let users tailor the instances and generate SMPS files as they want.

Problem	**Instance name**	Remark
AIRLIFT	**AIRLIFT**_$\mathcal{S}$	$\mathcal{S}$: number of scenarios
CARGO	**CARGO**_$\mathcal{S}$	$\mathcal{S}$: number of scenarios
CHEM	**CHEM**_$\mathcal{S}$	$\mathcal{S}$: number of scenarios
DCAP	**DCAP**_R_N_T_$\mathcal{S}$	R: number of resources, N: number of tasks, T: number of time periods, $\mathcal{S}$: number of scenarios
MPTSPs	**MPTSPs**_d_N_$\mathcal{S}$	d: node distribution strategy, N: number of nodes, $\mathcal{S}$: number of scenarios
PHONE	**PHONE**_$\mathcal{S}$	$\mathcal{S}$: number of scenarios
SDCP	**SDCP**_k_p_d_$\mathcal{S}$	k: maximum number of dispatchable loads, p: wind penetration level (%), d: day type, $\mathcal{S}$: number of scenarios
SIZES	**SIZES**_$\mathcal{S}$	$\mathcal{S}$: number of scenarios
SMKP	**SMKP**_I_$\mathcal{S}$	I: number of types for item, $\mathcal{S}$: number of scenarios
SSLP	**SSLP**_I_J_$\mathcal{S}$	I: number of clients, J: number of server locations, $\mathcal{S}$: number of scenarios
SUC	**SUC**_d_$\mathcal{S}$	d: day type, $\mathcal{S}$: number of scenarios

Components of the problems

We classify the problems based on their stage-wise variable types.

	1st stage		2nd stage	
Problem	Variable	Constraint	Variable	Constraint
AIRLIFT	$\mathbb{I}$	IKN	$\mathbb{C}$, $\mathbb{I}$	BIN, GEN
CARGO	$\mathbb{I}$	IKN	$\mathbb{C}$	GEN
CHEM	$\mathbb{C}$, $\mathbb{I}$	VBD, GEN	$\mathbb{C}$, $\mathbb{B}$	GEN
DCAP	$\mathbb{C}$, $\mathbb{B}$	VBD	$\mathbb{B}$	PAR, M01
MPTSPs	$\mathbb{C}$, $\mathbb{B}$	PAR, GEN	$\mathbb{B}$	GEN
PHONE	$\mathbb{C}$	IVK	$\mathbb{C}$, $\mathbb{I}$	GEN
SDCP	$\mathbb{I}$	IKN	$\mathbb{C}$	GEN
SIZES	$\mathbb{I}$	VBD, GEN	$\mathbb{B}$, $\mathbb{I}$	IKN
SMKP	$\mathbb{B}$	KNA	$\mathbb{B}$	KNA
SSLP	$\mathbb{B}$	IVK, GEN	$\mathbb{C}$, $\mathbb{B}$	GEN
SUC	$\mathbb{C}$, $\mathbb{B}$	VBD, GEN	$\mathbb{C}$, $\mathbb{B}$	VBD, GEN

- $\mathbb{C}$: continuous, $\mathbb{B}$: binary, $\mathbb{I}$: integer
- We have 10 non-redundant combinations.
- Constraint type notation is adopted from [4]MIPLIB 2010.

[4] Available at: http://miplib.zib.de/miplib2010.php

Contents

1. Overview
2. Stochastic Integer Programming
3. Main tools: SMPS and StructJuMP
4. Implementation of a Julia Package: Siplib.jl
5. Computational Experiments (in progress)

Computing environment

Computing facility	[5]*Bebop* with each node - CPU: Intel Xeon Processor E5-2695 v4 (36 cores, 36 threads) - Clock speed: 2.10GHz (maximum 3.30GHz) - Memory:128GB (45MB Smart cache)
Solver	General purpose MIP solver: CPLEX 12.8 Open-source Dual Decomposition based SIP solver: DSP
Multi-threading per instance	36
Time limit per instance	1 hours

[5] 1024-node computing cluster in Argonne National Lab (ANL): https://www.lcrc.anl.gov/systems/resources/bebop/

Computational report (in progress)

Table: Computational report (tentative)

Instance	Objective value		Optimality gap		LP2-relax gap	REVPI	RVSS
	CPLEX (SD)	DSP (SD)	CPLEX (time)	DSP (time)	CPLEX	CPLEX	CPLEX
AIRLIFT_200 AIRLIFT_300 AIRLIFT_500 AIRLIFT_1000							
CARGO_10 CARGO_50 CARGO_100							

- **DSP's optimality gap**: Relative Lagrangian duality gap
- **LP2-relax gap**: Relative gap between the EF and LP-relaxation (2nd stage only)
- **REVPI**: Relative Expected Value of Perfect Information
- **RVSS**: Relative Value of Stochastic Solution

Thank you!

Any suggestions are welcomed.

Contact point: **kimk@anl.gov**