A library for solving multiobjective multiple-choice knapsack problems

Dmitry Podkopaev
podkop@ibspan.waw.pl

System Research Institute, Polish Academy of Sciences

14.12.2020

Reproducibility in science

A result should be presented in such a way that other researchers could reproduce it.

In OR, we publish methods and results of computational experiments. In order to use a method and verify results, one needs to have a software implementation.

Theoretically, it is enough to publish a clear description of a method and results.

In practice, the easier for other researchers to use our methods, the better for us:

- facilitate reviewers' work;
- encourage other researchers to use our work.

The crisis of reproducibility

...as of 2020, an ongoing methodological crisis in which it has been found that many scientific studies are difficult or impossible to replicate or reproduce.[1]

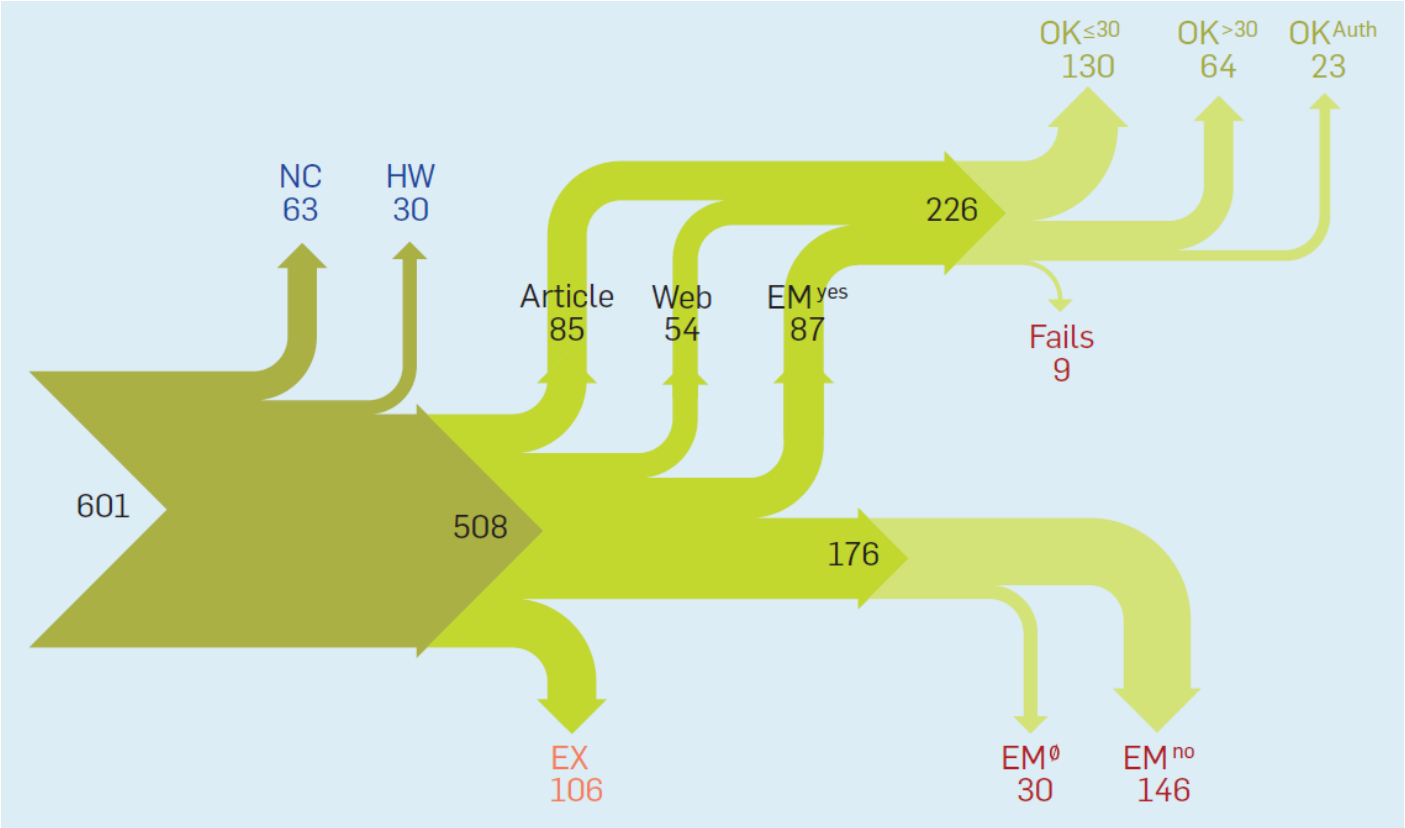
Current state of "reproducible research crisis" in CS is discussed[2]

[1] `https://en.wikipedia.org/wiki/Replication_crisis`

[2] **Cacho J. R. F., Taghva, K.** The state of reproducible research in computer science. *In: Latifi S. (Ed.) Proceedings of ITNG 2020, Springer.*

The crisis of reproducibility

Practical examination of 601 papers published in ACM[3]:



[3] **Collberg C., Proebsting T. A.** Repeatability in computer systems research. *Communications of the ACM, 2016, 59, 62–69.*

Strategy

While doing research, maintain the source code so that it will be ready for publication and deployment.

How much extra time should be spent?

Multiobjective multiple-choice knapsack problem

There are m classes. In each class i, there are $n(i)$ items. For each class, choose one item.

Binary variables in the 2D-array: m rows, each i-th row has $n(i)$ elements. Special case: $n(i) = n$, matrix $m \times n$.

$$\text{Minimize } f_l(\mathbf{x}) := \sum_{i=1}^{m} \sum_{j=1}^{n(i)} c_{ij}^l x_{ij},\ l = 1, \ldots, k,$$

$$x_{ij} \in \{0, 1\},$$

$$\sum_{j=1}^{n(i)} x_{ij} = 1,\ i = 1, \ldots, m,$$

$\langle$possible additional constraints$\rangle$.

Research projects using MO MC knapsack problems

Forest landscape planning.
Jointly with the University of Jyväskylä, Finland.

Classes: forest stands ($m \approx$ tens of thousands).

Objects of a class i: management regimes that can be applied to the i-th stand ($n(i) \approx$ from 3 to 7).

Objective functions: maximize the amounts of biodiversity services, such as timber production, nature conservation, carbon sequestration, etc.

Research projects using MO MC knapsack problems

File management in mass storage systems.
Together with P. Juszczuk, I. Kaliszewski, J. Miroforidis for a Polish IT company (by NCBR).

Classes: files to be stored ($m \approx$ hundreds of thousands).

Objects of a class i: storage devices where the i-th file can be stored ($n \approx$ up to few tens).

Objective functions: maximize performance characteristics, minimize cost characteristics.

Solution approaches

To solve the problem means to find an efficient (Pareto optimal) solution $\mathbf{x}^*$, which is *most preferred* for the DM in terms of $(f_1(\mathbf{x}^*), \ldots, f_k(\mathbf{x}^*))$

Procedures to be implemented:

- Derive the ideal and nadir objective vectors – individual minima and individual maxima over the Pareto optimal set.
- Solve scalarized problems – (parametric) weighted sum, (parametric) Chebyshev metric, lexicographic.

General information

- Programmed in Python 3.
- Calls external MILP solvers for solving scalarized problems:
  a. create the model as a solver-related object;
  b. modify the model to obtain a specific scalarization;
  c. solve the problem;
  d. go to b.
- Uses only common libraries, such as Numpy, and solver-specific libraries;
  - currently supports only CBC and Gurobi, but can be easily extended due to the modular structure.
- Freely available, published in Github
  `https://github.com/podkop/MO-MC-knapsack`

The issue of model modification speed

$$[\text{ Minimize } y_1, y_2, \ldots, y_k \text{ }]$$

$$y_l \geq f_l(\mathbf{x}) := \sum_{i=1}^{m} \sum_{j=1}^{n(i)} c_{ij}^l x_{ij}, \ l = 1, \ldots, k,$$

$$x_{ij} \in \{0, 1\}, \ \sum_{j=1}^{n(i)} x_{ij} = 1, \ i = 1, \ldots, m,$$

$$\langle \text{possible additional constraints} \rangle.$$

Weighted sum scalarization:

$$\text{Minimize } a_1 y_1 + a_2 y_2 + \cdots + a_k y_k$$

The issue of model modification speed

$$[\ \text{Minimize } y_1, y_2, \ldots, y_k\ ]$$

$$y_l \geq f_l(\mathbf{x}) := \sum_{i=1}^{m} \sum_{j=1}^{n(i)} c_{ij}^l x_{ij},\ l = 1, \ldots, k,$$

$$x_{ij} \in \{0, 1\},\ \sum_{j=1}^{n(i)} x_{ij} = 1,\ i = 1, \ldots, m,$$

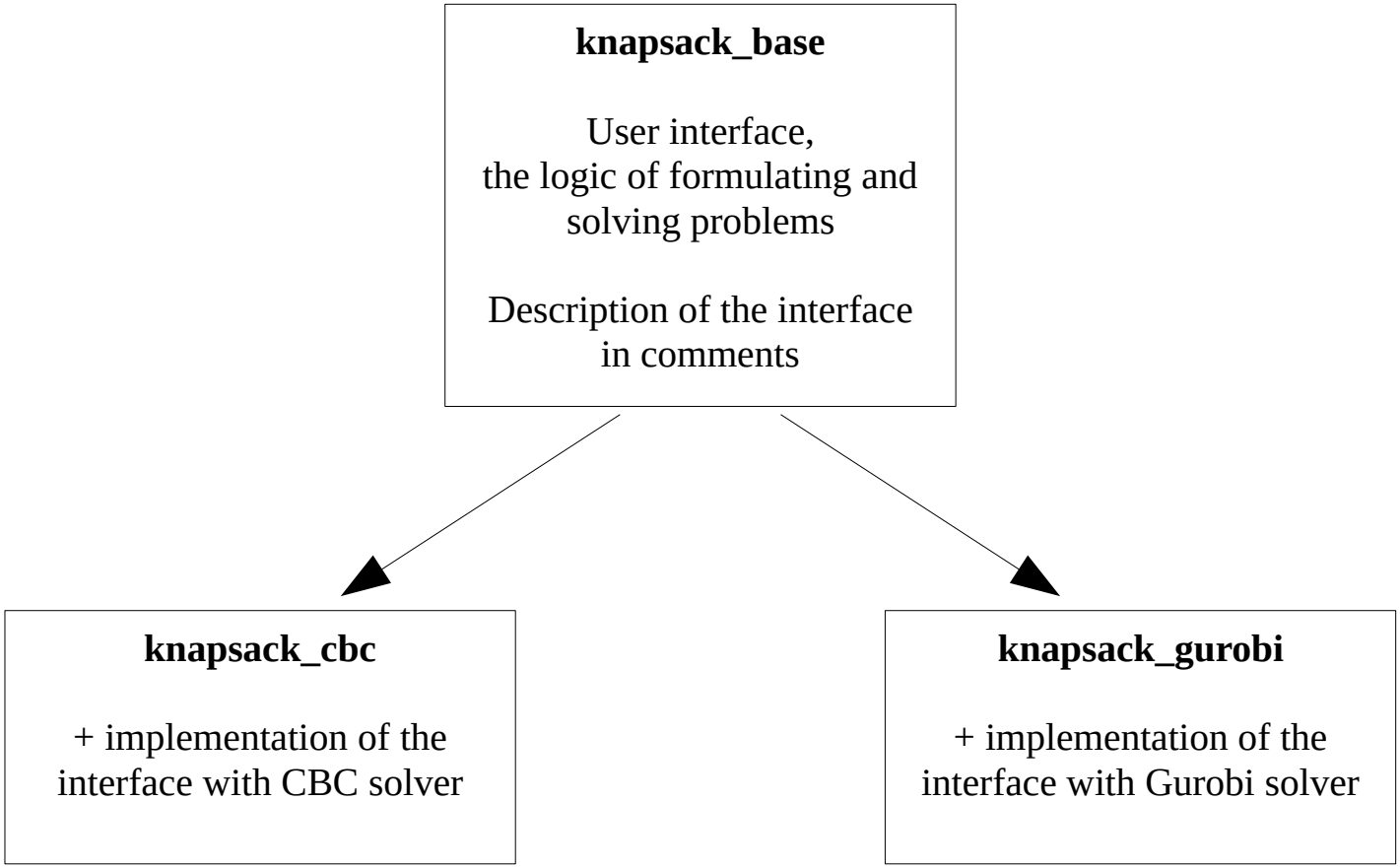
$$\langle \text{possible additional constraints} \rangle.$$

Chebyshev scalarization:

$$\text{Minimize } z + \rho(y_1 + y_2 + \cdots + y_k)$$

$$z \geq \lambda_l \left(y_l - z_l^{\text{ref}} \right),\ l = 1, \ldots, k.$$

Structure of classes



Conclusion

Python is a good choice for scientists:

- easy to learn (except few counter-intuitive features);
- extensive set of packages covering many fields of science;
- integration with libraries in other languages such as R, C/C++, etc.
- open source – free to use.

Modern software tools and IT ecosystems boost the productivity, and have low entry threshold:

- Anaconda – a free Python distribution for Scientists;
- Github ecosystem – for publishing code and collaboration;
- various online forums with active communities.

Conclusion

A considerable amount of time should be spent on learning modern instruments and good practices. However, this investment pays off with higher work efficiency.