Measurement and Research Department Reports 2004-4

IRT Test Assembly Using Genetic Algorithms

Angela J. Verschoor

Cito groep

Measurement and Research Department Reports 2004-4

IRT Test Assembly Using Genetic Algorithms

Angela J. Verschoor

Citogroep
Arnhem, 2004

IRT Test Assembly Using Genetic Algorithms

Angela J. Verschoor
National Institute for Educational Measurement
Cito
Arnhem, The Netherlands

April 5, 2004

Abstract

This paper introduces a new class of optimisation methods in test assembly: Genetic Algorithms (GAs). In the first part an overview is given of the concepts and principles of GAs, in the second part they are applied to three commonly used test assembly models using Item Response Theory. Simulation studies are performed in order to find conditions under which GAs can be successfully used.

1 Introduction

Test assembly has always been an important issue in the field of educational testing. The introduction of modern techniques by Birnbaum (1968) caused a gradual shift from using Classical Test Theory towards using Item Response Theory during test assembly. Lacking computation power item selection was initially done by hand, which made Birnbaum's approach theoretically attractive but little used in real-life situations. Feuerman and Weiss (1973) were the first ones to use Mathematical Programming methods in a somewhat different context, but even then it took more than a decade before Theunissen (1985) brought Birnbaum's ideas into practice. Since then computerised test assembly methods using item selection from an item bank have been gaining an increasing popularity. Various optimisation models have been developed, using different solving techniques: van der Linden and Boekkooi-Timminga (1988, 1989), Adema and van der Linden (1989), Armstrong, Jones and Wu (1992), Luecht and Hirsch (1992), Swanson and Stocking (1993), van der Linden (1996), Sanders and Verschoor (1998).

All test assembly models have several elements in common: *decision variables*, *objective* and *restrictions*. The decision variables determine whether or not items are selected in a test and thus form the heart of the model. The objective is a function of these decision variables expressing a desirable property of the test to be assembled. This function is maximised or minimised, depending on the kind of property it expresses. Examples of objectives are the number of

items in the test or the test measurement error as expressed in the Test Information Function. Typically, the first objective will be minimised, resulting in a test of minimum length, and the second objective will be maximised, resulting in the most reliable test available. The restrictions are the conditions which the test has to comply with. Examples of these restrictions are the test length, content specification or inter item relations such as enemy sets.

Various software packages are available today to apply these methods in practical situations like OTD, (cf. Verschoor, 1991) and ConTEST, (cf. Timminga et al., 1996). All solution methods that are currently developed can be divided in two categories: exact methods and heuristics. An example of an exact method is Branch-and-Bound, which is frequently implemented in commercially available LP-solvers. Heuristic algorithms are usually very fast providing 'good' solutions without proving that the provided solution is optimal. A popular class of heuristics are greedy algorithms in which items are selected sequentially. In a greedy algorithm, an item is never removed from a prospective test once it is selected.

This paper discusses a new class of optimisation methods that has become popular in recent years for its capability to efficiently solve non-linear integer programming models: Genetic Algorithms. By mimicking evolutionary processes as can be found in biology, a population of solutions mate and compete in order to drive the population towards the optimum. Contrary to most other traditional methods they assume no information of the structure of the problem to be solved and are therefore amongst the most versatile optimisation methods available. They can be classified as a group of heuristic local search algorithms that are capable of finding solutions without explicit search direction. The fact that Genetic Algorithms do not find optimal solutions to problems, but merely very good ones, should not be considered a limitation. In most practical situations, the major goal is not finding an optimal solution, but finding a good solution in which all restrictions are met.

In the first part of this paper some general principles of Genetic Algorithms are presented. In the second part a few widely-used test assembly models are solved using a Genetic Algorithm and are compared with solutions obtained from other algorithms.

2 Genetic Algorithms

Genetic Algorithms (GAs) form a family of general purpose search algorithms that are successfully employed to solve optimisation problems, amongst many others. First studied by Holland (1968, 1973, 1975) they are iterative methods inspired by analogy with evolution theory in biology as presented by Darwin (1859). A group of individuals (candidate solutions to the problem), or population, generate offspring and compete for survival. Individuals are represented by strings, called *chromosomes*. The values that the individual string positions, the *genes*, can take is called the *alphabet*. The search starts with a population of randomly selected individuals, and from these the next generation is created by

using operators that represent recombination and competition for survival. At each iteration the chromosomes are evaluated according to a *fitness function*. Based on this fitness function, individuals are selected to create the next generation applying genetic operators like *crossover* and *mutation*. In this process a good individual, one with a high fitness, will have a higher chance of survival and generation of offspring than a not-so-good individual with lower fitness. In this way the fitness function is related to the objective function of the optimisation problem: The more favourable a solution is (higher objective function value for maximisation models, lower objective function value for minimisation models), the higher the fitness function value is. This provides an implicit search direction for the optimisation process, thus 'automatically' steering the population towards the optimum, (cf. Goldberg 1989).

A coding and decoding scheme provides a mapping between the individual and its chromosome. Usually, this mapping is straightforward in binary models. Each decision variable is represented by a gene assuming values of 0 or 1 corresponding with the decision variable value. In other models a more comprehensive mapping is needed using several neighbouring bits to represent a decision variable value.

2.1 Genetic operators

Each iteration of a GA is composed of three steps: mate selection, generation of new individuals and survival. In the first step, the mate selection, individuals are selected into a reproduction pool in order to create new individuals. With a population of N chromosomes, $N/2$ pairs of parents are selected for mating. This process is directed by the fitness function. In the standard GA as described by Holland the selection chance is proportional to the individual's fitness. The probability to be selected for reproduction for an individual $\mathcal{S}$ in population $\Re$ is given by:

$$P(\mathcal{S}) = \frac{f(\mathcal{S})}{\sum_{\mathcal{S} \in \Re} f(\mathcal{S})}$$

where $f(\mathcal{S})$ is the fitness function. In all variants the rule holds that the higher an individual's fitness, the higher its chance to mate. Some individuals may mate multiple times during an iteration, while others might not mate at all.

After mate selection, offspring is generated using the two operators crossover and mutation. With crossover, two individuals mate to generate two new individuals. To this purpose, the chromosomes of the parents are copied onto the chromosome of the offspring, after which a position is chosen at random. The new chromosomes are cut at this position, the second chromosome parts are swapped and reconnected, as shown in Figure 1. This type of crossover is called one-point crossover. Alternative types of crossover can be used as well like the multiple-point crossover. Mutation is a bitwise operation: each bit flips value with chance p_μ. Fogarty (1989) has shown that best results are to be expected with $p_\mu = 1/I$, where I is the number of genes in a chromosome.

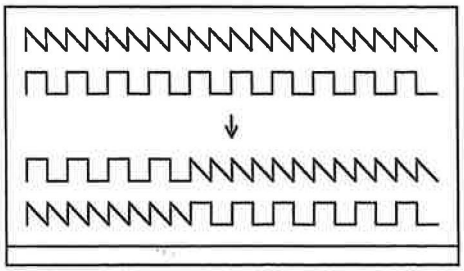

Figure 1: The crossover operator

In the last step, the population is reduced to its original size in order to form the initial population for the next iteration. This reduction process is also directed by the fitness function. Similar to mate selection, an individual with a high fitness has a larger chance for survival than an individual with a low fitness.

The notion that this procedure leads to improvement is given by Holland's Schema Theorem (1973, 1975). The idea behind the Schema Theorem is that individuals representing better solutions have certain things in common. The concept of schemata is an efficient way to describe similarities between individuals at certain genes. Next to the `0` and `1` the symbol `*` is used to indicate indifference in gene value, that is, the schema `1*0` can mean both `100` as well as `110`.

It is clear that all individuals can be referred to by 3^I schemata. Each individual is itself representative of 2^I schemata while the whole population in a particular generation represents at most $N2^I$ schemata. It is obvious that not all schemata are equally specific. For example, the schema `011*1**` is a more specific statement about similarity than `**1****`. Define the *order* $o(\xi)$ of schema ξ as the number of fixed positions of ξ, and the *defining length* $\delta(\xi)$ as the distance between the first and last fixed position. So the order of `1**0` is 2 and the defining length is 3.

Suppose that there are m representatives of schema ξ in the population $\Re_t$ at generation t, written as $m_t(\xi)$. Let $f_t(\xi)$ be the average fitness of all representatives of ξ and let $\overline{f}_t$ be the average fitness of the whole population $\Re_t$. Assuming a GA without crossover or mutation, ξ can be expected to gain representatives in the next generation if $f_t(\xi) > \overline{f}_t$. Selection for reproduction depends on fitness, and in the standard GA it can be expected that there will be $m_t(\xi)f_t(\xi)/\overline{f}_t$ representatives of ξ in the reproduction pool and thus in the next generation $\Re_{t+1}$.

Selection on its own, however, creates no new material. This is done by crossover and mutation. Crossover disrupts schemata with varying probability, depending on their defining length. The smaller the defining length of a schema, the smaller the chance that the randomly selected crossover site falls within it and thus disrupts the schema. The probability that a schema will contain the

crossover site is $\delta(\xi)/(I-1)$. Therefore the upper bound for the probability of ξ losing a representative is $\delta(\xi)/(I-1)$. Please note that the chance that both parents are representatives of ξ is not taken into account here and that ξ will not be disrupted in such a case even if the crossover site falls with ξ. Therefore, $1-\delta(\xi)/(I-1)$ is an upper bound of the probability of survival during the crossover.

Furthermore, every gene has mutation chance of p_μ or a chance of surviving the mutation of $1-p_\mu$. Therefore ξ survives mutation with probability $(1-p_\mu)^{o(\xi)}$. For small p_μ this is approached by $1-o(\xi)p_\mu$. Summarising, the representation of ξ will increase according to the following expression:

$$\mathcal{E}(m_{t+1}(\xi)) \geq \{m_t(\xi)f_t(\xi)/\overline{f}_t\}\{1-\delta(\xi)/(I-1)-o(\xi)p_\mu\}$$

The conclusion is that short, above average and low order schemata receive increasing representation in subsequent generations. These short and low order schemata are usually called building blocks and form the basis of the search direction towards the optimum. Bridges and Goldberg (1987) extended the Schema Theorem by taking into account that the crossover not only disrupts schemata but also has a probability of creating them.

One can typically observe the following behaviour in the representation of a certain schema. In first instance the representation is very small with only individuals with high fitness. The representation then rapidly grows with more highly fit representatives replacing lowly fit non-representatives, but at the same time this causes the average fitness $\overline{f}_t$ to rise. The growth rate decreases until $f_t(\xi) = \overline{f}_t$. From that moment onwards the representatives are gradually replaced by individuals that represent competing schemata with higher fitnesses. The representation declines again and in some cases the schema might even become extinct in the population.

One of the alternatives for the one-point crossover is the uniform crossover as proposed by Syswerda (1989) where every gene is swapped with chance 0.5. The one-point crossover, however, has certain advantages over the uniform crossover in many models. In more intricate mappings several neighbouring genes represent a single decision variable. In those mappings many neighbouring genes have a distinct relationship with each other, thus forming somewhat larger building blocks in the schema theorem in a natural way. The chance of disruption of a schema by uniform crossover is substantially larger than the one-point crossover. Exception to this are schemata of defining length 0, which can never be disrupted. In binary models it is not always the case that neighbouring variables are related. It is to be expected that only very few building blocks with defining length larger than 0 exist anyway. In those cases the uniform crossover has the advantage of reaching unexplored areas of the feasible region faster while still maintaining the necessary diversity in the population of solutions.

2.2 Fitness function

The fitness function and objective function are closely related. During the optimisation process in the Genetic Algorithm the fitness function is maximised and

can be regarded as a utility function. It provides the implicit search direction in the optimisation. By finding a maximum for the fitness function a solution is found with an optimal value for the objective function. The fitness function must be defined in such a way that the following two demands are fulfilled: First, it must be defined for each individual that can be generated. Especially in models with many restrictions this demand requires some precaution. Second, it must reach its maximum at the same point where the objective function reaches its optimal value, cf. Michalewicz (1994).
There are a few options to construct a fitness function in such a way that it is properly defined for any individual that could be generated:

- Rejection of infeasible individuals. This approach is simple, and works reasonably well in cases when the feasible search space is convex and constitutes of a large part of the whole search space. This is only the case in models with just a few restrictions. In more complex models too many individuals are generated that appear to be infeasible and only occasionally an individual is accepted.

- Use of special representations and operators to guarantee feasible solutions. For some models it could be a complicated task to design a special representation such that every chromosome maps into a feasible solution and vice versa. It involves a dedicated approach whereby each individual restriction has to be taken into account and frequently special crossover and mutation operators are needed as well. Achterkamp (1993) took this approach in test assembly. She noticed that in the standard mapping, fixed-length tests are frequently disrupted by crossover and mutation. In order to overcome this phenomenon she devised a special coding that does not state for every item whether or not it is in the test, but that mentions the items in the test explicitly. Chromosomes have a length equal to the test length each gene representing an item in the test. Genes are not binary here, but the alphabet contains all items in the item bank. This approach works very well for simple test assembly models but extension with extra restrictions is very hard to incorporate.

- Use of a repair algorithm as presented by Orvosh and Davis (1993). This algorithm constructs a feasible solution given any individual, but has as a drawback that sometimes many individuals are repaired into the same feasible solution, thus yielding a high chance of degraded performance caused by *epistasis*. Epistasis, as defined by Davidor (1991), is a condition in which some genes do not have any direct influence on the fitness and many chromosomes therefore have the same fitness. In that case the GA lacks a search direction which hinders the optimisation process.

- Use of a penalty function for infeasible solutions. Allow any offspring to be generated, but use a fitness function based on a relaxation of the optimisation model. The major question is how the penalty function should be designed. The intuitive answer is straightforward: the penalty should

be kept as low as possible, just above the limit above which the individual with the maximum fitness is feasible. In order for the process of propagation to converge to a feasible solution, a necessary condition can be given. For all infeasible solutions having directly neighbouring feasible solutions, there must be at least one such neighbour with a higher fitness. Should this not be the case the process might converge further away from the feasible region through such an infeasible solution. The penalties, however, should not be chosen too high. If infeasible solutions are eliminated too quickly, the process might not pass to feasible solutions with higher fitnesses, resulting in a premature convergence. Le Riche et al. (1995) confirm the effectiveness of this so-called *minimal penalty rule*. The use of penalty functions poses another problem: find a penalty function that is robust under a wide variety of optimisation problems as they occur in practical situations. This is why it is difficult to implement this rule. A hypothesis was formulated by Richardson et al. (1989) that provides some help: "Penalties which are functions of the distance from feasibility are better performers than those which are merely functions of the number of violated constraints."

An example of such a penalty function for a linear restriction that is offended, is $\lambda\,(b - \sum_i a_i x_i)$. Coefficient λ is called the penalty multiplier. In many practical cases, a simulation study must be used to determine appropriate values for the penalty multipliers. Furthermore, Siedlecki and Sklanski (1989) suggest that "the genetic algorithm with a variable penalty coefficient outperforms the fixed penalty factor algorithm." This scheme has the following outline: consider k consecutive iterations and their best solutions. If all these solutions appear to be infeasible, raise the penalty multiplier. If all these solutions are feasible, lower the multiplier. In other cases leave the penalty multiplier unchanged.

2.3 Reproduction and survival

In the standard GAs, the probability to reproduce for an individual $\mathcal{S}$ in population $\Re$ is given by:

$$P(\mathcal{S}) = \frac{f(\mathcal{S})}{\sum\limits_{\mathcal{S}\in\Re} f(\mathcal{S})}$$

where $f(\mathcal{S})$ is the fitness function. It is obvious that the relation $\forall\mathcal{S} : f(\mathcal{S}) \geq 0$ must hold in order to define all reproduction chances properly. If this is not the case, either the fitness or the reproduction chances must be redefined. There are several alternative reproduction chance schemes, the most important being *Linear Scaling* and *Rank Based Scaling* (Baker, 1985). In Linear Scaling, a linear transformation $f^*(\mathcal{S}) = a\ f(\mathcal{S}) + b$ is applied. Coefficients a and b are chosen in such a way that the transformed fitness values are within a prespecified range. In Rank Based Scaling all individuals are sorted according to their fitness. The reproduction chance $P(\mathcal{S})$ is assigned to an individual according to its rank

within the sort order. An alternative to these scaling methods is ensuring non-negativity. Especially when using penalty functions, fitness values could easily be negative and thus some attention must be paid to this requirement.

The standard survival strategy is to let the N best individuals survive to the next iteration. A stochastic strategy similar to mate selection strategies is possible as well. The essential difference between the two strategies is *elitism:* the individual with the highest fitness is guaranteed to survive to the next iteration. As De Jong (1975) has pointed out, "elitism improves local search at the expense of global perspective." In other words, elitism accelerates the optimisation at the cost of a higher chance to stop in a local optimum instead of the global optimum.

2.4 Stopping criteria

Preferably, the GA stops when it can be shown that the best individual is within a small bandwidth from the optimum. A detection mechanism, comparable to Lagrangian Relaxation, is absent, so other criteria must be used. The simplest rule is to stop after a fixed number of iterations. Early research concentrated on other criteria somewhat related to the Schema Theorem based on the extent of convergence within the population. Recognising convergence on the level of individual genes is rather straightforward. By its very definition, convergence manifests itself in a large number of converged genes. Convergence of a gene is measured by the largest percentage of the population having the same value. *Bias* is defined as the average convergence of each gene, thus defining an overall extent of convergence. Comparing the *online* performance, that is, the average fitness of all individuals that have been generated, with the *offline* performance, the average fitness of a small number of the best individuals, is a second indication of convergence. Thierens and Goldberg (1994) showed that the convergence can be estimated for sufficiently large populations. The chance that the population contains the optimal solution is asymptotically given by: $P_{opt}(t) = 1 - \left\{1 - (1 - 0.5e^{-t/I})^I\right\}^N$

The Schema Theorem, however, provides insight in the average behaviour of a GA. It states that as time passes building blocks with above average fitness will increase in representation. It does not tell us anything about the individual building blocks in the next generation. The GAs' heuristic nature suggests that it might not be safe to use stopping rules like "stop when no significant improvement is found in n iterations", since the process may be temporarily stuck in a local maximum.

More recent work by Goldberg and Segrest (1987), Eiben, Aarts and van Hee (1991) was done in the field of Markov Chain analysis. Their approach concentrates on the analysis of steady states and conditions under which these exist and contain the optimal solution. Suzuki (1993) derived a lower bound on the probability of the fittest individual being included in the steady state solutions. From here an upper bound on the number of iterations to find the optimal solution with certainty δ can be calculated. Aytug and Koehler (1996) showed that this upper bound is $\ln(1-\delta)/\ln(1-{p_\mu}^{IN})$. Although these analyses

are theoretically very important, evaluating this expression for a given instance shows that they provide us with no practical stopping rule.

3 Test Assembly under Item Response Theory

In recent years, test assembly has been a prominent theme in Item Response Theory (IRT). Several models for test assembly have been formulated, all aimed at various test purposes.

Theunissen (1985) proposed a minimisation model based on IRT. Given the context of the test, for example, a decision to be made based upon an examination, a certain reliability is required. This reliability is usually formulated as a maximum error of measurement at certain ability levels. Instrumental for this is the Test Information Function (TIF). The purpose of the model is to construct a test that exceeds the target TIF at some prespecified ability points against minimal effort. Examples of this effort are test length, administration time or production costs. The function defining this effort forms the objective which is to be minimised. The target TIF forms the basis for the restrictions in the model.

Van der Linden and Boekkooi-Timminga (1989) described a maximin model suitable for situations that are more resource-driven: assemble the best, that is, the most reliable, test given limited resources such as test length or administration time and given a desired shape of the TIF. Here, the objective is formed by the minimum ratio between the realised and target TIF. The limited test resources are formulated in the restrictions.

Swanson and Stocking (1993) presented a model in which infeasibility plays an important role. Their Weighted Deviation Model can be used in situations where no feasible solution for the test assembly model is found. It aims at minimising the distance to the feasible region. Key to this approach is weighing restrictions according to the importance to fulfill them.

Models for multidimensional test assembly were formulated by van der Linden (1996) and Veldkamp (1998). These models concentrate on testing situations in which multiple abilities play a role.

Van der Linden and Luecht (1998) proposed an Observed-Score model also using Item Response Theory. This model is particularly suited for criterion referenced tests or other tests whereby a fixed cut-off score is required at a predetermined ability level.

3.1 IMAX model

The IMAX model presented here is a slight reformulation of the Maximin model introduced by van der Linden and Boekkooi-Timminga (1989). The IMAX model expresses the wish of the test assembler to construct a test that maximises the Test Information Function (TIF), given a desired shape as expressed in the target TIF (1), and subject to quantitative restrictions (2), to classification restrictions (3), and to inter item restrictions (4) or logical restrictions as

proposed by Theunissen (1996):

$$
\begin{array}{lll}
\text{maximise} & y & \\
\text{subject to:} & y \leq \dfrac{\sum_i I_{ik} x_i}{T_k} & \forall k \qquad (1) \\
& \sum_i q_{in} x_i \leq Q_n & \forall n \qquad (2) \\
& C_m^{\ell} \leq \sum_i c_{im} x_i \leq C_m^{u} & \forall m \qquad (3) \\
& \sum_i l_{ir} x_i \leq L_r & \forall r \qquad (4) \\
& x_i = \begin{cases} 1, \text{ item } i \text{ in the test} \\ 0, \text{ else} \end{cases} & \forall i
\end{array}
$$

Variables x_i are the decision variables indicating whether an item is selected or not. I_{ik} is the item information in ability point ϑ_k while T_k is the target information. Coefficients q_{in} are the item's quantitative parameters, c_{im} the classification parameters having value 1 if item i belongs to category m and 0 otherwise. Q_n, C_m^{ℓ} and C_m^{u} are the test's desired quantities and number of items representing the classification categories, respectively. Equation (4) expresses restrictions on item level, the inter item restrictions. Common examples of inter item restrictions are enemy sets and friend sets. The purpose of the model is to maximise the information of the test at that ability point for which the ratio reached/target is minimal. Thus the information of the test in total is maximised while adhering to the preferred shape as much as possible.

A fitness function based upon a penalty function seems to be the most appropriate option. Other types of fitness functions will be effective only for the simplest models. When the model contains more than just one or two restrictions, the feasible region is so small that many infeasible solutions will be rejected. A special representation, for example the one presented by Achterkamp, is also limited to models with relatively few restrictions. A repair algorithm will have the drawback that it yields identical solutions too frequently, thus causing epistasis. On the other hand, a penalty function is versatile in the sense that many restrictions can be added without increased complexity of the algorithm. The proposed fitness function for the IMAX model is composed of the minimal ratio reached/target TIF and of penalty terms for each restriction that is not met:

$$f(x) = y - g(x) \tag{5}$$

$$\begin{aligned} g(x) = & \lambda \sum_n h\left(\sum_i q_{in}x_i - Q_n\right) \\ & +\mu \sum_m h\left(C_m^{\ell} - \sum_i c_{im}x_i\right) + \mu \sum_m h\left(\sum_i c_{im}x_i - C_m^{u}\right) \\ & +\varphi \sum_r 1 - p_r(x) \end{aligned} \tag{6}$$

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

Note that the inter item restrictions are reformulated as relations here. Usually, these restrictions are formulated as relations in first instance, then converted to linear restrictions. Verstralen (1990) noticed that a large number of restrictions is needed for the conversion of some relations, thus hampering the optimisation process. Functions $p_r(x)$ are the payoff functions for the inter item relations, according to De Jong and Spears (1989). The use of payoff functions avoids conversion to Conjunctive Normal Form, and from there on conversion to linear restrictions. The payoff functions are derived from relations by using AVE to evaluate AND clauses and MAX for OR clauses. However, the only type of relations that can successfully be treated are the ones in which the NOT has only the scope of individual variables. Conversion from other types of relations is simply a matter of applying De Morgan's laws.

3.1.1 Penalty multipliers

The advantage of a variable penalty scheme above a fixed one can be made clear from a simple example. Consider a simple neighbourhood structure defined by all solutions that differ just one item from the original solution: either an item is added to or removed from the test. Consider an infeasible solution with feasible neighbours. This means that there must exist at least one item whose addition or removal yields a feasible solution, that is a solution in which penalty function $g(x) = 0$, as formulated in (6). At the same time, the objective function value must change by such a quantity that the fitness function (5) of the feasible solution is higher than the fitness of the original solution.

Now consider an IMAX model with only one restriction defining the desired shape of the TIF:

$$y \leq \frac{\sum_i I_i x_i}{T}$$

and only one quantitative restriction:

$$\sum_i q_i x_i \leq Q$$

For all infeasible solutions in the neighbourhood of a feasible solution the property holds that there is at least one item j whose removal yields a feasible solution. Let $\mathcal{S}$ be such a solution. Let $y(\mathcal{S})$ be the objective function value reached by $\mathcal{S}$ and let $Q^*(\mathcal{S})$ be $Q - \sum_i q_i x_i$, the value by which the quantitative restriction is exceeded. The fitness of $\mathcal{S}$ is given by $f(\mathcal{S}) = y(\mathcal{S}) - \lambda Q^*(\mathcal{S})$. By removing item j, the objective function value will decline to $y(\mathcal{S}) - \frac{I_j}{T}$. Since this solution is feasible, its fitness is equal to the objective function value. The penalty multiplier λ must be chosen so that:

$$\forall \mathcal{S} : \exists j : y(\mathcal{S}) - \lambda Q^*(\mathcal{S}) < y(\mathcal{S}) - \frac{I_j}{T}$$

or:

$$\forall \mathcal{S} : \exists j : \lambda Q^*(\mathcal{S}) > \frac{I_j}{T}$$

For a given solution $\mathcal{S}$ the choice of item j is obvious: it is that item in the test whose item information at the requested ϑ is lowest while $q_j > Q^*(\mathcal{S})$. Item j, however, can be different for each solution $\mathcal{S}$ and in the worst case it is the item with the highest information in the item bank that must be removed in order to reach feasibility:

$$\forall \mathcal{S} : \lambda Q^*(\mathcal{S}) > \max_j \frac{I_j}{T}$$

In general, finding the infeasible solution with the smallest $Q^*(\mathcal{S})$ causes a problem. One can imagine that for some restrictions $Q^*(\mathcal{S})$ might be infinitesimally small, causing λ to go to infinity. Only for certain quantitative restrictions this can be guaranteed not to happen. In case the restriction has only integral coefficients, for example a maximum-number-of-items restriction, it holds that $\forall \mathcal{S} : Q^*(\mathcal{S}) \geq 1$. It can be inferred that λ is bounded by:

$$\lambda > \max_j \frac{I_j}{T}$$

When adding other restrictions and reverting to the original definition of neighbourhood as being the collection of all solutions that can be created in one single iteration, the situation becomes less clear. But here the same principle still holds: although $Q^*(\mathcal{S})$ can be very small, removal of highly informative items might be the only way to reach the feasible region. In this case estimation of λ becomes excessively high while the GAs implemented with high penalty multipliers tend to converge prematurely. For this reason it seems better either to use penalty multipliers found experimentally or the variable penalty scheme as proposed by Siedlecki and Sklanski. Such a scheme can be seen as an adaptive penalty scheme adhering to the minimum penalty rule.

Note that the fitness function as described in equation (5) yields negative values for certain individuals when the penalty is larger than y. For the scaling procedures like linear scaling and rank based scaling this poses no problem. But for mate selection and survival without scaling the fitness function has to be altered. Therefore the fitness function is redefined as

$$f(x) = \frac{y}{1 + g(x)} \tag{7}$$

in order to prevent the occurrence of negative values.

3.1.2 Simulation studies

Simulations were conducted in three phases. In the first phase some preliminary questions had to be answered, such as choice of the crossover operator, mate selection and survival strategies. In the second phase the variable penalty scheme was determined. The variable penalty scheme follows this outline: Consider for k consecutive iterations the individuals with the highest fitness. If for all these k individuals all quantitative restrictions are met, multiply λ by $1 - \delta$. If for all individuals some quantitative restrictions are not met, multiply λ by $1 + \varepsilon$. Leave λ unchanged in other cases, that is, that for some individuals all quantitative restrictions are met while for other individuals there are offending restrictions. Follow the same rule for the other penalty multipliers μ and φ. In the third phase a stopping rule is devised and a comparison is made with other optimisation methods to test the GA for its usefulness.

Two different test assembly models were used. They were based upon the same item bank consisting of 500 items with simulated parameters according the two-parameter model with $\log(\alpha) \sim N(0, 0.4)$ and $\beta \sim N(0, 1)$. All items are classified on two different abstract classifications. The first classification consists of 4 categories 101, 102, 103 and 104. These categories are filled with approximately 250, 125, 85 and 40 items, respectively. The second classification contains categories 201 .. 210, to which the items are assigned uniformly.

The simplest test assembly model, model 1, has four restrictions related to the target TIF: $T_{\vartheta=-1.5} = 4, T_{\vartheta=-0.5} = 8, T_{\vartheta=0.5} = 8, T_{\vartheta=1.5} = 4$ and one quantitative restriction: $\sum_i x_i \leq 40$. No content restrictions based on the classification structure described above are used. Model 2 is an extension of model 1, having a test grid with the columns representing categories 101..104, the rows representing categories 201..210, and each cell requiring exactly one item. Next to this it has two inter item relations that exclude combinations of items observed to be included in the vast majority of good tests constructed without these relations.

3.1.3 Primary simulations

In the first phase some preliminary questions are answered:

- Is there a difference in performance of the variable penalty scheme and the fixed penalty scheme?

- What crossover operator should be used, the one-point crossover or the uniform crossover?
- What strategy for mate selection should be used: no scaling, linear scaling or rank based scaling?
- What strategy for survival should be used, a deterministic one allowing the best N individuals to survive or a stochastic one similar to mate selection?

All results mentioned below are based on 400 test assemblies for each condition. If nothing else is specified, a variable penalty scheme with $k = 20$, $\delta = 0.11$ and $\varepsilon = 0.08$ is used, together with the uniform crossover, no scaling and the deterministic survival. Two models are used: model 1 and model 5 as described earlier. The procedure for the simulations of this phase was to check at certain iterations for the current-best solutions as well as the iteration in which this solution was created. Sometimes, a good solution will be found after which a long period follows in which no improvement was found. The algorithm was stopped at 8000 iterations after which the feasible solution with the highest fitness was reported as the optimum as well as the iteration of creation. This was also the procedure for model 5, except for the fact that the process was stopped at 32000 iterations. In all cases an initial population of 100 individuals was randomly generated with each gene having a chance of $\frac{40}{500}$ of assuming value 1. Elitism is used throughout the simulations. For the variable penalty schemes the elitist strategy needs to be commented upon. As the variable penalty scheme modifies the penalty function every k iterations and with it the fitness function, the best solution might not stay the best solution during multiplier update. Then the elite position shifts towards a different individual. If the best solution is feasible at the update procedure, a copy of it is preserved outside the actual algorithm and reported as the optimal solution if necessary. In Table 1 the progress of the algorithm is given for the base line settings in the first simulation phase for models 1 and 5.

Table 1: Base line condition for Model 1 and 5

	Model 1		Model 2	
iter.	fitness	created at iter.	fitness	created at iter.
250	1.969 ± 0.169	4.3 ± 2.0	3.636 ± 0.078	240 ± 6.9
500	4.563 ± 0.029	440 ± 51	3.712 ± 0.078	400 ± 68
1000	4.583 ± 0.025	760 ± 180	3.741 ± 0.070	680 ± 220
2000	4.592 ± 0.022	1200 ± 460	3.758 ± 0.066	1100 ± 500
4000	4.597 ± 0.019	2000 ± 1000	3.771 ± 0.063	1900 ± 1100
8000	4.601 ± 0.017	3300 ± 2200	3.781 ± 0.058	3300 ± 2300
16000			3.792 ± 0.055	6500 ± 4800
32000			3.805 ± 0.052	13200 ± 10000

A fitness value of 1.969 in Table 1 means that the average fitness of the best feasible solution measured at iteration 250 is 1.969. Since penalty function $g(x)$

in equation (6) has value 0 for all feasible solutions, the fitness function value $f(x)$ coincides with y in equation (1).

Thus the TIF of the current-best solution exceeds, on average, $I_{\vartheta=-1.5} = 7.876, I_{\vartheta=-0.5} = 15.752, I_{\vartheta=0.5} = 15.752, I_{\vartheta=1.5} = 7.876$. It can be seen that these solutions were found very fast, in approximately 4 iterations, but that until iteration 250 no feasible solution with higher information was reported. The population has evolved into the infeasible region. After some time the population has evolved back into the feasible region again, but now in a part with much higher fitnesses. Feasible solutions with high objective function values were found again.

The relatively large standard deviation of the iterations of creation is an indication of the sometimes long intervals between improvements.

In the following sections only the final results are given. The fitness growth during the iterations is not shown, but instead a graphical display of the best solutions and the iterations of creation during the optimisation process for some conditions is given.

Penalty Four conditions were investigated: a variable penalty scheme with $k = 20$, $\delta = 0.11$ and $\varepsilon = 0.08$ and three fixed penalty schemes with multiplier values in the same order of magnitude as the multiplier values in the last iterations of the variable scheme. At the end of the simulations for the variable penalty scheme the average penalty multiplier for model 1 was measured at $\lambda = 0.02 \pm 0.002$. For model 2 the multipliers were $\lambda = 0.02 \pm 0.004$, $\mu = 0.03 \pm 0.006$, $\varphi = 0.05 \pm 0.02$. The values of one of the fixed strategies are chosen close to these averages. The rationale behind this approach is the assumption that the optimal values for a fixed penalty scheme adhere to the minimal penalty rule, while the variable penalty scheme adheres to the minimal penalty rule dynamically. The penalty multiplier values converge to these optimal fixed values. Usually in real-world situations a scheme with somewhat different multipliers will be chosen since these optimal values cannot be known in advance. Therefore, two additional schemes are considered with multipliers somewhat lower and higher. In total three fixed strategies were surveyed, low: $\lambda = 0.004, \mu = 0.006, \varphi = 0.01$; medium: $\lambda = 0.02, \mu = 0.03, \varphi = 0.05$ and high: $\lambda = 0.1, \mu = 0.15, \varphi = 0.25$.

Table 2 presents the final results for the penalty schemes. The multipliers of the low penalty scheme are clearly too low: the solutions that were presented as optimal appeared to be infeasible. The optima for the fitness and objective do not coincide any more. For model 1 only in the very first iterations feasible solutions were found and subsequently reported here. After those first iterations the population developed well into the infeasible region towards the point where the optimal fitness is to be found. For model 2 no feasible solution was found at all.

On the other hand the medium scheme seems to be a proper strategy for a fixed penalty scheme. It should be noted, however, that in 8 cases the medium penalty scheme failed to find a feasible solution for model 2 within 32000 iter-

Table 2: Penalty schemes for Model 1 and 2

	fitness	iteration
Model 1, variable	4.601 ± 0.017	3300 ± 2200
Model 1, low	1.866 ± 0.142	2.9 ± 1.4
Model 1, medium	4.583 ± 0.132	3400 ± 2200
Model 1, high	4.550 ± 0.049	5300 ± 2000
Model 2, variable	3.805 ± 0.052	13200 ± 10000
Model 2, low	–	–
Model 2, medium	3.721 ± 0.079	19000 ± 9000
Model 2, high	3.501 ± 0.129	23800 ± 7000

ations at all. For the high scheme all simulations produced a feasible solution, but the average fitness is lower. So the medium scheme is probably somewhat too close to the estimated minimal values to be useful in all circumstances.

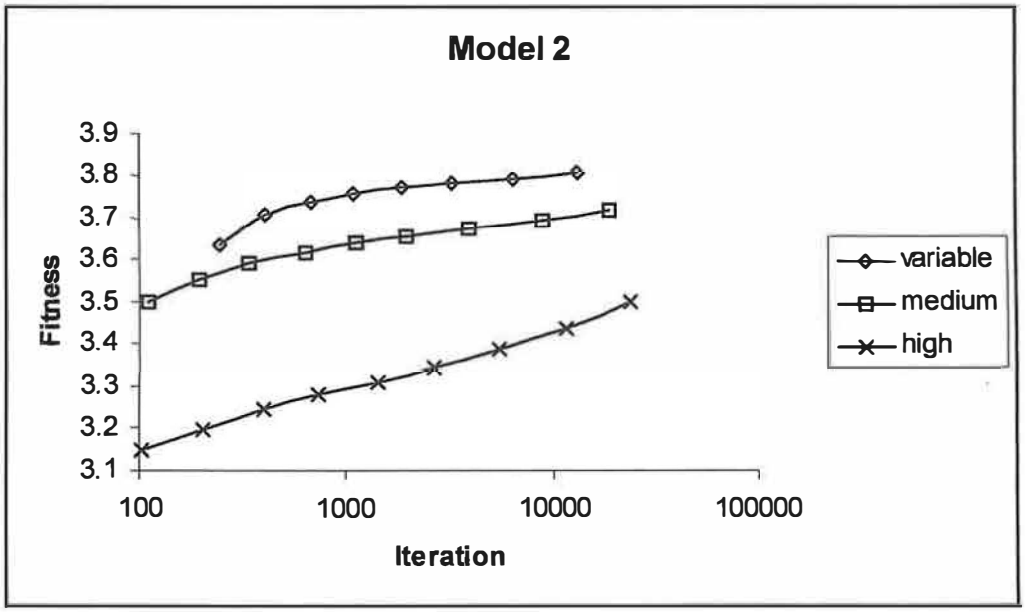

Figure 2: Penalty schemes for Model 2

Although the results suggest that the differences between the schemes are not very great, the superiority of the variable penalty scheme becomes clear in Figure 2 in which the current-best fitness is shown as function of the iteration in which its corresponding solution was created. Combining data in Table 2 and in Table 1 shows that, even while the medium penalty scheme is the optimum fixed penalty scheme, the fitness found at approximately 19000 iterations is comparable with the fitness found at approximately 500 iterations with the variable penalty scheme. In the high penalty scheme, the performance is even considerably worse: it cannot be expected that, on average, within 24000 itera-

tions a solution will be found comparable to the solution found in 240 iterations of the variable scheme. Therefore, fixed penalty schemes were not considered in later simulations.

Crossover Two crossover operators were considered: the uniform crossover and the one-point crossover. Since neighbouring items in the bank have in general no relationship with each other, neighbouring bits in the chromosomes will not form building blocks in a natural way. Thus it can be assumed that the uniform crossover might propagate at least as fast as the one-point crossover while still maintaining the necessary variation in the population.

Table 3: Crossover operators for Model 1 and 2

	fitness	iteration
Model 1, uniform	4.601 ± 0.017	3300 ± 2200
Model 1, one-point	4.591 ± 0.022	5000 ± 2100
Model 2, uniform	3.805 ± 0.052	13200 ± 10000
Model 2, one-point	3.772 ± 0.071	16900 ± 9600

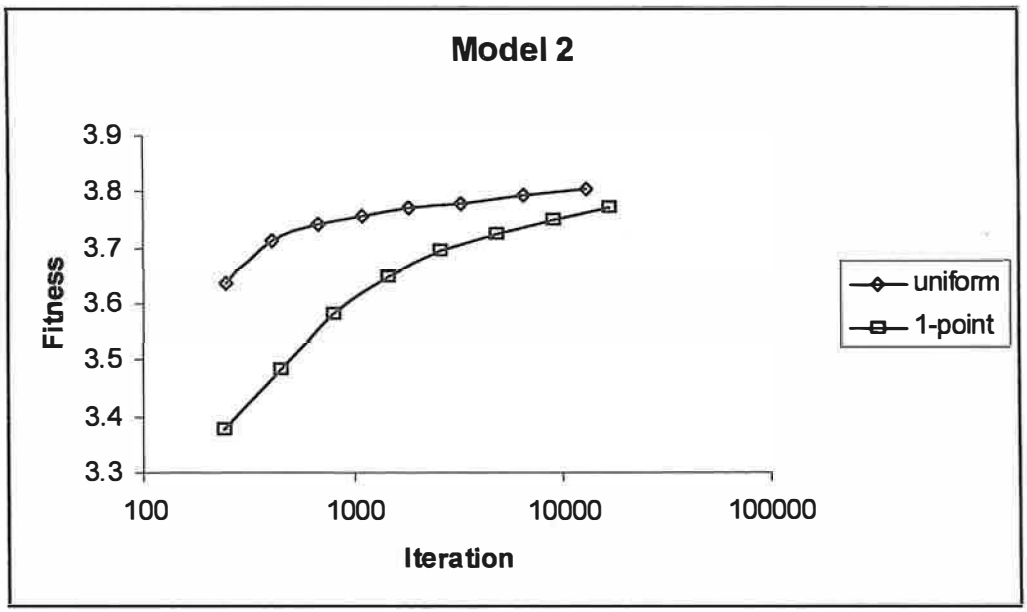

Figure 3: Crossover operators for Model 2

Although it can be inferred from Table 3 that differences are statistically significant, they are relatively small in terms of fitness. Since the optimisation process is rather slow, a large difference in number of iterations is to be expected. This can also be seen in Figure 3 or by combining the data in Table 3 and in Table 1: the one-point crossover reaches a fitness of 3.772 in on average

16900 iterations while the uniform crossover reaches the same in just over 1900 iterations. Therefore the choice is made for the uniform crossover operator.

Scaling The base line condition here is using no scaling technique. This condition was compared with two variants of the scaling techniques mentioned earlier: an aggressive variant with high selection pressure in which the worst individual is not selected at all and a more conservative variant leaving more room to explore the search space. So in total five different conditions were investigated:

1. no scaling
2. linear scaling (linear 1), the worst individual in the population will not be selected
3. linear scaling (linear 2), the worst individual has a chance half as high to be selected as the best individual
4. rank based scaling (rank 1), the worst individual will not be selected
5. rank based scaling (rank 2), the worst individual has a chance half as high to be selected as the best individual.

Table 4: Scaling techniques for Model 1 and 2

	fitness	iteration
Model 1, no scaling	4.601 ± 0.017	3300 ± 2200
Model 1, linear 1	4.594 ± 0.023	3900 ± 2200
Model 1, linear 2	4.597 ± 0.018	3400 ± 2200
Model 1, rank 1	4.595 ± 0.022	3700 ± 2200
Model 1, rank 2	4.600 ± 0.014	3400 ± 2200
Model 2, no scaling	3.805 ± 0.052	13200 ± 10000
Model 2, linear 1	3.784 ± 0.064	15600 ± 9900
Model 2, linear 2	3.805 ± 0.052	14500 ± 9900
Model 2, rank 1	3.791 ± 0.060	14700 ± 9600
Model 2, rank 2	3.805 ± 0.052	15000 ± 10100

Differences in performance regarding scaling techniques are minimal as reported in Table 4. Especially the difference between the conservative scaling variants and not using any scaling at all seems to be negligible. This is also illustrated in Figure 4 for model 2. It should be noted that a high selection pressure by providing a large differentiation between the good and bad individuals is counterproductive. Mate selection must be more explorative so that the genetic material of individuals of low fitness will be recombined into superior offspring. The conclusion is that the choice between the more explorative alternatives does not seem to have a large influence on the optimisation process. Therefore the simplest technique, using no scaling at all, is chosen.

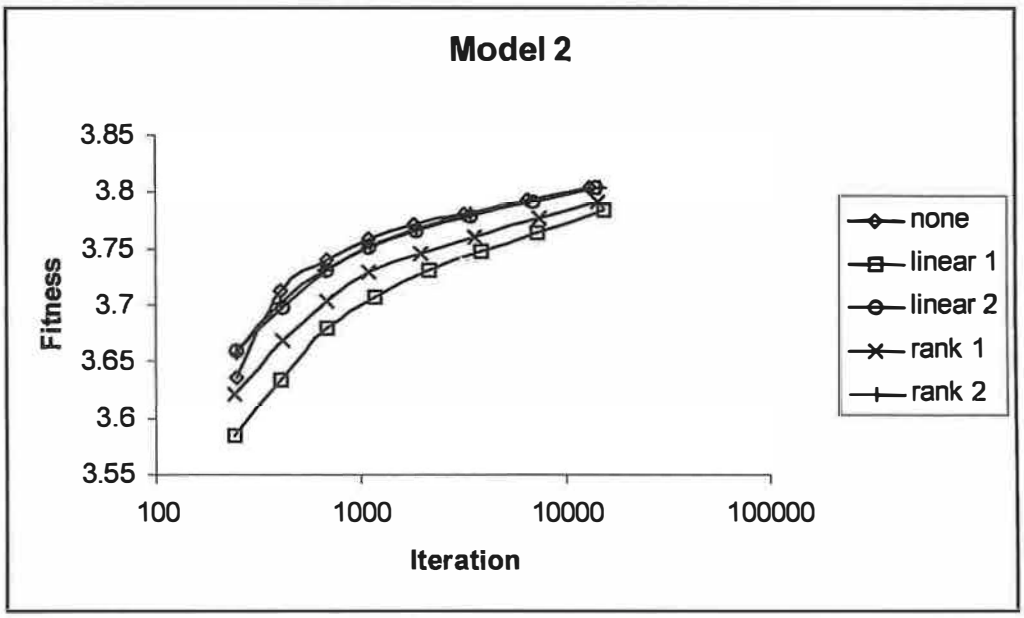

Figure 4: Scaling techniques for Model 2

Survival The first issue regarding survival is whether identical individuals should be allowed or not. Allowing duplicates to survive to the next generation might result in the situation that a highly fit individual generates many duplicates even to the extent that the whole population consists of only duplicates. In terms of test assembly models this means that all proposed tests in the population are identical, and no search direction can be found. Therefore creation of duplicate chromosomes was not allowed in any of the strategies, thus maintaining a minimal level of variety in the population.

All survival strategies start with the removal of newly-created duplicates. After removal of these duplicates two conditions were surveyed, a deterministic one with survival of the N best solutions and a stochastic survival scheme making use of the fitness function in a way similar to mate selection. In the stochastic approach, a linear scaling technique is used such that the chance of survival for the best individual is twice as high as for the worst individual. The rationale behind this scaling procedure is that after only a few iterations the population has been converged to such an extent that the ratio between the highest and lowest fitness is close to 1. In that case all individuals have about equal chance to survive and the external pressure to improvement has disappeared. The linear scaling magnifies the small differences in fitness and restores the pressure.

The stochastic strategy is clearly less efficient than the deterministic strategy as can be inferred from Table 5 and Figure 5. Therefore only the deterministic strategy will be used from here.

Table 5: Survival strategies for Model 1 and 2

	fitness	iteration
Model 1, deterministic	4.601 ± 0.017	3300 ± 2200
Model 1, stochastic	4.473 ± 0.048	6300 ± 1500
Model 2, deterministic	3.805 ± 0.052	13200 ± 10000
Model 2, stochastic	3.612 ± 0.103	24300 ± 7300

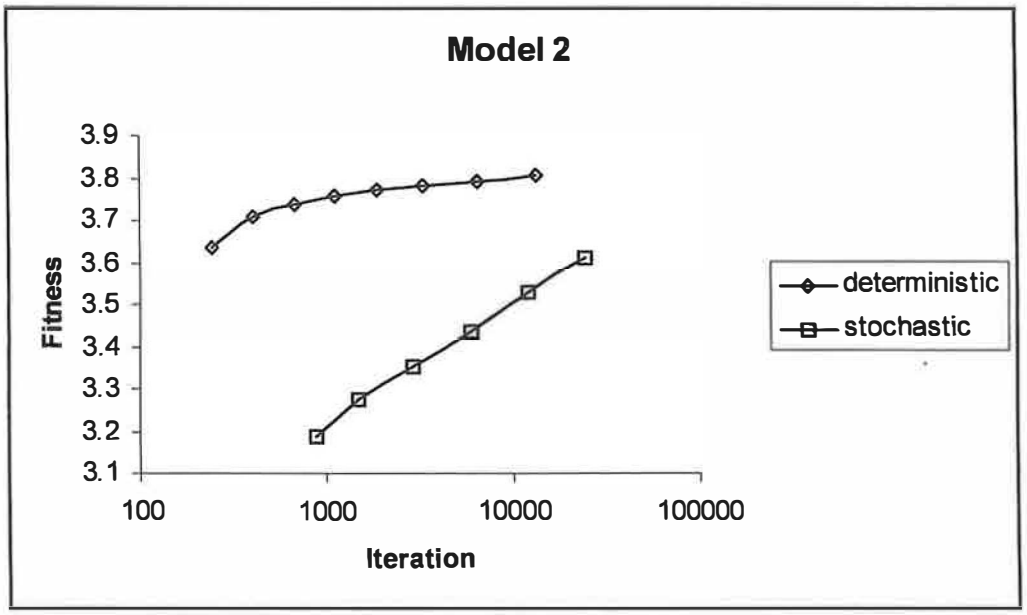

Figure 5: Survival strategies for Model 2

3.1.4 Secondary simulations

In the second phase the variable penalty scheme is investigated, using the best strategies as found in the first phase. The question to be answered here is what values should k, δ and ε assume in order to give a good performance. Specifically, can a fixed set of values be found or is there a dependency with model parameters such as its complexity expressed in the number of restrictions? Six variable penalty schemes were investigated. Three iteration cycles with $k = 10$ (fast), $k = 20$ (moderate) and $k = 40$ (slow) were combined with two adaptation schemes with $\delta = 0.11, \varepsilon = 0.08$ (high) and $\delta = 0.03, \varepsilon = 0.02$ (low). The algorithm was stopped after 16000 iterations for model 1 and 32000 iterations for model 2. Similarly to the preceding simulations the best solution is reported as well as the iteration in which it was created.

Table 6 shows that in general the slow iteration cycles perform somewhat better than the fast and moderate cycles. Similarly the low adaptation schemes seem to win from the high adaptation schemes. The slow high scheme performs somewhat better than the slow low scheme. Although this is not strictly true

Table 6: Variable penalty schemes

		low adaptation		high adaptation	
		fitness	iteration	fitness	iteration
1	fast	4.603 ± 0.016	5600 ± 4500	4.601 ± 0.017	5400 ± 4400
	mod.	4.605 ± 0.014	5600 ± 4100	4.603 ± 0.015	5300 ± 4400
	slow	4.605 ± 0.013	6200 ± 3600	4.606 ± 0.013	5200 ± 4400
2	fast	3.811 ± 0.050	14300 ± 10300	3.791 ± 0.057	15400 ± 9900
	mod.	3.829 ± 0.040	12800 ± 9700	3.805 ± 0.052	13200 ± 10000
	slow	3.829 ± 0.039	13100 ± 9800	3.818 ± 0.049	13400 ± 9800

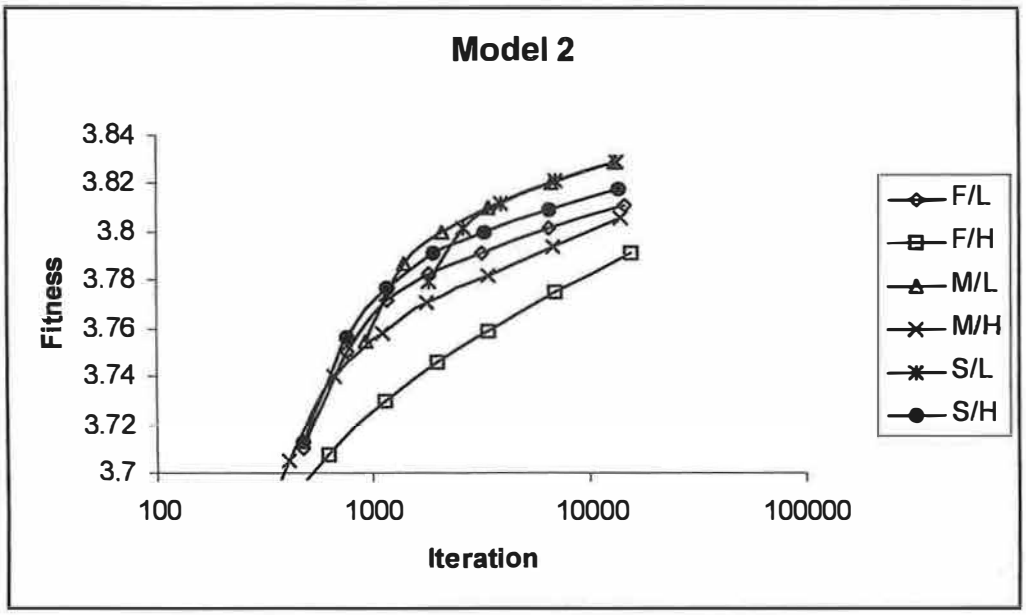

Figure 6: Variable penalty schemes for Model 2

for model 1, it should be noted that model 1 is by far the most simple model needing considerably fewer iterations than other models. Therefore the slow low penalty scheme seems to be the most appropriate choice for all cases, thus applying the same regime to all models. One phenomenon should be considered, however. As can be seen in Figure 6 the slow low and medium low schemes tend to produce feasible solutions rather late in the process while the other schemes produce feasible solutions earlier. Especially in combination with very quick stopping rules this might lead to the unsatisfactory situation that no solution is found at all. In these cases it might be wise to consider one of the other schemes or to modify the stopping in order to accommodate this situation.

3.1.5 Tertiary simulations

As a start in the third phase a stopping criterion is developed. Until now the simulations were based on a fixed stopping rule. After a fixed number of iterations the process was stopped after which the best solution was considered. However, before a comparison can be made with other optimisation methods a stopping rule must be devised. Various criteria can be examined for their usefulness either for a quick or a more thorough optimisation.

The simplest criteria are similar to the fixed rule as used before: stop after a fixed number of iterations, now depending on the model complexity as expressed in I, the number of items in the bank, and K, the total number of restrictions in the model.

Other criteria are based on the convergence of the optimisation process. Three types of convergence can be distinguished. The first is local convergence as expressed in the bias. With a random start the bias is approximately 0.92, rapidly rising to 0.99 in about 100 iterations after which it moves seemingly erratically roughly between 0.99 and 0.995. Although the population seems to be converged at this stage, the optimisation process has not converged at all. The second type is global convergence as expressed in the ratio online/offline performance. This ratio seems to be a better base for a stopping criterion, especially for a rapid optimisation process where a solution with a fitness 5 - 10% below the optimum is sufficient. Usually the ratio starts at about 0.70 rising to about 0.995 or slightly higher at the moment that the current-best solution might not be too far off the optimum. The third type of convergence is based on the creation of new feasible individuals. If for a certain amount of time no feasible solutions are created that would rank somewhere in the top of current-best solutions, then stop. All stopping criteria based on convergence need a fixed rule as a kind of emergency in case the criterion is never met.

In total 5 stopping criteria were examined:

1. Stop at IK iterations
2. Stop at $IK \ln(K)$ iterations
3. Stop at $IK \ln(IK)$ iterations
4. Stop when online / offline performance $= 0.995$
5. Stop when no new solutions are found in $\frac{IK}{10}$ iterations.

Two item banks are used: the same item bank as used before and a second item bank used for Turkish reading comprehension containing 205 items calibrated under the OPLM model. An optimisation model is formulated in model 3 according to the test specifications that involve an equal measurement error at three important ability levels as well as a content balancing over five context categories.

It can be inferred from Table 7 that the stopping criteria based on convergence are outperformed by the fixed stopping rules. Frequently a lower fitness

Table 7: Stopping criteria

	fitness	iterations
Model 1		
1	4.599 ± 0.019	2500
2	4.606 ± 0.016	4024
3	4.615 ± 0.008	19560
4	4.597 ± 0.021	2700 ± 760
5	4.602 ± 0.019	3100 ± 700
Model 2		
1	3.827 ± 0.041	23500
2	3.843 ± 0.035	90478
3	3.852 ± 0.033	236522
4	3.813 ± 0.055	28900 ± 33100
5	3.832 ± 0.040	45200 ± 28500
Model 3		
1	11.483 ± 0.013	1845
2	11.484 ± 0.012	4054
3	11.486 ± 0.009	13875
4	11.483 ± 0.014	1300 ± 55
5	11.484 ± 0.013	2600 ± 600

is reached while, on average, using more iterations. Moreover, since the standard deviation of the number of iterations is rather high the required calculation time becomes unpredictable. The convergence stopping rules are not considered further.

Table 8: Optimal fitnesses and percentages of deviation

Model	optimum	IK	$IK\ln(K)$	$IK\ln(IK)$
1	4.620	0.4%	0.3%	0.1%
2	3.920	2.4%	2.0%	1.7%
3	11.488	$< 0.1\%$	$< 0.1\%$	$< 0.1\%$

When adhering to the norm of stopping when a solution is found within a 5%-bandwidth from the optimum, stopping rule 1 will generally be sufficient, although no indication can be made from within the GA itself and the deviations presented in Table 8 do not hold for individual cases, only for the averages. When a solution is preferred within 2% from the optimum, stopping rule 3 could be chosen for more complex models.

The optimal values reported in Table 8 were obtained using CPLEX ®, a commercially available LP-solver.

3.2 IMIN model

Whereas the IMAX model reflects a resource-driven test assembly approach, the IMIN model (8) is more suitable for situations in which it is important to reach a certain error of measurement, like high stakes testing. Important decisions have to be made based on the outcome of the test. Given the acceptance of a certain maximum measurement error, the purpose of the model is to assemble a test against minimal effort. Thus the IMIN model expresses the wish of the test constructor to construct a 'minimum' test with respect to a quantitative function, for example test length, subject to a target TIF, to quantitative restrictions, to classification restrictions and to inter item restrictions, similar to the IMAX model.

$$
\begin{array}{lll}
\text{minimise} & \sum_i q_{i1} x_i & \\
\text{subject to:} & \sum_i I_{ik} x_i \geq T_k & \forall k \\
& \sum_i q_{in} x_i \geq Q_n & \forall n \\
& C_m^{\ell} \leq \sum_i c_{im} x_i \leq C_m^{u} & \forall m \\
& \sum_i l_{ir} x_i \leq L_r & \forall r \\
& x_i \in \{0,1\} & \forall i
\end{array}
\tag{8}
$$

The proposed fitness function for the IMIN model is similar to the fitness function of the IMAX model. Since the objective function has to be minimised while the fitness function has to be maximised, a transformation is needed in such a way that negative fitness values cannot occur:

$$
f(x) = \left(\frac{1}{1+\sum_i q_{i1} x_i}\right)\left(\frac{1}{1+g(x)}\right) \tag{9}
$$

$$
\begin{aligned}
g(x) = {} & \kappa \sum_k h\left(T_k - \textstyle\sum_i I_{ik} x_i\right) + \lambda \sum_n h\left(Q_n - \textstyle\sum_i q_{in} x_i\right) \\
& + \mu \sum_m h\left(C_m^{\ell} - \textstyle\sum_i c_{im} x_i\right) + \mu \sum_m h\left(\textstyle\sum_i c_{im} x_i - C_m^{u}\right) \\
& + \varphi \sum_r 1 - p_r(x)
\end{aligned}
$$

3.2.1 Epistasis and the IMIN model

The IMIN model is highly susceptible to epistasis. Especially when using the test length as the objective, there are many solutions with equal objective function values. Since all feasible solutions with the same objective function value will have the same fitness, epistasis is the result. The GA cannot make a distinction between solutions and the process might stagnate lacking a search direction. 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 distinction will be restored. These differences should not be assigned randomly to the solutions. They should have some meaningful value in order to propagate the optimisation process.

It is easy to see that, given two feasible solutions, it is easier to remove an item from the most informative test than from the least informative test while still retaining feasibility. Thus assign a reward γ to each solution for each surplus unit of information it has and replace the fitness function as described in equation (9) with equation (10):

$$f(x) = \left(\frac{1}{1+\sum_i q_{i1}x_i}\right)\left(\frac{1+\gamma\sum_k h\left(\sum_i I_{ik}x_i - T_k\right)}{1+g(x)}\right) \tag{10}$$

The reward should not be too high. If adding an item to a feasible solution is rewarded in such a way that it compensates for the increase in objective function and subsequent loss in fitness, the optimum for the fitness does not coincide with the optimum for the model anymore. If the removal of an item results in a feasible solution, it must always be more profitable than the reward for the surplus of information caused by keeping the item in the test.

3.2.2 Simulations

In order to investigate the reward scheme's usefulness, simulations are performed. As a first step, several reward schemes are simulated using three models defined for the same item bank as for the IMAX models: models 4, 5 and 6 respectively. Model 4 has four restrictions pertaining to the target TIF: $T_{\vartheta=-1.5} = 12, T_{\vartheta=-0.5} = 24, T_{\vartheta=0.5} = 24, T_{\vartheta=1.5} = 12$. For model 8 two sets of classification restrictions are defined: for categories 101..104 the maximum percentage of representation is 55, 30, 20 and 10, while for categories 201..210 the maximum percentages are 20. The same principle is used for model 6, but now the representation of categories 101..104 should be between 20% and 30% while for categories 201..210 it should be between 5 % and 15%. Note that the classification restrictions in the IMIN model need a small modification in order to accommodate a representation in terms of fractions of the test length instead of a fixed number of items:

$$C_m^{\ell} \leq \frac{\sum_i c_{im}x_i}{\sum_i x_i} \leq C_m^{u} \qquad \forall m \tag{11}$$

A fixed stopping rule is applied, 8000 iterations for model 4 and 32000 iterations for models 5 and 6. Furthermore, a slow and low adaptation scheme is used.

Table 9: Reward schemes for the IMIN models

	fitness	test length	iteration
Model 4, $\gamma = 0$	0.6641 ± 0.0054	25.30 ± 0.62	1600 ± 2100
Model 4, $\gamma = 0.0001$	0.6742 ± 0.0033	24.17 ± 0.38	2700 ± 2000
Model 4, $\gamma = 0.001$	0.6752 ± 0.0025	24.15 ± 0.36	2600 ± 2000
Model 4, $\gamma = 0.01$	1.0632 ± 0.0026	44.06 ± 0.23	730 ± 1200
Model 5, $\gamma = 0$	0.6636 ± 0.0063	25.36 ± 0.72	6300 ± 7600
Model 5, $\gamma = 0.0001$	0.6734 ± 0.0038	24.26 ± 0.44	8900 ± 8800
Model 5, $\gamma = 0.001$	0.6744 ± 0.0029	24.23 ± 0.42	6500 ± 8100
Model 5, $\gamma = 0.01$	1.0636 ± 0.0016	44.02 ± 0.14	2000 ± 3800
Model 6, $\gamma = 0$	0.6489 ± 0.0045	27.05 ± 0.54	5400 ± 6400
Model 6, $\gamma = 0.0001$	0.6571 ± 0.0026	26.11 ± 0.32	10600 ± 9000
Model 6, $\gamma = 0.001$	0.6583 ± 0.0019	26.11 ± 0.31	11000 ± 9600
Model 6, $\gamma = 0.01$	1.0414 ± 0.0016	46.02 ± 0.15	3400 ± 5300

From Table 9 it can be seen that the reward scheme effectively prevents epistasis and improves the performance of the process. Only the fact that a reward scheme is used seems to be important, as long as the order of magnitude for γ is appropriate. It can be clearly seen that a too high value of γ causes the optimal solutions for the fitness and the objective to be different. Therefore, in subsequent simulations $\gamma = 0.0001$ will be used.

In the second phase the stopping criteria are investigated. Two fixed stopping rules are used, stopping at IK iterations for a rapid optimisation and stopping at $IK \ln(IK)$ iterations for a more thorough optimisation.

Table 10: Optimal test lengths and percentages of deviation

Model	stop at	iter.	test length	optimum	deviation
4	IK	2500	24.35 ± 0.48	24	1.4%
	$IK \ln(IK)$	19560	24.14 ± 0.34		0.6%
5	IK	9500	24.46 ± 0.50	24	1.9%
	$IK \ln(IK)$	87011	24.10 ± 0.30		0.4%
6	IK	9500	26.19 ± 0.39	26	0.7%
	$IK \ln(IK)$	87011	26.07 ± 0.26		0.3%

From Table 10 it can be concluded that for all models investigated here a 2% bandwidth is reached. Note that this holds for the average test lengths only but not for individual cases: only integer values can occur. In 54% of the cases for model 8 using the fast stopping rule a test of 24 items was found, while in 46% of the cases the best test length was 25. This is well outside the 2% bandwidth. On the other hand, if one would accept a 25-item-test, Figure 7 suggests that in many cases this will be reached well before the stopping criterion.

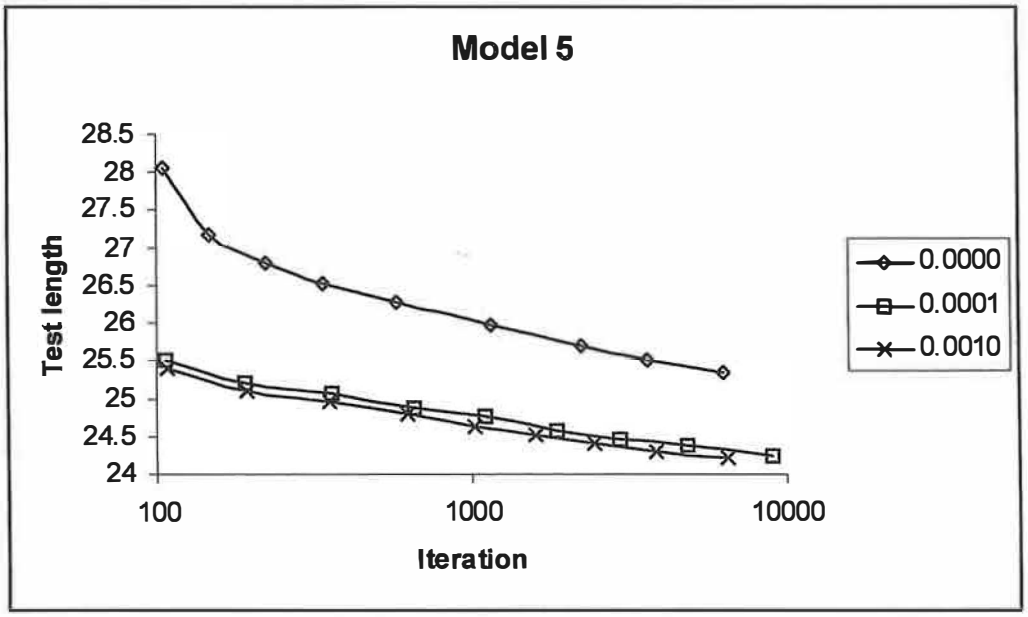

Figure 7: Reward schemes for Model 5

3.3 TCC model

An observed-score model was proposed by Van der Linden and Luecht (1998). Here a variant is proposed using the Test Characteristic Curve. The TCC model (12) can be used in situations where a high information function is not the predominant factor in the optimisation process. The expected TCC is part of the test specifications in situations where a fixed cut-off score is specified. Other situations are those in which the shape of the realised TIF should approximate the shape of the target TIF of a seed test as closely as possible. Both IMAX and IMIN models do not punish solutions whose information is well above its target at some ability points, thus suggesting a test with quite a different shape. This causes a situation in which the cut-off score cannot be fixed properly at a prespecified ability point. The TCC model can be used here, since the TCC and TIF are closely related and the TCC model is more focused on the shape rather than the height of the TIF. The model minimises the distance from a target TCC, subject to quantitative restrictions, to classification restrictions and to inter item restrictions. Here, S_{ik} is the expected score of item i at ϑ_k, W_i the maximum weighted score of item i and T_k the target TCC value at ϑ_k.

$$
\begin{array}{lll}
\text{minimise} & y & (12) \\
\text{subject to:} & y \geq \text{abs}\left(\sum_i (S_{ik} - W_i T_k)\, x_i\right) & \forall k \\
& \sum_i q_{in} x_i \leq Q_n & \forall n \\
& C_m^{\ell} \leq \sum_i c_{im} x_i \leq C_m^{u} & \forall m \\
& \sum_i l_{ir} x_i \leq L_r & \forall r \\
& x_i \in \{0,1\} & \forall i
\end{array}
$$

Similar to the IMIN model, the TCC model is a minimisation model. Therefore the fitness function will be based on a similar transformation of the objective function in order to guarantee only non-negative values. Thus the proposed fitness function for the TCC model is:

$$
f(x) = \left(\frac{1}{1+y}\right)\left(\frac{1}{1+g(x)}\right)
$$

$$
\begin{aligned}
g(x) = {} & \lambda \sum_n h\left(\sum_i q_{in} x_i - Q_n\right) \\
& + \mu \sum_m h\left(C_m^{\ell} - \sum_i c_{im} x_i\right) + \mu \sum_m h\left(\sum_i c_{im} x_i - C_m^{u}\right) \\
& + \varphi \sum_r 1 - p_r(x)
\end{aligned}
$$

3.3.1 Simulations

Simulations were conducted in two phases using two models: model 7 and 8. The simplest model, model 7, has 4 restrictions pertaining to the target TCC defined as: $T_{\vartheta=-1.5} = 0.075, T_{\vartheta=-0.5} = 0.25, T_{\vartheta=0.5} = 0.75, T_{\vartheta=1.5} = 0.925$ as well as one quantitative restriction limiting the number of items used to 40. In model 8 the same classification restrictions are added that were used for model 2.

In the first phase the variable penalty scheme is investigated. Similar to the secondary phase in the IMAX simulations the optimisation is stopped after a fixed number of iterations. Model 7 was stopped at 8000 iterations and model 8 at 32000 iterations. The current-best individual and the iteration in which this individual was created are reported.

The results in Table 11 show that differences in performance are rather small. Although it is strictly speaking not in all cases the optimal strategy, in the subsequent phase the slow and low penalty scheme is used.

Table 11: TCC models

		low		high	
		fitness	iteration	fitness	iteration
7	fast	0.9842 ± 0.0004	1900 ± 1100	0.9842 ± 0.0004	1700 ± 1200
	mod.	0.9841 ± 0.0004	2200 ± 1200	0.9841 ± 0.0005	1700 ± 1100
	slow	0.9840 ± 0.0004	1600 ± 1100	0.9842 ± 0.0004	2000 ± 1100
8	fast	0.9430 ± 0.0055	24900 ± 6600	0.9439 ± 0.0057	24800 ± 6300
	mod.	0.9432 ± 0.0060	24000 ± 7100	0.9454 ± 0.0049	24500 ± 6500
	slow	0.9452 ± 0.0059	23500 ± 7600	0.9464 ± 0.0048	24000 ± 7200

In the second phase the stopping criteria are evaluated. Both fast and thorough stopping rules at IK and $IK \ln(IK)$ iterations, respectively, are examined.

Table 12: Optimal fitnesses and percentages of deviation

Model	stop at	iter.	fitness	optimum	deviation
7	IK	2500	0.9840 ± 0.0005	0.9848	0.1%
	$IK \ln(IK)$	19560	0.9844 ± 0.0004		$< 0.1\%$
8	IK	22500	0.9439 ± 0.0062	0.9575	1.4%
	$IK \ln(IK)$	225479	0.9515 ± 0.0035		0.6%

In Table 12 it can be clearly seen that for all models the fast stopping rule presents a solution well within the 5% bandwidth from the optimum.

4 Conclusions

The results presented in this paper show that test assembly problems can be successfully solved by Genetic Algorithms. Important issues in optimisation, like convergence criteria and speed, are theoretically very hard to deal with, if possible at all. Therefore, simulation studies are used to determine appropriate parameter settings and to demonstrate the Genetic Algorithms' usefulness in solving test assembly problems.

In general variable penalty schemes with longer iteration cycles ($k = 40$), and low adaptation ($\delta = 0.03, \varepsilon = 0.02$) give good results, together with a uniform crossover operator, a more explorative mate selection and a rigorous survival scheme. Stopping after IK iterations usually yields a feasible solution within 5% of the optimum.

Being very versatile methods, GAs can successfully be employed to solve test assembly problems. Although they are in general not extremely efficient on problems that can be solved by traditional methods, their strength lies in the ease to accommodate restrictions that appear to be cumbersome for other methods, like inter item restrictions or nonlinearities.

References

[Achterkamp, 1993] Achterkamp, M. (1993). Toetsconstructie met behulp van genetische algoritmen (test construction with genetic algorithms). Master's thesis, University of Twente.

[Adema and van der Linden, 1989] Adema, J. and van der Linden, W. (1989). Algorithms for computerized test construction of parallel tests using classical item parameters. *Journal of Educational Statistics*, 15:129–145.

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

[Aytug and Koehler, 1996] Aytug, H. and Koehler, G. (1996). Stopping criteria for finite length genetic algorithms. *INFORMS Journal of Computing*, 8:183–191.

[Baker, 1985] Baker, J. (1985). Adaptive selection methods for genetic algorithms. In Grefenstette, J., editor, *Proceedings of an International Conference on Genetic Algorithms and their Applications*, pages 101–111. Pittsburgh: Carnegie-Mellon University.

[Birnbaum, 1968] Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. In Lord, F. and Novick, M., editors, *Statistical Theories of Mental Scores*. Reading, MA: Addison-Wesley.

[Bridges and Goldberg, 1987] Bridges, C. and Goldberg, D. (1987). An analysis of reproduction and crossover in a binary-coded genetic algorithm. In Grefenstette, J., editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 9–13. Hillsdale, NJ: Erlbaum.

[Darwin, 1859] Darwin, C. (1859). *The Origin of Species by Means of Natural Selection*. London: John Murray.

[Davidor, 1991] Davidor, Y. (1991). Epistasis variance: a viewpoint on GA-hardness. In Rawlins, G., editor, *Foundations of Genetic Algorithms*, pages 23–35. San Mateo, CA: Morgan Kaufmann.

[De Jong, 1975] De Jong, K. (1975). *Analysis of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan.

[De Jong and Spears, 1989] 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.

[Eiben et al., 1991] Eiben, A., Aarts, E., and van Hee, K. (1991). Global convergence of genetic algorithms: a Markov chain analysis. In Schwefel, H.-P. and Männer, R., editors, *Parallel Problem Solving from Nature I*, pages 4–12. Berlin: Springer.

[Feuerman and Weiss, 1973] Feuerman, F. and Weiss, H. (1973). A mathematical programming model for test construction and scoring. *Management science*, 19:961–966.

[Fogarty, 1989] Fogarty, T. (1989). Varying the probability of mutation in the genetic algorithm. In Schaffer, J., editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 104–109. San Mateo, CA: Morgan Kaufmann.

[Goldberg, 1989] Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization & Machine Learning*. Reading, MA: Addison-Wesley.

[Goldberg and Segrest, 1987] Goldberg, D. and Segrest, P. (1987). Finite Markov chain analysis of genetic algorithms. In Grefenstette, J., editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 1–8. Hillsdale, NJ: Erlbaum.

[Holland, 1968] Holland, J. (1968). Hierarchical description of universal spaces and adaptive systems. Technical Report ORA projects 01252 and 08226, Ann Arbor: University of Michigan.

[Holland, 1973] Holland, J. (1973). Genetic algorithms and the optimal allocations of trials. *SIAM Journal of Computing*, 2:88–105.

[Holland, 1975] Holland, J. (1975). *Adaptation in Natural and Artificial Systems*. Ann Arbor: University of Michigan Press.

[Le Riche et al., 1995] Le Riche, R., Knopf-Lenoir, C., and Haftka, R. (1995). A segregated genetic algorithm for constrained structural optimization. In Eschelbaum, L., editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 558–565. San Mateo, CA: Morgan Kaufmann.

[Luecht and Hirsch, 1992] Luecht, R. and Hirsch, T. (1992). Computerized test construction using average growth approximation of target information functions. *Applied Psychological Measurement*, 16:41–52.

[Michalewicz, 1994] Michalewicz, Z. (1994). *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin: Springer, 2nd. edition.

[Orvosh and Davis, 1993] Orvosh, D. and Davis, L. (1993). Shall we repair? genetic algorithms, combinatorial optimization and feasibility constraints. In Forrest, S., editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, page 650. San Mateo, CA: Morgan Kaufmann.

[Richardson et al., 1989] Richardson, J., Palmer, M., Liepins, G., and Hilliard, M. (1989). Some guidelines for genetic algorithms with penalty functions. In Schaffer, J., editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 191–197. San Mateo, CA: Morgan Kaufmann.

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

[Siedlecki and Sklanski, 1989] 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.

[Suzuki, 1993] Suzuki, J. (1993). A Markov chain analysis on a genetic algorithm. In Forrest, S., editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 146–153. San Mateo, CA: Morgan Kaufmann.

[Swanson and Stocking, 1993] Swanson, L. and Stocking, M. (1993). A model and heuristic for solving very large item selection problems. *Applied Psychological Measurement*, 17:151–166.

[Syswerda, 1989] Syswerda, G. (1989). Uniform crossover in genetic algorithms. In Schaffer, J., editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 2–9. San Mateo, CA: Morgan Kaufmann.

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

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

[Thierens and Goldberg, 1994] Thierens, D. and Goldberg, D. (1994). Convergence models of genetic algorithm selection schemes. In Davidor, Y., editor, *Parallel Problem Solving from Nature - PPSN III*, pages 119–129. Berlin: Springer.

[Timminga et al., 1996] Timminga, E., van der Linden, W., and Schweizer, D. (1996). *ConTEST 2.0: A decision support system for item banking and optimal test assembly.* Groningen: iec ProGAMMA. Software and User's Manual.

[van der Linden, 1996] van der Linden, W. (1996). Assembling tests for the measurement of multiple abilities. *Applied Psychological Measurement*, 20:373–388.

[van der Linden and Boekkooi-Timminga, 1988] van der Linden, W. and Boekkooi-Timminga, E. (1988). A zero-one programming approach to Gulliksen's matched random subtests method. *Applied Psychological Measurement*, 12:201–209.

[van der Linden and Boekkooi-Timminga, 1989] van der Linden, W. and Boekkooi-Timminga, E. (1989). A maximin model for test design with practical constraints. *Psychometrika*, 54:237–247.

[van der Linden and Luecht, 1998] van der Linden, W. and Luecht, R. (1998). Observed-score equating as a test assembly problem. *Psychometrika*, 63:401–418.

[Veldkamp, 1998] Veldkamp, B. (1998). Multidimensional test assembly based on Lagrangian relaxation techniques. Technical Report 98-08, University of Twente.

[Verschoor, 1991] Verschoor, A. (1991). *Optimal Test Design.* Arnhem: Cito. Software and User's Manual.

[Verstralen, 1990] Verstralen, H. (1990). Semantic and syntactic simplification of truth functions. Technical Report OIS 8, Arnhem: Cito.