Measurement and Research Department Reports 2006-2

Automated Assembly of Testsets: Fit in all Seasons

Angela J. Verschoor

Measurement and Research Department Reports 2006-2

Automated Assembly of Testsets: Fit in all Seasons

Angela J. Verschoor

Cito
Arnhem, 2006

Abstract

A model for the assembly of testsets is introduced. This model concerns the construction of several tests from a shared item pool, whereby each test may have its own specifications and either a limited overlap is allowed or no overlap at all.

In cases where a limited overlap is allowed, either a non-linear formulation must be chosen, or a linearization with many extra dummy variables and restrictions. Therefore, a Genetic Algorithm to solve these models is developed. A frequently occurring complication is epistasis, and several strategies to prevent this are evaluated.

1 Introduction

Since Theunissen (1985) showed how to apply mathematical programming methods in test assembly by formulating a target test information defined at a number of discrete ability points, this topic has gained both theoretical as well as practical attention. Testing agencies try to reduce costs when producing their test forms by constructing item banks, from which items are selected. Usually, no single test forms are produced from item banks, but testsets, consisting of various test forms designed for several purposes. A model for the assembly of parallel test forms, in the sense that test characteristics such as test information function (TIF) are identical, is reported by Boekkooi-Timminga (1990). A related paper on a heuristic approach is Ackerman (1989), while Armstrong, Jones and Wu (1992) concentrated on the assembly of tests similar to an existing seed test using a network flow model. Sanders and Verschoor (1998) took a somewhat different approach by using a greedy heuristic to minimize the distance between the item parameters in the different test forms.

2 Test Assembly in Item Response Theory

A key notion in item response theory (IRT) is that of a latent scale on which both the item difficulties and the test takers' abilities are defined. An IRT model defines the probability on an item score obtained by test takers as a function of the latent ability. A widely used IRT model is the two parameter logistic (2PL) model for dichotomous items, with score 0 for a wrong answer and 1 for a correct answer. The 2PL model assumes the probability on a correct answer for item i and a candidate with ability ϑ to be

$$P(X_i = 1|\vartheta) = \frac{\exp(a_i(\vartheta - b_i))}{1 + \exp(a_i(\vartheta - b_i))} \tag{1}$$

where a_i is referred to as the discrimination parameter and b_i as the difficulty parameter of item i. A usual way to build item banks from which tests are assembled, consists of two phases. First, items are constructed and, through pretests, data are gathered to estimate the item parameters. In the next phase, the items are selected for the tests while the item parameters are considered to be known without any error. Purpose of this selection is the assembly of a set of tests with minimal error of measurement, against minimal effort as expressed in, for example, test length.

A statistical property of an item is Fisher information, also called the item information function, which for the 2PL model (Hambleton and Swaminathan, 1985) is given by

$$I_i(\vartheta) = \frac{a_i^2 \exp(a_i(\vartheta - b_i))}{(1 + \exp(a_i(\vartheta - b_i)))^2}. \tag{2}$$

The variance of the ability estimator $\widehat{\vartheta}$ is asymptotically equal to the inverse of the TIF value evaluated at ϑ. Under the assumption of local independence

between item scores, the TIF is the sum of the information functions of the items in the test:

$$I(\vartheta) = \sum_i I_i(\vartheta). \tag{3}$$

As a general aim of tests is to measure candidates abilities as accurately as possible, the variance of $\widehat{\vartheta}$ should be minimized and, hence, the TIF should be maximized. Note that not only the TIF has a relation with the error of measurement, also the difficulties of the items has a relation with the TIF. As the item information function reaches its maximum at $\vartheta = b_i$, the joint difficulties of the items control the shape of the TIF.

Models for parallel test assembly can be described in a rather straightforward way: Assuming an item pool of size L, assemble J tests with the same test characteristics but with no items in common. Boekkooi-Timminga (1990) was the first to formulate such a model, using decision variables x_{ij} to indicate whether item i is selected for test j. The model formulated by van der Linden and Adema (1998), which allowed other restrictions in their model as well, is the basis of the PARIMAX model formulated below. Let variables x_{ij} denote the decision variables indicating whether an item is selected in test j or not. I_{ik} is the item information in ability point ϑ_k, while T_{kj} is defined as the target information for test j at this point. Note that formulating the TIF target for each test separately gives the possibility to specify different characteristics for each test. Coefficients q_{in} are the resource parameters of item i, indicating how many resources are needed to use it in the test, c_{im} the classification parameters having value 1 if item i belongs to category m and 0 otherwise. Q_{nj}, C^{ℓ}_{mj} and C^{u}_{mj} are the desired use of resources and number of items in the classification categories for test j, respectively:

$$\text{maximize} \quad y \tag{4}$$

$$\text{subject to:} \quad y \leq \frac{\sum_i I_{ik} x_{ij}}{T_{kj}} \qquad \forall k, j \tag{5}$$

$$\sum_i q_{in} x_{ij} \leq Q_{nj} \qquad \forall n, j \tag{6}$$

$$C^{\ell}_{mj} \leq \sum_i c_{im} x_{ij} \leq C^{u}_{mj} \qquad \forall m, j \tag{7}$$

$$p_r(x) = 1 \qquad \forall r, j \tag{8}$$

$$\sum_j x_{ij} \leq 1 \qquad \forall i \tag{9}$$

$$x_{ij} = \begin{cases} 1, & \text{item } i \text{ in test } j \\ 0, & \text{else} \end{cases} \qquad \forall i, j. \tag{10}$$

The PARIMAX model formulated in (4) - (10) expresses the wish of the test assembler to produce a series of tests that maximize the TIF at selected ability levels, given desired shapes in the TIF targets in (5), and subject to various restrictions. The restrictions in (6) put a limit on the use of certain resources.

As can be derived from the additivity of the item information functions, the easiest way to maximize the TIF is to use all items in the pool. So, the resource restrictions in (6) limit the resources that the tests use, for example, test length or the total time allotted for test taking. In addition, the restrictions in (7) define a desired taxonomic makeup of the tests. Items are classified, for example, with respect to content domains or behavioral aspects. For all these categories minima and maxima are specified. A third group of restrictions in (8) concerns the item level. When building large item pools, it is almost inevitable that some items form a relation with others. These relations are referred to as interitem relations. Common examples of these item level relations are enemy sets and testlets. In enemy sets, one item may give away a clue to the answer on the other items, clearly an unwanted situation. Testlets are structures within the item pool that contain several items that should stay together. An example of a testlet is a reading passage with a group of test questions. Theunissen (1996) has shown that more complex relations exist frequently. These can be formulated as well, using Boolean operators $\vee$, $\wedge$ and $\neg$. These relations can be transformed into restrictions based on differential payoff functions, denoted by $p_r(x)$, as proposed by De Jong and Spears (1989). A last group of restrictions in (9) is added in order to stipulate the requirement of non-overlap, that is, an item is allowed to be used in no more than one test.

The purpose of the model is to maximize the information in the tests at those ability points for which the ratio between the TIF and its target is minimal. Thus, the total information of each test is maximized while adhering to the preferred TIF shapes as much as possible.

These models can be large for programs such as the NIVOR testing program for Dutch as a Second Language, produced by Cito (de Jong, 1998). For each of the four language skills Reading, Writing, Listening and Speaking comprehension, testsets had to be developed for four levels of mastery. The first test was a placement test to assign an incoming learner to a course level. Then the progress of the learners was monitored during the courses by a number of tests. Finally, a certification test concluded the various courses. The placement tests were relatively short and had a broad and flat TIF in order to determine the appropriate course level. The monitoring tests had increasing difficulty to reflect the growing proficiency during the courses. These were short tests with narrow TIFs. The certification tests aimed at deciding whether the candidate had passed or failed the appropriate mastery level as accurately as possible. For this purpose, for each skill and each level an item bank was built with sizes ranging from 400 to 600 items, from which the tests were assembled. In those cases, standard integer linear programming techniques should be applied carefully.

In order to control calculation times, Armstrong (1992) proposed a heuristic for the assembly of parallel test forms. As a first step, a group of items is selected into a seeding test. In practical situations, this might be an existing test. Thereafter, parallel forms are created by minimizing the distance between the item parameters of the new forms and the parameters of the seed test. The problem is modelled as a network flow problem. This approach, however, might impose rather strict demands on the item bank from which the items are drawn:

Numerous items must be present, identical both in classification as well as in psychometric parameters. A more flexible approach in which items are selected in such a way that their parameters do not have to be identical, but in which the overall test characteristics are, would be welcome. Moreover, incorporation of other types of restrictions is generally not possible with network flow models. On the other hand, Verschoor (2004) has shown that genetic algorithms (GAs) can be applied successfully to test assembly problems while still retaining a large degree of flexibility.

The mapping between decision variables x_{ij} and the chromosomes in a GA is straightforward: Each variable forms a gene with alphabet $\{0,1\}$. Furthermore, Verschoor (2004) has found that a population of 100 individuals, mate selection proportional to fitness, uniform crossover, and a survival scheme in which the best unique individuals survive, is an efficient combination to solve the class of test assembly models. The fitness function is comprised of the objective function and a penalty function related to the restrictions, in such a way that only positive values are possible:

$$f(x) = \frac{y}{1+g(x)} = \frac{\min\limits_{k,j}\left\{\frac{\sum_i I_{ik}x_{ij}}{T_{kj}}\right\}}{1+g(x)} \tag{11}$$

$$\begin{aligned} g(x) = {} & \lambda \sum_{n,j} h\left(\sum\nolimits_i q_{in}x_{ij} - Q_{nj}\right) \\ & + \mu \sum_{m,j} h\left(C^{\ell}_{mj} - \sum\nolimits_i c_{im}x_{ij}\right) + \mu \sum_{m,j} h\left(\sum\nolimits_i c_{im}x_{ij} - C^{u}_{mj}\right) \\ & + \varphi \sum_{r,j} 1 - p_r(x) + \xi \sum_i h\left(\sum\nolimits_j x_{ij} - 1\right) \end{aligned}$$

$$h(u) = \begin{cases} u, & u > 0 \\ 0, & u \leq 0 \end{cases}.$$

Coefficients λ, μ, φ and ξ denote the penalty multipliers that are determined dynamically similar to the strategy of Siedlecki and Sklanski (1989). Consider for τ consecutive iterations the individuals with the highest fitness. If for all these τ individuals all resource restrictions are met, multiply λ by $1-\delta$. If for all individuals some resource restrictions are violated, multiply λ by $1+\varepsilon$. Leave λ unchanged in all other cases, that is, that for some individuals all resource restrictions are met while for other individuals there are violated restrictions. The same rule is followed for the other penalty multipliers μ, φ and ξ.

Variations to the PARIMAX model can easily be formulated in cases, for example, where a limited overlap is allowed. Define test overlap V_{jl} as the maximum number of items to be included simultaneously in tests j and l. Next

to the test overlap, the item exposure must be controlled to prevent items from appearing in too many tests. Define W_i as the maximum number of tests that item i is allowed to appear in. Replace the restrictions in (9) by

$$\sum_i x_{ij}x_{il} \leq V_{jl} \qquad \forall j, l \qquad (12)$$

$$\sum_j x_{ij} \leq W_i \qquad \forall i. \qquad (13)$$

Penalty function $g(x)$ is modified in order to reflect the incorporation of the overlap and exposure restrictions. Note that the restrictions in (12) are nonlinear, but can be linearized by the introduction of a dummy variable z_{ijl} for each combination of item i and pair of test forms j and l, and related restrictions:

$$\sum_i z_{ijl} \leq V_{jl} \qquad \forall j, l \qquad (14)$$

$$z_{ijl} \geq x_{ij} + x_{il} - 1 \qquad \forall i, j, l. \qquad (15)$$

This, however, would result in the growth of the complexity of the model to such an extent that for sizeable item pools it is questionable that problems based on this linearization can be solved within reasonable time limits.

3 Compact Coding

As can be seen above, using the conventional model formulations implies the introduction of many variables as well as restrictions on overlap control. This causes a significant increase in problem size compared to single test assembly. Various approaches have been studied in other fields to investigate the efficiency of alternative representations (Hornsby and Pollack, 2001; Rothlauf and Goldberg, 2003; Toussaint, 2005), leaving no firm conclusion whether alternative representations improve the performance. If certain restrictions are made redundant, the performance might be improved.

If no overlap between the tests should be allowed at all, there is an alternative problem formulation that uses a special representation scheme for the chromosomes. Define a coding scheme that maps gene x_i to the test assembly model:

$$x_i = \begin{cases} 0, \text{ if item } i \text{ is not selected} \\ 1, \text{ if item } i \text{ is selected in test 1} \\ 2, \text{ if item } i \text{ is selected in test 2} \\ 3, \text{ etc.} \end{cases}$$

Now, the chromosome length is equal to the size of the item pool, one gene mapping onto a single item and vice versa, while a mapping directly based on the decision variables x_{ij} would cause the chromosome length to be the number

of items times the number of test forms to be assembled. On the other hand, the alphabet should accommodate all possible decisions regarding an item: values 1, ..., J to designate the test form for which the item is selected, or 0 in case the item is not selected at all. In this way, overlap is not possible and the corresponding restrictions in (9) are redundant. The other restrictions in (4) - (8) remain unchanged.

Even though it may be expected that a GA based on compact coding would consume considerably more time than for a similar single test assembly problem using a comparably large item pool, compact coding might deliver results faster than traditional coding. Especially in the early phase of the optimization process, compact coding might have a distinct advantage over traditional coding as the majority of restrictions have become redundant.

4 Epistasis

Since the fitness is based upon y, and thus on the combination of ϑ_k and test j for which y is minimal, it will be no surprise that the fitness function is epistatic if no precautions are taken. All feasible solutions that have the same minimum test in common have the same fitness. The individual with the highest fitness, and thus with the highest minimum test, will have the greatest probability to procreate and to survive. Within a few iterations all individuals will be based upon the same minimum test having only differences in the other tests. From that iteration on, all individuals have the same fitness. With all differences located in the non-minimum tests, a search direction for improvement has been lost, and the population has converged to a local maximum.

An obvious way to reduce the epistasis is to allow small differences in fitness. This way, every solution will have a unique fitness and the search direction will be restored. These differences should have some meaningful value in order to propagate the optimization process.

Consider two different feasible solutions with equal fitness. These two candidate testsets share the same minimum test, while differences are located in (at least one of) the non-minimum tests. It is easy to see that, given these two testsets, it is easier to improve the minimum test, and hence the fitness, and retain feasibility by an item migrating from a high non-minimum test to the minimum test than by migrating from a low non-minimum test. Even if there is no direct reason to do so, the fitness function should be based on at least one non-minimum test in order to favour higher non-minimum tests.

Two strategies can be considered to overcome the epistasis:

- The use of a reward scheme that involves the non-minimum tests in the fitness. For the PARIMAX model, this reward scheme could be devised as

$$f(x) = \frac{y + \gamma \sum_j y_j}{1 + g(x)}$$

whereby restrictions in (5) are replaced by

$$y \leq y_j \qquad \forall j$$
$$y_j \leq \frac{\sum_i I_{ik} x_{ij}}{T_{kj}} \qquad \forall k, j.$$

Solutions that have high non-minimum tests will get a higher reward than those with relatively low non-minimum tests. The purpose of a reward scheme is to favour those individuals that have a higher chance to produce offspring with more favorable objective function values. When the non-minimum tests have a high TIF, it will be easier for crossover and mutation to establish an exchange of items so that the minimum test is improved, than when the non-minimum tests have a low TIF. The reward, however, should be chosen carefully. A small increase in the TIF of the minimum test, and therefore in objective, should be more profitable than an increase in TIF of the non-minimal tests.

- The use of several alternating fitness functions in successive iterations. A cycle of iterations is formed in which all fitness functions are evaluated sequentially. If the cycle is very short, only individuals that do well according to all fitness functions will survive. This approach can be regarded as the introduction of seasons, analogous to seasons in biology. Individuals that do well in all seasons have a larger chance of survival than individuals that thrive in one season but encounter problems in the other. For multiple test assembly, a cycle of two seasons can be used. In the odd iterations, the original fitness function as described in (11) is evaluated while in the even iterations a fitness function is evaluated that also takes the non-minimum tests into account. Define this alternative fitness as

$$f^*(x) = \frac{\sum_j y_j}{1 + g(x)}.$$

In effect, individuals that have a high TIF for the minimum test as well have a high information function for the other tests will be favoured. This approach has the advantage of being more robust than the reward scheme, since too high a reward could interfere with the optimization process. The disadvantage, however, is that in half of the iterations a fitness function is evaluated that has only an indirect bearing on the objective function. Therefore, the process might slow down somewhat and more iterations are needed to reach good solutions.

5 Simulations

Simulations have been conducted in order to investigate the effectiveness of the various approaches in preventing epistasis and solving the test assembly problems. Two questions have to be answered: What strategy is most effective

in solving test assembly problems: reward, seasons, or a combination thereof? Second, does compact coding accelerate the optimization process in case of non-overlap?

To answer these questions, three different test assembly problems were used. All three problems were based upon an item pool consisting of 500 items with simulated parameters according to the two-parameter model with $\log(\alpha) \sim N(0, 0.4)$ and $\beta \sim N(0, 1)$. All items were classified on two different dimensions. The first dimension consisted of 4 categories labelled as 101, 102, 103 and 104. These categories were filled with approximately 250, 125, 85 and 40 items, respectively. The second dimension contained categories labelled as 201, ..., 210, which contained approximately equal numbers of items.

The simplest test assembly problem, Problem 1, involved the assembly of two parallel tests. For each test, four restrictions on the shape of the TIF ($\vartheta_1 = -1.5$, $\vartheta_2 = -0.5$, $\vartheta_3 = 0.5$, $\vartheta_4 = 1.5$ with $T_{\vartheta_1} = 4$, $T_{\vartheta_2} = 8$, $T_{\vartheta_3} = 8$, $T_{\vartheta_4} = 4$) and one resource restriction, $\sum_i x_{ij} \leq 40$, were defined. No content restrictions based on the classification structure described above were used and item overlap was not allowed. Problem 2 was an extension of Problem 1, in that it involved assembly of three parallel tests with a similar TIF as in Problem 1 with zero overlap. All tests had 40 classification restrictions. From each combination of two categories, the first from the range 101, ..., 104, and the second from the range 201, ..., 210, exactly one item was required.

Furthermore, the two coding schemes were combined with the four combinations of epistasis prevention strategies, resulting in eight different conditions. The optimization process was stopped after $LK\log(LK)$ iterations, where L is the number of decision variables, and K is the total number of restrictions, excluding the overlap restrictions. Verschoor (2004) has shown that in general for test assembly models this stopping criterion gives satisfactory solutions. Thus, Problem 1 was stopped after 92103 iterations while Problem 2 was stopped after 750592 iterations, after which the best solution found so far was presented. For Problem 1, 400 replications were performed. For Problem 2, this number was 200. In the dynamic penalty adaptation scheme, τ was chosen to be 80, while δ was set equal to 0.03, ε to 0.02, while reward γ was equal to 0.0001.

The average best fitnesses and their standard deviations are presented in Table 1. As all strategies gave feasible solutions in all replications and all fitnesses were based on feasible solutions, it follows that $f(x)T_{\vartheta_k}$ is a lower bound on the TIFs of solution x realized at ϑ_k. Therefore, $\frac{1}{\sqrt{f(x)T_{\vartheta_k}}}$ is an upper bound on the standard error of measurement at ϑ_k. Note that this argument does not hold for the strategies involving a reward: In the best solutions on which the data in Table 1 are based, a total reward in the order of magnitude of $0.001 - 0.002$ was observed, and this reward should be subtracted from the fitness before estimations of the TIFs can be made.

Furthermore, it can be seen that there is no strategy that is clearly best overall. For Problem 1, a relatively small problem, compact coding offered no advantages over binary coding. For Problem 2, a larger problem, compact coding performed better in combination with either the seasons approach or

Table 1: Best Fitness Function Values

	Best Fitness			
	Problem 1		Problem 2	
Strategy	M	SD	M	SD
None-Binary	3.911	0.025	3.105	0.038
None-Compact	3.836	0.050	3.022	0.072
Reward-Binary	3.919	0.018	3.101	0.037
Reward-Compact	3.844	0.054	3.016	0.085
Seasons-Binary	3.914	0.017	3.101	0.022
Seasons-Compact	3.902	0.011	3.123	0.029
Both-Binary	3.869	0.029	3.101	0.021
Both-Compact	3.899	0.011	3.122	0.023

with both seasons and reward scheme.

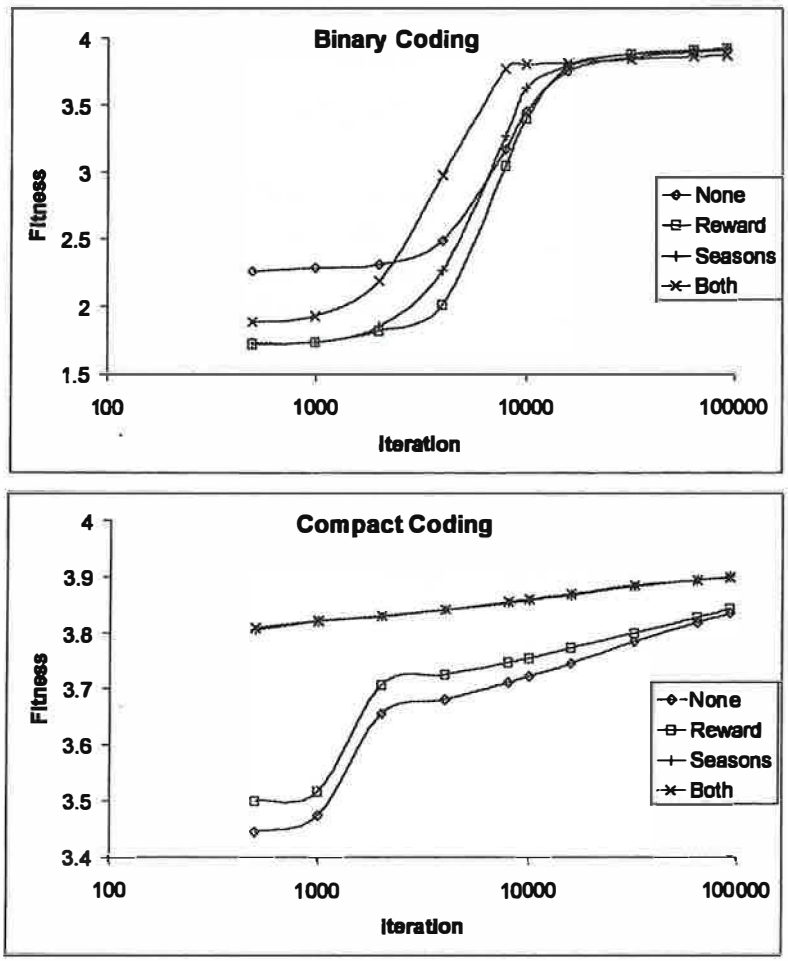

FIGURE 1. Average Best Fitness during Optimization after Reaching Feasibility (Problem 1)

Figure 1 shows the average fitness of the best solution during the iterations. It can be seen that compact coding gave good solutions much faster than binary coding. While in Problem 1 all strategies yielded feasible solutions within 500 iterations, the average fitness for the strategies that involved seasons was over 3.8. For binary coding, these fitness values were reached after 16000 iterations.

Note that for compact coding the performance of the seasons approach coincided almost entirely with the combination of both approaches. Apparently, adding a reward scheme to the seasons approach had no effect for compact coding. It is interesting to see that this was different for binary coding. In the first 10000 iterations, the combination performed better than either single approach, after which the other strategies caught up and eventually performed somewhat better.

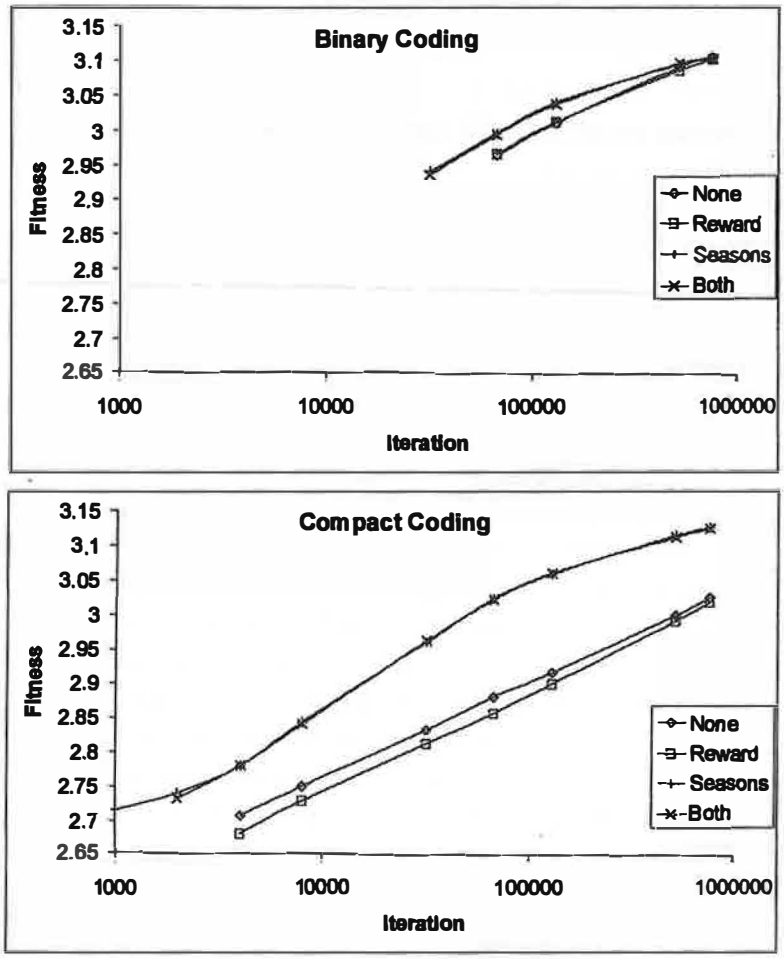

FIGURE 2. Average Best Fitness during Optimization after Reaching Feasibility (Problem 2)

Figure 2 shows that for Problem 2 the difference between binary coding and compact coding was even larger than for Problem 1. The GA with compact coding found feasible solutions within 2000 iterations for the strategies involving seasons, but it lasted 32000 iterations before all replications of the binary coding with seasons found a feasible solution, and it took 64000 iterations without seasons. From that point, however, the best fitnesses that were found for the seasons approach were of the same order of magnitude as those for the compact coding scheme. For the approach without seasons, binary coding produced slightly better solutions than compact coding at this stage. Note that for Problem 2, the results for the seasons approach coincided almost fully with those for the combination of approaches. This could be observed not only for compact coding, as with Problem 1, but also for binary coding. Even more, also the performance for using no scheme coincided with the reward scheme in case of binary coding. In case of compact coding, the reward scheme performed even slightly worse than using no scheme at all.

5.1 A Testset with Overlap

The third problem in the simulation studies concerned the assembly of three tests with different characteristics, similar to the case of the monitoring tests in the NIVOR testing program. All three tests had length 40 and were subjected to the same content restrictions: Category 101 had to be represented by 15 items, category 102 and 103 by 10 items each, and category 104 by 5 items. The tests had an increasing difficulty in order to represent a growth in ability level over the test administrations. For each test, a TIF target was defined at three ability points with $T_{\vartheta_1} = 8$, $T_{\vartheta_2} = 8$, $T_{\vartheta_3} = 4$, while the ability points varied for each test. For test 1 these were: $\vartheta_1 = -1.5$, $\vartheta_2 = -0.5$, and $\vartheta_3 = 0.5$; for test 2: $\vartheta_1 = -1.0$, $\vartheta_2 = 0$, and $\vartheta_3 = 1.0$, and for test 3: $\vartheta_1 = -0.5$, $\vartheta_2 = 0.5$, and $\vartheta_3 = 1.5$. The test overlap was restricted to a maximum of four items and each item was allowed to appear in at maximum two tests.

Since Problem 3 allowed for overlap between the tests, compact notation could not be used. Thus, four epistasis prevention strategies were simulated: the reward scheme, the seasons approach, both, and using no strategy. Similar to Problem 1 and Problem 2, the optimization process was stopped after $LK\log(LK)$ iterations, where L is the number of decision variables, and K is the total number of restrictions, excluding the overlap restrictions. Problem 3 was stopped after 588424 iterations, after which the best solution found so far was reported. For each strategy, 200 replications were performed. In the dynamic penalty adaptation scheme, τ was chosen to be 80 while δ was 0.03 and ε was 0.02.

The average best fitnesses and their standard deviations are presented in Table 2.

Table 2: Best Fitness Function Values for Problem 3

	Best Fitness	
Strategy	M	SD
None-Binary	4.413	0.106
Reward-Binary	4.449	0.116
Seasons-Binary	4.495	0.032
Both-Binary	4.482	0.045

From Figure 3 it can be seen that the strategies did not differ very much in effectiveness. The strategies involving seasons performed somewhat worse during the first 10000 iterations, after which they performed somewhat better than the strategies without seasons.

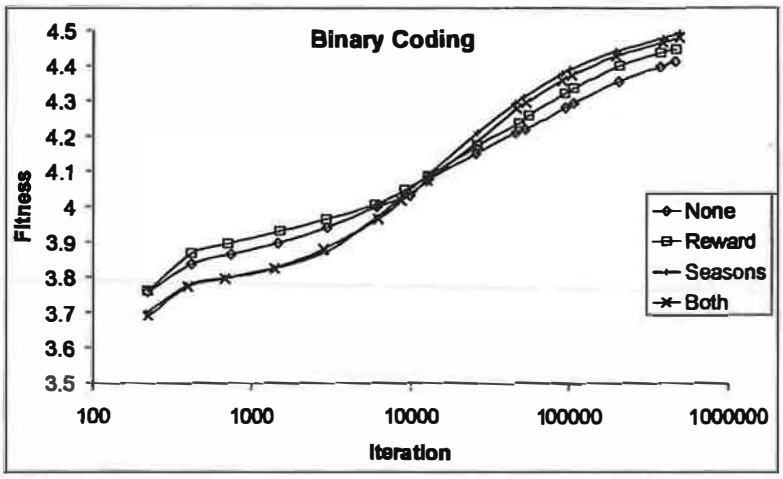

FIGURE 3. Average Best Fitness during Optimization after Reaching Feasibility (Problem 3)

The question which strategy prevents epistasis best for the assembly of testsets cannot be answered simply. If the optimization is stopped after a relatively large number of iterations, as was the case in our simulation study, using a reward scheme seems to be the best choice for relatively simple models like Problem 1. For more complicated models, such as Problem 2, compact coding combined with either the seasons approach or both seasons and reward schemes, seemed to be the favorable choice. In cases where overlap restrictions do not allow the use of compact coding, such as in Problem 3, the seasons approach seemed to perform best.

From a practical point of view, however, the stopping criteria seemed to be rather strict, and it might be preferable to give a reasonably good solution at an earlier moment. When the optimization is stopped at an earlier moment, compact coding combined with seasons or with both the seasons and reward schemes seemed to be the best choice.

6 Conclusions

A model for the assembly of testsets was proposed in this study. The idea behind the model is the observation that in some testing programs, sets of test forms are developed all originating from a common item pool, and where a limited amount of overlap is allowed. The test specifications may vary between test forms, as each may serve a different purpose within the testing program. It is shown that a genetic algorithm is capable of solving these models efficiently. Especially if a limited overlap is allowed, the resulting test assembly models are either nonlinear or need many extra dummy variables and restrictions, which may cause algorithms traditionally used in the field, based upon integer linear programming, to fail.

Should no overlap between tests be allowed, a compact coding using a more

elaborate alphabet improves the performance significantly, especially when combined with an approach that mimics seasons, evaluating different fitness functions in consecutive iterations. Compact coding is an effective method in preventing epistasis inherent to the model formulations of the assembly of testsets.

References

Ackerman, T. (1989). An alternative methodology for creating parallel tests using the IRT-information function. Paper presented at the annual meeting of the National Council on Measurement in Education, San Francisco, CA.

Armstrong, R., Jones, D., and Wu, I. (1992). An automated test development of parallel tests from a seed test. *Psychometrika*, 57:271–288.

Boekkooi-Timminga, E. (1990). The construction of parallel tests for IRT-based item banks. *Journal of Educational Statistics*, 15:129–145.

de Jong, J. (1998). *NIVOR toetsen 1998 handleiding.* Arnhem: Cito. in Dutch; Test Manual.

De Jong, K. and Spears, W. (1989). Using genetic algorithms to solve NP-complete problems. In Schaffer, J., editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 124–132. San Mateo, CA: Morgan Kaufmann.

Hambleton, R. and Swaminathan, H. (1985). *Item Response Theory: Principles and Applications.* Boston: Kluwer-Nijhoff.

Hornsby, G. and Pollack, J. (2001). The advantages of generative grammatical encodings for physical design. In *Proceedings of the 2001 Congress on Evolutionary Computing*, pages 600–607. IEEE Press.

Rothlauf, F. and Goldberg, D. (2003). Redundant representations in evolutionary computation. *Evolutionary Computation*, 11:381–415.

Sanders, P. and Verschoor, A. (1998). Parallel test construction using classical item parameters. *Applied Psychological Measurement*, 22:212–223.

Siedlecki, W. and Sklanski, J. (1989). Constrained genetic optimization via dynamic reward-penalty balancing and its use in pattern recognition. In Schaffer, J., editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 141–150. San Mateo, CA: Morgan Kaufmann.

Theunissen, T. (1985). Binary programming and test design. *Psychometrika*, 50:411–420.

Theunissen, T. (1996). *Combinatorial Issues in Test Construction.* PhD thesis, University of Amsterdam.

Toussaint, M. (2005). Compact genetic codes as a search strategy of evolutionary processes. In Wright, A., Vose, M., De Jong, K., and Schmidt, L., editors, *Foundations of Genetic Algorithms 2005*, pages 75–94. Berlin: Springer.

van der Linden, W. and Adema, J. (1998). Simultaneous assembly of multiple test forms. *Journal of Educational Measurement*, 35:185–198.

Verschoor, A. (2004). IRT test assembly using genetic algorithms. Measurement and Research Department Reports 2004-4, Arnhem: Cito.