Infeasibility in automatic test assembly

Huitzing, Hiddo Arnold

CHAPTER 7: CONCLUSION

7.1 Summary

In Automatic Test Assembly (ATA), tests are assembled by selecting items from an item bank where the items are stored with their characteristics, such as word count, Item Response Theory (IRT) parameter estimates, content, etc. With Linear Programming Test Assembly (LPTA) techniques optimal tests can be assembled. A problem arises when the mathematical model has no solution. It is then said to be infeasible. Reasons for infeasibility can be a typing error, a logical error, a wrong transcription of a stated demand into mathematical constraints, or an item bank that cannot fulfill the demands. When the mathematical model is infeasible, then no solution is presented, leaving the test assembler empty-handed.

Infeasibility in LPTA models can be caused by a variety of sources, which can be difficult to track. Sometimes one is not really interested in the reasons for the infeasibility, but a fast remedy is needed. This is especially true if time is a factor, e.g., in Computer Adaptive Testing. Some violations of the original demands are then unavoidable, causing a loss of quality, and a test assembler will want to minimize these, preferably in a smart way. We will refer to this as a short-term strategy.

If there is more time, a test assembler might wish to investigate the causes of infeasibility and, if possible, repair the defects in the model or shortcomings in the item bank. Analyzing the causes of infeasibility caused by model shortcomings will be considered to be a short-term to medium-term strategy, while looking into item bank deficiencies, and possibly repairing these, is thought to be long-term planning.

Chapters 1 and 2 and Appendix 1 are introduction chapters to Test Assembly and Linear Programming Test Assembly in which also infeasibility in ATA is discussed. Chapters 3, 4 and 5 treat short-term approaches and medium—term approaches to infeasibility in LPTA models, while Chapter 6 presents a long-term strategy for the solution / prevention of infeasibility. Appendix 2 discusses the software program NuzLight, specifically programmed for this thesis, with some guidelines on utility and usability of software packages. Here we will give summaries of Chapters 3, 4, 5 and 6.

Ch. 3: Using the Irreducible Inconsistent Sets of Constraints

This chapter offers a heuristic, called the IIS-Solver, which is able to repair an infeasible LPTA model by use of Irreducible Inconsistent Sets of constraints (IISs), powerful analyzing tools for infeasibility. By searching IISs, it is possible to indicate constraints which are crucial to make the LPTA model feasible.

By means of the Deletion Algorithm, an IIS is found and one of its constraints is modified to make this subset of constraints (i.e., the IIS) feasible. Several ways are presented to make a choice on which constraint to adapt. A loss function that calculates a percentage of violation is introduced, giving the test assemblers a decision criterion. Once an IIS is fixed (i.e., the set of constraints is made feasible), but the model is still infeasible, the IIS-Solver will continue to search for another IIS, repeating this procedure until the whole modified set of constraints is made feasible. The feasible set of constraints can then be solved with a LP solver using the original objective function.

The IIS-Solver is a short-term to medium-term approach to infeasibility in LPTA in the sense that it modifies the constraint bounds to force a solution. The IIS-Solver can both be used as a stand-alone procedure (as a short-term strategy) or interactively (as a medium-term strategy). Drawbacks of this heuristic are also studied in this chapter, such as the computer time needed for the deletion algorithm to find an IIS. Another weak point is the heuristic nature of the IIS-Solver, which does not guarantee a fast or best solution (e.g., in terms of number of constraints violated or in terms of total percentage violation). However, in the numerical examples, this theoretical problem does not seem to play a role of importance. Finally, the IIS-Solver is compared to a Weighted Goal Programming (WGP) model solved with a heuristic. In a WGP model all the constraint bounds are allowed to be violated, but with a penalty. By giving weights to the constraints (where a larger weight represents a more important constraint bound to be met, i.e., not to be violated), it can be more or less be controlled which constraints are violated. In an objective function the weighted sum of the deviations is minimized. Both the WGP model solved with a heuristic and the IIS-Solver have their merits. The first method is fast and usually gives acceptable solutions, but the IIS-Solver has been really developed with the purpose of forcing a solution in the case of infeasibility with a lowest cost possible (although a global minimum of the cost cannot be guaranteed, this being only a heuristic). We see that, in terms of total

percentage violations (i.e., by how much percent a constraint bound is exceeded, summed over all constraints), the IIS-Solver does much better.

Ch. 4: Set Covering with Item Sampling and Infeasibility Analysis

As argued in Chapter 3, finding IISs can take much computer time. The Set Covering with Item Sampling (SCIS) procedure provides an alternative for the medium term. By sampling sets of items, representing the tests from the item bank, and checking these against the constraints of the LPTA model, one can see which constraints are never satisfied and which are easy to satisfy. Moreover, by means of a set-covering model, tentative IISs can also be identified. Because of the fact that these tests are sampled and will usually not cover all the possible combinations, such IISs in the sampled item sets are not necessarily IISs in the original problem, but they can suggest "real" IISs and are indeed helpful in pointing to the constraints that are most influential in causing the infeasibility of the LPTA model.

In Chapter 4 it also shown that one can use item sampling to search for indications of Maximum Feasible Subsets (MFSs) of constraints or Minimum Cardinality IIS Set Covers (MCISCs). A MFS in an infeasible LPTA model is a largest set of constraints that together is still feasible. There can be several MFSs which differ in one or more constraints, but have the same number of constraints. Its complement is the Minimum Cardinality Set Cover of IISs (MSISC), which is a smallest set of constraints that has to be deleted from the infeasible LPTA model so that the remaining set of constraints is a feasible set. A MFS can then be regarded as the set of constraints that can all be set as hard (i.e., they are not allowed to be violated) while only the constraints of the MCISC are set as soft constraints (i.e., they can be violated if need be). This presents an advantage in terms of the number of soft constraints and more security for the test. Setting all constraints as hard is not helpful in analyzing an infeasible LPTA model but one would like to restrict the set of soft constraints as much as possible. In a large numerical example these methods are then further explored.

Ch.5: A Case Study of Short to Medium-Term Strategies for Infeasibility

Here the methods introduced in Chapters 3 and 4 are applied to two real cases coming from the Cito Group in The Netherlands (National Institute for Educational Measurement). The methods researched are the IIS-Solver of Chapter 3, the SCIS of Chapter 4, and a WGP model solved either with a Greedy Heuristic, or with a LP solver. For the LP solver two situations are distinguished. In the first situation only one objective function is used, which is to minimize the total unweighted sum of violations, and in the second situation a second objective function is added, which is to maximize the test information function (the original objective function). Furthermore, two versions of the Deletion Algorithm (DA) are used. The Relaxed Ordered Deletion Algorithm (RODA) is a version of the DA programmed in CPLEX (Ilog, 2001) to detect an IIS, but uses the relaxed LPTA model (i.e., the binary variables are relaxed and can take on any real value between 0 and 1). The Integer Randomized Deletion Algorithm (IRDA) is also capable of finding an IIS using the binary LPTA model.

Two types of methods are distinguished: analyzing methods, which have as function to find the causes of infeasibility; and forcing methods, which have as primary goal to offer a solution to the infeasible model by forcing (i.e., violating) some of the constraints. In each of the two cases from the CITO group, two causes of infeasibility are introduced and it is checked whether the methods are able to detect those causes and how they deal with the situation. The goal of this chapter is to see how well the newly developed short-term and medium-term methods perform in practice. Methods using a combination of an analyzing method and a forcing method, such as the IIS-Solver of Chapter 3, gave very good results in terms of a high objective function value and a small number of constraints violated. However, there appeared to be a trade-off between a good result and forcing a solution.

Ch. 6: Adding Hypothetical Items to the Item Bank to Prevent Infeasibility

A long-term strategy to prevent infeasibility in LPTA is explored in this chapter by focusing on item bank deficiency, i.e., infeasibility caused by the shortcomings of the item bank. It is argued that in the long run the main issue is not to act upon the symptoms of the problem which appears in the form of infeasible LPTA models, but upon the real causes of infeasibility, which is (in almost all cases) a deficient item bank.

The Item Bank Extension Model works as follows. Hypothetical items are added to the item bank and the (previously infeasible) LPTA model. If the model has become feasible, we have an idea of which type of items might be added to the item bank to prevent infeasibility. The difficulty is how to define these hypothetical items, i.e., which characteristics they must have.

A first suggestion is to have the hypothetical items resemble the real items as much as possible, without being actual copies. Therefore of each real item, to be called original item, an imitation item is constructed, which has the same numerical characteristics (e.g., word count, IRT parameter estimates, etc.), but a different content (e.g., while the original content is a question on Napoleon, the imitation is a question on Louis XIV). The reason to have the imitation items resemble the original items is that we know that such items have been written in the past and are likely to be written again. In a second step the item characteristics are allowed to vary within certain bounds. A numerical example concludes the chapter and also shows some of the problems encountered. Special attention should be paid to how the item characteristics in the IBE model can vary to be of practical value. For example, if the solution of the IBE model is to add several items with only one or two words to the item bank, to solve the infeasibility problem, such a solution is unrealistic.