Open-source Automated Test Assembly: the Challenges of Large-sized Models

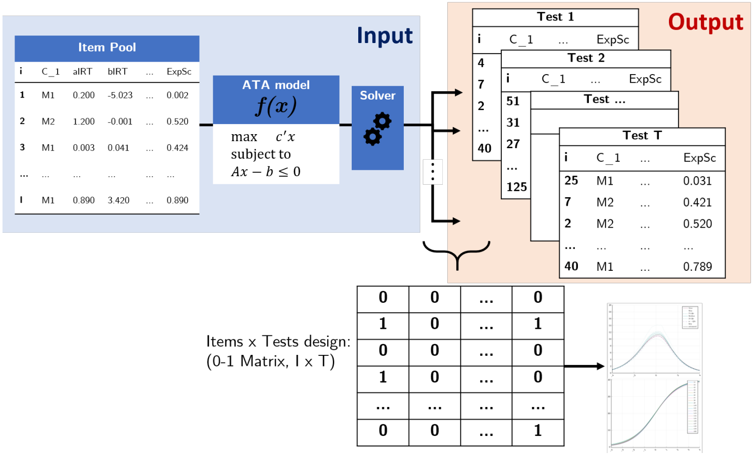
L'assemblaggio dei test automatizzato e a sorgente aperto: le sfide dei modelli di grandi dimensioni

Giada Spaccapanico Proietti

Dipartimento di Scienze Statistiche P. Fortunati

Outline

A depiction of the problem



Item Pool

i	C_1	aIRT	bIRT	...	ExpSc
1	M1	0.200	-5.023	...	0.002
2	M2	1.200	-0.001	...	0.520
3	M1	0.003	0.041	...	0.424
...	...	...	...	...	...
I	M1	0.890	3.420	...	0.890

ATA model

$f(x)$

$$\begin{aligned} &\max \quad c'x \\ &\text{subject to} \\ &Ax - b \le 0 \end{aligned}$$

Test 1

i	C_1	...	ExpSc
4			
7			
2			
...			
40			

Test 2

i	C_1	...	ExpSc
51			
31			
27			
...			
125			

Test T

i	C_1	...	ExpSc
25	M1	...	0.031
7	M2	...	0.421
2	M2	...	0.520
...	...	...	...
40	M1	...	0.789

Items x Tests design:
(0-1 Matrix, I x T)

0	0	...	0
1	0	...	1
0	0	...	0
1	0	...	0
...	...	...	...
0	0	...	1

The linear model

Examples of classical ATA models in: van der Linden [1], Yan, von Davier, and Lewis [2]

Given a set of I items (item bank) calibrated through an IRT model, an automated test assembly (ATA) model for assembling T tests with **minimum expected θ estimation error** can be represented by a linear mixed-integer optimization (MILP) model of the following kind:

$$\begin{array}{lll} \text{maximise} & \sum_{i=1}^{I} IIF_i(\theta_{j_t}) x_{it} \quad \forall t, j_t & \text{(objectives)} \\ \text{subject to} & \sum_{i=1}^{I} a_{ic} x_{it} - b_{tc} \leq 0 \quad \forall t, c & \text{(constraints)}, \\ x_{it} \in \{0,1\} \text{ .(binary optimization variables)} & & \end{array} \tag{1}$$

where $IIF_i(\theta_{j_t})$ is the estimated item information function (IIF) at certain ability, θ_{j_t}.
A model is **feasible** if it has at least a solution $\boldsymbol{x}_t^*$ which satisfies the entire set of constraints. A solution is **optimal** if it achieves the maximum theoretical objective.

MAXIMIN ATA model

If $T > 1$ (multiple test assembly), the model becomes a multi-objective model. Since MILP solvers is able to handle only one objective, the **maximin principle is applied**. Thus, the minimum across tests and ability points θ_{j_t} is maximized using the following model:

$$\begin{aligned}
&\text{max} && w \\
&\text{subject to} && \sum_{i=1}^{I} IIF_i(\theta_{j_t}) x_{it} \geq w \quad \forall t, j_t \\
& && \text{(other constraints)} \\
&x_{it} \in \{0,1\}\,. &&
\end{aligned} \tag{2}$$

Challenges of large-sized models 1/2

Model size (length of x) increases with:

- number of items in the bank *(banks supporting large-scale assessments are usually very large, $I > 300$)*
- *number of test forms to assemble (in testing, security is a primary concern, $T > 10$)*
- auxiliary variables coming from the linearization of non-linear constraints or objectives *(overlap constraints add $I\binom{T}{2}$ decision variables)*

$\Rightarrow$ **High computational complexity since x has length $IT + I\binom{T}{2}$. Model not solvable in reasonable time.**

Challenges of large-sized models 2/2

Infeasibility:

- several constraints to ensure test validity (domain distribution), fairness (parallel tests), and security (overlap, item use, etc...).
- incompatibility between two or more constraints (irreducible infeasible sets, IIS; Chinneck [3] and Huitzing, Veldkamp, and Verschoor [4])
- restrictive lower and/or upper bounds of constraints

$\Rightarrow$ **Intricated interactions between constraints, difficult to disentangle. Tests are not produced.**

Open-source solvers

A **solver** is a software which takes an optimization model as an input and tries to find the feasible and optimal solutions. Most of open-source software (`xxIRT`, `eatATA`, `mstR`, `ATA.jl`) wrap open-source solvers, such as `cbc`, `GLPK`, and `lpSolve`.

ADVANTAGES

- Open-source solvers are **free**. Commercial licenses are expensive (`CPLEX`, `Gurobi`).
- **Open science** requires open-source software.

DISADVANTAGES

- Slow, suitable for **small models** with few constraints.

Solutions

Thus, in order to use open-source solvers, the model must be kept as small as possible and infeasibilities must be detected prior to the solving phase.

In literature:

1. Huitzing, Veldkamp, and Verschoor [4] and Huitzing [5] provide modified ATA models which try to identify the IISs using the theory of infeasibility analysis and minimize the weighted total violation of the soft constraints.
2. Spaccapanico Proietti, Matteucci, and Mignani [6] propose two sequential heuristics, called additive and subtractive unraveling strategies, inspired by on-field experience and the stepwise regression procedure.

Additive and Subtractive strategies

1. Create a list of constraints sorted in descending order of priority.
2. Prepare backup plans for the soft (relaxable) constraints.

Additive

3. It starts by loading only essential constraints in the model.
4. Optimize the model.
5. If the model is feasible, the following constraint is added. While, if infeasibility happens, the backup plans are implemented $\rightarrow$ 4
6. It stops when the model is infeasible and constraints cannot be relaxed further or the model is feasible and all the constraints have been added.

Subtractive

3. It starts by loading all constraints in the model.
4. Optimize the model.
5. As long as the model is infeasible and the list of constraints is not empty, the back-up plans are implemented on the least important constraint $\rightarrow$ 4
6. It stops as soon as the model becomes feasible.

Issues

- **Additive**: the deletion of the constraint which creates the infeasibility does not allow to identify incompatibilities with the previously added constraints (higher priority).
- **Subtractive**: if, after relaxing or removing some constraint, the model is feasible, the algorithm is stopped rather than adding the other requirements with lower priorities again. This implies that it is likely that the final model will not meet most of the listed requirements, even if it would be feasible to meet those requirements.

A **mixed strategy** would improve the two approaches in:

- **detecting conflicts between constraints**
- **detecting item bank deficiencies**
- **providing the solution which satisfies most of the constraints**
- **removing the arbitrariness (one approach instead of two)**

Mixed strategy

1. Create a **list of constraints** sorted in descending order of priority.
2. Prepare **backup plans** for the soft (relaxable) constraints.
3. It starts by applying the **additive** strategy.
   - If the solver cannot find a solution (infeasibility) at the end of an additive step, the subtractive algorithm is implemented. So, the last constraint added, A, is relaxed. If A has reached its most relaxed version and the model is still infeasible $\rightarrow$ 4
   - If the model is feasible, the algorithm continues with the additive mechanism. $\rightarrow$ 3
4. The **repair phase** starts using the **subtractive** approach, and the previously added constraints with priority higher than A, are relaxed.:
   - If relaxing a further constraint B, the model is feasible, B is kept as relaxed, and the subsequent constraints with lower priority are restored using the **additive** technique (note that A is restored in its relaxed version). $\rightarrow$ 3
   - If the feasibility cannot be reached before arriving to the essential model, A must be deleted from the list of constraints since it is incompatible with the other more significant requirements. $\rightarrow$ 3

An application - Data and Specifications

A MAXIMIN ATA model is solved using 325 science items coming from the Trends in International Mathematics and Science Study (TIMSS) item bank. In particular, the items have been calibrated following a unidimensional 3PL model using the dichotomous responses of the 2015 and 2019 survey data on Italian 8th grade students. The items are grouped in friend sets and they are categorized by *content* and *cognitive domain*, *item type* and *cycle* in which they have been created.

Specifications:

- $T = 14$ test forms with 45 items.
- The TIFs are maximized at $\theta = 0$.
- Constraint, in order of priority (s = security , cv = content validity):
    - I. (s) the items must be in at most 2 different tests.
    - II. (s) the majority of items in a test must be created in the last 2 assessments (cycles 6 and 7),
    - III. (cv) test forms must have the same distribution of content and cognitive domains
    - IV. (cv) between 30 and 40 multiple choice items
    - V. (cv) at least 3 items for each combination of content/cognitive domain.
    - VI. (s) maximum overlap between test forms is 15 items.

The **backup plan** for the item use is to relax the upper bound to 3 for specific problematic items. For the other constraints, we allow a relaxation of the lower and upper bounds of -1 and $+1$, respectively.

An application - Results

We choose the software `ATA.jl` and the `cbc` solver with a time limit of 500 seconds[1].

The results obtained by applying the **mixed algorithm** reveal that, given all the previously imposed specifications (I-II), the requirements over the content domains (III) presents internal conflicts. In particular, the constraints on *physics* items disagree with the bounds enforced on the *earth-science* items.

Moreover, the deficiency of *reasoning* items required to increase their maximum item use, possibly raising the items circulation and consequently compromising the security of the test.

Furthermore, the relaxation on the item use contributed to a 11.5% increment of the minimum TIF across the tests revealing that this deficiency caused a substantial worsening in the measurement precision of the tests.

Finally, giving to the maximum overlap (VI) constraint the lowest priority, allowed to keep the model small dramatically decreasing the solving time. Also, it is not needed to be added to the model since the latest solution already satisfies the requirement.

[1] Code, detailed specifications and results are available at https://github.com/giadasp/SIS2021.

Bibliography I

- W. J. van der Linden, *Linear Models for Optimal Test Design*. New York: Springer, 2005.
- D. Yan, A. von Davier, and C. Lewis, *Computerized Multistage Testing : Theory and Applications*. Chapman and Hall/CRC, 2016.
- J. W. Chinneck, "Finding a useful subset of constraints for analysis in an infeasible linear program," *INFORMS Journal on Computing*, vol. 9, no. 2, pp. 164–174, 1997. DOI: `10.1287/ijoc.9.2.164`.
- H. A. Huitzing, B. P. Veldkamp, and A. J. Verschoor, "Infeasibility in automated test assembly models: A comparison study of different methods," *Journal of Educational Measurement*, vol. 42, no. 3, pp. 223–243, 2005.

Bibliography II

- H. A. Huitzing, “An interactive method to solve infeasibility in linear programming test assembling models,” *Journal of Educational Measurement*, vol. 41, no. 2, pp. 175–192, 2004. DOI: 10.1111/j.1745-3984.2004.tb01113.x.
- G. Spaccapanico Proietti, M. Matteucci, and S. Mignani, “Automated test assembly for large-scale standardized assessments: Practical issues and possible solutions,” *Psych*, vol. 2, no. 4, pp. 315–337, 2020, ISSN: 2624-8611. DOI: 10.3390/psych2040024. [Online]. Available: https://www.mdpi.com/2624-8611/2/4/24.