UNIVERSITY OF TRIESTE

Mathematical Optimisation project

“ Using 3D-printing in disaster response: The two-stage stochastic 3D-printing knapsack problem ”

Denise D. Tönissen, Loe Schlicher

Federico Boni

20/09/2022

Introduction

- An aspect that complicates the allocation of basic survival resources in natural disasters is the **level of uncertainty** in the demand.
- **3D-printing** can **anticipate the actual demand**.
- Printing material can be **packed more efficiently** than items.
- Printers nowadays have an **high weight and volume** and they require **long printing times**.

It is convenient to bring 3D printers in disaster response operations?

Abstract

- The paper introduce a new type of problem, called the **two-stage stochastic 3D-printing knapsack problem or TSS-3DKP**.
- The paper provide a **two-stage stochastic programming formulation** for the problem, for which both the first and the second stage are NP-hard integer linear programs.
- The authors reformulate this formulation to an **equivalent integer linear program**, which can be efficiently solved by standard solvers.
- Their numerical results illustrate that for most situations **using a 3D-printer is beneficial.**

Methodology

- The model has been implemented both with **Gurobi** and **Xpress** libraries in Python.
- Datasets have been constructed in accord to the **distributions given by the authors**.
- All the analysis carried in the paper have been performed again using Gurobi.
- In addition new analysis and **scalability analysis** have been carried.

TSS-3DKP

TSS-3DKP – Two stage stochastic problem

Uncertainty
Some parameters with uncertainty (i.e. demand for each item).

Stochastic programming [0]
One way to model *uncertainty*: it is assumed that the probability distributions of the uncertain parameters are *known a priori*. The uncertainties are usually characterized by some discrete realizations of the uncertain parameters as an approximation to the real probability distribution, called *scenarios*.

Two stage stochastic problem

Two stage [0]
Stochastic problems divided in two stage, where:
- Stage 1 – First decision.
- Stage 2 – Second decision, *taken after the resolution of uncertainty.*

Deterministic equivalent [1]
A standard two stage stochastic problem can be written as a *deterministic problem* with constraints and variables of Stage 2 for each scenario.

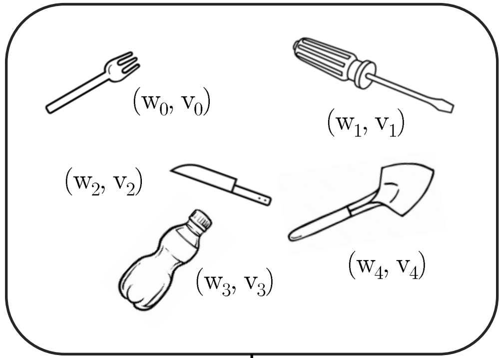
[0] Can Li and E. Grossman, 2021. A Review of Stochastic Programming Methods for Optimization of Porcess Systems Under Uncertainty. Comput. Front. Chem. Eng., 28 January 2021.
[1] Determinist equivalent of a two stage stochastic problem, Wikipedia

TSS-3DKP – The first stage

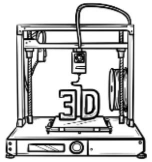
- Decision maker has to fill a multidimensional knapsack (weight capacity $\mathbf{W}$, volume capacity $\mathbf{V}$)
- Each item (i) from the set of pysichal items $\mathbf{N}$ has a weight $\mathbf{w_i}$ and a volume $\mathbf{v_i}$
- Each unit of printing material has weight $\mathbf{w_b}$ and volume $\mathbf{v_b}$
- Each 3D printer has weight $\mathbf{w_p}$ and volume $\mathbf{v_p}$
- The first stage decision is a = (($\mathbf{a_i}$)$_{\mathbf{i \in N}}$, $\mathbf{a_p}$, $\mathbf{a_b}$)

$$\left[\begin{array}{l} (\mathbf{a_i})_{\mathbf{i\in N}} : \text{number of i items to bring} \\ \mathbf{a_p} : \text{number of printers to bring} \\ \mathbf{a_b} : \text{number of units of printing material to bring} \end{array}\right.$$

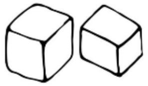
N



(W,V)



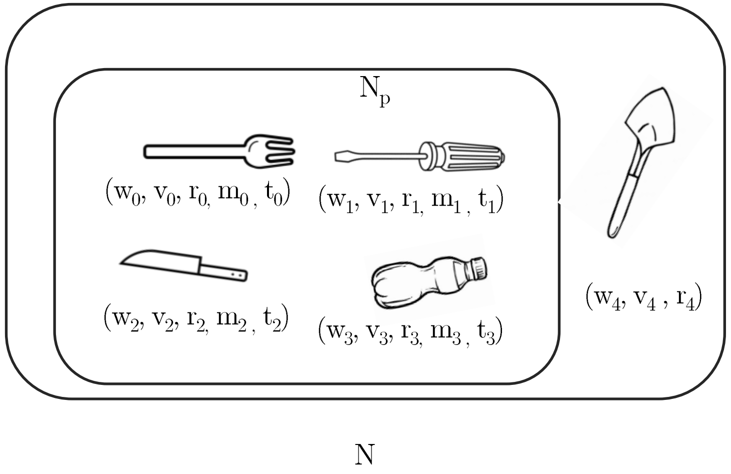
(w_p, v_p, T)

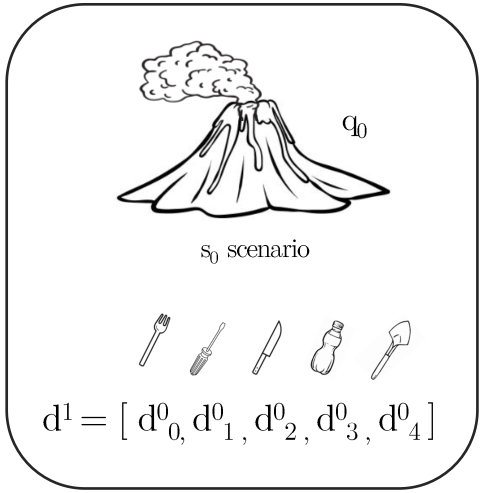


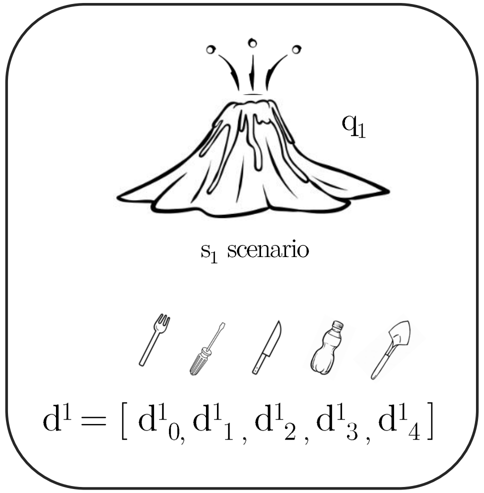
(w_b, v_b)

TSS-3DKP – The second stage

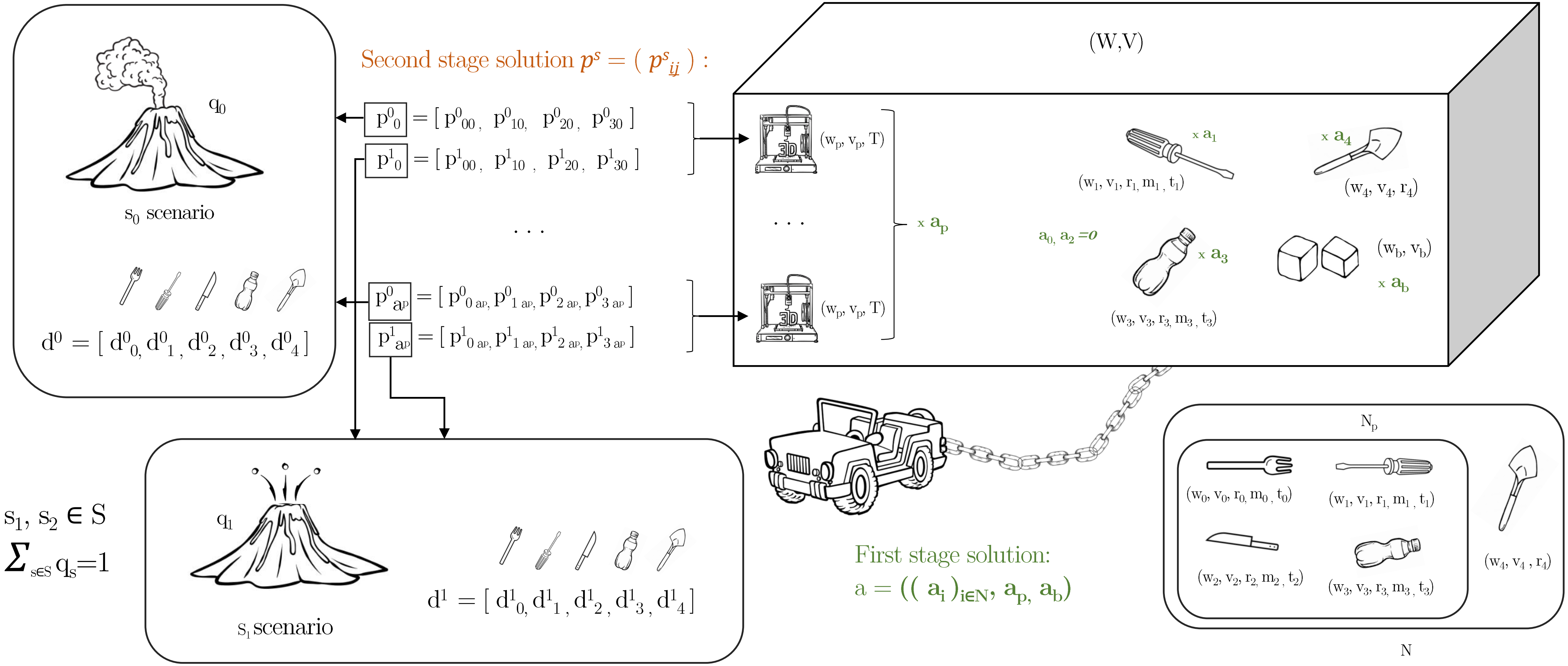
- Demands is modelled with a set of scenarios **S** where **s** $\in$ **S** has probability $\mathbf{q_s} \in [\mathbf{0,1}]$
- Each item **i** in the set of printable items $\mathrm{N^p} \subseteq \mathbf{N}$ require $\mathbf{m_i}$ units of printing material and a printing time $\mathbf{t_i}$
- A reward $\mathbf{r_i}$ is obtained for each $\mathbf{r_i}$ carried, a reward $\boldsymbol{\alpha}\ \mathbf{r_i}$ for each item **i** printed.
- $\boldsymbol{p^s_{ij}}$: number of times item **i** is printed by printer **j** in scenario **s**
- The second stage decision is $\mathbf{p}^s = (\ \boldsymbol{p^s_{ij}}\ )$ with $\boldsymbol{i} \in \mathrm{N^p}$ and $\boldsymbol{j} \in \boldsymbol{P}(\mathrm{a_p})$ where $\boldsymbol{P}\ (\boldsymbol{x}) = \{\boldsymbol{i} \in \mathbb{N}_+ | \boldsymbol{i} \leq \boldsymbol{x}\}$







TSS-3DKP – The second stage



TSS-3DKP – Formulation

First stage

Decision variables

- a_p: number of printers to take.
- a_b: number of units of printing material to take.
- $(a_i)_{i\in N}$: number of items to take.

Constraints

$$\sum_{i\in N} a_i w_i + a_p w_p + a_b w_b \le W, \quad (1)$$

$$\sum_{i\in N} a_i v_i + a_p v_p + a_b v_b \le V, \quad (2)$$

$$a_b \le a_p \mathcal{M}, \quad (3)$$

$$a_p, a_b, a_i \in \mathbb{N}_0 \qquad \forall i \in N, \quad (4)$$

Objective function

$$\max \sum_{s\in S} q_s \mathrm{Q}(\boldsymbol{a}, s)$$

Second stage

Decision variables

- a^s_i : number of item i to take in scenario s.
- p^s_{ij} : number of item i printed by printer j in scenario s

Constraints

$$a_i^s \le d_i^s \qquad \forall i \in N \setminus N^p, \quad (5)$$

$$a_i^s + \sum_{j\in P(a_p)} p_{ij}^s \le d_i^s \qquad \forall i \in N^p, \quad (6)$$

$$a_i^s \le a_i \qquad \forall i \in N, \quad (7)$$

$$\sum_{i\in N^p} \sum_{j\in P(a_p)} p_{ij}^s m_i \le a_b, \quad (8)$$

$$\sum_{i\in N^p} p_{ij}^s t_i \le T \qquad \forall j \in P(a_p), \quad (9)$$

$$a_i^s \in \mathbb{N}_0 \qquad \forall i \in N, \quad (10)$$

$$p_{ij}^s \in \mathbb{N}_0 \qquad \forall i \in N^p, \quad \forall j \in P(a_p). \quad (11)$$

Objective function

$$\mathrm{Q}(\boldsymbol{a}, s) := \quad \max \sum_{i\in N} a_i^s r_i + \sum_{i\in N^p} \sum_{j\in P(a_p)} \alpha p_{ij}^s r_i$$

TSS-3DKP – Upperbound

- It is well-known that a standard two-stage stochastic programming problem can be modelled as a large integer linear programming called the **deterministic equivalent.**
- However, the **TSS-3DKP** has the non-standard feature that the number of constraints **(Constraints (9))** and variables **(Constraints (11))** in the second-stage problem depend on the first-stage decisions (a_p).

$$\sum_{i \in N^p} p_{ij}^s t_i \leq T \qquad \forall j \in P(a_p), \qquad (9)$$

$$p_{ij}^s \in \mathbb{N}_0 \qquad \forall i \in N^p, \quad \forall j \in P(a_p). \qquad (11)$$

$$\Downarrow$$

- An **upper bound is identified on the total number of 3D-printers** (a_p) that can be packed, and include printer related constraints and variables of the second-stage problem into our deterministic equivalent as if the **knapsack would be filled with this upper bound of 3D-printers**.

TSS-3DKP – Upperbound

- For each scenario $s \in S$, we determine **how many 3D-printers are needed to print all demand d^s**. We assume that items are allocated one by one in order of their indices to the 3D printers and **we go to the next 3D-printer if adding another item would exceed the printing T of the 3D-printer**. Taking the maximum number of 3D-printers over all possible scenarios then gives us our first upper bound U.

$$\Big\} \; U$$

- Sometimes, this upper bound exceeds the total knapsack weight or volume. In those cases, we select the **maximum number of 3D-printers that fit the knapsack as upper bound.**

$$\Downarrow$$

$$Z = \min \{ \lfloor \frac{W}{w_p} \rfloor, \lfloor \frac{V}{v_p} \rfloor, U \}.$$

TSS-3DKP – Deterministic equivalent

$$\text{ILP-3DKP}(Z) \quad \max \sum_{s\in S} q_s \left[\sum_{i\in N} a_i^s r_i + \sum_{i\in N^p} \sum_{j\in P(Z)} \alpha p_{ij}^s r_i\right]$$

$$\text{s.t.} \sum_{i\in N} a_i w_i + a_b w_b + \sum_{j\in P(Z)} y_j w_p \le W, \tag{12}$$

$$\sum_{i\in N} a_i v_i + a_b v_b + \sum_{j\in P(Z)} y_j v_p \le V, \tag{13}$$

$$a_b \le \sum_{j\in P(Z)} y_j M, \tag{14}$$

$$a_i^s \le d_i^s \qquad \forall i \in N \setminus N^p, \quad \forall s \in S, \tag{15}$$

$$a_i^s + \sum_{j\in P(Z)} p_{ij}^s \le d_i^s \qquad \forall i \in N^p, \quad \forall s \in S, \tag{16}$$

$$a_i^s \le a_i \qquad \forall i \in N, \quad \forall s \in S, \tag{17}$$

$$\sum_{i\in N^p} \sum_{j\in P(Z)} p_{ij}^s m_i \le a_b \qquad \forall s \in S, \tag{18}$$

$$\sum_{i\in N^p} p_{ij}^s t_i \le T y_j \qquad \forall j \in P(Z), \quad \forall s \in S, \tag{19}$$

$$y_j \le y_{j-1} \qquad \forall j \in P(Z) \setminus \{1\}, \tag{20}$$

First stage variable $\mathbf{a_p}$ became set of dummy variables y:

$$y_j = \begin{cases} 0 & \text{if printer number j will be present in the knapsack.} \\ 1 & \text{otherwise.} \end{cases}$$

$$y_j \in \{0,1\} \qquad \forall j \in P(Z), \tag{21}$$

$$a_b \in \mathbb{N}_0 \tag{22}$$

$$a_i \in \mathbb{N}_0 \qquad \forall i \in N, \tag{23}$$

$$a_i^s \in \mathbb{N}_0 \qquad \forall i \in N, \tag{24}$$

$$p_{ij}^s \in \mathbb{N}_0 \qquad \forall i \in N^p, \quad \forall j \in P(Z), \quad \forall s \in S. \tag{25}$$

Instance generation

Parameter	Distribution
Weight of item i	$w_i \sim U[1,R]$
Revenue of item i	$r_i \sim U[1,R]$
Quality factor alpha	$\alpha = 0.8$
Weight and volume of a printer	$w_p = v_p = 5000$
Number of printable items	$N_p = 0.5 \cdot \lvert N \rvert$
Knapsack weight capacity	$W \sim \lfloor U[0.5, 1] \cdot \sum_{s \in S} q_s \sum_{i \in N} w_i d_i^s \rceil$
Demand for each scenario	$d^s_i \sim \lfloor U[0, U_i] \rceil$, $U_i \sim \lfloor U[1, D] \rceil$
. . .	

R=100

- We generate the weight w_i and the reward r_i according to the class **uncorrelated knapsack class** of Pisinger[0] (2005).
- **This class is realistic for disaster response missions**: it considers many different items for which the weight, volume, and rewards vary heavily.

Maximum demand for each item in every scenario

Number of scenarios

Number of items

$$\boldsymbol{N}\ x\ \boldsymbol{D}\ y\ \boldsymbol{S}\ z$$

[0] Pisinger, D., 2005. Where are the hard knapsack problems? Comput. Oper. Res. 32(9), 2271–2284.

Computational results

Computational results

- We will test our ILP-3DKP(Z) on five instance sets of 100 instances:
  - **N100D100S50**
  - **N200D100S50**
  - **N100D200S50**
  - **N100D100S100**
  - **N200D200S100**

Computational results - The performance criteria of the 5 instance sets.

Instance Set	Cuts (#)	Nodes (#)	Fails (#)	Avg time (s)	Max time (s)
N100D100S50	1191	1.0	0	2.97	6.4
N100D200S50	1501	1.0	0	6.19	13.15
N200D100S50	1569	1.0	0	6.19	22.89
N100D100S100	1637	1.0	0	7.29	20.77
N200D200S100	2393	1.0	0	18.45	43.5

Table 1: *The performance criteria of the 5 instance sets.*

- Average execution time is less than a minute.
- Most of the instance do not require branching.
- No computation turned out to be a failure.

Computational results - Different relative gaps

Relative gap	Cuts (#)	Nodes (#)	Fails (#)	Avg time (s)	Max time (s)
0.01%	1658	1.0	0	8.22	43.5
0.1%	1265	1.0	0	7.24	30.88
1.0%	40	1.0	0	5.53	20.66
10.0%	0	1.0	0	5.45	20.54

Table 2: *Comparison of the ILP-3DKP(Z) method for different relative gaps*

- The standard stopping criteria of the Gurobi MIP solver is the relative gap of 0.01%.
- Relative gap between upperbound and incumbent solution found with Gurobi heuristic is always less then 10%.
- As per the paper, we will use the **relative gap of 0.1%** in the remainder of the analyses.

Computational results – EVPI and VSS

- We will evaluate the ability of 3D printers to hedge against demand uncertainty by evaluating the **EVPI (expected value of perfect information)** and the **VSS (value of the stochastic solution)** for the N100D100S50 instance set in two cases: when packing 3D printers is allowed and when it is not.
- EVPI and VSS are well defined in the book Introduction to Stochastic Programming [2]

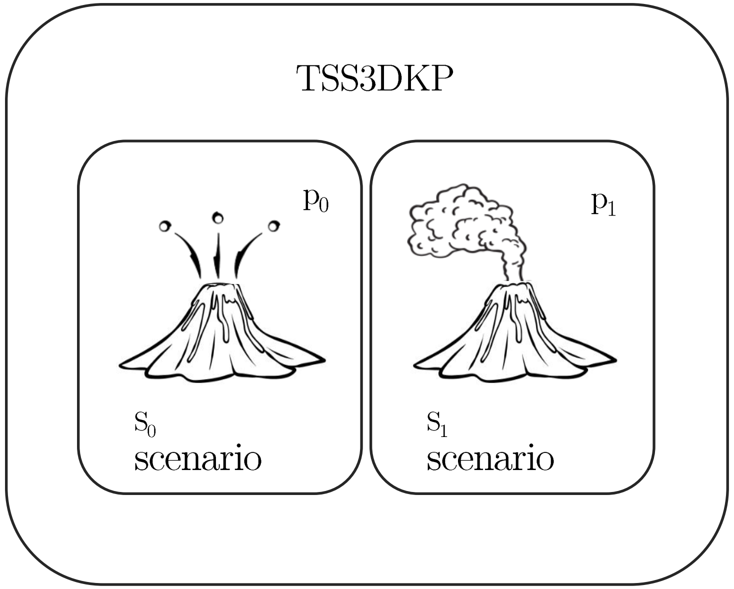
[0] Birge, J.R., Louveaux, F., 2011. Introduction to Stochastic Programming. Springern Science & Business Media.

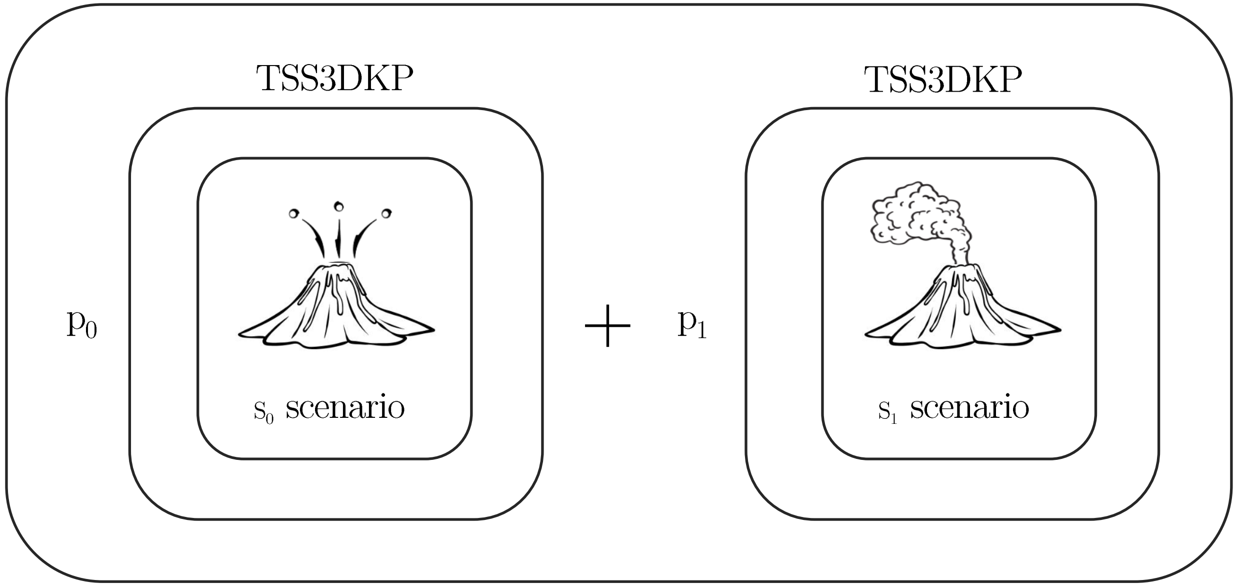
Computational results – EVPI

$$\text{EVPI} = \frac{\text{WS - TSS3DKP}}{\text{TSS3DKP}}$$

$$\text{WS} = \sum_{s \in S} q_s \max \text{Q}(\boldsymbol{a_s}, s)$$

a_s is an independent packing decision for each scenario $s \in S$





Computational results – EVPI

EVPI (printers)	EVPI (no printers)
10.07%	19.48%

Table 3: *EVPI with the instance set N100D100S50 with the possibility of packing printers and without.*

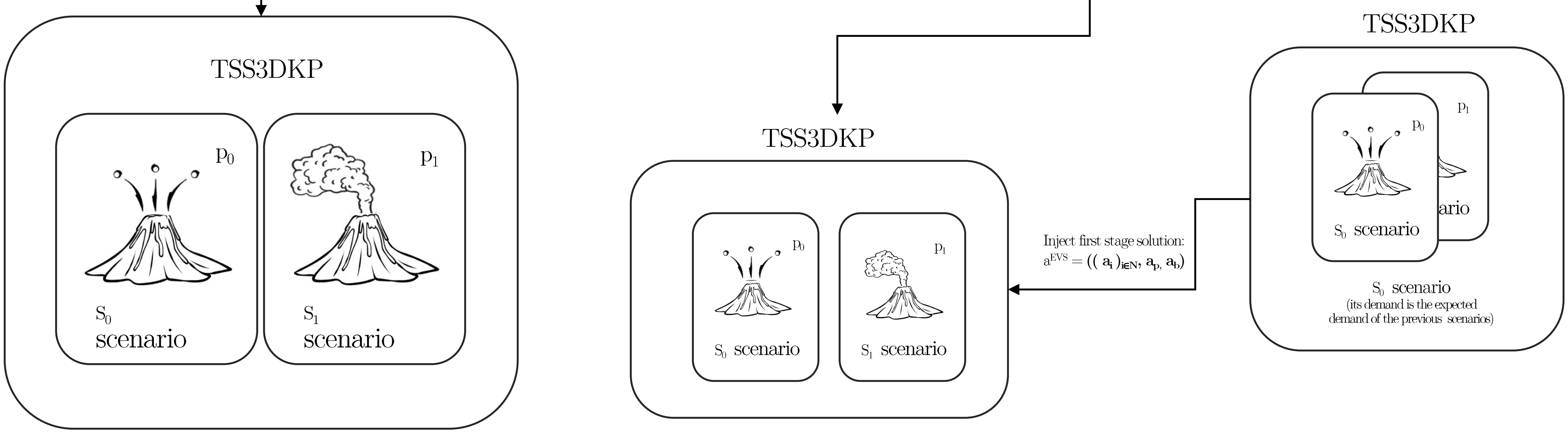
- The EVPI without 3D-printers is 19.48%, i.e., we are willing to pay up to 19.48% of the profit to gain perfect information.
- When we allow for packing 3D-printers the EVPI decreases to 10.07%. This indicates that packing 3D-printers reduces the need for perfect information.

Computational results – VSS

$$\mathrm{VSS} = \frac{\mathrm{TSS3DKP} - \mathrm{EEVS}}{\mathrm{EEVS}}$$

$$\mathrm{EEVS} = \max \sum_{s \in S} q_s \, \mathrm{Q}(\boldsymbol{a}^{\mathrm{EVS}}, s)$$

a^{EVS} is the solution where the demand scenarios for each item are replaced by their expected demand. It is sufficient to solve the TSS-3DKP with only one "scenario" where the demand of each item is its expected demand.



Computational results – VSS

VSS (printers)	VSS (no printers)
5.87%	11.97%

Table 3: *VSS with the instance set N100D100S50 with the possibility of packing printers and without.*

- The VSS without 3D-printers is 11.97%. Hence, our two-stage stochastic programming approach increases the profit by 11.97%. compared to the EEVS.
- When we allow for packing 3D-printers the VSS decreases to 5.87%: there is less need to take into account the uncertainly when we allow to pack 3D printers.

To print or not to print?

To print or not to print?

Should we bring 3D-printers? What is the added value of using 3D-printers?

We will determine, per instance, the percentage increase in total reward if bringing 3Dprinters is allowed. In other words, we calculate:

$$\text{Reward} = \frac{\text{ILP-3DKP(Z)}-\text{ILP-3DKP(0)} \times 100\%,}{\text{ILP-3DKP(0)}}$$

where Z is the upper bound on the number of 3D-printers that can be packed.

We will take into account 6 different aspects:

- ~~Quality of printed item~~
- ~~Printer weight and volume~~
- ~~Storage efficiency of material~~
- ~~Printing time~~
- Demand uncertainly
- ~~Knapsack classes~~

To print or not to print? The effect of demand uncertainty

- It has been studied the N100DXS50 with D ∈ $\{2^0, 2^1, 2^2, \ldots, 2^{17}\}$ and $W = V = 100,000$ and $T = 4000$.

D	Median (3D Printers)	Min-Max(average)(3D Printers)	Median (Reward %)	Min-Max(average)(Reward %)
1	1	0 - 1 (0.9)	0.0	0 - 0 (0.0)
2	1	0 - 1 (0.9)	0.0	0 - 0 (0.0)
4	1	0 - 1 (0.9)	0.0	0 - 0 (0.0)
8	1	0 - 1 (0.9)	0.0	0 - 0 (0.0)
16	0	0 - 1 (0.5)	0.0	0 - 1 (0.0)
32	1	1 - 1 (1.0)	5.0	1 - 16 (5.6)
64	1	1 - 1 (1.0)	16.0	7 - 25 (15.6)
128	1	1 - 3 (1.4)	23.0	13 - 37 (23.6)
256	2	1 - 3 (2.1)	25.5	13 - 45 (26.1)
512	3	2 - 4 (2.7)	22.0	10 - 43 (23.2)
1024	3	1 - 5 (3.0)	17.5	5 - 39 (18.8)
2048	3	1 - 7 (3.3)	13.0	4 - 40 (14.8)
4096	3	1 - 7 (3.2)	9.5	0 - 43 (11.6)
8192	3	0 - 8 (2.8)	6.0	0 - 56 (8.9)
16384	2	0 - 10 (2.4)	2.0	0 - 78 (7.1)
32768	0	0 - 10 (1.5)	0.0	0 - 69 (5.6)
65536	0	0 - 11 (1.2)	0.0	0 - 79 (3.8)
131072	0	0 - 14 (1.0)	0.0	0 - 101 (3.6)

Table 8: *An overview of the median, minimum, maximum and average number of both the packed 3D-printers, and the percentage increase in reward, for varying values of D*

Other analyses

Other analyses – Printable items (weights)

|N_p| %	Median (Reward %)
10.0	0.0
20.0	3.4
30.0	7.3
40.0	11.9
50.0	17.1
60.0	21.6
70.0	26.1
80.0	30.7
90.0	36.0
100.0	41.5

Table 11: *Reward for different values of N_p with item choosen in accord to inverse weight distribution.*

|N_p| %	Median (Reward %)
10.0	2.0
20.0	7.2
30.0	13.2
40.0	18.9
50.0	23.6
60.0	28.2
70.0	32.6
80.0	35.5
90.0	40.7
100.0	41.5

Table 12: *Reward for different values of N_p with item choosen in accord to weight distribution.*

- It has been studied the N100D100S50 for different values of N_p.
- Table 11: lightweight items are more likely to be printable.
- Table 12: heavyweight items are more likely to be printable.

Other analyses – Printable items

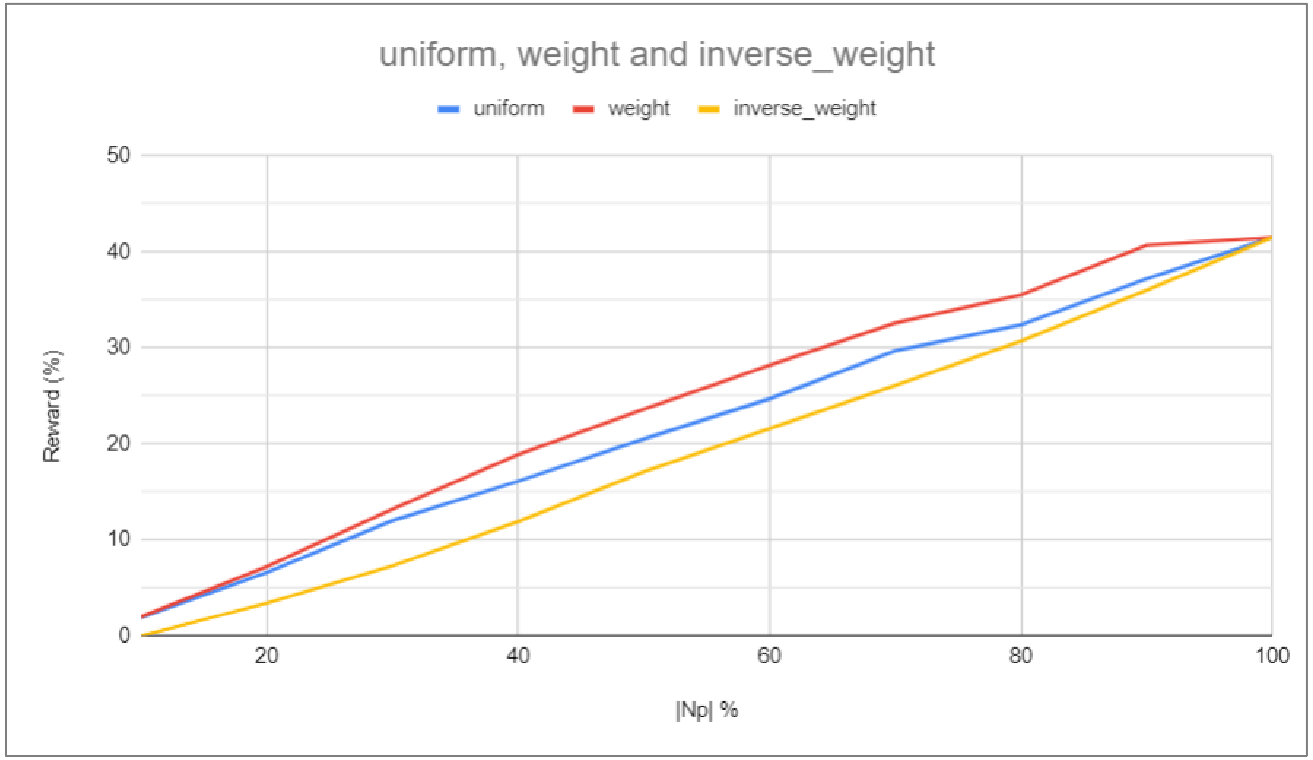


Figure 1: *Reward value for varying |Np| as percentage of N.*

- Printable heavyweight items seems to be more helpful in fighting the demand uncertainty then printable lightweight items.

Scalability analyses

Scalability analyses – NXD100S50 (20 instances) - Number of items

N	Max Time (s)	Min Time (s)	Avg (s)	# Cuts	# Nodes
100.0	6.0	1.7	2.8	1253.5	1.0
200.0	13.3	3.2	5.9	1433.5	1.0
300.0	23.9	5.6	10.1	1410.7	1.0
400.0	23.5	8.6	12.2	756.0	1.0
500.0	31.4	11.7	18.2	1067.8	1.0
600.0	49.8	15.3	24.1	532.5	1.0
700.0	77.2	22.0	29.8	474.8	1.0
800.0	229.5	24.3	41.6	402.8	1.0
900.0	53.2	31.1	39.1	0.0	1.0
1000.0	298.9	32.0	64.7	464.1	1.0
1100.0	91.3	33.0	57.5	$\boxed{0.0}$	1.0
1200.0	86.0	39.5	68.1	$\boxed{0.0}$	1.0
1300.0	142.0	48.3	74.2	274.9	1.0
1400.0	122.7	59.3	83.4	$\boxed{0.0}$	1.0
1500.0	134.5	62.9	93.2	$\boxed{0.0}$	1.0

Table 13: *Comparison of the ILP-3DKP(Z) method for different values of N*

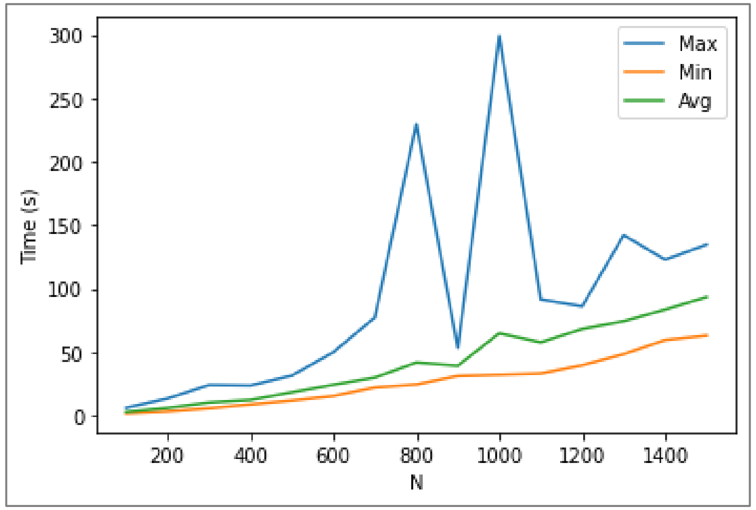


Figure 2: *Trends of min, avg and max execution time for different values of N*

- The relative GAP used for the Gurobi solver is the same of the other analyses (0.001). With the relative gap default value (0.0001) on N1500D100S50 we have an average of 5825 cutting planes and 3.41 minutes on 20 instances.

Scalability analyses – N100D100SX (20 instances) - Number of scenarios

S	Max Time (s)	Min Time (s)	Avg (s)	# Cuts	# Nodes
100.0	3.3	1.6	2.5	1222.8	1.0
200.0	12.1	3.3	6.2	1643.3	1.0
300.0	37.1	5.2	13.9	1907.5	1.0
400.0	88.8	8.2	28.3	2564.7	1.0
500.0	261.5	20.0	53.2	3147.3	1.2
600.0	288.1	16.4	100.5	4393.7	4.0
700.0	409.9	25.3	125.7	4660.3	3.8
800.0	1001.4	23.7	213.0	5428.4	3.4
900.0	598.3	34.5	183.5	4393.6	6.0
1000.0	756.1	31.8	230.9	5264.6	4.2

Table 14: *Comparison of the ILP-3DKP(Z) method for different values of S*

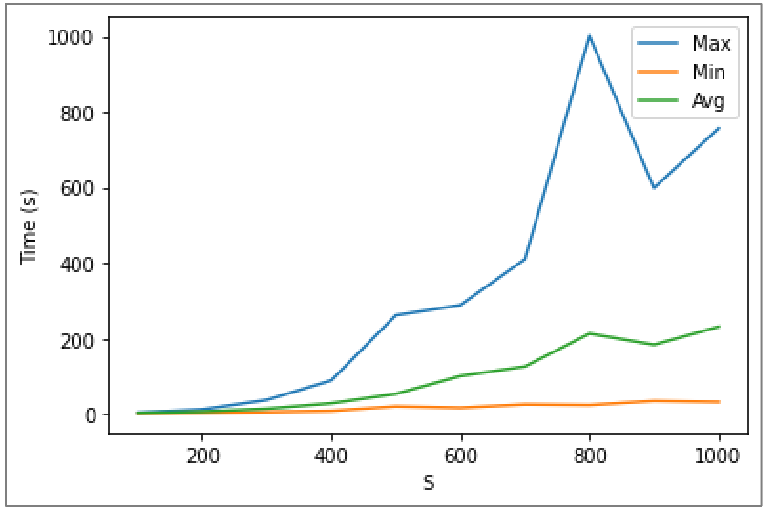


Figure 3: *Trends of min, avg and max execution time for different values of S*

- The relative GAP used for the Gurobi solver is the same of the other analyses (0.001).
- The increment of S is more expensive (with respect to computation time) then the increment of N.
- We have a positive trend on the number of cuts since this time the order of magnitude of the optimal value of objective function does not change (differently from the previous case)

Scalability analyses – N200D100SX (20 instances) - Number of scenarios

S	Max Time	Min Time	Avg	# Cuts	# Nodes
100.0	10.7	3.2	5.6	1198.3	1.0
200.0	48.4	6.6	16.6	2534.2	1.0
300.0	269.1	14.3	63.7	3402.3	3.1
400.0	542.1	23.6	177.6	5207.4	7.9
500.0	952.3	29.3	243.8	6011.4	3.4
600.0	1421.5	29.9	268.0	5436.9	2.0
700.0	715.3	58.2	263.0	5972.6	2.3
800.0	1188.8	66.7	364.1	6621.6	4.3
900.0	$\boxed{4434.9}$	133.0	733.4	10427.2	2.8
1000.0	2843.2	157.4	$\boxed{777.6}$	10899.4	2.9

Table 15: *Comparison of the ILP-3DKP(Z) method for different values of S*

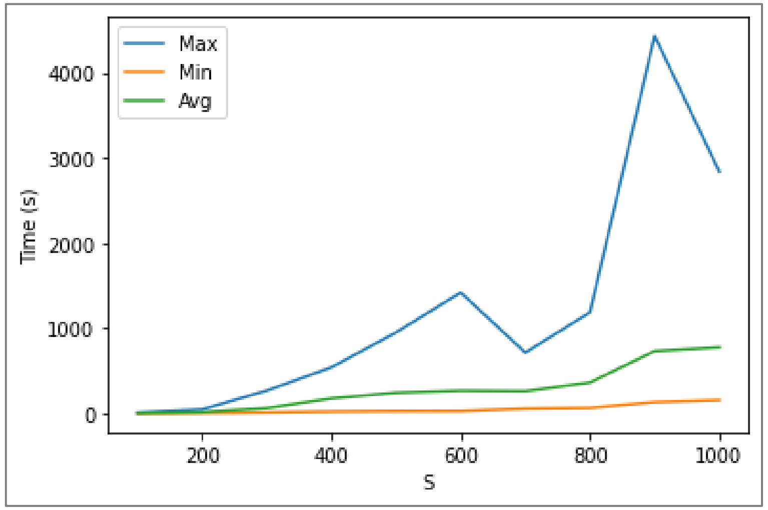


Figure 4: *Trends of min, avg and max execution time for different values of S*

- With N=200 and S=1000 we have an average execution time of 13 min.
- With N=200 and S=900 the execution with the maximum execution time has registered 1h 15min.
- The total execution time of the 20x10 instances required more than 16 hours.

Thank you