UNIVERSITY OF TURKU

New metaheuristic approaches for optimal stimuli selection

Timi Niemensivu

MSc thesis
September 2024

Reviewers:
Prof. Marko Mäkelä
PoP Pekka Räsänen

DEPARTMENT OF MATHEMATICS AND STATISTICS

UNIVERSITY OF TURKU
Department of Mathematics and Statistics

Niemensivu Timi: New metaheuristic approaches for optimal stimuli selection
MSc Thesis, 69 pages, 2 appendix pages
Applied mathematics
September 2024

In psychological testing, specifically with repeated testing of the same subjects, the optimal choice of used stimuli is essential. In repeated testing, two similar test versions are necessary since the test subject always learns something from the test itself, which means conducting the same test twice would lead to errors in estimates. Other reasons for the need for two test versions include research settings where interest is in the effect of a certain stimuli characteristic.

Optimal stimuli selection has received surprisingly little attention from the academic community, and many state-of-the-art optimization methods have not been used to tackle the issue. In this thesis, two new metaheuristic approaches for stimuli selection are proposed. The proposed methods will be based on iterated local search and scatter search metaheuristics. The new methods are compared to the simulated annealing-based method, which has previously seen its fair share of use.

The methods were compared with various real-life and simulated datasets in terms of optimality of solutions and used computational time. Resulting test versions were also compared by inspecting descriptive statistics, as the mathematically optimal solution is not guaranteed optimal in terms of practical research use. A comparison of the methods showed that scatter search seemed the best at finding the optimum with the drawback of the computing times getting out of hand in the larger dataset. Simulated annealing showed its strengths as a good all-rounder, while the iterated local search was fasted with the least-optimal solutions.

Keywords: Stimuli selection, Test item selection, Metaheuristics, Combinatorial optimization

Contents

1 Introduction

Measuring change or development in skills or abilities is common in psychological and educational sciences. However, using the same scale or test to measure things more than once may cause some complications. The complication here is that many carry-over effects from the first testing can contaminate the scores of the second testing (Allen and Yen, 1979, p. 77). This contamination may be due to test takers remembering and reusing their previous answers, or it could be the practice effect due to learning about the test situation the first time.

Different test items can be used to prevent such contamination. This presents the issue of the scores on these two different tests being comparable. To guarantee comparability, the stimuli items in the test need to be carefully chosen so that the tests behave equally. Ensuring this involves choosing similar stimuli for both test versions.

In most situations, selecting the optimal or good stimuli by hand is demanding. The more information we have about the item features, the more difficult it will be to use this information in selection. Therefore, automatic methods for stimuli selection have been proposed instead of the manual heuristic approach. In this thesis, the stimuli selection problem is modelled mathematically as an integer nonlinear programming problem, which is solved with metaheuristics methods.

Metaheuristics are extensions of heuristic methods that incorporate a heuristic approach and widen them to find solutions that compare to the global optimum. Metaheuristics are used here, as the problem at hand is not too difficult to solve exactly. These metaheuristics find approximatively optimal solutions in most cases and therefore give a good balance between the quality of the solutions produced and the needed computational load.

This thesis is done in cooperation with Turku Research Institute for Learning Analytics (later TRILA), which requested a case study of the methods that could be used to create parallel test versions for the lexical decision-making task Lexize (Salmela et al., 2021). The task is included in the reading test battery of the FUNA measurement tools (Räsänen et al., 2021). FUNA is a digital test battery originally made for online assessments of dyscalculia, but later there has been development to study reading and cumulation of learning difficulties.

In addition to constructing equal test versions for Lexize, the methods can be used in other similar repeated testing scenarios that frequently arise in the test development work conducted and psychological measurements. Repeated assessments are not the only application of these methods, as they can be used in studies where the purpose is to examine the effects of some singular features of the stimuli items. This is done by having two lists being otherwise equal but different in the fixed feature. With slight modifications, the methods could also be used e.g. to match the subjects in treatment and control groups.

The structure of this thesis is as follows. Chapter 2 describes the psychological testing context, in which the methods will be applied later. Chapter 3 is for formulating the mathematical optimization problem modelled to correspond to the practical problem at hand. In Chapter 4, the different metaheuristics methods are presented. Out of the methods presented, two of them are chosen to be tested against previously used simulated annealing (Armstrong et al., 2012a). In Chapter 5, I describe the effects the different parameterizations of these methods have on the results. Finally, in Chapter 6, the main results of the thesis are presented. These include the quality of solutions and the CPU time used by the compared methods. The practical quality of these mathematically best solutions is also discussed. In Chapter 7, the conclusions of the results and ideas for follow-up studies are discussed.

2 Stimuli selection in repeated testing

2.1 Psychological testing

"Function of the psychological tests is to measure differences between individuals or between reactions of the same individual on different occasions" (Anastasi, 1961, p. 3). Abilities that psychological tests measure are often abstract, such as intelligence or linguistic skills. This contributes to certain ambiguity of the tests, as the measures for the abilities cannot be defined exactly. For assessing the test accuracy, the evidence is collected from a combination of theory, previous psychological knowledge, and other available evidence (The British Psychological Society, 2017, p. 11).

Psychological tests themselves do not tell much without means to interpret the result. The interpretation can be either done through a table of norms or having some criterion score as a reference (The British Psychological Society, 2017, p. 9). For valid interpretations, a test user must consider possible test taker-related background effects such as age, gender, ethnicity, and primary language (American Psychological Association, 2020, p. 17). This is often done by calculating different norms, such as age- or language-group-based norms.

The most common way of standardizing the scores is to calculate z-scores, subtract the group-wise mean from the raw score and divide the result with group-wise standard deviation. If there is only one tested group, the z-score can be presented as $z_i = \frac{x_i - \bar{x}}{sd(x)}$, where the $\bar{x}$ is the sample mean and $sd(x)$ is the standard deviation of the sample. As the mean and standard deviation are constant, this transformation from raw score to z-score can be seen as a linear transformation, so it will only change the scale and positioning of the distribution of scores.

In the case of normally distributed scores, which often is the case at least approximately (Allen and Yen, 1979, p. 20), this will mean that z-scores will be standard normally distributed. The standard normal distribution has many beneficial properties for statistical modelling, while also allowing the switching to a more interpretable scale like the PISA-scale (e.g. OECD, 2019). In addition, many distributions can be approximated by a normal distribution, and large samples tend to be close to normally distributed as the central limit theorem states will asymptotically happen for the distribution of sample means. This means that if the sample is representative of the population in question and the distribution of the scores is relatively normal, quantiles of the standard normal distribution can be used to compare the result of an individual to the rest of the population.

$$Score_O = Score_T + Score_E \tag{1}$$

In the classic test theory framework, the observed scores of an individual can be decomposed to true score and measurement error (Allen and Yen, 1979, p. 57), such as in (1). Other decompositions are also possible, such as one where the error has been divided by its source or might be in some ways dependent on the true score. As the scores are not the main point of this thesis, this simple version is sufficient. More complex ways include item response theory (IRT) models, where the scores are calculated through a more complex mathematical function that involves estimating the amount of latent ability needed to get a certain item correctly (e.g. Birnbaum, 1968). IRT models will be discussed more in the section 2.4.

Here, it is assumed that the expected value of the observed score is the true score, and there is no correlation between the true score and the measurement error. The variance of the true scores σ_T^2 is always smaller or equal to the variance of the observed scores σ_O^2 because $\sigma_O^2 = \sigma_T^2 + \sigma_\varepsilon^2$, where σ_ε^2 is the variance of the measurement error (Allen and Yen, 1979, p. 65). This also means that if the variances are equal, there exists no measurement error as then $\sigma_\varepsilon^2 = 0$ and because $E[O] = T$ therefore $E[\varepsilon] = 0$. From previous facts, we can construct a measure $\varrho = \frac{\sigma_T^2}{\sigma_O^2}$ of how reliably observed scores reflect the true scores. It can be easily seen that ϱ gets the maximum value of 1 if there is no measurement error, and it is a decreasing function of the magnitude of the measurement errors. This measure is called *reliability* of the test, and as the true scores of the test are unknown, it must be estimated from data through some coefficient such as Cronbach's alpha (Cronbach, 1951), greatest lower bound estimator (Woodhouse and Jackson, 1977) or McDonald's omega (McDonald, 1999).

The fact that the test is consistently measuring something does not yet imply that it is good. Another important thing to consider is how the test scores can be interpreted. The test creator always has an intent of what the test measures and how it is to be interpreted, the degree to which the test achieves these intents regarding theory and collected evidence is referred to as the validity of the test (American Educational Research Association et al., 2014, p. 22). Reliability and validity are the most important measures of goodness of psychological test but neither of them works alone. In other words, a good test is both valid and reliable.

Unlike reliability, validity is not straightforward to measure but is about collecting evidence that supports the intended interpretations being valid. Validity has often been divided into subcategories: content, predictive, concurrent, and construct validity (American Psychological Association, 1954). However, the newest Standards for Educational and Psychological Testing by American Educational Research Association et al. (2014) consider validity a unitary concept and these subcategories as different aspects of the validity seen through different sources of evidence rather than individual subcategories.

High validity and reliability are still insufficient for a psychological test to accurately depict some ability. As for the test to work, the test taker must complete it to the best of their abilities. This is especially prevalent in psychological and educational tests where the results hold no consequences for the test taker, referred to as low-stakes testing. The opposite of this is consequential or high-stakes testing, which has been shown to lead to higher scores than low-stakes testing (Sundre, 1999; DeMars, 2000).

Test-taking motivation is associated with test performance, which is particularly problematic in a low-stakes setting, where motivation is often lower (Wise and DeMars, 2005). Additional incentives such as financial benefits from good performance and results feedback have been tried to increase the test-taking motivation, yielding varying degrees of success (Oneil et al., 1995; Baumert and Demmrich, 2001). More demanding test items such as essays and open-ended questions are more affected by the low-stakes setting than the multiple choice questions (DeMars, 2000). Low test-taking motivation is a relevant issue in testing, leading to lower test validity, as the scores no longer represent the actual level of the test takers' ability.

The purpose of this section was to give the reader a quick summary of psychological testing better to understand the context and motivation of this thesis. In the next section, the specific area of psychological testing in which methods presented later will be applied is described.

2.2 Lexical decision-making task

In education, the ability to read is central to all learning. Not only does it have a positive association with studying motivation (Toste et al., 2020), but more importantly, reading achievement can predict later academic success (Herbers et al., 2012; Mitchell, 2024). Additionally, with communication increasingly transitioning to written form, the importance of reading skills is high.

To construct good measures for a certain skill, there needs to be some theoretical understanding of said skill. Foremost, there must be a clear definition of reading skills. One used here will be based on A Simple View of Reading (later SVR), where reading is seen as a multiplicative process $R = D \times C$ of decomposition D and comprehension C (Gough and Tunmer, 1986). The definition of decomposition varies a bit. Gough and Tunmer argue it is not the same as word recognition but is closely tied to it while also being closely tied to knowledge of letter-sound correspondence rules. Linguistic comprehension is more straightforward, as it refers to the process of interpreting lexical information, sentences and discourses.

Apart from SVR, there exist many other theories on reading, many of which extend SVR with additional components. One such extension is Scarborough's Reading Rope (Scarborough, 2001), which attempts to visualize the learning of reading skills based on the SVR. The reading rope conceptualizes decoding and linguistic comprehension as strands of rope that are built from smaller subareas such as vocabulary or phonological awareness. The idea of a rope stems from the fact that these strands are all seen as necessary building blocks woven together like a rope to form a fluent reading skill.

SVR has since been developed into The Active View of Reading (AVR) model, which adds emphasis on the processes bridging word recognition and language comprehension together (Duke and Cartwright, 2021). These bridging processes include reading fluency and vocabulary knowledge, for example. The reason for adding these bridging processes is that even though the studies show that decoding and language comprehension explain most of the variance in reading comprehension, a large portion of this variance is shared between the two (Lonigan et al., 2018). Tunmer and Chapman (2012) showed similar results and found that language comprehension skills may directly contribute to word recognition. Therefore, it makes sense that SVR needed an extension where the shared variance is taken into consideration, such as in AVR.

Finnish language in its standard written form follows closely the standard spoken language, which in turn makes the burden of learning decoding skills low compared to other languages (Lyytinen et al., 2019). As the decoding of written Finnish is less challenging, the language comprehension skills get additional focus in comparison. Later analyses for SVR by Tunmer and Chapman (2012) divided language comprehension into two factors, which are listening comprehension and vocabulary knowledge. Unlike in SVR, the Reading Rope and AVR already mention vocabulary as a factor contributing to reading comprehension. In the Reading Rope, it is considered as a part of language comprehension and in AVR as a part of the bridging processes. The highlighted weight of language comprehension in the Finnish language and the central role of vocabulary as a contributor to reading comprehension skills make it an important factor to be researched in the context of Finnish reading research.

Even though the aforementioned models were all built more or less on top of the SVR model, the vocabulary has a similarly central role in other theoretical frameworks. In the Direct and Inferential Mediation model (DIME) (Cromley and Azevedo, 2007) vocabulary is considered a contributing to reading comprehension and was found to have a medium effect size as defined by Cohen (1988). Another model challenging SVR called the Direct and Indirect Effect (DIER) (Kim, 2017) model had vocabulary as a part of its structural regression, even though the connection was not as straightforward as with DIME.

The importance of vocabulary in the theoretical frameworks of reading stems from the empirical results of the connection between vocabulary and reading comprehension. One study estimated that knowing around 95 % of the words is sufficient for comprehensive reading (Laufer, 1989), which would explain the connection between the two. A systematic review on the effects of vocabulary interventions on reading comprehension found that teaching text-related words improves comprehension, but no evidence that teaching vocabulary affected overall reading comprehension (Wright and Cervetti, 2017). Even though the vocabulary interventions did not affect general reading comprehension, this does not mean that vocabulary isn't a central part of reading comprehension, as the amount of vocabulary knowledge a word intervention can teach is limited. The text-specific interventions did have a clear effect. Similar results were acquired in Finland when Harkio and Pietilä (2016) found strong correlations between vocabulary breadth, width, and reading comprehension in English.

No studies have been done on the relationship between vocabulary and reading comprehension in the Finnish language due to no vocabulary assessment tools made for the Finnish language until recently. The only existing tool is Lexize (Salmela et al., 2021), which is a visual lexical decision-based test for the assessment of vocabulary size. In the test, a word or a pseudoword is given to the test taker, and the task is to determine whether it is a word. Since then, an auditive version of Lexize has also been developed.

The main philosophical idea behind the Lexize task is that if one does not know a word, one does not think of it as a word, at least when it is shown by itself without any context. This idea can be taken advantage of in measuring vocabulary, as it implies that if a test subject does not think some word is a word, that word does not belong to their vocabulary. This is why using pseudowords instead of non-words is necessary, as those are indistinguishable from the words the test taker does not know.

Lexize is based on an English vocabulary test LexTALE (Lemhöfer and Broersma, 2012), which has since been used as a base for multiple vocabulary tests in other languages (e.g. Brysbaert, 2013). Unlike the original LexTALE, which had only 60 items (40 words and 20 pseudowords), the first version of Lexize consisted of 68 words and 34 pseudowords. This larger number of items has since allowed the creation of parallel lists for the FUNA assessments.

Lexize has now been integrated as a part of the reading and writing segment of the FUNA test battery. FUNA test battery is an assessment environment widely used in large-scale research projects. Even if Lexize is the test that sparked the idea, the methodology later described in this thesis will be used for other tests that TRILA uses in research. As the context of stimuli items used in practical selection problems is now provided, the motivation for the selection process. The next section discusses the problems arising in repeated testing and their implications for the selection of stimuli items.

2.3 Repeated testing and parallel test versions

Measurement of the evolution of psychological abilities is done through repeated measurements. The complication here is that many carry-over effects from the first testing can contaminate the scores of the second testing (Allen and Yen, 1979, p. 77). This contamination may be due to test takers remembering and reusing their previous answers, or it could be the practice effect due to learning about the test situation the first time.

Let's consider the classic true score decomposition (1) and how carry-over effects affect it. The first situation is one where the ability measured is evolving rapidly, and the test interval is short enough for test takers to remember their answers. Here $Score^1_O$ and $Score^2_O$ are the observed scores of the first and second measurements, respectively. Because the test takers remember the answer, the answer is the same both times and $Score^1_O = Score^2_O$. As the ability measured evolves, we can assume that $Score^1_T < Score^2_T$. With the previous decomposition, this means that $Score^1_E < Score^2_E$, which means that there is more measurement error on the second test. The bigger issue here is that the ability measured has changed, but the test does not indicate it, as the test taker only repeats the previous answers.

Next, let's demonstrate the practice effect and problems related to it by analyzing a situation where the ability measured does not evolve (or degrade) with time, which means $Score^1_T = Score^2_T$. Here it is assumed that the interval between the tests is long enough that answers are not remembered exactly, but taking the test improves the score of the next test, meaning $Score^1_O < Score^2_O$. Once again, this means that $Score^1_E < Score^2_E$. In addition to adding measurement error, the practice effect can be interpreted as an evolution of the ability, while it is only something learned from the previous test.

Previous examples highlight that the same test items cannot be used in repeated testing without introducing multiple possible sources of measurement error. In the optimal case, where no result-altering carry-over effects exist, the same test could be used. Otherwise, the use of different stimuli is recommended. Different stimuli are used by using multiple versions of the same test or having a bank of items from which the individual test forms are constructed. An issue with using different stimuli is that there is no existing base for interpreting the relationship between the two tests done with different stimuli. For ease of interpretation, the first and the second tests are either constructed to be equal in terms of the scores or are transformed to a mutual scale.

In the classical true-score framework, the tests are considered parallel if $Score_T^1 = Score_T^2$ and $\sigma_{\varepsilon 1}^2 = \sigma_{\varepsilon 2}^2$ (Allen and Yen, 1979, p. 57). This is the desired scenario, as it would mean that the population level distributions of the observed scores differ only by the possible change between the tests. Therefore, the results reflect the change better if the tests are indeed parallel. Experimental evidence of parallel lists reducing the practice effect can be found for example in (Crawford et al., 1989).

Chelune (2003) argues that alternate forms do not work for avoiding carry-over effects, as at least some of the effect is due to learning the test procedures and not the items themselves. This is backed up by studies showing practice effects even on alternative forms (Uchiyama et al., 1995; Ruff et al., 1996). Part of the argument is that constructing parallel lists with comparable scores is a difficult task.

Apart from parallel tests, the research on assessing the change over time has also created tools for assessing the reliability of the change. Attempting to separate the measurement error and the actual change involves the calculation of a reliable change index (RCI) (Jacobson and Truax, 1991). The original formulation has been debated (e.g. Speer and Greenbaum, 1995; Hsu, 1999), but the general idea of reliable change has seen its fair share of clinical use (Iverson, 2011). In its original form, RCI assumes that there is no practice effect, which is unlikely true, and therefore, multiple modifications that account for the practice effect have been proposed (Chelune, 2003).

Continuing the evaluation of avoiding the practice effect with alternative tests, assume that two test versions have the same true score for the same level of ability. Still, the variances of the error may not exactly be the same. For simplicity, the practice effect P is assumed to be constant over the population. With addition of the change in ability ΔT, the observed scores have relation

$$Score_O^2 = Score_T^2 + Score_E^2 = Score_O^1 + P + \Delta T - Score_E^1 + Score_E^2 \tag{2}$$

As the error terms have expected value of zero and variances $\sigma_{\varepsilon 1}^2$, $\sigma_{\varepsilon 2}^2$, the sum $-Score_E^1 + Score_E^2$ has also zero expectation and variance $\sigma_{\varepsilon 1}^2 + \sigma_{\varepsilon 2}^2$, as the error terms are assumed to be independent. This means that the observed change $Score_O^2 - Score_O^1$ consists of the practice effect, actual change of the true score, and error, a zero-mean random variable. With these assumptions, if the size of the practice effect could be accurately estimated beforehand, the average change should be close to the population-level change.

In practical situations, estimating the effects would be difficult, because of the costs and efforts related to acquiring samples. The issue with the estimation approach is that it must be estimated within a time frame in which change in ability does not occur. This means that the change and the size of the practice effect cannot be estimated in the same time frame as the measurement of change, which likely leads to inaccurate estimates. Another issue that could arise from this approach is that if the practice effect is large and the test takers are already at the higher end of the scale, the ceiling effect is likely to occur. This means that the test takers obtain very high or maximal scores, which means that their ability can be above the scale and, therefore, underestimated (Banks, 2011).

With the aforementioned issues in estimating the practice effect, it seems advisable to build the test setting so that the practice effect is minimal. Use of the alternate forms has been shown to completely negate the practice effect at times (Crawford et al., 1989), but the evidence of only a smaller reduction of the practice effect by alternate forms also exists (Benedict and Zgaljardic, 1998). Benedict and Zgaljardic (1998) suggest that the alternate forms have less practice effect, negating effectiveness on the tests measuring a novel concept, visuospatial learning, or graphomotor responding. In general, alternate forms reduce the practice effect, but the size of the reduction seems to depend on the type of the tests (Beglinger et al., 2005).

Multiple studies show the practice effect being substantially larger between the first and the second trial than the subsequent trials (Theisen et al., 1998; Ivnik et al., 1999; Collie et al., 2003). This indicates that the practice effect could be reduced by having a practice test before repeated testing. Though, having an extra test is usually not practical. In FUNA measurements, a similar idea is implemented by having a few practice items before the actual test, which must be done right before proceeding. This way, the test logic becomes more familiar, and scores better represent the real skill levels.

With the prevalence of the practice effect, strategies for handling it are in order. Here, the approach chosen to be the most suitable is the use of alternate or, in the best-case scenario, parallel test versions. The question becomes how to make the used alternate lists equal and comparable. The methods for constructing these lists are discussed in the next section.

2.4 Optimal selection of stimuli items

To construct alternate lists, some strategies for picking fitting stimuli have to be determined. The picking strategy depends on the desired result, which here is to construct equal lists from a certain predefined set of stimuli items. But before the optimal strategy can be determined, there must first be a quantifiable measure of the optimality of these lists.

For the sake of simplicity, we start by defining optimality for the situation where the lists are desired to be equal and ignore the cases where the intervention study needs lists that differ in some aspect. The most important characteristic of equal lists is that they give the same scores to the same people. In the previous true-score framework, this would mean that the true scores are the same, but here, we introduce another, more probabilistic framework for looking at the scores.

The interest of this framework is the probability P_i of getting some item i correct. These probabilities depend not only on the item's specifics but also on the test takers' abilities. These probabilities can be estimated accurately with a large enough sample of test takers. A parametric model will be specified for the estimation. This parametric model will be the one-parameter logistic IRT model or Rasch model (Baker and Kim, 2014, p. 21), where the probability P_i is modelled as a function of the latent ability θ as

$$P_i(\theta) = \frac{1}{1 + e^{-(\theta - \beta_i)}}. \tag{3}$$

Parameter β_i in the (3) is called the difficulty parameter of the item. It is easy to see that if $\theta = \beta_i$, the probability of getting the item correct is exactly half, which is the common interpretation of the difficulty parameter. In the IRT framework, there are other more complex models, such as two- and three-parameter logistic models, that often better reflect the reality (Baker and Kim, 2014, p. 19-21), but as the purpose of the model is just to illustrate the concepts, the simpler model is preferred here.

With the probabilistic approach, the relationship between the true score and the observed score can be characterized similarly to (1), as the expected score for n items $E = \sum_{i=1}^{n} P_i$ can be substituted for the true score yielding decomposition for observed score

$$Score_O = \sum_{i=1}^{n} c_i = E + \varepsilon = \sum_{i=1}^{n} P_i + \varepsilon, \tag{4}$$

where c_i is a binary variable of whether item i was answered correctly and the ε is the error of the score. It is obvious here, that like in the true-score framework, the mean of ε is zero, but its variance is dependent on the probabilities, which adds difficulty to estimation.

With this framework, we can define the equality of the lists. In the optimal scenario, the expected scores E, E' and the errors $\varepsilon, \varepsilon'$ for both lists are equal. Since both the score and the error depend on θ, more exact would be to say that they must be equal for any given θ. Creating even small lists where this would be exactly true is a practically impossible task, and for this reason, approximative equality is used. If the errors for both lists are fairly small, their equality is not as important as the expected scores being equal.

As neither the expected score nor the error can be calculated from the observed score, other methods for estimating these must be used. One simple way would be to use averages, as the variance of the average error gets closer to zero as the sample size grows, which means that averages get closer to the population mean of the expected score. The problem is that errors can have different variances between lists, and the needed sample size for errors to become irrelevant could be large. Therefore, using IRT-based difficulties to conduct a proxy for the expected score can be a better alternative. The same difficulties lead to the same expected scores, matching the difficulties of the items between the lists.

To understand previous research on item selection, one must first be familiar with the concept of information function, as it is commonly used to select items. The information functions can be categorized into related concepts of item and test information function. The item information function is a function of θ that gets higher values for those θ that can be estimated more precisely with the item and lower values for those θ that can be estimated less precisely (Baker, 2001, p. 104). The test information function is the sum of the individual item information functions, and its values represent test precision on measuring a certain level of θ similar to item information function (Baker, 2001, p. 107). Under the Rasch model, the item information function is, (Baker, 2001, p. 110)

$$I_i(\theta) = P_i(\theta)(1 - P_i(\theta)). \tag{5}$$

Automated test construction started with the idea of constructing the shortest possible test that contained at least a specified amount of information in certain θ values (Theunissen, 1985). This practical problem was formulated as a 0/1-knapsack optimization problem, where all the items had equal weights, meaning the objective function to minimize was a sum of binary variables telling if the item was included. The framework itself was simple but allowed for building further automation for list construction, such as having some fixed number of items from different subcategories (Theunissen, 1986). This formulation had the weakness that it could produce a test information function with steep spikes on the chosen points, which was negated by a different formulation where the largest deviation from the target information was minimized (Van Der Linden, 1987).

The automated test construction framework was a powerful tool for a time, but its relation to parallel list creation was still unclear. This was the case until Boekkooi-Timminga (1987) extended the framework to create multiple tests simultaneously. Like the original, the information had certain thresholds in some predefined points, and the total number of items in all the tests was minimized. The article also presented an alternative formulation in which the absolute distances between the two test information functions were minimized.

Even if the Theunissen (1985) originally suggested the use of mathematical optimization formulation in test construction, it was not the first instance of using test information in test construction, as Lord (1977) suggested such almost a decade earlier. Based on the ideas of Lord, Ackerman (1989) proposed an alternative methodology for automated parallel test construction, which did not involve mathematical optimization, as Ackerman considered it too computationally demanding for the time. This method involved adding items iteratively by choosing the item that would fill the areas of the test information function still needed to reach the target curve.

One issue with IRT-based approaches is the interpretability of the test information, which makes it difficult for the test constructor to choose the adequate target information function. For this reason, Adema and van der Linden (1989) made an alternative formulation using classical test theory (CTT) by using reliability as an objective function. This line of research was extended by Armstrong et al. (1994), who presented a more efficient network-flow algorithm for solving the CTT-based optimization problem.

The algorithm's efficiency had a great emphasis on the following research on the topic, as binary optimization problems are notoriously hard to solve. Boekkooi-Timminga (1990) used the idea of clustering similar items to one equivalent class to limit options to choose from so the computational load could be reduced. The major emphasis of the research turned towards developing heuristic methods for solving the binary optimization problem related to item selection (Swanson and Stocking, 1993; van der Linden, 1996, 1998). Heuristics are methods that provide solutions fast but do not guarantee their optimality. Heuristics and their extension of metaheuristics will be further discussed later.

The research of automated test construction reached its logical conclusion as (Bejar et al., 2003) developed an automated on-fly test generation program. Instead of generating different test versions, the program created tests adaptively for each test taker. The tests generated this way were promising, as they had high test-retest reliability and high correlation with a non-adaptively generated test. This may not be the best approach for repeated testing, as generating tests on the fly could result in two tests that are not comparable if the method somehow fails. However, it must be noted that, with these approaches, the stimuli items could be guaranteed to be new every time.

All the IRT-based approaches are great for creating parallel test versions, but they have a major weakness: They require an item bank where each item has IRT parameters estimated for it. This implies that at least some data on each item is needed beforehand. For large item banks, collecting such data becomes an enormous task. Some of the models presented in this section put constraints on the content of the items. Generally, this approach ignores the additional characteristics of the items and only focuses on the estimated information function. This lack of generalizability is another major weakness of IRT-based test construction.

Due to these weaknesses, an alternative, more general framework for stimuli selection will be presented here. The basic idea of this framework is that the objective function is easily adaptable to any situation. In its most basic form, the optimization problem attempts to construct lists as equal as possible but is not limited to only this scenario. In intervention studies, for example, it can be desired that certain characteristics of the stimuli differ between the lists. Therefore, the objective function must be able to adapt accordingly. IRT parameters and information functions can still be included, but the optimization process is not dependent on them, as any item characteristic can be used as the basis of equating.

This general framework still has its flaws, as the reliability of the resulting test versions is unknown and may vary. In an IRT-based framework, the use of information functions helps with making tests similarly reliable. Here no such guarantee can be given, as there is often no data yet on how the stimuli behave. When data already exists, the reliability coefficients can be incorporated into the objective function. Validity is a harder question, as it is not easily quantifiable like reliability. This is an important reminder that even with automation, some additional consideration is in order. In the next chapter, a mathematical optimization problem corresponding to the previous description of the general stimuli selection framework is provided and discussed.

3 Stimuli selection as mathematical optimization problem

3.1 Formulating the problem

At the beginning of the list optimization, there is a dataset $\boldsymbol{X}$ that contains different attributes of all the n stimuli items. As there are items, features and lists which all need indexing, these indices will be fixed from the start, with i referring to items, j to features and k to lists. This data matrix has m rows and here m can be higher than the number of variables relevant to the optimization procedure, as the effect of any variable on the objective function can be mitigated with weights. This dataset is also assumed to have no missing data, as it would make little sense to use an unknown attribute as the basis of choosing a word.

To formulate a mathematical optimization problem corresponding to the practical stimuli selection problem described in the last section, some notation needs to be declared. The intention is to find l lists of q stimuli that satisfy some conditions concerning the similarity of certain features between lists. Conditions of features of interest are formulated as some objective function $f : \mathbb{R}^{n \times l} \to \mathbb{R}$ which gives smaller values in situations where conditions are better satisfied and vice versa. The decision variable used as input for an objective function is a boolean matrix $\boldsymbol{U}$ that holds information about which items from $\boldsymbol{X}$ belong to which lists.

For a real-life counterpart to correspond to the optimization problem, some constraints need to be set. Variables u_{ik} are boolean, meaning they get values of True/False. If u_{ik} happens to be true, the stimuli corresponding to the row i of $\boldsymbol{X}$ belongs to the list k. Mathematically, they are encoded as 0/1 with ones meaning that the item belongs to the list. Now it is possible to set list- and stimuli-wise constraints. No stimuli can belong to multiple lists at once as it would clash with the point of stimuli selection which was to make similar lists with different items. This problem can be avoided by adding a constraint where row-wise sums of $\boldsymbol{U}$ can be only one or zero (less or equal to one). Lastly, these lists must be the same predefined length. This can be achieved by setting up a constraint where column-wise sums of $\boldsymbol{U}$ need to be equal to q. Combining these defines the stimuli selection optimization problem in the most general form

$$
\begin{aligned}
\min \quad & f(\boldsymbol{U}) \\
\text{s. t.} \quad & \boldsymbol{U} \in [0,1]^{n \times l} \\
& \sum_{k=1}^{l} u_{ik} \leq 1, \quad i = 1, \dots, n \\
& \sum_{i=1}^{n} u_{ik} = q, \quad k = 1, \dots, l.
\end{aligned} \tag{6}
$$

As the constraints have been formulated, now the next thing is to choose a fitting objective function f. This task is difficult in general and usually, the best objective function can be quite case-specific. Here, we attempt to formulate an objective function that would be useful for most of the stimuli selection situations. The objective function will be formulated in a general form, with additional parameters that help to adapt to the exact situation at hand.

The usual statistical way of measuring the equality of two samples is the Student's t-test. More precisely, the t-test tests the null hypothesis that the two samples came from the same normal distribution. Originally, the two-sample t-test was formulated by Ronald Fisher in (Fisher, 1925). As Fisher's version's assumption of the equal variance was limiting in some cases, the t-test was generalized for two different populations by Welch (Welch, 1947). In the case of non-normal distributions, one would have to turn to a non-parametric test, such as Wilcoxon's signed-rank test (Wilcoxon, 1945), but it will not be discussed here.

Let now n_1 and n_2 be the sample sizes of samples 1 and 2 respectively. These samples come from populations that have means of μ_1 and μ_2 and variance of σ_1^2 and σ_2^2, respectively. An unbiased estimator for the population mean is the sample mean (proven in appendix A)

$$\bar{x}_k = \frac{1}{n}\sum_{i=1}^{n_k} x_{ik}, \quad k = 1, 2. \tag{7}$$

This means that the unbiased estimator of the population variance is the sample variance (proven in appendix A)

$$s_k^2 = \frac{1}{n_k - 1}\sum_{i=1}^{n_1}(\bar{x}_k - x_{ik})^2, \quad k = 1, 2 \tag{8}$$

Now we can define Welch's t-statistic as

$$t = \frac{(\bar{x}_1 - \bar{x}_2) - (\mu_1 - \mu_2)}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}. \tag{9}$$

Welch t-statistic follows approximately Student's t-distribution with degrees of freedom of *df* (Welch, 1947),

$$df(n_1, n_2, \sigma_1^2, \sigma_2^2) = \frac{\left(\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}\right)}{\frac{\sigma_1^4}{n_1^2(n_1-1)} + \frac{\sigma_1^4}{n_1^2(n_1-1)}}. \tag{10}$$

The important thing to note about this formulation of the t-test is that when we use the most common null hypothesis that the means of two populations are equal, the numerator of (9) simplifies to $(\bar{x}_1 - \bar{x}_2)$. Considering the desire to make lists as equal as possible, the desirable property for sample means and variances is to be as equal as possible. As the t-distribution has most of its probability mass close to 0, the extreme values are those that have large absolute values and as such they are undesirable.

In a stimuli selection situation when we change stimuli within samples to optimize the similarity of them, t-statistic changes mostly because of changes in difference of means. This is because the sample size is constant and large compared to changes in sample variance, as the stimuli features have been standardized to zero mean and unit variance. For this reason, even larger changes in the standard deviation of the sample have marginal effects on the t-statistic. Based on the t-statistic and previous observations, we can conclude that one good objective function or at least part of it could be the absolute difference of sample means $|\bar{x}_1 - \bar{x}_2|$.

In the most basic setting, we want to have two similar lists, and then the objective function would be the sum of absolute differences in list-wise means over all relevant features. The situation becomes a bit more complex if there are some features where we want to have a clear difference between lists. So instead of minimizing the difference, we want to maximize it. Helpfully, the principles of mathematical optimization tell us that the minimum of f is the same as the maximum of $-f$ (formally proven in appendix A). So by changing the sign to minus for those features that are desired to be maximized, we can put individual weights b_j for each feature.

In addition to giving the correct sign, the weights b_j can be used to give more importance to certain features if we allow other values than just 1 or -1. There are also some cases where it is desirable for a penalty of the increased difference in means to be something other than constant. This can be achieved through raising the absolute difference to some power p_j. Now we can finally formulate the basic form of the general objective function to be used later. By taking the absolute difference of list-wise means for each of the features j as $\bar{x}_{kj} = \frac{1}{n}\sum_{i=1}^{n} u_{ik}x_{ij}$ we can write the general objective function as

$$f(\boldsymbol{U}) = \sum_{j=1}^{m} b_j \left| \frac{1}{n}\sum_{i=1}^{n} u_{i1}x_{ij} - \frac{1}{n}\sum_{i=1}^{n} u_{i2}x_{ij} \right|^{p_j}. \tag{11}$$

This form of the objective function will be later used in numerical comparisons of the different methods. In numerical comparisons, there will be a few different values for b_j to see how robust the results are for changes in the objective function. Other variations of the objective function will not be tested numerically, but some possible extensions for previously derived form will be discussed in the next section.

3.2 Extensions of the objective function

The basic form of the objective function, as in (11) is good enough for the most basic cases where the desirable outcome is two lists that are similar or differ only in some of the features. In the case of more than two lists, deviation from overall mean or pairwise mean differences can be used. As the features are assumed to be normally distributed, by adding the absolute difference of the standard deviations to the objective function, distributions could be exactly matched. This is the case because the normal distribution is characterized only by mean and variance.

In situations where the specifications for the lists are more complex, the objective function needs to also be more complex. Additional ideas for objective function include a term for pairwise distances between lists or correlations between different features. These and many more can be found in Appendix 1 of (Armstrong et al., 2012a). As those mentioned before have already been implemented and well documented which means it is pointless to go through them. For this reason, this section focuses on giving a couple of ideas that have not been presented yet but could be useful.

One desirable property of the stimuli list in a practical situation could be that there is a steady difficulty curve inside the list. The difficulty here means the item response theory-based interpretation, where difficulty tells the amount of latent ability the test subject would at least need to have better than chance probability of success in the stimuli item. Let us assume that the difficulty of the item is the feature j, which means that the difficulty gap can be defined as

$$\min_{i\neq i'}|u_{ik}x_{ij} - u_{i'k}x_{i'j}|, \quad i, i' = 1, \ldots, n. \tag{12}$$

Now, as the intention was to minimize all gaps, the logical addition to the objective function is the maximum of these gaps. The mean of the gaps could also work, but it gives the possibility of there being one or more extreme outliers, which is undesirable. One challenge this will pose is that it favours the lists where all the stimuli have a similar difficulty, which in most cases is not practically fit for use, as the point was to make a steadily increasing difficulty instead of just minimal differences between difficulties. This challenge can be tackled by an additional penalty term for short difficulty ranges. This can be achieved by taking the difference of maximum and minimum or more simply maximum of absolute differences of the difficulties inside the list as defined

$$\max_{i,i'=1,\ldots,n}(u_{ik}u_{i'k})|u_{ik}x_{ij} - u_{i'k}x_{i'j}|. \tag{13}$$

To be useful in the objective function both of the previously presented terms have to be combined. Similarly to the original objective function, here the weights c_j are added. These are different and completely independent from previous weights b_j, as it makes little sense to have two different aspects of the objective function forced to be weighted equally. A combination of previously presented two terms and weights yields an additional term for the objective function

$$\sum_{j=1}^{m} c_j \left(\sum_{k=1}^{l} \left(\max_{i,i'=1,\ldots,n} \min_{i\neq i'}|u_{ik}x_{ij} - u_{i'k}x_{i'j}| - \max_{i,i'=1,\ldots,n}(u_{ik}u_{i'k})|u_{ik}x_{ij} - u_{i'k}x_{i'j}| \right) \right). \tag{14}$$

As properties of distributions of these features go, for some of them it would be desirable to have a "natural" distribution. This could be interpreted as an aim for as normal distribution as possible, as the normal distribution is more often than not considered the most "natural" out of all distributions. For many features stimuli could have, the normality is also intuitively the most likely distribution they follow at the population level. For example, height and intelligence are variables that approximately follow the normal distribution (Allen and Yen, 1979, p. 20).

As all features have already been standardized to have zero mean and unit variance, it means we can only look at deviation from the standard normal distribution. For defining measure for normality, a more simple notation is used where X is some random variable and $x_i,\ i = 1, \ldots, n$ are realizations of it. Only once we have defined this measure fully, the notation of the optimization problem will be brought to the formula.

Definition 1. Let X be some continuous random variable with a probability distribution of $f_X : \mathbb{R} \to \mathbb{R}$. *Differential entropy* $h(X)$ is defined as

$$h(X) = -\int_{-\infty}^{\infty} f_X(x) \log f_X(x) dx. \tag{15}$$

For our purposes, the most important feature of differential entropy is the fact that assuming constant variance it is maximized by a normally distributed random variable as stated by Theorem 1. This means that the larger the value of differential entropy we get, the more normal the underlying random variable is.

Theorem 1. *Let $S_\sigma = \{X \text{ is a continuous random variable} \mid E[X] = 0, Var(X) = \sigma\}$ and $Z \sim N(0, \sigma)$. The maximum of $h(X)$ in the set S_σ is $h(Z)$.*

Proof. The proof is presented in (Cover, 1991, p. 234) □

As all features are standardized to unit variance inside the data, the variance of the corresponding random variables can also be assumed to be one and the maximal differential entropy comes from the standard normal distribution and can be calculated exactly to be $h(Z) = \frac{1}{2}\log(2\pi e)$, when $Z \sim N(0, 1)$ (Cover, 1991, p. 225). The problem here is that we are dealing with samples instead of random variables, which in turn means that probability density functions are unknown, and some approximation is needed.

For the sake of the optimization problem set up previously, we would like to have a similarity measure between random variables behind the values of certain features inside one of the lists.

Definition 2. Let X be a continuous random variable for which $E[X] = 0$ and $Var(X) = 1$. Also, let the random variable $Z \sim N(0, 1)$. Now the *negentropy* of X is defined by

$$J(X) = h(Z) - h(X) = \frac{1}{2}\log(2\pi e) + \int_{-\infty}^{\infty} f_X(x) \log f_X(x) dx. \tag{16}$$

Negentropy has many useful properties for measuring the normality of a sample, as presented in the following theorem.

Theorem 2. *Let X be a continuous random variable for which $E[X] = 0$ and $Var(X) = 1$. Also, let the random variable $Z \sim N(0, 1)$. Now, negentropy has the following properties:*

(i) $J(X) \geq 0$

(ii) $J(X) = 0 \Leftrightarrow X \sim N(0, 1)$

(iii) $J(X) \approx \frac{1}{12}\left(E[X^3]^2 + \frac{1}{4}(E[X^4] - 3)^2\right)$.

Proof. (i) Theorem 1 states that $h(Z) \geq h(X)$ where $Z \sim N(0, 1)$. Therefore $J(X) = h(Z) - h(X) \geq 0$.

(ii) As proven in Theorem 1 $h(X) \leq \frac{1}{2}\log(2\pi e)$ with equality holding only if $Z \sim N(0,1)$ Therefore $J(X) = 0 \Leftrightarrow h(X) = h(Z) \Leftrightarrow X \sim N(0,1)$.

(iii) Proof is presented in (Jones and Sibson, 1987).

□

Now with properties stated in Theorem 2, especially (iii), we can see that negentropy could have many desirable properties to measure normality. However, the approximation is not a robust or accurate one, as there exist more complex and better alternatives (Hyvärinen, 1997). Still, for our purposes, it is a good enough measure of normality that has a simple estimator

$$J(\boldsymbol{x}) = \frac{1}{12}\left(\left(\frac{1}{n}\sum_{i=1}^{n} x_i^3\right)^2 + \frac{1}{4}\left(\frac{1}{n}\sum_{i=1}^{n} x_i^4 - 3\right)^2\right). \tag{17}$$

Here the random variable X has been replaced with the vector $\boldsymbol{x}$ containing n independent realizations of X.

It is important to note that the presented estimator of negentropy uses such estimators for skewness and kurtosis that are biased for non-normal samples. This is a major shortcoming as it means we are adding additional bias to an approximation which greatly decreases the accuracy. As this is a measure of normality, it would be a desirable property for the estimator to be accurate specifically for non-normal samples. Denoting previously used estimators as $g_1 = \frac{1}{n}\sum_{i=1} n(x_i - \bar{x})^3$ and $g_2 = \frac{1}{n}\sum_{i=1} n(x_i - \bar{x})^4$ we can have correction terms that make them unbiased for normal samples. Hence, we get new estimators

$$G_1 = \frac{\sqrt{n(n-1)}}{n-2} g_1 \tag{18}$$

and

$$G_2 = \frac{n-1}{(n-1)(n-3)}((n+1)g_2 + 6) \tag{19}$$

that are unbiased for normal samples and have the smallest mean-squared error in small samples ($n < 100$) of the commonly used estimators for skewness and kurtosis (Joanes and Gill, 1998). With these, we get a slightly more accurate approximation for negentropy as

$$J(\boldsymbol{x}) = \frac{1}{12}\left(\left(\frac{\sqrt{n(n-1)}}{n(n-2)}\sum_{i=1}^{n} x_i^3\right)^2 + \frac{1}{4}\left(\frac{(n-1)\left(\frac{n+1}{n}\sum_{i=1}^{n} x_i^4 + 6\right)}{(n-1)(n-3)} - 3\right)^2\right). \tag{20}$$

This approximation looks complicated, but for a computer, it involves only some basic arithmetic, which makes it quite usable for our purposes. To fit in our objective function, we need to harmonize the notation accordingly and add sums, as there is a need for control over which lists are desired to be normal and which are not. Doing this yields the form

$$\sum_{j=1}^{m}\sum_{k=1}^{l}\frac{d_j}{12}\left(\left(\frac{\sqrt{n(n-1)}}{n(n-2)}\sum_{i=1}^{n}u_{ik}x_{ij}^3\right)^2+\frac{1}{4}\left(\frac{(n-1)\left(\frac{n+1}{n}\sum_{i=1}^{n}u_{ik}x_{ij}^4+6\right)}{(n-1)(n-3)}-3\right)^2\right), \tag{21}$$

where the d_j is the individual weight that controls how much the normality of a certain feature is valued. By using negative weights, the features can be forced to be non-normal.

As shown in this section, almost every desirable property of the lists to be constructed can be transformed into mathematical form, which can then be used to optimize the lists accordingly. On the flip side, leaving the possibility for an arbitrary objective function means the methodology has to be able to adapt to them. As will be seen in the next section, this will lead to heuristic approaches, as it would be impossible to make an exact approach that could tackle such a wide variety of different objective functions.

3.3 Difficulty of global optimization

Considering integer nonlinear programming problem (6), binary variables in combination with possibly very complex objective functions make for a hard optimization problem. This section describes the difficulties that are common in global optimization in a general case, and additional challenges arising in the context of discrete optimization problems. The notation and definitions are similar to the ones used by Bagirov et al. (2014).

Creating efficient methods for discrete optimization similar to linear or convex optimization is almost impossible, as most of the problems are NP-hard (Sergienko and Shylo, 2006). An additional challenge with these problems is the fact that they can have a plethora of local minima, which are found instead of the global optimum. The challenges of local minima are first described considering a general continuous constrained optimization problem. Afterwards, the additional challenges posed by the discrete variables are discussed.

Definition 3. Let $f : \mathbb{R}^n \to \mathbb{R}$ and d be the Euclidean distance function. Then $\boldsymbol{x}^*$ is called *local minimum* of f if there exist such $\varepsilon > 0$ that

$$f(\boldsymbol{x}^*) \leq f(\boldsymbol{y}),\ \text{for all } \boldsymbol{y} \in \{\boldsymbol{y} \mid d(\boldsymbol{x},\boldsymbol{y}) < \varepsilon\}.$$

As these local minima are abundant, it means that finding the global minimum among these will be challenging. Finding the global minimum is always the ultimate goal, as many local minima can be substantially worse than the global minimum.

Definition 4. Let $f : \mathbb{R}^n \to \mathbb{R}$. Then $\boldsymbol{x}^*$ is called *global minimum* of f if

$$f(\boldsymbol{x}^*) \leq f(\boldsymbol{y}),\ \text{for all } \boldsymbol{y} \in \mathbb{R}^n.$$

As the local and global optima have been defined, the logical next step is to have tools for finding them. For finding these, one common way is to start from some point and continue in a direction where the function decreases. The set of these directions is often referred to as the cone of descending directions and follows the subsequent definition.

Definition 5. Let $f : \mathbb{R}^n \to \mathbb{R}$, $\boldsymbol{x} \in \mathbb{R}^n$ and $S \subseteq \mathbb{R}^n$ be the set of feasible points of the problem in question. Set

$$F_S(\boldsymbol{x}) = \{\boldsymbol{d} \in \mathbb{R}^n \mid \boldsymbol{x} + t\boldsymbol{d} \in S, \text{ for all } t \in (0, \varepsilon], \text{ for some } \varepsilon > 0\}$$

is called the *cone of feasible directions in* $\boldsymbol{x}$.

Definition 6. Let $f : \mathbb{R}^n \to \mathbb{R}$, $\boldsymbol{x} \in \mathbb{R}^n$. Set

$$D(\boldsymbol{x}) = \{\boldsymbol{d} \in \mathbb{R}^n \mid f(\boldsymbol{x}) < f(\boldsymbol{x} + t\boldsymbol{d}), \text{ for all } t \in (0, \varepsilon], \text{ for some } \varepsilon > 0\}$$

is called the *cone of descending directions in* $\boldsymbol{x}$.

Based on this definition, we can easily see that a point being local minimum is equivalent to the intersection of cone of feasible directions and cone of descending directions $F_S(\boldsymbol{x}) \cap D(\boldsymbol{x})$ being empty set at the current iteration's point. Many iterative optimization algorithms use a cone of descending directions as a base for choosing the point for the next iteration (Bagirov et al., 2014, p. 122). Many of these kinds of algorithms such as Hooke-Jeeves (Hooke and Jeeves, 1961) and gradient descent (Curry, 1944) follow some of the general ideas shown in algorithm 1. In cases where a plethora of local minima exist, this becomes problematic, as following only descending directions may lead to getting stuck in the local minimum. Following only a descending direction also makes the choosing of a starting point crucial for the optimal result. Figure 1 illustrates a simple example where not all starting points lead to finding the global optimum with the descending directions method.

Algorithm 1 General descent algorithm

```
Require: Some starting point $\boldsymbol{x}^* \in S$, objective function $f$, set of feasible points
  $S$.
  while $D_S \neq \varnothing$ do
      $D_S \leftarrow \varnothing$
      $F_S \leftarrow \{\boldsymbol{d} \in \mathbb{R}^n \mid \exists \varepsilon_1 > 0 : \boldsymbol{x}^* + t\boldsymbol{d} \in S,\ \forall t \in (0, \varepsilon_1]\}$
      for $\boldsymbol{d} \in F_S$ do
          if There exists $\varepsilon_2 > 0$ so that $f(\boldsymbol{x}^* + t\boldsymbol{d}) < f(\boldsymbol{x}^*),\ \forall t \in (0, \varepsilon_2]$ then
              $D \leftarrow \{D, \boldsymbol{d}\}$
          end if
      end for
      Choose $\boldsymbol{d}$ among $D$ by some method
      Choose some $t < \min(\varepsilon_1, \varepsilon_2)$
      $\boldsymbol{x}^* \leftarrow \boldsymbol{x}^* + t\boldsymbol{d}$
  end while
  return $\boldsymbol{x}^*$
```

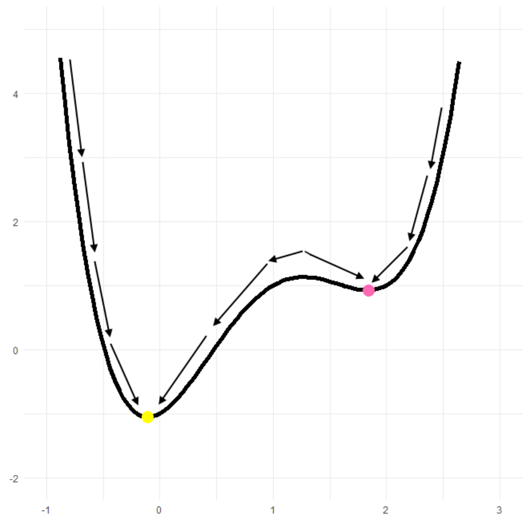

Figure 1: Example of the behaviour of the descending directions optimization algorithm

Previously discussed challenges are just the ones that are present in continuous optimization. As the problem studied here has binary variables, poses some additional difficulties for solving the problem. Especially, the equality constraints limit the direction it is possible to move at any iteration, though they are not an unsolvable issue, as they can be dealt with by penalty function methods (e.g. Di Pillo and Grippo, 1989). The greater challenge here is the binary decision variables, as many of the methods and even definitions depend on the continuity of the variables to function.

However, binary optimization problems have a simple exact-solving algorithm known as an exhaustive search where every single possible solution is checked and evaluated. The drawback of this approach is the curse of dimensionality, as the number of solutions to be checked grows exponentially. In the case of the problem (6) the number of possible solutions is

$$\binom{n}{lq} = \frac{n!}{(lq)!(n-lq)!}. \tag{22}$$

For a simple case such as choosing two 16-item lists among 50 possible items, there exist well over 10^{13} possible combinations. This means that if we would like to go through all of them in an hour, the computer would need to be able to calculate billions of objective function values per second, which would be a lot of computational power for such a simple task.

As the exhaustive search is out of the question, the other exact option would be the Branch-and-Bound algorithm (e.g. Kolesar, 1967), which limits explored search space through upper and lower bounds. The drawbacks here are similar to exhaustive search as the computational complexity can get out of hand and in the worst case the algorithm is equal to exhaustive search in terms of the computational complexity (Axehill and Morari, 2010).

The classical approximation technique of relaxation where discrete variables are turned into continuous is also often used for combinatorial problems as it can benefit from smoothness, convexity or continuity of the objective function for quicker convergence. In some cases, relaxation can be quite a powerful technique as it expands the variety of tools that can be used in solving the problem. If the objective function is both convex and smooth with constraints being well-behaved, this means that finding the global optimum of the relaxed problem is fairly straightforward. Even in the convex and smooth case, relaxed problem can have substantially different minimum than the original problem (example in Appendix A).

In the case of the stimuli optimization problem (6) with the most basic form of the objective function (11) difficulties arise from the equality constraints and the absolute value function in the objective function. As the problem has been already relaxed, it should not worsen the solution all too much if the equality constraint is turned to two inequality constraints, for example having the sum be in $[q-0.5, q+0.5]$. The challenge posed by the absolute value function can be easily dealt with by having $p_j = 2, \forall j = 1, \dots, m$ which makes the objective smooth, with the partial derivative for item i and list k is

$$\frac{\partial f(\boldsymbol{U})}{\partial u_{ik}} = \sum_{j=1}^{m} b_j (-1)^{k+1} 2 x_{ij} \left(\frac{1}{n} \sum_{i=1}^{n} u_{i1} x_{ij} - \frac{1}{n} \sum_{i=1}^{n} u_{i2} x_{ij} \right), \quad k = 1, \dots, l. \quad (23)$$

Even though it has been shown that derivative-based methods could be used for certain parameter choices, some other parametrization or objective functions will make things more difficult. The basic form of the objective function can not be guaranteed to be convex even for $p_j = 2$, which limits the solving of the relaxed problem. Even more, challenges are posed for $p_j \neq 2$ and the extensions presented in the section 3.2, as they can make the objective function nonsmooth and nonconvex. With this, it is clear that there is no way of making a reliable all-around optimizer for stimuli selection problems with the relaxation of the problem.

With exact approaches and relaxation out of the question, heuristics are the go-to way of solving hard combinatorial problems. The definition of heuristic as a word is "serving to discover" or "aid to discovery" (Michael, 1972). Here the heuristic is a method, which uses a rule-of-thumb or some strategy of simplifying the solving process of some problem. The drawback of heuristics is that by simplifying the solving process, there is no way to guarantee that the optimal solution is or even can be found. Heuristics often use greedy and local searching procedures that are problematic when applied to problems with many local optima.

The key to solving this challenge is widening the scope by accepting sometimes worse solutions than the one in the current iteration. Methods that incorporate heuristics in combination with a more global search scope are referred to as metaheuristics (Blum and Roli, 2001). The next section will discuss the previously attempted methods for the stimuli selection problem, some of which use metaheuristics that are presented in more depth in Chapter 4.

3.4 Previously used methods for stimuli selection

Previous to the development of a more sophisticated methodology, the selection of stimuli was done by experts using heuristic measures to construct lists that were similar enough to the human eye. Therefore, the methods have been strongly based on objective functions measuring the similarity of different lists. Used objective functions differ between methods, as some used difference in list-wise means to measure similarity and others have paired stimuli to use more exact matches between lists.

Automatized test construction depicted in section 2.4 cannot be considered stimuli selection in a context this thesis attempts to establish, as they are too reliant on IRT estimates. For this reason, stimuli selection will be defined now as an act of creating two or more test versions, which are created by matching (or differentiating) items based on their features. These features can be IRT parameters, but the focus is on finding general methods that do not need to rely on IRT.

The first example of automatized stimuli selection is the matching of items into pairs based on the Euclidean distance between items. The simplest method of this sort is EQUIWORD (Lahl and Pietrowsky, 2006). The lists are constructed by calculating all pairwise distances and choosing those with the lowest distances. EQUIWORD included a few other measures of similarity in addition to Euclidean distance, such as Mahalanobis distance and correlation coefficient (Lahl and Pietrowsky, 2006), and allowed for maximization of list-wise differences or some pairing of the items between lists.

A more complex approach similar to EQUIWORD is Match (van Casteren and Davis, 2007) where instead of taking one dataset from where to construct the list, it takes one dataset for each of the lists. This change allows features to be different between lists, as it is possible to have one list created from a set which includes only high values of certain features and another one from a set with low values. Match uses a similar approach to EQUIWORD in the sense that it makes tuples of items from different sets, intending to have the best possible matching items from each of the sets in each tuple. In each iteration, the best matching set of items in some tuple is chosen, and those items are deleted from other tuples. This continues as long as there is either a ready solution or one of the tuples no longer has any items. A possible stopping condition is also if the solution in the making starts to look inferior to the current best solution. Then the algorithm backtracks to the last matching decision and takes a turn to a new direction. The search goes on as long as there are new directions to take, or when the researcher decides that the algorithm has been going on for too long. The weakness of this kind of global search algorithm is that it gets computationally really intensive even for simple problems with small initial sets of stimuli. Match tries to combat this with an additional heuristic that decides when a direction is deemed inferior, which on the other hand could cause global optimum to never be found.

Previous methods were designed to specifically create two or more balanced lists, where balance meant minimal pairwise dissimilarity between matched items. In certain research settings, it is desirable to have certain features of the test items to be matched and others as far apart from each other as possible. The problem posed by such settings can be tackled with previous methods by creating high and low lists for said features. For researchers, this means extra work and added subjectivity to the results, as thresholds for low and high values in these features have to be set. To combat such a problem, the following methods use the objective function described in section 3.1 to measure the optimality of lists.

As the number of available stimuli increases, the number of possible lists grows exponentially. Constraints of the problem combined with possibly nonsmooth objective function (only smooth if $p_j = 2$ for all $j = 1, \dots m$ and no extra terms added to the (11)) make the stimuli selection problem impossible to solve exactly, even for small sets of stimuli. This means that heuristic methods need to be used to solve the optimum. With heuristics, there is no guarantee of finding a global optimum, but the methods still give good enough solutions for most of the problems. These methods often have trade-offs between the computational time used and the quality of the solution.

Most basic heuristics include such algorithms as local search, which does an efficient search of a certain area where the optimum could lie based on some rule of thumb. For more complex problems, these simple heuristics might not provide good solutions, as they easily get stuck on a local optimum. Metaheuristics on the other hand are much better at finding local optima closer to the global optimum, as they add in some way to widening the scope of the search (Blum and Roli, 2001). This widening scope could mean conducting local searches in multiple areas or forcing the search to move on if it has been stuck in a certain area for too long.

One of the most common types of metaheuristics is so-called bio-inspired metaheuristics (Almufti et al., 2019), which try to mimic the functions of real biological organisms or processes. The mimicked organism could for example be an ant colony or a flock of birds. Some bio-inspired methods try to mimic the evolution process itself by the means of taking different solutions and taking parts of them to produce more optimal solutions. These evolution-mimicking methods are referred to as genetic algorithms (e.g. Goldberg, 1989) and have been used to solve the stimuli selection problem (Coupé, 2011). As the program BALI made by Coupé is no longer available online and the conference paper has no clear explanation of the details of said method, it will not be discussed further here.

Fortunately, BALI is not the only instance of using a genetic algorithm in stimuli selection problems, as (Lintz et al., 2021) developed another list balancing method based on it. Instead of the objective function, the p-values of the t-test are to measure the fitness of a solution. Here, a certain parent solution is used in iterations as long as it takes for the child solution to be superior. Then this child solution will be used as a parent for the next iteration. The child solution here is generated by swapping one of the words from the parent solution to the one from the bank of unused stimuli. The weakness of this kind of simple genetic algorithm is that it cannot escape local optimum, as that would mean accepting an inferior solution.

Another type of real-life-inspired metaheuristics is simulated annealing, which takes its inspiration from material physics, where the temperature of a substance is lowered slowly to achieve a more optimal structure. As the optimization problems do not naturally have this kind of temperature in them, the idea of temperature is included in the method, where it serves the purpose of dictating how easily we accept worse than the current solutions. The key idea here is to start with a higher temperature value and anneal to a value where only improving solutions are accepted, as proposed in (Kirkpatrick et al., 1983). Simulated annealing has been used in stimuli selection in (Armstrong et al., 2012a) and it will be discussed in detail in the next chapter as it will be the baseline method to which the proposed method shall be compared.

In addition to metaheuristic approaches, the stimuli selection problem has been tried to solve by reformulating it into a clustering problem. In clustering, the data is divided into similar groups called clusters based on some similarity measure. One of the most common types of clustering is so-called k-means clustering (MacQueen, 1967), where data is divided to k clusters by minimizing the sum of squares of within-cluster Euclidean distances. This approach was used for stimuli selection in (Guasch et al., 2017) where an additional algorithm was proposed to combat the biggest challenge of the k-means algorithm which is the selection of the k. After the clusters have been acquired, it is possible to divide them into different lists with expected results of similar lists, as the clusters are supposed to have the most similar items that can be grouped within the data. The clustering problem is a very difficult optimization problem as the number of local optima can be very large and k-means is a simple local search heuristic, which means there are many drawbacks in this approach.

As the clustering aims for the different clusters to be as well separated as possible by minimizing within-cluster variability, there has also been a philosophically different approach to clustering where the aim is to have clusters be similar to each other. This way of thinking is opposite to clustering as now the goal is to maximize within-cluster variance and fittingly this way of clustering has been labeled as anticlustering (Valev, 1998). For stimuli selection purposes, anticlustering is a potential candidate, as it can quickly and reliably divide small datasets into equal parts (Papenberg and Klau, 2021). The problems with this approach arise from the fact that the method partitions the whole dataset, which means that the size of the lists is determined by the size of the dataset divided by the number of the lists. Also, anticlustering cannot account for the desire for any other properties of the lists other than similarity.

The common feature for most of these methods is that researchers behind them are more concerned about the ease of use their software gives for other researchers than how well the used method performs. This is even more clear as the researchers even today still develop methods where ease of use is considered high as seen by (Göbel et al., 2023) where principal component analysis (PCA) is used to reduce the dimension of the data to two, hence giving the possibility of visually grouping similar items to be included in different lists. This approach after all is mostly just a pen-and-paper counterpart for EQUIWORD and Match with the addition of information loss introduced by PCA, not to mention the subjective nature of visually choosing stimuli similar to each other.

The anticlustering example has a unique take considering the rest of the field as different clustering approaches were compared in the numerical examples (Papenberg and Klau, 2021), but as the datasets were small, and the method had strict limitations regarding its use cases, there is no real information about which method is performance-wise the best for stimuli selection and is there even a one-for-all the best method. Considering that methods for hard optimization problems such as the travelling salesman problem have been studied for decades without finding a superior method, it is highly unlikely that any of the methods could reign supreme in all or even most of the cases.

4 Metaheuristic optimization methods

4.1 Simulated annealing

As seen in the last chapter's review of the previous methodology applied to the stimuli selection problem, it is easy to see that most methods have some limitations which could affect acquired solutions negatively. From these, the SOS!-algorithm (Armstrong et al., 2012a) shows potential and versatility that other methods cannot rival. The possibility to greatly modify the objective function to meet the needs of the situation at hand is a commodity that almost every stimuli selection algorithm should include, but it is rarely implemented or even thought of. In addition to being the most versatile of the methods, the simulated annealing approach also gives the method a better ability to conduct more global searches to find near-optimal solutions for harder problems. All-in-all SOS! seems to be a gold standard in the field where additional research feels particularly necessary.

As the original SOS! is written as a MATLAB program with GUI in mind, for the sake of comparability it is necessary to rewrite the program in the same programming language as the proposed method. Implementation might differ a bit from the original SOS! so it is essential to go through the exact algorithm to be implemented here.

The original simulated annealing algorithm introduced in (Kirkpatrick et al., 1983) tightly follows the idea of sometimes "going uphill" to widen the search scope. This is implemented by accepting proposed swaps always if it improves the solution and otherwise with a probability that is a decreasing function of both temperature and the change of the objective function $\Delta f = f(\boldsymbol{U}') - f(\boldsymbol{U})$, where $\boldsymbol{U}$ is the solution before the swap and $\boldsymbol{U}'$ is after. The probability is then $p_{original}(\Delta f, T) = e^{-\frac{\Delta f}{T}}$, where compared to the original notation Boltzmann constant is included in T. In (Armstrong et al., 2012a) the approach is different as the probability weights come from a sigmoid function

$$p_{sos}(\Delta f, T) = \frac{1}{1 + e^{\frac{\Delta f}{T}}}, \tag{24}$$

which means that at very high temperature values the swaps that improve and reduce the objective function (negative and positive Δf values) are almost equally likely to happen.

This definition for probability weights of the swaps means that compared to the original simulated annealing this approach will likely make more reducing swaps as in the original the improving swaps were guaranteed to happen. Now, regardless of the used probability weight, the method is presented in the following algorithm.

While algorithm 2 is mostly straightforward, it introduces yet ambiguous terms such as stopping criterion and thermal equilibrium. The stopping criterion can very well be anything defined by a researcher, yet the most common choice is to stop when the algorithm reaches some equilibrium or otherwise, the result has not changed in the recent iterations. In SOS! there are two stopping criteria. The first one is if the predefined statistical criteria have been met, such as the p-value of the t-test

Algorithm 2 Simulated annealing for stimuli selection

```
Require: Temperature $T > 0$, annealing rate $0 < r < 1$
  Generate starting solution $\boldsymbol{U}$ at random
  while Stopping criterion is not met do
      Choose randomly $u_{ik} = 1$ for a swap, where $i = 1, \dots, n, k = 1, \dots, l$
      Choose randomly $u_{i'k'} = 0$ for a swap, where $i' = 1, \dots, n, k' = 1, \dots, l$
      Calculate $p(\Delta F, T)$ for the swap $u_{ik} = 0$, $u_{i'k'} = 1$
      if $p(\Delta f, T) > u$, where $u \sim U(0, 1)$ then
          Set $u_{ik} = 0$ and $u_{i'k'} = 1$
      end if
      if Thermal equilibrium or fixed number of iterations is reached then
          $T \leftarrow r \cdot T$
      end if
  end while
  return Resulting solution $\boldsymbol{U}$
```

is above 0.5 for those features where the difference is minimized and below 0.05 for ones to be maximized. Depending on the goals of the optimization, there can also be other statistical tests involved. The secondary stopping criterion is if there has not been a single swap for a fixed number of iterations, the algorithm terminates. Since one of the goals is to measure the goodness of solutions these methods produce, it makes little to no sense to have the process terminate when a good enough solution has been found. For that reason here the only stopping criterion will be that if no swaps happen for some number of iterations, this indicates that we have likely found at least a local minimum.

Thermal equilibrium refers to a point where the algorithm has reached a stationary state for a certain temperature, or more precisely "Thermal equilibrium is the point at which the probability of swapping to a given item has stabilized at a particular value across all items"(Armstrong et al., 2012b). Evaluation of the probability distribution of all swaps is difficult, which leads to the use of a proxy for this distribution. The used proxy will be the distribution of the objective function changes Δf for each of the swaps. Even though not very accurate, a t-test will be used to measure if the distributions at two different iterations will be equal with the statistical criterion of $p > 0.5$.

Calculation of Δf is a time-consuming process and as it is only a proxy for the thermal equilibrium, it is not guaranteed to be an accurate measure of when the temperature should be lowered. For this reason more simple approach of lowering temperature will be used here. This does have the downside of annealing temperature at a fixed rate instead of a dynamic rate. The positive side is that there is no need to estimate whether or not thermal equilibrium is reached, which adds extra steps and parameters.

4.2 Other single-point search metaheuristics

The field of metaheuristics consists of a wide variety of methods that employ different tactics to the problem at hand. Variety means that the classification of metaheuristics is an interesting question in itself. Out of all the different classifications presented in (Blum and Roli, 2001), the most useful for this thesis is the classification between single-point and population-based approaches. The distinction between these two classes is whether the algorithm operates on a single solution or a population of solutions at each iteration.

Only two of the presented methods will be used for stimuli selection, which means there is little sense in presenting them all in a specific form. For this more general approach, the following more general optimization problem is used

$$\begin{aligned} \min \quad & F(\boldsymbol{x}) \\ \text{s. t.} \quad & \boldsymbol{x} \in S. \end{aligned} \tag{25}$$

With simulated annealing presented in the previous section, one of the most used single-point algorithms is tabu search (Glover, 1986). In tabu search, the way to avoid getting stuck in the local minimum is through a so-called tabu list that is excluded from the possible choices of the solution for the next iteration. In its most basic form as presented in algorithm 3, tabu search chooses the best possible neighbouring solution that is not on the tabu list and adds the current solution to the tabu list where it stays for some fixed number of iterations or indefinitely. The definition of neighbouring solutions and the neighbourhood $N(\boldsymbol{x})$ depends on the problem. For continuous problems $N(\boldsymbol{x})$ can be all points that are within a certain distance of $\boldsymbol{x}$ and in combinatorial problems the natural way to define $N(\boldsymbol{x})$ is those solutions that can be reached with one swap to $\boldsymbol{x}$. The more complex versions of tabu search might incorporate a dynamically sized tabu list or different strategies for deleting older items from tabu list (Glover, 1990).

Algorithm 3 Tabu search algorithm

```
Require: Feasible set for the problem $S$, used neighbourhood structure $N(\boldsymbol{x}) \subseteq S$
  Generate random solution $\boldsymbol{x} \in S$
  Initialize the tabu list $\mathcal{T} \leftarrow \varnothing$
  while Stopping criterion is not met do
      Choose $\boldsymbol{x} \leftarrow \arg\min_{\boldsymbol{x} \in N(\boldsymbol{x}) \setminus \mathcal{T}} F(\boldsymbol{x})$ from neighbourhood $N(\boldsymbol{x}) \setminus \mathcal{T}$ of previous $\boldsymbol{x}$
      Update tabu list $\mathcal{T}$ with $\boldsymbol{x}$ and drop older out if necessary
      if $F(\boldsymbol{x}) < F(\boldsymbol{x}^*)$ then
          $\boldsymbol{x}^* \leftarrow \boldsymbol{x}$
      end if
  end while
  return $\boldsymbol{x}^*$
```

Greedy randomized adaptive search procedure (Feo and Resende, 1995) or GRASP for short is a simple metaheuristic where the solution is found by starting a local search from different promising points sequentially. In every iteration, the solution is started from scratch and constructed through an adaptive heuristic that chooses the next part of the solution among the list of the best solutions called the restricted candidate list randomly. This means that the solution is constructed iteratively by randomly choosing part of the solution from some defined number of the most promising solutions, after which the heuristic of determining the promising solutions is changed according to the already constructed solution. Combining these steps yields the algorithm 4.

Algorithm 4 GRASP algorithm

```
Require: The size of restricted candidate list (RCL) $m$
  Initialize the best solution $\boldsymbol{x}^* \in S$ at random
  Initialize RCL
  while Stopping criterion is not met do
      Construct new solution $\boldsymbol{x}$ by taking parts of random solutions in RCL while
  updating RCL after every new part
      Conduct a local search to improve solution $\boldsymbol{x}$
      if $F(\boldsymbol{x}) < F(\boldsymbol{x}^*)$ then
          $\boldsymbol{x}^* \leftarrow \boldsymbol{x}$
      end if
  end while
  return $\boldsymbol{x}^*$
```

Iterated local search (Martin et al., 1992) is likely the simplest local search-based metaheuristic as it only involves conducting sequential local searches by perturbing the previously acquired solution to escape the possible local optimum. Perturbation can be in the most simple form a fixed step in a random direction, but adaptive step sizes and non-random directions are possible also. The solution can also be accepted with differing criteria, with one extreme being accepting all solutions and the other one being accepting only improving ones. To achieve the best outcome, it is important to find a balance between accepting some worse solutions to escape the local optimum and at the same time not wasting time searching far away from optimal solutions. One way of doing this is incorporating an annealing schedule similar to simulated annealing as an acceptance criterion. In general, the iterated local search can be presented as the following algorithm.

Algorithm 5 Iterated local search algorithm

```
  Initialize some solution $\boldsymbol{x} \in S$ at random
  Conduct a local search to $\boldsymbol{x}$ and note solution as $\boldsymbol{x}^*$
  while Stopping criterion is not met do
      Perturb $\boldsymbol{x}^*$ in accordance to iteration history to obtain $\boldsymbol{x}$
      Conduct local search starting from $\boldsymbol{x}$ to obtain new $\boldsymbol{x}$
      if Solution found by local search $\boldsymbol{x}$ is good enough then
          $\boldsymbol{x}^* \leftarrow \boldsymbol{x}$
      end if
  end while
  return $\boldsymbol{x}^*$
```

In variable neighbourhood search (Mladenović and Hansen, 1997) the idea is similar to iterated local search where sequential local searches are constructed starting at the perturbed solution of the last local search. The acceptance criterion here is improving solutions only, but the perturbation strategy differs greatly. As the name would suggest, the perturbation is selected randomly from some fixed number k of neighbourhoods and every one of them is searched after another until a stopping criterion is met. These neighbourhoods are not fixed, as they are a structure that depends on the last acquired solution. A common approach is to have the sequence of the neighbourhood structures increase in size (Blum and Roli, 2001). Like in tabu search, the neighbourhood structures depend on the characteristics of the problem. For continuous problems, they can be all points in some distances within the current solution and for discrete problems all solutions that can be acquired with some fixed number of swaps. With these, we end with the algorithm 6.

Algorithm 6 Variable neighbourhood search algorithm

```
Require: A set of neighbourhood structures $N_1(\boldsymbol{x}), \dots, N_k(\boldsymbol{x}) \subseteq S$, for any $\boldsymbol{x} \in S$
  Initialize some solution $\boldsymbol{x}^* \in S$ at random
  while Stopping criterion is not met do
      $j \leftarrow 1$
      while $j \leq k$ do
          Generate some $\boldsymbol{x}' \in N_j(\boldsymbol{x}^*)$ randomly
          Generate $\boldsymbol{x}''$ by conducting local search starting from $\boldsymbol{x}'$
          if $F(\boldsymbol{x}'') < F(\boldsymbol{x}^*)$ then
              $\boldsymbol{x}^* \leftarrow \boldsymbol{x}''$
              $j \leftarrow 1$
          else
              $j \leftarrow j + 1$
          end if
      end while
  end while
  return $\boldsymbol{x}^*$
```

Similarly to other local search-based methods, guided local search (Tsang and Voudouris, 1997) takes sequential local searches with a perturbation method. Here, the perturbation method involves having a penalized objective function that changes from iteration to iteration depending on the solutions acquired. The penalty is added to the features that are present in the current solution to force the search out of the local optimum. The meaning of the feature here is ambiguous and dependent on the type of problem at hand. In graph-related problems, it could refer to certain arches, and in combinatorial problems, it could be the items in solution. For a general problem (25), defining penalized objective function is hard and an additive structure $g(\boldsymbol{x}) = F(\boldsymbol{x}) + p(\boldsymbol{x})$ is used. In a general form, the guided local search is presented as an algorithm 7.

Single-point metaheuristics presented in this section were the classical ones that often get the most attention. The field of single-point metaheuristics is vast, but most of them are inspired by the methods presented here. Other single-point metaheuristics include for example the demon algorithm (Zimmermann and Salamon, 1992), the great deluge algorithm (Dueck, 1993), threshold accepting (Dueck and Scheuer, 1990) and the noising method (Charon and Hudry, 1993).

Algorithm 7 Guided local search algorithm

```
Require: A rule for updating penalization function $p(\boldsymbol{x})$
  Initialize some solution $\boldsymbol{x} \in S$ at random
  while Stopping criterion is not met do
      Conduct local search from $\boldsymbol{x}$ with updated $g$ to get a new $\boldsymbol{x}$
      Update penalization function $p(\boldsymbol{x})$
      if $F(\boldsymbol{x}) < F(\boldsymbol{x}^*)$ then
          $\boldsymbol{x}^* \leftarrow \boldsymbol{x}$
      end if
  end while
  return $\boldsymbol{x}^*$
```

4.3 Population-based metaheuristics

As opposed to single-solution metaheuristics, population-based methods involve a plethora of solutions that form a so-called population at every iteration. This group of solutions is refined at each iteration to form a new generation of solutions, with the end goal of creating a generation that includes the global optimum. As the idea of a population evolving through generations comes from nature, it is not a surprise that almost every method of this class is inspired by biological phenomena. Some of the methods in this class are closer to a subclass of methods than singular methods, as the ideas behind them can be interpreted in many ways and some of these methods are highly use-case sensitive. Another key prospect of population-based methods is that they usually have plenty of parameters and can be quite sensitive to the initial choice of them.

The idea of making a machine learn through an evolutionary process was first introduced by Alan Turing in his famous article about "The Imitation Game" (Turing, 1950). Not long after this proposition, the first steps were taken in the simulation of the evolution by a computer (Fraser, 1957). These simulations paved the way for a class of optimization algorithms known as genetic algorithms. As the genetic algorithm is more of an umbrella term than any particular algorithm it is difficult to define, but generally, they at least include the steps presented in the algorithm 8 (Goldberg, 1989). As for how each step of the algorithm is done depends heavily on the application and chosen approach. In the case of the combinatorial problems the definition through the metaphor of evolution is easier as the solution resembles more closely the actual DNA but approaches for continuous optimization problems exist too (e.g. Chelouah and Siarry, 2000).

Algorithm 8 Genetic algorithm

```
  Generate initial population of solutions
  while Stopping criterion is not met do
      Reproduce new solutions from the solutions of the last generation
      Crossover the solutions to create a new generation
      Mutate the created solutions to add randomness
  end while
```

As the algorithm 8 has been written in a very general form, it might be difficult to understand the meaning of every step. Reproduction in its most simple form is cloning the solutions of the previous generation with the proportion of the fitness. Crossover means shuffling parts of the reproduced solutions to create new solutions that take on average the best parts of the previous generation, as those should be the most represented in the pool of reproduced solutions. Together these two create in a sense a heuristic population counterpart to greedy local search as mostly the best parts of the previous solutions are combined to create new ones and therefore aspects of certain fit solutions get enriched through the generations. Problems the metaheuristics target are usually too complex for this kind of greedy algorithm to work, so for that reason, the mutations are introduced to escape the local optima the algorithm would otherwise get stuck in.

Scatter search (Glover, 1977) is another classical population-based method that is more of a blend between the ideas of genetic algorithm and local search-based metaheuristics, as it employs the local search procedure to each generation but instead of directly forcing the iteration out of local minima, it employs a population of solutions that are in different areas of the search space to avoid such problem. By taking linear combinations of the solutions in different areas, even if all the solutions are in local minima their combination might be out of their area of influence and that way can reach a new area of the search space. Algorithm 9 has a vague description of the specific process scatter search optimization follows.

Algorithm 9 Scatter search optimization algorithm

```
Initialize a population of solutions
Choose a subset of these solutions to become a reference solutions
while Stopping criterion is not met do
    Create a new population as the linear combinations of the reference solutions
    Apply the local search procedure to each of the generated solutions
    The best solutions found by local search become the new reference solutions
end while
```

Ant colony optimization (ACO) (Dorigo and Di Caro, 1999) is one of the earliest examples of an optimization algorithm that tries to mimic the behaviour of a group of biological organisms. As the name would suggest, ACO mimics the behaviour of a colony of ants as they look for the optimal routes. The key idea here is that the choice of the path of an ant is a random-weighted draw from the set of possible paths. The weights here are the pheromone intensity of the said paths. As each ant lays down pheromones while traversing a path and shorter paths mean that the ant gets to lay pheromones to more paths, this leads to shorter paths having higher pheromone intensities and therefore higher likelihood to be chosen by other ants. Pheromones also deteriorate, so if no ants choose a certain path for a while, the trail goes cold and is even less likely to be chosen in the future. With a clear metaphorical connection to path optimization, as the algorithm is quite dependent on graph structure, it has still been used for different classes of problems such as prediction (Zhang et al., 2013) even though these approaches need to somehow convert at least part of the problem to a graph-form. The concept of the ACO algorithm is presented as algorithm 10.

Algorithm 10 Ant colony optimization algorithm

```
while Stopping criterion is not met do
    while Resources available do
        Create a new ant
        while Ants solution is incomplete do
            Ant moves to a neighbouring solution probabilistically based on
            pheromone trails and ants' memory
            Ant lays down pheromones on taken path
        end while
        Ants dies and frees the allocated resources
    end while
    Decrease the intensity of the old pheromone trail
    Carry out daemon actions that cannot be done by a single ant (optional)
end while
return $\boldsymbol{x}$
```

Particle swarm optimization (Kennedy and Eberhart, 1995) is another example of taking the behaviour of a group of animals and turning it into an algorithm. Here the inspiration is the flock of birds flying in sync. The basic idea here is that there are m particles moving with velocity, $\boldsymbol{v}_i^j$ where the velocity is the product of the speed and direction. The position is updated at every iteration by $\boldsymbol{x}_{i+1}^j = \boldsymbol{x}_i^j + \boldsymbol{v}_i^j$. The original simple formulation of the velocity update formula is

$$\boldsymbol{v}_{i+1}^j = \boldsymbol{v}_i^j + 2u_1(\boldsymbol{x}^j - \boldsymbol{x}_i^j) + 2u_2(\boldsymbol{x}^* - \boldsymbol{x}_i^j), \tag{26}$$

where $\boldsymbol{x}^j$ is the position of the best solution seen so far by the particle in question, $\boldsymbol{x}^*$ is the position of the best solution seen by any of the particles and $u_1, u_2 \sim U(0, 1)$ are random numbers.

Even though the basic PSO algorithm 11 is really simple at its core, the velocity updating rule can be formulated in more complex manners to further improve the method. The updated formula can in a sense be divided into global and local search parts where the term $\boldsymbol{v}_i^j$ is the term which amplifies the more global search procedure, so it is possible to control the locality (or globality) of the search by introducing an inertia weight (Shi and Eberhart, 1998). This weight can also be changed over the iterations, resembling the temperature annealing of the simulated annealing approach. As might be apparent at this point, the PSO algorithm is intended to be used for unconstrained and continuous optimization problems. For discrete problems, the modified version by (Chen et al., 2010) works fine, and in the case of constrained optimization, one can use penalty function methods to turn the problem into an unconstrained one.

Inspiration for different population-based metaheuristics does not end with ants and birds, as different metaphor-based methods are abundant. These include the algorithms inspired by flashing of fireflies (Yang, 2009), behaviour of bee colonies (Pham and Castellani, 2009; Karaboga, 2005), echolocation of the bats (Yang, 2010), improvisation of the musicians (Geem et al., 2001) and movements of the water (Eskandar et al., 2012). Frequent use of metaphors to justify methodological choices has also attracted criticism, and the novelty of the never-ending list of methods

Algorithm 11 Particle swarm optimization algorithm

```
Initialize $m$ particles to random positions $\boldsymbol{x}_1^j$
Initialize $\boldsymbol{x}^j = \boldsymbol{x}_1^j$ and $\boldsymbol{x}^* = \arg\min_{j=1,\dots m} F(\boldsymbol{x}_1^j)$
while Stopping criterion is not met do
    for Every single particle $j = 1, \dots, m$ do
        Update velocity with formula (26)
        Update position $\boldsymbol{x}_{i+1}^j = \boldsymbol{x}_i^j + \boldsymbol{v}_i^j$
        if New position is better than $\boldsymbol{x}^j$ then
            $\boldsymbol{x}^j \leftarrow \boldsymbol{x}_i^j$
            if New position is also better than $\boldsymbol{x}^*$ then
                $\boldsymbol{x}^* \leftarrow \boldsymbol{x}_i^j$
            end if
        end if
    end for
end while
return $\boldsymbol{x}^*$
```

closely resembling each other has been questioned (Sörensen, 2015). It is hard to argue that focusing on the metaphorical roots of the methods would not hurt the development of new methods, as it makes researchers justify the method more with analogue instead of performance in solving the problem it is supposed to.

4.4 New metaheuristic approaches for stimuli optimization

The field of stimuli selection has seen a multitude of approaches, some of which involve researchers more and some that can be considered fully automatic. Metaheuristic approaches represent the more automated side of the spectrum, as they only need to be given the objective function and a few parameters to construct a list fitting the researcher's needs. However, it has to be noted that the objective function corresponding to the situation at hand can be difficult to formulate. From this section onward, the problem considered is again (6) instead of the general minimization problem (25) that was used in the past two subsections.

Previous work on applying metaheuristic approaches to stimuli selection is thin, and only attempted methods include simulated annealing and genetic algorithms. Even though the (Lintz et al., 2021) did follow the idea of a genetic algorithm, the simplified version of reproduction and lack of mutation makes it only a rebranding of the greedy algorithm. As source code (Coupé, 2011) is no longer available and the poster did not explain the exact algorithm, only (Armstrong et al., 2012a) have truly applied a metaheuristic approach to the stimuli selection problem. This leaves much room for new ideas, with still a clear benchmark for comparison.

The field of metaheuristics is too vast and ever-growing to get a good grip of only randomly choosing different methods. Most of the newer methods are complex metaphor-based methods that are difficult to implement and may need loads of CPU time. As the aim of the stimuli selection is to have an easily usable and efficient tool for aiding researchers, it makes little to no sense to use these modern complex and

CPU-heavy approaches. For this reason, methods are chosen among the classics in the field. To get a good overall picture without needing to use a ton of methods, one of the proposed approaches will be a single-solution method and the other one will be population-based. As the methods proposed include a local search phase, we need to first define the used local search algorithm before introducing it to the actual methods proposed.

For local search, the most common approach is the greedy search algorithm which looks for the descending directions and moves in them just like in the algorithm 1. As with combinatorial optimization, this means making swaps that are improving the objective function. Swaps can be chosen at random and accepted only if they are improving, or then one could calculate the change of all swaps and choose the one with the greatest improvement. The weakness of the latter approach is that calculating objective function values is often complicated, and with the vast number of possible swaps the computational complexity quickly gets out of hand. Especially in this case, as the local search is only part of the iteration, it has to be made as efficient as possible. Therefore, swaps will be chosen at random and accepted if they are improving.

Additionally, to limit computational complexity as much as possible, the stopping criterion has to be made loose in the sense that the iteration is stopped prematurely rather than letting it go for too long. Two kinds of stopping criterion will be introduced to the greedy search, with the first one being that there will be no improving swap for a certain number of iterations. The secondary stopping criterion will be a fixed limit for the number of iterations. This can be done without hurting the ability to find global optimality substantially as the population of the solutions will be larger, which means that if one solution does not converge on local search some other solution will. Combining previously described steps yields the following algorithm.

Algorithm 12 Greedy search for stimuli selection

```
Require: Iterations accepted without change $\iota_c$ and maximum iterations $\iota_{\max}$
  Generate the initial solution $\boldsymbol{U}$ if not given
  while Change has happened in last $n_{\text{change}}$ iterations or less than $n_{\max}$ iterations
  in total do
      Choose randomly $U_{ij}$ to be swapped.
      Choose a random $i' \in \{1, \dots, n \mid U_{ij} \neq 1\}$
      Denote $\boldsymbol{U}'$ as a solution where $U_{ij}$ is swapped to $U_{i'j}$
      if $f(\boldsymbol{U}') < f(\boldsymbol{U})$ then
          $\boldsymbol{U} \leftarrow \boldsymbol{U}'$
      end if
  end while
  return $\boldsymbol{U}$
```

The first of the methods proposed will be the single-solution one. Out of the single-solution algorithms described in the previous section, many have some special characteristics making them hard to implement effectively for stimuli selection. Algorithm 3 (tabu search) is not fit for stimuli selection as there is no natural formulation for the tabu list as the solution is more about the combination than the singular items in it. If the tabu list were defined through the items it would be too restricting and with a combination approach the size of the tabu list will explode even for smaller problems. The same difficulty is somewhat true for the restricted candidate list in algorithm 4 (GRASP), as the optimality of items is highly dependent on other items chosen. This could be tackled by calculating the centre of already chosen items in the certain list and then choosing the item closest to the centre. This could work but likely would need a lot of computational power, as there would need to be n calculations of distance in every iteration of solution building.

Algorithm 7 (guided local search) seems unfit for stimuli selection, as there does not exist a natural choice for penalized objective function. This difficulty could be handled for the basic form of objective function, but in more complex cases there is likely not a working formulation for penalized objective function. Therefore, as the goal is to develop a general tool for stimuli selection, it would not make sense to choose a specific method for certain types of problems even if they are the most usual case. This leaves us with algorithms 5 and 6 (iterated local search and Variable neighbourhood search), out of which algorithm 5 is the simpler and therefore preferred choice. Additionally, though the natural choice of neighbourhood structures would be to have them be all solutions that can be acquired by swapping some number of items from the current solution, algorithm 6 would become a variation of algorithm 5 with changing perturbation size.

Changing perturbation size would mean that there would be an abundance of parameters to be chosen, which is undesirable. Therefore, the obvious choice out of these is iterated local search. Now the only thing left is to define the ambiguous steps in algorithm 5. The local search step is easy, as we previously defined greedy search as the local search to be used. Lastly, we need to define the stopping criterion and perturbation type. For the stopping criterion, we are going to use a simple version where the algorithm terminates if no new best solution has been found in some fixed number of iterations. Perturbations are also going to be simpler form as the history is ignored here and the perturbation will be some fixed number of random swaps. The last thing to define is when the solutions are good enough to be accepted as a new starting point for the next iteration. Here a simple and general rule will be implemented where if the new solution is at most c units worse in terms of objective function, it will be accepted. Threshold c is left as a parameter and choosing it is explored more in the next chapter. With these, we end up with the final version of the method as the following algorithm.

For population-based algorithms the choice is much easier as out of the presented ones there are only two potential options which are genetic algorithms or scatter search. Particle swarm is out of the question because it is designed for continuous and unconstrained problems. There does exist a version of discrete particle swarm optimization and constraints could be handled by a penalty function, but this would pose too much unnecessary complexity. Ant colony optimization is more oriented towards graph optimization and there are too many parameters that will affect the performance. Therefore, the aforementioned two will be the most suitable ones for the stimuli selection problem.

The genetic algorithm and scatter search are similar in many ways, though the genetic algorithm introduces more randomness than the scatter search does. The

Algorithm 13 Iterated local search for stimuli selection

```
Require: Size of perturbation $n_{pert}$, iterations accepted without change $\iota_c$, maximum iterations $\iota_{\max}$ and threshold for accepting new solutions $c$
  Initialize some solution $\boldsymbol{U}$ at random
  Conduct a greedy search to $\boldsymbol{U}$ and note solution as $\boldsymbol{U}^*$
  $\boldsymbol{U}_{best} \leftarrow \boldsymbol{U}^*$
  while Change to $\boldsymbol{U}_{best}$ has happened is last $n_{\text{change}}$ iterations or less than $n_{\max}$ iterations in total do
      Perturb $\boldsymbol{u}^*$ by swapping $n_{pert}$ items randomly
      Conduct greedy search starting from $\boldsymbol{U}$ to obtain new $\boldsymbol{U}$
      if $f(\boldsymbol{U}) - f(\boldsymbol{U}^*) < c$ then
          $\boldsymbol{U}^* \leftarrow \boldsymbol{U}$
      end if
      if $f(\boldsymbol{U}) < f(\boldsymbol{U}_{best})$ then
          $\boldsymbol{U}_{best} \leftarrow \boldsymbol{U}$
      end if
  end while
  return $\boldsymbol{U}_{best}$
```

challenge of genetic algorithms is that the definition is vague and different steps can be done in a multitude of ways. As the scatter search is much more straightforward with each of its steps, it reigns supreme. As scatter search combines ideas of the most single-point methods with genetic algorithms' way of evolving population over the iterations, it sounds promising for a stimuli selection problem.

The other way to look at scatter search is that it randomly tests different areas of the search space to determine those areas that have the most potential to contain optimal solutions. When these are determined, the search can be concentrated on these areas and in between them. In some very peculiar optimization problems, this way might not ever get even close to optimum if it happens to be very far away from the local optima and the descent around it is very steep. Though in the situation described it is also unlikely for any other metaheuristic to perform well.

Scatter search as it was presented in algorithm 9 is not as suitable for binary variables as it involves taking linear combinations of the reference solutions. Here, the used get-around will be inspired by the crossover step of the genetic algorithm. It would be possible to have combining solutions be weighted by their fitness, but for ease of implementation, simple random sampling shall be used to create new solutions. As stated before, the local search part of the algorithm will be the greedy search as defined in algorithm 12.

Previously, we have defined a suitable local search algorithm and a method for creating the population of solutions. Now the only thing that is left is to define a stopping criterion for the scatter search. The most common stopping criterion for metaheuristics involves looking at the change happening between the iterations. The scatter search might not improve the solution every turn, but it does no either take "uphill" steps at least in the main loop, still, the result might be worse than previous iterations and iteration by iteration change does not tell the full picture. In certain situations, scatter search might also get stuck in a periodic cycle which means that even if change happens from iteration to iteration the algorithm might be stuck. Therefore, the most logical stopping criterion is whether the algorithm has found the new best solution in the last ι_c iterations. By combining everything, we end up with the algorithm 14.

Algorithm 14 Scatter search for stimuli optimization

```
Require: The size of the solution population $n_s$, the reference list $n_r$ and iterations
  accepted without change $\iota_c$ and maximum iterations $\iota_{\max}$
  Generate randomly $n_s$ solutions
  Choose $n_r$ solutions with the lowest objective function values
  $f(\boldsymbol{U}^k)$ to be the reference list $\mathcal{R}$
  while Change to $\boldsymbol{U}_{best}$ has happened is last $\iota_c$ iterations or less than $\iota_{\max}$ iterations
  in total do
      for $k = 1, \ldots, n_s$ do
          for $i = 1, \ldots, n$ do
              for $j = 1, \ldots, l$ do
                  Choose $u$ randomly among the items of the solutions in $\mathcal{R}$
                  $U_{ij}^k \leftarrow u$
              end for
          end for
      end for
      Apply the greedy search from algorithm 12 to all acquired $\boldsymbol{U}^k$
      to get a new generation of solutions $\boldsymbol{U}^{k'}$
      Choose $n_r$ solutions with the lowest objective function values
      $f(\boldsymbol{U}^{k'})$ to be the reference list $\mathcal{R}$
  end while
  return $\boldsymbol{U}_{best}$
```

Now that the proposed algorithms have been described, the next logical step is to start testing their performance. As simulated annealing is the only real competitor in the metaheuristics side of stimuli optimization, it will be used as a benchmark to which these two will be compared to. Before the final comparison, there needs to be an evaluation of the role of parameters for each of the methods, as there is a high likelihood of methods not being robust to change of parameters.

5 Evaluating the effect of the parameters

Metaheuristic methods always have some random part in them, which often involves some parameters set by the user. This means an additional subjectivity which makes it too hard to evaluate, especially new proposed methods. For this reason, there is a need to inspect the effect these parameters have on the result. Randomness can also affect this inspection, for which reason it has to be done through some set of seeds given to the random number generator. If the set of seeds is too small, there can be some bias in the results, for which reason the number has to be as high as it can be while keeping the computational time used relatively small.

In an ideal world, the methods could determine the parameter values themselves, or at least it would be possible to infer from the problem what are the optimal values for the parameters. Sadly this is not the case and the parameter choice is hard as there are no clear guidelines on how a certain parameter should be chosen. One way could be trying many different values to see which performs the best. This approach will be used in this chapter to develop some heuristic ideas on the selection of the parameters. The difficulty here is that for an even a little more complicated problem, the amount of CPU time needed to run any metaheuristic with a bunch of different parameter values will be substantial.

To see the effects parameters have on the result, there needs to be some numerical testing. For these purposes, some neutral and big enough data is fitting. Here simulated data will be used where there are five independent variables following a standard normal distribution. For a simple setting, four of these will have differences between lists maximized and the last one will be minimized. This should be a more difficult task than just equating lists or maximizing one feature and for that reason will be used to have more differences in results. The exact objective function used is (11) with parametrization of $p_j = 2$, for all $j = 1, 2, 3, 4, 5$ and $\boldsymbol{b} = (-1, -1, 1, -1, -1)$. Some uncontrolled and undocumented testing of methods was done with the previously described setting in the implementation phase. Insights gained from this testing will be used to narrow down the parameter candidates and will be referred to as pilot testing.

Methods were all implemented in the R programming language (R Core Team, 2023). Even though R is a mostly efficient language for a lot of things, the objective function values are calculated in C++ through the RCPP-package (Eddelbuettel and Francois, 2011). In the tests, we noticed that the use of C++ saves 85% CPU time which is a lot considering the objective function values are calculated possibly hundreds of thousands of times during the run of the algorithm. Results were produced by R version 4.3.0 and RCPP-package version 1.0.11 running on the Windows 10 operating system with Intel® Core™ i5-1135G7 processor. All the codes and datasets used to produce results in this thesis can be found in (Niemensivu, 2023).

5.1 Temperature and annealing schedule on simulated annealing

On the implementation of simulated annealing done for this thesis, there exist four parameters that can be played around with. First is the temperature parameter that unlike what one would expect does not affect the solution much as the annealing mostly guarantees enough temperature range as long as the starting temperature is large enough and temperature changes are not too drastic. This means that instead of all that could be gained from meddling with temperature can be gained more effectively with other parameters. For this reason from now on, the temperature will be fixed to $T = 10$, which seems to be a good starting value in the early testing phase.

The second parameter of simulated annealing is the annealing rate r, which dictates how quickly the temperature decreases. This parameter is the key to simulated annealing, as too low values could mean premature convergence and too high values could lead to the algorithm converging too slowly. Here the annealing rate is fixed meaning the temperature will be decreased after some fixed number of iterations. The schedule being fixed means that the optimal schedule needs to be tested similarly to other parameters. Hypothetically, a high number of iterations between decreases should be combined with larger decreases and vice versa.

Previous hypothetical statements are backed up with numerical findings presented in the table 1 where the objective function values and CPU times for different iteration and annealing rate values are listed. These values are all averages over 20 different seed values. Here the last parameter which is the maximum number of iterations is fixed at 500 000. This upper limit seems to lead to the algorithm not converging for the highest values, as mostly the objective function values seem to increase with the annealing rate and annealing schedule.

annealing	temperatures					
schedules	0.1	0.3	0.5	0.7	0.8	0.9
2000	-21.1309	-21.1119	-21.0894	-21.0530	-21.0636	-21.3440
	2.1900	2.4404	2.6945	3.5285	4.7071	8.7613
4000	-21.3639	-21.0130	-21.5014	-21.3689	-21.4813	-21.7164
	3.0365	3.6188	4.6229	7.3319	9.5034	17.6531
6000	-21.5004	-21.3237	-21.4177	-21.4199	-21.5729	-21.7429
	3.9394	5.3727	6.3071	9.7086	14.2127	26.1392
8000	-21.3855	-21.5323	-21.7400	-21.5084	-21.7541	-21.6305
	4.8025	6.4296	8.3297	12.1089	17.2556	29.6592
10000	-21.3987	-21.4261	-21.5111	-21.5567	-21.8640	-20.8976
	6.4354	7.2209	9.3240	14.9757	22.0423	30.8714

Table 1: Objective function values and CPU times for different temperatures and annealing schedules

To investigate the possibility of the algorithm not converging, another numerical comparison is constructed. Here the other parameters are set to the maximum of the last experiment which means $r = 0.9$ and iterations between decreases is 10000. Now, as the number of iterations can get quite astronomical, in the worst case the number of seeds used will be limited to 6 out of the previous 20 to keep CPU time somewhat manageable. Here the CPU time is still not the point of interest, as the idea is just to figure out how many iterations it would take for the algorithm to converge.

For the experiment, the maximum number of iterations will go from 500 000 up to 1 500 000 as there is no way to know beforehand how long it will take for the algorithm to converge. As table 2 suggests, the best value of each seed is found already around 700 000 to 800 000 iterations. After the number of maximal iterations where the minimum value is obtained, CPU times no longer rise, meaning that the stopping criterion cuts the run before the maximum of iterations is reached. This is good, as it means that maximum iterations do not need to be considered as long as they are high enough to not be reached before convergence. In large problems, the situation is different, as convergence could take a lot of time.

maximum	seeds					
iterations	620	409	795	61	535	288
500 000	-21.22655	-20.92157	-21.19603	-21.187	-21.07047	-20.61033
600 000	-21.53964	-21.53649	-21.47911	-21.50282	-21.33627	-21.65044
700 000	-21.96379	-21.92871	-21.72897	-21.7128	-21.72379	-21.99299
800 000	-21.96379	-21.92632	-21.77495	-21.74138	-21.73229	-21.98043
900 000	-21.96379	-21.92632	-21.77495	-21.74138	-21.73229	-21.98043
1 000 000	-21.96379	-21.92632	-21.77495	-21.74138	-21.73229	-21.98043
1 100 000	-21.96379	-21.92632	-21.77495	-21.74138	-21.73229	-21.98043
1 200 000	-21.96379	-21.92632	-21.77495	-21.74138	-21.73229	-21.98043
1 300 000	-21.96379	-21.92632	-21.77495	-21.74138	-21.73229	-21.98043
1 400 000	-21.96379	-21.92632	-21.77495	-21.74138	-21.73229	-21.98043
1 500 000	-21.96379	-21.92632	-21.77495	-21.74138	-21.73229	-21.98043

Table 2: Effects of the number of maximum iterations on different seeds

The previous result is evidence that the results of table 1 do not tell the whole truth about the algorithm, and further investigation is necessary. For the second experiment, the conditions will be kept mostly similar to the first except for the higher number of maximum iterations. As the data is the same as used to investigate the effect of maximum iterations, we can use this gained knowledge and safely set the maximum number of iterations to 1 000 000. Thanks to previous results, the number of annealing rates and iterations between changes can be narrowed down without losing much information. In this experiment, the used annealing rates will be 0.1, 0.5 and 0.9. For the number of iterations between decreases the used values will be 1000, 5000 and 10000.

This setup yields us the results found in table 3. Here it can be seen that higher temperatures and breaks between annealing contribute to systematically better results when the maximum number of iterations is not a problem. Unlike on the examination of maximum iterations, here we have returned to the original 20 seeds. It has to be noted that used CPU time seems to be increasing in tandem with increases to both parameters. Therefore, there is a clear trade-off between the quality of solutions and the used computational power. For the benchmark cases the combination of 0.9 temperature and annealing every 10000 iterations will be used. For a case where a quick solution is desired, lower parameter values are recommended.

All previous results were calculated using (Kirkpatrick et al., 1983) version of the calculation of probability, p which means that for improving swaps $p = 1$. On the contrary, if (Armstrong et al., 2012a) version of the p was used the probability for even increasing swaps is often less than 1 meaning that increasing swaps are not guaranteed to happen. This is both a big philosophical and practical choice that is bound to affect the results in some way. As the different annealing schedules and temperatures seem to have the same effect on the results, the temperature will be fixed for this testing and the annealing schedule will alter from 1000 to 10000 with the steps of 2000.

Running the experiment depicted above, averaging over 20 previous seeds, we get results shown in table 4. Here we can see that in terms of objective function

annealing schedules	temperatures 0.1	0.5	0.9
1000	-20.49403 1.208057	-20.77672 1.385596	-21.0448 4.642841
5000	-21.18843 3.551749	-21.43273 5.607007	-21.63132 22.458017
10000	-21.39868 6.697095	-21.51106 9.519296	-21.75934 44.364596

Table 3: Objective function values and CPU times for different temperatures and annealing schedules with higher max iterations

values, the SOS!-version of the p function greatly outperforms the original. In CPU time it does take at least double the time, but positively the difference shrinks when the CPU times go up. Especially in the case of SOS!, both objective function values and CPU times act interestingly, as they do not always go up when increasing the annealing schedule. Regardless of this, previously established parameters shall be used in the benchmark test in combination with the SOS!-version of p.

temperature function	annealing schedules 1000	3000	5000	7000	10000
Kirkpatrick et. al.	-21.0448 4.323912	-21.00942 4.324535	-21.35432 12.78944	-21.53006 12.17409	-21.63132 21.2282
Armstrong et. al.	-21.67838 20.175777	-21.84424 29.115103	-21.65462 28.01449	-21.75934 42.6373	-21.7454 39.111

Table 4: Objective function values and CPU times for different p function and annealing schedules

5.2 Size of population and reference list on scatter search

Out of the methods evaluated, the scatter search is by far the most problematic in terms of parameter selection. With certain choices of parameters such as high population size, scatter search can be quite CPU-heavy. From the testing perspective, this poses the problem of being even able to run satisfactory test cases in even relatively sensible time. For these reasons, the test will be simplified to achieve a lower total time spent. This means that tests will be run only on singular seeds instead of the previous multi-seed approach, which will limit the generalizability of the results.

Scatter search has a multitude of parameters including population size, reference list size, accepted iterations without change, maximum iterations and maximum iterations for the greedy part. Out of these, the population and reference list sizes are the ones that cannot be set arbitrarily large, even for pre-testing. Therefore, they are the best candidates to be in the first test. Maximum iterations are set to default 100, as in the pilot testing the algorithm never ran for over 50 iterations. Iterations accepted without change will be set at 10 as it showed to be enough to prevent the most premature stopping in the pilot tests. For the greedy part, the maximum iterations will be kept at default 10000.

Now, the first two tested parameters will be population and reference list sizes. To see how population affects results, there should be as wide a range of different values as possible. Limited computational capacity does limit this though. Reference list size is harder as if we want to run the same sizes for all population sizes, then $\max n_r \leq \min n_s$ as otherwise there are not enough options to even choose the reference list in the first place. For these reasons, the smallest tested population size is 10, which will also be the maximal reference list size. As the smaller reference list sizes are more likely to have greater differences than the higher ones, the starting size will be 2 from which we are advancing to 3, 5 and 8 before the aforementioned 10. For population sizes, the tested ones will be 10, 20, 30 and 50.

The result of the runs with previously mentioned parameter values and the first seed of the previously used list (620) can be found in table 5. There we can see first that the running of the scatter search can vary greatly and secondly, as only one seed was used, the result does not follow consistently logical patterns. Inference based on these results will be difficult, but as computational times get easily out of hand, the extensive study done to simulate annealing seems unnecessarily time-consuming.

population	reference list sizes				
sizes	2	3	5	8	10
10	-21.5929 16.9700	-21.4276 50.6650	-21.5887 73.2659	-21.5541 92.9042	-21.5929 116.2994
20	-21.5149 22.9091	-21.9038 29.6412	-21.5574 174.4580	-21.5722 112.5521	-21.7687 176.5041
30	-21.5373 27.1496	-21.6034 69.7672	-21.8993 58.7571	-21.5821 187.9294	-21.6497 262.6139
50	-21.6510 97.0144	-21.5661 72.7864	-20.7437 114.3850	-21.7582 108.9625	-21.7171 349.3039

Table 5: Objective function values and CPU times for different population and reference list sizes

Previous tests showed that higher population and reference list sizes are not automatically better, though increasing either seems to at least make objective function values a little better. These better values come at a high cost, as the CPU time starts to get out of hand quickly. Based on these results, population sizes 20 and 30 seem to be good compromises combined with a reference list of 3 or 5 solutions. For the test of the effect of accepted iterations without change, a population size of 20 will be used in combination with reference list sizes of both 3 and 5. Tested iteration numbers will be 3, 5, 10 and 20. As the test will now be less computationally demanding, the average of 3 differently seeded runs will be used for a slightly more representative sample.

With the previously defined test setup, we get the result shown in the table 6. We can first see that out of these perturbation sizes, 5 is by far superior in terms of the objective function values produced. The increase of the CPU times with increasing the accepted iterations is curious, as the 5 has higher CPU times than 10 even though one would expect it to be another way around as there should be on average at least 5 more iterations per run. This fact is likely due to one of the seeds being an outlier and taking way more CPU time than the others.

The most surprising of the results is that even by accepting only 3 iterations without changes, we get results comparable to a higher number while using less CPU time. The bigger reference lists do always give better results here, but with a cost of at least doubling the CPU time. Therefore, for the benchmarks values will be the population size of 20, reference list size of 5 and iterations without change accepted of 10. For practical use, where the use of CPU time is of concern, a reference list size of 3 is recommended.

reference	iterations accepted without change			
list size	3	5	10	20
3	-21.5517 25.3834	-21.5205 47.0150	-21.6091 35.8500	-21.5309 59.1373
5	-21.6432 48.8047	-21.6072 104.9917	-21.7184 94.8770	-21.6551 178.3254

Table 6: Objective function values and CPU times for different iterations accepted without change and reference list sizes

5.3 Different perturbations on iterated local search

Similarly to simulated annealing, the iterated local search includes 4 parameters that can be used to tune the algorithm. The most important of these is the perturbation size, as it dictates how far away the start of the local search is pushed from the current solution. With too small values the search fails to escape the local minimum, and for too large values all the information already gained is lost. As with all metaheuristics a good balance between greediness and globality of the search has to be found.

Other parameters of the algorithm include the acceptance criterion, the number of iterations between changes and the maximum number of iterations. In the pilot testing, it was noted that maximum iterations were never needed, as the stopping criterion handled the possibility of the algorithm being stuck on the cycle of the smallest of improvements. Therefore, it can be set arbitrarily large without affecting the functioning of the algorithm.

The acceptance criterion and number of iterations without change are harder ones to decide, as they directly affect the functioning of the algorithm. Acceptance criterion can vary from the extremes of accepting only improving solutions to accepting any new solutions. By accepting only improving solutions and small perturbation sizes, the algorithm is highly likely to get stuck in the local minimum. On the other hand, accepting any solution will lead to loss of already gained information might mean that the algorithm never converges to good solutions. To find a balance between these, some testing is in order. Pilot testing showed that 20 iterations without a change mostly prevented premature stopping. Even though it seemed that this choice led sometimes to prolonged run times. Additional testing is in order, but first, it will be fixed at the aforementioned 20 for purposes of evaluating the other two parameters.

Now that the stopping criterion has been fixed, it is time to evaluate how the perturbation size and acceptance criterion affect the solution. Previous tests have shown that the objective function value is around -20 and the fluctuation of it is around one unit. For the acceptance criterion, the extremes will be included in the test. One extreme is accepting only improving solutions corresponding to $c = 0$ and the other is accepting all of them which would correspond to for example $c = 10$ as the local search should never even get close to these big fluctuations. Lastly, we want a criterion where only those that are somewhat close to the current one in fitness are accepted, for which a suitable criterion is $c = 1$. This might be not good for all situations because it highly depends on the magnitude of the objective function.

Perturbation sizes should range from not much changing to almost everything changing. As the list size for these tests is 32 and therefore the total number of items is 64, almost everything changing extremes should be 48. Going higher than that would make the algorithm almost sequential greedy searches from different random starting points, which is undesirable as the aim is to keep some of the previous information. The smallest perturbation size should not be smaller than 4 as the solution needs to be shaken to escape the local minimum. The other two values should be on uniform intervals between these, which means a good choice would be 16 and 32.

Now with the tested perturbation sizes and acceptance criteria chosen, we can proceed to testing itself. The setting is the same as before with 20 different seeds used on runs from which the averages of objective function and used CPU time will be returned. Results of the test are shown in the table 7. Based on these results, the acceptance criterion seems to not matter that much as long as at least some decreasing solutions are accepted. Perturbation size seems to have a high impact, as can be expected. Test results indicate that perturbation of at least half of the items seems the best.

acceptance	perturbation size			
criterion	4	16	32	48
0	-20.86253 9.416565	-21.06365 18.5751	-21.24856 21.23736	-21.44295 24.71411
1	-21.11924 12.626256	-21.19094 15.37382	-21.47766 22.16805	-21.48287 21.57559
10	-21.13531 12.121576	-21.29341 17.15387	-21.47547 23.48131	-21.47053 20.85157

Table 7: Objective function values and CPU times perturbation sizes and acceptance criteria

The next point of interest is how the stopping criterion will affect the results. Too many iterations accepted without changes will mean that the algorithm will run for longer than necessary. On the other hand, too few iterations mean that the run is aborted before convergence. Balancing between these is difficult since there is nothing to guarantee that convergence has happened after some number of iterations unless the algorithm is left to run for an infinite amount of time. Therefore, multiple different iteration numbers will be used to see where the premature stopping happens and how big is the trade-off between better solution and CPU time used.

Previously used 20 iterations without change seemed to produce mostly good results and therefore will be the starting point of the examination. Smaller values will be 5 and 10 to get the fullest extent of the possible premature stopping. When going higher the only used value will be 40 as going even higher would make the computational load quite massive. Still, if the results show that there could be a need for even larger values, those will be tested too.

The results of table 8 show that the longer we let the algorithm run, the better the result is. The growth of CPU time used seems to be almost linear to the number of iterations accepted without changes. Perturbation size does not have a clear linear relation, but 32 seems to be the best out of these. Going forward, the default perturbation will be set at the group size. As the iterations accepted had more of a trade-off between objective function value and CPU time, similar defaults will not be set. For benchmarks, the maximal number of 40 will be used, but in a practical sense, 20 seems more efficient.

accepted iterations	perturbation sizes		
without change	32	40	48
5	-21.14183 5.943093	-21.18408 5.617944	-21.23297 5.433167
10	-21.41531 9.857227	-21.30362 9.019729	-21.32208 9.143402
20	-21.45017 15.804736	-21.41544 15.531507	-21.46074 16.802306
40	-21.53472 35.049985	-21.49289 29.160858	-21.483 29.452234

Table 8: Objective function values and CPU times perturbation sizes and accepted iterations without change

6 Results

6.1 Data and methodology

The datasets used in this study can be separated into two categories based on the origin. A total of six datasets will be used to evaluate the methods, three of which contain simulated data and three of which contain real data. The simulated data is simple in a way that all the distributions sampled are common and only one of the variables includes any arithmetic operations between them. Therefore, these datasets can and will be characterized accompanied by results in the next section. The real-life datasets contain variables that need further explanation and distribution unknown beforehand. For these reasons, the descriptions of variables and descriptive statistics will be provided separately from the results in this section.

With word-related datasets, the most common included variable is the length of the word in terms of characters, which also is seen here. The second variable is word frequency, which is another common one in word-related data. Frequency here is a raw frequency of a lemma extracted from newspaper Turun Sanomat corpus from January 1994 to March 1996. The corpus consisted of 22.7 million word forms and was assessed by the lexical search program called WordMill (Laine and Virtanen, 1999).

The next three variables are all raw frequencies of certain parts of the word. Bigrams are all combinations of two consecutive characters that the word includes. The value here is calculated as a mean of frequencies of all bigrams in the word in question. Initrigam refers to the raw frequency of the first three letters of the word and as an opposite the fintrigram refers to the raw frequency of the last three letters of the word. It is important to note that the corpus of 22.7 million word forms is relatively small and frequencies might not be the most accurate, especially on the lower end. For the context of this thesis, this is not a problem, as the focus is more on the mathematical properties of solutions. Descriptive statistics for this dataset are presented in table 9. The Lexize dataset is not included in the Gitlab repository, as it will be included in an unpublished article.

In addition to variables related to the characteristics of the words, data also includes estimated item discriminations and difficulties. These were estimated using the 3-parameter logistic from the previously discussed IRT framework using the data gathered by TRILA as part of their FUNA measurements (Turku Research Institute for Learning Analytics, 2022). To sum it up quickly, the 3PL-IRT model is a latent variable model, where an unobserved latent variable is a test subject's ability in a tested subject. The probability of getting the item correct with a certain level of ability is estimated using the logistic curve for dichotomous items such as those we have here.

The parameters of this curve have clear interpretations such as the difficulty, discrimination and the guessing probability of the item. Difficulty tells how much ability the test subject needs to have to have a 50% chance of getting the said item correct or if the guessing parameter is greater than zero, then the probability is halfway between guessing probability and 100%. Item discrimination tells how well the item differentiates between those under and above the item difficulty. Higher difficulty means that the logistic curve has a steeper slope. The guessing parameter estimates the probability of a test subject getting the item correct, even if their ability would otherwise not be enough. This is also the lower asymptote of the logistic curve.

Lexical decision-making tasks are a common type of stimuli for list optimization, as there are many variables available for the items. Therefore, the second real dataset used will also be a set of words. The dataset is a subset of the English Lexicon Project (Balota et al., 2007), which is a large-scale English language lexicon. Lexicon itself contains a multitude of variables, but only a couple of them will be used in this analysis. It is important to note that in this kind of dataset, many of the variables will measure the same thing a little bit differently, as the attributes of the words are not absolute, but depend on the way they are measured. The best example of this is word frequency, which will highly depend on which format and

	Length	LemFreq	Bigram	Initrigram	Fintrigram	Disc.	Diff.
Min.	4.00	2.00	234.70	0.07	0.86	-2.66	-2.19
1st Qu.	5.00	26.75	666.80	7.12	26.43	2.01	0.54
Median	6.00	100.00	852.20	29.35	47.45	2.81	0.40
Mean	5.77	127.51	904.40	54.13	86.79	2.96	0.24
3rd Qu.	6.00	191.75	1128.30	72.96	107.92	3.73	1.04
Max.	9.00	595.00	1938.80	355.84	478.85	6.11	1.69

Table 9: Descriptive statistics of Lexize dataset

	Length	Freq_HAL	Concreteness	NPhon	NSyll
Min.	11.00	5006	1.33	7.00	2.00
1st Qu.	11.00	6507	1.92	9.00	4.00
Median	11.00	9062	2.37	10.00	4.00
Mean	11.78	17253	2.52	9.96	4.13
3rd Qu.	12.00	16925	3.04	11.00	5.00
Max.	15.00	467659	4.75	14.00	7.00

Table 10: Descriptive statistics of English lexicon dataset

time it is measured from.

Instead of the classic word frequency based on corpus constructed by Kučera and Francis (1967) that is also included in the data of English Lexicon Project, here we have opted for HAL frequencies (Lund and Burgess, 1996). The concreteness of the word measures how much it refers to something that can be experienced through human senses and was crowdsourced from over 4000 people rating the words on a scale of 1 to 5 (1 means abstract and 5 means concrete) (Brysbaert et al., 2014). The last two variables are quite self-explanatory, as NPhon and Nsyll stand for the number of phonemes and the number of syllables in the word, respectively.

The original dataset includes over 40000 words, which would likely need computational power that is not attainable for the author. Therefore, a smaller subset of the dataset is selected by filtering out incomplete cases and fixing word length at over 10 with a HAL frequency of over 5000. With these limitations, we are left with 265 words, which is a more manageable amount in terms of the computational load. Descriptive statistics for this filtered dataset can be found in table 10.

The last dataset that will be examined is a subset of the MRC Psycholinguistic Database (Wilson, 1988). Here we follow the setup that was used to originally demonstrate the capabilities of SOS! (Armstrong et al., 2012a), where the study by Morrison and Ellis (1995) was replicated. Variables used here are the age of acquisition, imageability, word frequency and the number of letters. The dataset includes only those words that had all the aforementioned variables, which leaves us with 1689 words.

The frequencies used in this dataset are the raw frequencies from the corpus constructed by Kučera and Francis (1967). The age of acquisition ratings is the means of the ratings of 36 volunteering students collected by Gilhooly and Logie (1980). The scale goes from 1 representing 0–2 years to 7 which represents 13 years or older and has been scaled by 100. Imageability ratings are merged from three highly correlating norms (Paivio et al., 1968; Toglia, 1978; Gilhooly and Logie, 1980) by averaging after adjusting for differing means and standard deviations. The scale here is also from 1 to 7 scaled by a factor of 100, where 100 corresponds to words that arouse images with difficulty or not at all and 700 meaning that the word easily arouses images.

Looking at the descriptive statistics of this dataset in table 11, we can see that the dataset has a lot of variation in all variables. Age of acquisition and imageability have a good amount of variation even if they do not reach minimal or maximal possible values. Considering these are means of ratings made by humans, it would be unlikely that any word would get the minimum or maximum possible value. Age of acquisition seems to have a symmetric distribution based on the quantiles and median. Imageability on the other hand seems to be a bit skewed towards the higher end of the spectrum. Most of the frequencies are fairly small, and the distribution looks left-skewed, which is to be expected from the variable that has a lower limit but practically no upper limit as the sizes of used corpora are counted in at least millions. The distribution of the number of letters is also a bit left-skewed and most of the words are in the center, close to the mean and median.

	AOA	IMG	KFFRQ	NLET
Min.	125.00	129.00	1.00	2.00
1st Qu.	320.00	389.00	5.00	5.00
Median	408.00	472.00	17.00	6.00
Mean	412.80	465.20	48.77	6.59
3rd Qu.	497.00	555.00	52.00	8.00
Max.	697.00	655.00	967.00	15.00

Table 11: Descriptive statistics of ME95 dataset

Parameters used in the testing are based on the tests conducted in the previous chapter and can be found in table 12. It should be noted that these parameter choices are optimal on certain test data, but nothing guarantees that they are optimal for all tested datasets. Still, testing all possible parameter combinations on every dataset would be a tremendous task and is therefore not included. There likely exist other parameter combinations that are more optimal for the smaller datasets than the current ones. This is a thing to consider when interpreting the results. The fact is still that in a real-life scenario, there is no point in testing all methods and different parameter combinations, so there is a need to set some recommendations of parameters to be used that work for most uses.

	max_iter	check_change	temp	temp_change	pop	ref	pert_size
SOS!	1 500 000	10000	10	0.9	-	-	-
SSSO	100	10	-	-	20	5	-
ILS	500	40	-	-	-	-	list_size

Table 12: Parameters used in testing

The test setup used in all the six different datasets will be the same in terms of everything except the weights ***b*** used in the optimization, as these depend on the practical case we are interested in. If the variables are not at least approximately on the same scale, they will be scaled to have zero mean and unit variance. All the datasets will be tested in four different scenarios to see if the changes in $\boldsymbol{b}$ affect the outcomes. As each method contains some random elements, the methods will be run on 20 different seeds to take at least some of the randomness away from the results. For each of the seeds, the resulting objective function value and used CPU time will be stored. From there, the average of minimum objective function values on each run (later OBJ), the minimum objective function value of all runs (later MIN) and the average of the elapsed CPU time across all runs (later TIME) will be presented for each of the scenarios. Also, the best values for both objective function-related measures for each scenario will be bolded.

6.2 Results on simulated data

In theoretical consideration of the goodness of algorithm, it is not only important to apply it to practical scenarios, but also to theoretically interesting ones. As these theoretical scenarios might not be present in any currently existing dataset but still may appear in the future dataset, the scenarios must be simulated. Two of the most interesting theoretical questions involving stimuli selection algorithms are how they change their behaviour when the variables are highly correlated or the distribution is not symmetric. To answer these questions, we have to simulate some datasets where such scenarios exist.

The first simulated data will be the normally distributed data already established in the parameter evaluation chapter. The data has been generated in R with `rnorm` function. It includes five different random samples from the standard normal distribution with 1000 observations for each of the variables. As the normal samples are independent, there exists little to no correlation between variables, which should make the optimization easier than if the variables were not independent. The rest of the datasets have different kinds of inter-variable correlation structures to simulate some possible challenging situations. Scenarios here are equating other variables with 1-3 variables that have between-list differences maximized and as a last one equating the lists. Different $\boldsymbol{b}$ used in this experiment can be found in table 13

	Variable 1	Variable 2	Variable 3	Variable 4	Variable 5
(1)	1	1	1	1	-1
(2)	1	-1	1	-1	1
(3)	1	-1	1	-1	-1
(4)	1	1	1	1	1

Table 13: $\boldsymbol{b}$ vectors for simulated standard normal distributed dataset

Results acquired on the simulated normally distributed data can be found in the table 14. Here we can see that in terms of objective function values, the simulated annealing was superior to other tested methods except for the equating setting where it had the worst average and little higher minimal value than the scatter search. Scatter search does not seem a good method for anything other than equating the list as it uses almost double the CPU time compared to simulated annealing and fails to find good solutions. Iterated local search managed to be the fastest in every setting, but the values were never the best of the bunch. Equating lists differs a lot from others likely since simulated annealing suffers as changes in the objective function are small and therefore, probabilities of swaps are almost all equal, which lessens the greediness of the algorithm at the latter stages.

	Simulated annealing			Scatter search			Iterated local search		
	OBJ	MIN	TIME	OBJ	MIN	TIME	OBJ	MIN	TIME
(1)	**-20.208**	**-20.224**	47.0	-20.066	-20.132	65.7	-19.983	-20.086	**29.2**
(2)	**-20.556**	**-20.571**	45.2	-20.364	-20.458	74.5	-20.294	-20.465	**28.0**
(3)	**-20.893**	**-20.966**	42.7	-20.615	-20.825	81.6	-20.481	-20.695	**33.1**
(4)	1.00E-04	7.06E-06	73.54	**3.42E-05**	**6.10E-06**	23.0	5.64E-05	1.31E-05	**5.1**

Table 14: Standard normal distributed dataset results

The second simulated data used will be a set of correlated variables. The first two variables of this dataset are independent standard normal samples. The third variable $V_3 = V_1 + Z \cdot V_2, \quad Z \sim N(0,1)$, is the first one summed with the second that has been multiplied by a standard normal variable. The last variable of this dataset $V_4 = U \cdot V_3 + Z, \quad U \sim U(0,1), Z \sim N(0,1)$, is the third, scaled with a uniformly distributed variable plus a standard normal variable. The data consists of 250 observations of the previously described variables. The correlations of the variables are presented in the table 15.

The correlations in this dataset are supposed to mimic a scenario where the effects of some variable are the interest of the study, but it correlates highly with another confounding variable. Here, high and low lists are hard to conduct while keeping the confounding variable similar between lists. Therefore, it would seem that this optimization problem will be harder than the one with independent normal variables. The test will have four different settings where each one of the variables will have differences maximized while keeping the rest as similar as possible. Here the number in the table tells which of the variables is to be maximized in terms of the list-wise differences. Based on the correlations, the last two cases should be the hardest ones, as the correlation between the last two variables is almost 0.8.

	Variable 1	Variable 2	Variable 3	Variable 4
Variable 1	1	-0.0107	0.3000	0.2807
Variable 2	-0.0107	1	-0.0254	0.0135
Variable 3	0.3000	-0.0254	1	0.7763
Variable 4	0.2807	0.0135	0.7763	1

Table 15: Correlations between variables in correlated variables dataset

As the table 16 presents, the results are pretty similar to the ones with the previous standard normal dataset. The exception here is that scatter search manages to find the same best solution as simulated annealing, though the averages are worse in 3/4 cases. A surprising observation here is that case (3) seems the easiest out of these, as all the methods manage to find the same minimum. This is interesting as one could expect that maximizing the difference in a variable with almost 0.8

correlation with a variable to be minimized would seem more difficult than the other first two cases where correlations are lower. Considering the big picture, these results still paint the picture of simulated annealing being a superb method, even though the other ones are closer than they were in the previous dataset.

	Simulated annealing			Scatter search			Iterated local search		
	OBJ	MIN	TIME	OBJ	MIN	TIME	OBJ	MIN	TIME
(1)	**-9.523**	**-9.532**	38.02	-9.521	**-9.532**	50.59	-9.504	-9.528	**14.11**
(2)	**-9.910**	**-9.917**	36.49	-9.904	**-9.917**	65.41	-9.875	-9.908	**12.31**
(3)	-90.911	**-90.918**	28.52	**-90.913**	**-90.918**	42.38	-90.878	**-90.918**	**12.82**
(4)	**-23.579**	**-23.583**	32.72	-23.571	**-23.583**	42.08	-23.551	-23.577	**10.47**

Table 16: Results for correlated variables dataset

Another scenario which is interesting, aside from the high correlations between variables, is what would happen if the variable where high and low lists are constructed for was to have a skewed distribution. In theory, this would mean that from one of the ends of the spectrum, it is much harder to find suitable items for the corresponding list. Therefore, optimization could be harder in this kind of setting. To test it out, we will use a dataset of five standard log-normally distributed variables. Each of the variables will contain only 100 observations, to make it possibly even harder to find suitable items for the lists. The $\boldsymbol{b}$ weights for each of the test scenarios can be found in table 17.

	Variable 1	Variable 2	Variable 3	Variable 4	Variable 5
(1)	-1	1	1	1	1
(2)	-1	1	1	1	-1
(3)	1	-1	-1	-1	1
(4)	1	1	1	1	-5

Table 17: $\boldsymbol{b}$ vectors for log-normal dataset

As was expected, the optimization problem was a difficult one and scatter search managed to be way above the other two, as the table 18 shows. In all four scenarios, both simulated annealing and iterated local search find the same minimal value, but it is not even close to the minimal values found by scatter search. This suggests that there is a possibility that previous examples of minimal values found by simulated annealing are still far off from the global minimum. It is also to be considered that even if scatter search outperforms others here by a long shot, it does not imply that these minimal values would be close to the global minimum. The only way to guarantee a global minimum in a combinatorial setting would be to check all possible solutions, which is practically impossible due to the high number of possible solutions.

An interesting observation here is that both scatter search and iterated local search are much faster in the last datasets compared to simulated annealing when considering the benchmark set by the first dataset. At the same time, the number of observations has also dropped drastically, which would indicate that the simulated annealing is much less sensitive to the number of observations when the CPU time is considered. Adding the information gained by testing the methods with different parameters, it could be said that simulated annealing is more sensitive to the parameters than the size of the data. This is an important observation as it means that for the large dataset, in addition to seeming superior in terms of objective function values, also the CPU time used should grow less drastically than with the other two methods.

	Simulated annealing			Scatter search			Iterated local search		
	OBJ	MIN	TIME	OBJ	MIN	TIME	OBJ	MIN	TIME
(1)	-16.201	-16.253	28.1	**-18.696**	**-19.990**	24.2	-16.253	-16.253	**4.0**
(2)	-20.405	-20.413	23.9	**-22.142**	**-23.805**	27.8	-20.413	-20.413	**4.5**
(3)	-49.640	-49.663	24.9	**-55.758**	**-89.276**	27.2	-49.663	-49.663	**4.8**
(4)	4.89E-04	9.83E-05	40.9	**1.28E-04**	**3.20E-05**	16.9	2.30E-04	9.21E-05	**3.6**

Table 18: Results for log-normal dataset

Given the three diverse hypothetical scenarios constructed here, so far the strongest observation is the aforementioned differing sensitivity to dataset size regarding CPU time. In terms of the objective function, the factors determining the goodness of results are more unclear. In the case of equating lists, simulated annealing seems to be outperformed by others for technical reasons that could be avoided with different parametrization. In situations where there are both minimization and maximization objectives, simulated annealing is favoured in the larger datasets and scatter search in the smaller ones. In the next section, similar tests will be run on three real datasets, which should give additional insight into whether these phenomena are persisting or just occurring randomly.

6.3 Results on real data

Even if the theoretical scenarios are interesting for seeing how the algorithm behaves, the real interest here is their behaviour on real-life datasets. An algorithm could be the best in all theoretically crafted scenarios, but if it fails to perform in real scenarios, it is not that useful after all. Therefore, in addition to theoretical cases, the real-life dataset performance must also be examined. The three datasets examined in this section have previously been described in detail in the section 6.1. It is important to note that, unlike simulated data, these datasets are not in similar scales, which means that they will all be scaled with mean and standard deviation.

The first dataset examined will be the Lexize dataset. The original spark for this study was to find better ways to create two parallel Lexize lists automatically instead of doing it by hand. Therefore, it is fitting that the first test case is making two maximally equal lists. Other scenarios are not as practical but still could be used in research. These are creating equal lists of long and short words, well and poorly discriminating words and lastly easy and hard words. The weights $\boldsymbol{b}$ corresponding to these scenarios can be found in the table 19.

A pattern similar to the results in the simulated datasets emerges here, as from table 20 we can see that the scatter search outperforms the other two methods by a clear margin. Even iterated local search manages to tie or outperform simulated

	Length	LemFreq	Bigram	Initrigram	Fintrigram	Discr.	Diff.
(1)	1	1	1	1	1	1	1
(2)	-1	1	1	1	1	1	1
(3)	1	1	1	1	1	-1	1
(4)	1	1	1	1	1	1	-1

Table 19: ***b*** vectors for Lexize dataset

annealing in every scenario while using about 90% less CPU time. This comparison is likely unfair towards simulated annealing, as the problem here is that there is no need for such a long exploration phase, which means the algorithm could start a greedy search way earlier without affecting the performance. Regardless of this, the scatter search seems to show its strength on the smaller datasets, which could indicate that it would need larger populations or reference lists for the larger datasets. This could in turn lead to a massive need for computational load, which is not practical.

	Simulated annealing			Scatter search			Iterated local search		
	OBJ	MIN	TIME	OBJ	MIN	TIME	OBJ	MIN	TIME
(1)	2.28E-4	2.28E-5	54.7	**4.30E-5**	**1.53E-5**	17.9	1.00E-4	3.88E-5	**4.3**
(2)	-3.460	-3.460	41.6	**-3.752**	**-3.895**	21.6	-3.460	-3.460	**3.7**
(3)	-2.278	-2.278	39.5	**-2.656**	**-2.802**	15.6	-2.278	-2.278	**3.4**
(4)	-2.282	-2.284	39.9	**-2.688**	**-2.906**	23.8	-2.284	-2.284	**4.2**

Table 20: Results of the Lexize dataset

The second dataset tested will be the English Lexicon project subset (Balota et al., 2007). Here we are mimicking a study where interest would be the effect of different numbers of syllables. This means that the optimization problem is maximizing the difference in the number of syllables between the list and otherwise matching the variables. The set-up will be otherwise the same across the test, but the weights will vary to simulate different possible preferences in the real scenario. The exact weights ***b*** used in the test can be found in the table 21.

	Length	Freq_HAL	Concreteness	NPhon	NSyll
(1)	1	1	1	1	-1
(2)	1	1	1	1	-3
(3)	1	1	1	1	-5
(4)	5	5	5	5	-1

Table 21: ***b*** vectors for English lexicon dataset

The results with the English Lexicon dataset are on par with previous observations, as the scatter search manages to again outperform the other methods, though the differences are minimal. The third case is an outlier here, as the scatter search manages to find a minimum that others do not. As the margins between simulated annealing and scatter search are way smaller here than they were in Lexize or log-normal datasets while being similar to the correlated variable's dataset, this would indicate that the critical limit for scatter search to be superior is somewhere close to 250 observations. This limit is in no way absolute and is likely affected by the number of variables, exact parameters and the dependencies between variables. If using the parameters provided in this thesis, the limit of observations 250 could be a good guiding principle when choosing the method to use.

	Simulated annealing			Scatter search			Iterated local search		
	OBJ	MIN	TIME	OBJ	MIN	TIME	OBJ	MIN	TIME
(1)	-10.142	**-10.144**	38.3	**-10.144**	**-10.144**	37.4	-10.141	**-10.144**	**10.7**
(2)	-30.815	**-30.816**	36.6	**-30.816**	**-30.816**	27.3	-30.813	**-30.816**	**9.7**
(3)	-103.165	-103.167	35.7	**-103.425**	**-108.288**	26.4	-103.165	-103.167	**8.4**
(4)	-9.706	**-9.714**	33.9	**-9.711**	**-9.714**	34.5	-9.701	**-9.714**	**12.6**

Table 22: Results of English lexicon dataset

The last dataset examined will be the ME95 study dataset. This is a replica of Armstrong et al. (2012a) original test, but it will not be exact as they used the first version of the MRC Database and had extra metaconstraints in addition to standard minimization and maximization constraints. Regardless, as the SOS! is also used here, it can be a good benchmark of how it compares to proposed methods in the setting it was originally used. Originally, Morrison and Ellis (1995) only had experimented with low and high lists in age-of-acquisition and word frequency. For the sake of additional testing, a low and high setting for imageability and the scenario of creating equal lists will be included. The exact order of settings and weights $\boldsymbol{b}$ can be found in the table 23.

	AOA	IMG	KFFRQ	NLET
(1)	-1	1	1	1
(2)	1	1	-1	1
(3)	1	-1	1	1
(4)	1	1	1	1

Table 23: $\boldsymbol{b}$ vectors for English lexicon dataset

This dataset containing over 1500 words would imply that the simulated annealing is likely to reign supreme, which is confirmed by the results presented in the table 24. The simulated annealing is superior in all scenarios except equating where scatter search manages to beat the others. Interestingly also iterated local search manages to find the same minimum with only a fifth of the CPU time used. Here the simulated annealing seems to be in trouble with equating the lists, as the average CPU time used is about 90 seconds, which is triple the amount needed by scatter search, which is slower in other scenarios.

Overall, the tests on real data were most favourable towards scatter search, which is likely related to the fact that the datasets used here were smaller than the ones in simulated data tests. Especially with the Lexize dataset, scatter search managed to find minima smaller than the others. In the English Lexicon dataset, the same happened only in one of the settings and in the ME95 dataset simulated annealing was the superior method in 3/4 settings. Simulated annealing seemed the least sensitive to the number of items in terms of the used CPU time. Results

	Simulated annealing			Scatter search			Iterated local search		
	OBJ	MIN	TIME	OBJ	MIN	TIME	OBJ	MIN	TIME
(1)	**-11.801**	**-11.820**	54.7	-11.615	-11.714	70.7	-11.563	-11.634	**28.9**
(2)	**-37.354**	**-37.358**	65.5	-37.170	-37.334	115.8	-36.784	-37.204	**37.2**
(3)	**-13.510**	**-13.534**	51.6	-13.301	-13.372	77.4	-13.261	-13.381	**31.4**
(4)	9.14E-06	7.23E-07	90.6	**1.91E-06**	**2.84E-07**	33.4	3.31E-06	**2.84E-07**	**6.5**

Table 24: Results of ME95 dataset

could indicate that the scatter search is superior at finding the minimum, but as a drawback, it needs way more computational power to find the best solution it can. Simulated annealing seems like a good all-rounder, which can reliably find a good solution regardless of the situation. Iterated local search is fast but not as good as the others at finding good solutions.

6.4 Practical evaluation of mathematically optimal solutions

In mathematical modelling, the evaluation of how well the model and its solutions correspond to the situation being modelled is just as important if not more so as the solving process. Even if the methods used produce a global minimum for the formulated optimization problem, it is wholly irrelevant if the model does not have the main characteristic of the actual situation. The model used here uses the means as a similarity measure, which can potentially be inaccurate at times. As similarity is also measured with t-tests the mean should be fairly accurate, but it is still possible that the p-values do not meet the preset standards at the global minimum even if it would in some other solution.

In addition to statistical criteria, another important thing is to look at the produced lists and see if they meet the goals set for the solution. The best way to test the practical use of these lists would be to have them used in an actual test scenario, which sadly is not possible for this thesis at least. Since it is not possible to conduct tests on actual subjects, the lists are evaluated by comparing them based on descriptive statistics. As it would not make sense to evaluate the fitness of simulated data for practical situations, the evaluation is done only on real datasets, as they have some context for the evaluation.

The evaluation will be done here with the Lexize dataset to create more information for the TRILA researchers to back up the decisions regarding the parallel list creation process. The difference between methods in terms of the objective function values is rather small, which likely leads to not seeing any difference between generated lists by different methods. To amplify the differences here, the best-found solution is compared to the solution found with a greedy algorithm using the same seed. This way, the objective function values between compared solutions could differ enough that it would make a noticeable difference.

The greedy algorithm managed to find a solution with an objective function value of 2.47E-3. The best solution for this list equating problem found by scatter search was 1.53E-5. This means that the difference is not a massive one, but it could have an impact on how the lists turn out. To compare the solutions, means

Means

		Length	LemFreq	Bigram	Initrigram	Fintrigram	Discr.	Diff.
Best	List 1	0.0243	0.1313	0.0479	0.1313	0.0654	-0.0357	0.0405
	List 2	0.0243	0.131	0.0479	0.131	0.0666	-0.0382	0.0378
	p-value	1.0000	0.9992	1.0000	0.9992	0.9967	0.9930	0.9930
Greedy	List 1	-0.1261	0.0752	0.0082	0.0752	-0.0991	-0.0406	-0.1014
	List 2	-0.1261	0.0513	-0.0178	0.0513	-0.1093	-0.0307	-0.0803
	p-value	1.0000	0.9393	0.9253	0.9393	0.9703	0.9708	0.9288

Standard deviations

		Length	LemFreq	Bigram	Initrigram	Fintrigram	Discr.	Diff.
Best	List 1	1.1442	1.1818	1.0721	1.1818	1.1225	1.0096	1.3035
	List 2	1.0120	0.9370	0.7045	0.9370	0.7496	0.7236	0.7524
Greedy	List 1	1.1188	0.7860	0.7958	0.7860	0.8076	0.7930	0.8426
	List 2	0.7977	1.2652	0.9684	1.2652	1.1214	1.1472	0.8227

Table 25: List-wise means, t-test p-values and standard deviations with 2 differently generated stimuli sets in Lexize dataset

and standard deviations of the list generated by both methods were calculated for each of the variables, and they can be found in table 25. Standard deviation was particularly interesting as it was not included in the modelled objective function and, therefore, could differ substantially between the lists.

Looking at the results, the better solution has overall more equal means, which is to be expected. However, the differences are far from statistically significant as the smallest t-test p-value for greedy lists is around 0.925. The standard deviations have clear differences between lists, with the highest ones being around 0.5 for both lists. These are substantial differences as all the variables have unit variance. In difficulty within the best solution, the standard deviation of the first list is almost double what the first list has, which means that these lists are unlikely to seem equal in terms of item difficulty, even if the means are almost the same.

The observation about standard deviation is a reminder that formulating a practical problem into a mathematical optimization problem is not a straightforward one. The objective function needs to be carefully considered so that it fits the situation at hand. The simple approach chosen here raises the question of whether two lists can be called equal if they have the same means. The obvious answer here is no, as a list with no variation and a list where there is an equal number of items in both extremes can have the same mean. The difficulty here lies in defining equality, as it is mostly impossible to find exact matches for items and therefore, the equal lists have to be something other than the ones with the same characteristics.

Including additional descriptive statistics of the sample distribution like standard deviation in the objective function is possible. This approach still would raise the question of which descriptive statistic would one have to include in the objective function and which could be left out. Changing the objective function like this should not change how the optimization methods work, but as they calculate objective function values multiple times in each iteration, making the objective function more complex would lead to longer running times.

One different approach would be to match the items between lists using pairwise distances. The weakness of this approach would be that in the situation with a low number of good matches, the stopping condition could be met prematurely, as the swaps are rare to happen. Matching the pair's approach should be the gold standard for large stimuli databases with thousands of items. A potential approach for these databases could be initializing the list by calculating the closest pair for all items and constructing the initialization list from the non-overlapping pairs with the least distance between them. These initialized lists could then be tuned with either greedy search or one of the methods presented in this thesis.

Smaller datasets will need to be matched less exactly using something else than matching, as close matches are unlikely to happen. Here, the role of the researcher is highlighted, as the objective function has to be tailored exactly to match the research scenario. There is not a single correct way the objective function should be defined, but based on the results of the thesis, at least the standard deviations should be included in addition to the means. If the exact distributions of the variables are wished to be similar, adding skewness and kurtosis to the mix could be considered.

In the context of mathematical modelling, defining what it means for lists to be equal is not the most straightforward task. With matching the items, the definition is intuitive, but it leads to situations where the lists cannot be constructed for all datasets. The approach based on matching the distributions of variables using descriptive statistics is more universal but less exact. Choosing the approach that suits the situation at hand the most is a challenging task and needs to be done carefully to ensure a satisfying outcome.

7 Conclusions

This thesis presented two new metaheuristic approaches for stimuli selection problem. These methods were selected among the larger set of described metaheuristic methods. The selection was made based on the methods' strengths and the problem's special characteristics. The proposed methods were compared to the previously used simulation annealing-based approach in 3 simulated and 3 real-life datasets.

The main result of this thesis is that the proposed methods provide promising alternatives for the previously superior simulated annealing. None of the methods studied were good all-around, which gives more weight to the results, as not only do they propose new approaches, but they also give a base for choosing proper methods for different scenarios.

The greatest difference between simulated annealing and proposed methods is that the computational power needed for simulated annealing is less about the number of items and more about the parameters. This makes simulated annealing superior in scenarios, where the set of possible items is large. However, in the list of equating settings, simulated annealing was outperformed by the other two, which makes them better considering the original intentions of this thesis.

Multiple datasets with different characteristics were used in this thesis to see if said characteristics had any effect on which of the methods performed the best. Noticed differences between datasets seemed to be more about the sizes of the datasets than the distributions or the correlations of the different features. This is a positive observation, as it could imply that the guidelines of the method choice can be generalized to only consider the size of the data. However, it needs to be noted that different parameterizations could lead to different results and could need their own guidelines instead of one fixed set of values.

Another important thing studied was the practical implications of the differences. The evaluations of the optimal and greedy solutions showed that there is little to no difference between the tests generated by a metaheuristic and greedy algorithm. However, it should be noted that the evaluation was done in the smallest dataset, which could explain these minimal differences. If the standard deviations were also matched, the difference between solutions could have been far greater.

This thesis had quite a few limitations, some of which were dependent on the author's limited knowledge of psychological testing and scientific computing, and others were due to scarce amounts of previous research on the matter. The extent of this thesis did not allow accounting for all scenarios the author would have liked to address. Mostly this meant that the test scenarios were run on parameters predefined in testing with a single dataset. The strength of the approach is that at least there was some data to back up the choices instead of them being random or heuristically chosen. However, after conducting the research, it is safe to say that now there would be a better instinctive ability to select appropriate parameters for different scenarios. On the other hand, the ideas of in which situations which parameters work the best are also valuable information provided by this thesis.

The limited programming knowledge of the author leads to the use of R in the implementation of the algorithms, which means that lesser CPU times could be achieved through the use of more efficient programming languages such as C++ or Fortran. As the methods of this thesis and the implementation of the SOS!-algorithm were hoped to be used as a usable R package in the future, they are due to be recoded in the aforementioned languages before they can be considered a widely usable methodology. In addition to being overhauled to a more efficient programming language, the code also will be advanced to have some error handling to be easily usable by others than the author.

Important aspects of the generated test that were not considered were reliability and validity, as the focus was more on the methods instead of models. Calculating reliabilities would have required additional data from the test subject, which was inaccessible in this case. It would be an enlightening follow-up study to examine the reliabilities of automatically generated test versions. A similar could also be done with the validity of the test, yet this would be harder due to the less quantifiable nature of validity.

Another interesting future field of study regarding these methods is a way to make them able to self-parametrize. This means that the methods can determine at least somewhat optimal parameters for each use case automatically. How this should be implemented is still a question, but a simple approach could be gathering information of some data from all the scenarios that likely occur and using it to teach a decision tree for selecting each of the parameters. Depending on the performance of the decision trees, other machine learning or heuristic approaches should also be considered

When considering the methods themselves, simulated annealing was simplified here from the original SOS! to ease the implementation, apart from the exclusion of thermal equilibrium checks. This meant that temperature annealing likely happened at suboptimal times, at least for some of the runs. By annealing temperature only at times when thermal equilibrium had been found, we could be almost certain that continuing with said temperature would no longer yield any additional value. Instead, we had to rely on a heuristically approximating annealing schedule. However, at least in the case of SOS! the criterion of determining when the equilibrium was in a sense problematic and could need some revisioning.

Outside methods themselves, another area needing additional research is the modelling itself as at least to the knowledge of the author there have not been studies empirically testing the performance of the equated list. Testing like this would likely be done by conducting a single test where the stimuli of both lists are tested inside the same test in some randomized order. Testing both lists simultaneously is likely necessary to prevent any bias due to possible practice effects, doing one test list before the other could lead to.

In the case of even the most mathematically optimal lists failing to perform equally in the real test setting, the constructed model would have to be rebuilt from the ground up using a different way of measuring the similarity of lists than means. One possible approach could be the matching of the stimuli, often done in non-randomized trials, to ensure causal inference possibilities.

To conclude, this thesis proposed the alternatives to previously established SOS!-algorithm in solving stimuli selection problems. Additional new information was provided on the effect of the parameters for both proposed methods and SOS!, further increasing the knowledge that can be used to pick the optimal method and parameters for the situation at hand. As could be expected, none of the methods were superior to others in all scenarios, and they were shown to have their strengths and weaknesses. Further work is needed to refine the parameter selection and find optimal use cases for each of them.

References

Ackerman, T. A. (1989). An Alternative Methodology for Creating Parallel Test Forms Using the IRT Information Function. In *Annual Meeting of the National Council on Measurement in Education*, San Francisco, CA. ERIC Number: ED306279.

Adema, J. J. and van der Linden, W. J. (1989). Algorithms for Computerized Test Construction Using Classical Item Parameters. *Journal of Educational Statistics*, 14(3):279–290.

Allen, M. J. and Yen, W. M. (1979). *Introduction to Measurement Theory.* Brooks/Cole Publishing Company.

Almufti, S., Marqas, R., and Saeed, V. (2019). Taxonomy of bio-inspired optimization algorithms. *Journal of Advanced Computer Science & Technology*, 8:23.

American Educational Research Association, American Psychological Association, and National Council on Measurement in Education, editors (2014). *Standards* *for Educational and Psychological Testing.* American Educational Research Association, Washington, DC.

American Psychological Association (1954). Technical recommendations for psychological tests and diagnostic techniques. *Psychological Bulletin*, 51(2, Pt.2):1–38.

American Psychological Association, editor (2020). *APA Guidelines for Psychological Assessment and Evaluation.* American Psychological Association.

Anastasi, A. (1961). *Psychological testing.* Macmillan, New York, 2nd ed. edition.

Armstrong, B. C., Watson, C. E., and Plaut, D. C. (2012a). SOS! An algorithm and software for the stochastic optimization of stimuli. *Behavior Research Methods*, 44(3):675–705.

Armstrong, B. C., Watson, C. E., and Plaut, D. C. (2012b). Supplemental material for "SOS! An algorithm and software for the stochastic optimization of stimuli".

Armstrong, R. D., Jones, D. H., and Wang, Z. (1994). Automated Parallel Test Construction Using Classical Test Theory. *Journal of Educational Statistics*, 19(1):73–90.

Axehill, D. and Morari, M. (2010). Improved complexity analysis of branch and bound for hybrid MPC. In *49th IEEE Conference on Decision and Control (CDC)*, pages 4216–4222.

Bagirov, A., Karmitsa, N., Mäkelä, M. M., and Mäkelä, M. M. (2014). *Introduction to Nonsmooth Optimization: Theory, Practice and Software.* Springer International Publishing AG, Cham, Switzerland.

Baker, F. B. (2001). *The Basics of Item Response Theory.* ERIC Clearinghouse on Assessment and Evaluation, University of Maryland, College Park, MD, second edition.

Baker, F. B. and Kim, S.-H., editors (2014). *Item Response Theory: Parameter Estimation Techniques, Second Edition.* CRC Press, Boca Raton, FL, second edition.

Balota, D. A., Yap, M. J., Hutchison, K. A., Cortese, M. J., Kessler, B., Loftis, B., Neely, J. H., Nelson, D. L., Simpson, G. B., and Treiman, R. (2007). The English Lexicon Project. *Behavior Research Methods*, 39(3):445–459.

Banks, S. (2011). Ceiling Effect. In Kreutzer, J. S., DeLuca, J., and Caplan, B., editors, *Encyclopedia of Clinical Neuropsychology*, pages 506–507. Springer, New York, NY.

Baumert, J. and Demmrich, A. (2001). Test motivation in the assessment of student skills: The effects of incentives on motivation and performance. *European Journal of Psychology of Education*, 16(3):441–462.

Beglinger, L. J., Gaydos, B., Tangphao-Daniels, O., Duff, K., Kareken, D. A., Crawford, J., Fastenau, P. S., and Siemers, E. R. (2005). Practice effects and the use of alternate forms in serial neuropsychological testing. *Archives of Clinical Neuropsychology*, 20(4):517–529.

Bejar, I. I., Lawless, R. R., Morley, M. E., Wagner, M. E., Bennett, R. E., and Revuelta, J. (2003). A Feasibility Study of On-the-Fly Item Generation in Adaptive Testing. *The Journal of Technology, Learning and Assessment*, 2(3).

Benedict, R. and Zgaljardic, D. (1998). Practice Effects During Repeated Administrations of Memory Tests With and Without Alternate Forms. *Journal of clinical and experimental neuropsychology*, 20:339–52.

Birnbaum, A. (1968). Some latent trait models and Their in Inferring an Examinee's Ability. In Lord, F. and Novick, M., editors, *Statistical Theories of Mental Test Scores*, pages 397–479. Addison-Wesley, Reading.

Blum, C. and Roli, A. (2001). Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Comput. Surv.*, 35:268–308.

Boekkooi-Timminga, E. (1987). Simultaneous test construction by zero-one programming. *Methodika*, 1:101–112.

Boekkooi-Timminga, E. (1990). A Cluster-Based Method for Test Construction. *Applied Psychological Measurement*, 14(4):341–354.

Brysbaert, M. (2013). Lextale_fr A Fast, Free, and Efficient Test to Measure Language Proficiency in French. *Psychologica Belgica*, 53(1):23.

Brysbaert, M., Warriner, A. B., and Kuperman, V. (2014). Concreteness ratings for 40 thousand generally known English word lemmas. *Behavior Research Methods*, 46(3):904–911.

Charon, I. and Hudry, O. (1993). The noising method: a new method for combinatorial optimization. *Operations Research Letters*, 14(3):133–137.

Chelouah, R. and Siarry, P. (2000). A Continuous Genetic Algorithm Designed for the Global Optimization of Multimodal Functions. *Journal of Heuristics*, 6(2):191–213.

Chelune, G. (2003). Assessing reliable neuropsychological change. In Franklin, D., editor, *Prediction in Forensic and Neuropsychology : Sound Statistical Practices*, pages 115–138. Taylor & Francis Group, Mahwah, New Jersey.

Chen, W.-N., Zhang, J., Chung, H. S. H., Zhong, W.-L., Wu, W.-G., and Shi, Y.-h. (2010). A Novel Set-Based Particle Swarm Optimization Method for Discrete Optimization Problems. *IEEE Transactions on Evolutionary Computation*, 14(2):278–300.

Cohen, J. (1988). *Statistical Power Analysis for the Behavioral Sciences*. Routledge, New York, 2 edition.

Collie, A., Maruff, P., Darby, D., and McStephen, M. (2003). The effects of practice on the cognitive test performance of neurologically normal individuals assessed at brief test-retest intervals. *Journal of the International Neuropsychological Society : JINS*, 9:419–28.

Coupé, C. (2011). BALI: A software tool to build experimental material in psycholinguistics. *Proceedings of Architectures and Mechanisms for Language Processing (AMLaP) Conference 2011.*

Cover, T. M. (1991). *Elements of information theory.* Wiley series in telecommunications. John Wiley & Sons, New York.

Crawford, J. R., Stewart, L. E., and Moore, J. W. (1989). Demonstration of savings on the AVLT and development of a parallel form. *Journal of Clinical and Experimental Neuropsychology*, 11(6):975–981.

Cromley, J. G. and Azevedo, R. (2007). Testing and refining the direct and inferential mediation model of reading comprehension. *Journal of Educational Psychology*, 99(2):311–325.

Cronbach, L. J. (1951). Coefficient alpha and the internal structure of tests. *Psychometrika*, 16(3):297–334.

Curry, H. B. (1944). The method of steepest descent for non-linear minimization problems. *Quarterly of Applied Mathematics*, 2(3):258–261.

DeMars, C. E. (2000). Test Stakes and Item Format Interactions. *Applied Measurement in Education*, 13(1):55–77.

Di Pillo, G. and Grippo, L. (1989). Exact Penalty Functions in Constrained Optimization. *SIAM Journal on Control and Optimization*, 27(6):1333–1360.

Dorigo, M. and Di Caro, G. (1999). Ant colony optimization: a new meta-heuristic. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 2, pages 1470–1477 Vol. 2.

Dueck, G. (1993). New Optimization Heuristics: The Great Deluge Algorithm and the Record-to-Record Travel. *Journal of Computational Physics*, 104(1):86–92.

Dueck, G. and Scheuer, T. (1990). Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. *Journal of Computational Physics*, 90(1):161–175.

Duke, N. K. and Cartwright, K. B. (2021). The Science of Reading Progresses: Communicating Advances Beyond the Simple View of Reading. *Reading Research Quarterly*, 56(S1):S25–S44.

Eddelbuettel, D. and Francois, R. (2011). Rcpp: Seamless R and C++ Integration. *Journal of Statistical Software*, 40:1–18.

Eskandar, H., Sadollah, A., Bahreininejad, A., and Hamdi, M. (2012). Water cycle algorithm – A novel metaheuristic optimization method for solving constrained engineering optimization problems. *Computers & Structures*, 110-111:151–166.

Feo, T. and Resende, M. (1995). Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization*, 6:109–133.

Fisher, R. A. (1925). Applications of "Student's" Distribution. *Metron*, 5:90–104.

Fraser, A. S. (1957). Simulation of Genetic Systems by Automatic Digital Computers I. Introduction. *Australian Journal of Biological Sciences*, 10(4):484–491.

Geem, Z. W., Kim, J. H., and Loganathan, G. (2001). A New Heuristic Optimization Algorithm: Harmony Search. *SIMULATION*, 76(2):60–68.

Gilhooly, K. J. and Logie, R. H. (1980). Age-of-acquisition, imagery, concreteness, familiarity, and ambiguity measures for 1,944 words. *Behavior Research Methods & Instrumentation*, 12(4):395–427.

Glover, F. (1977). Heuristics for Integer Programming Using Surrogate Constraints. *Decision Sciences*, 8(1):156–166.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549.

Glover, F. (1990). Tabu search—part II. *ORSA Journal on Computing*, 2:4–32.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., USA, 1st edition.

Gough, P. B. and Tunmer, W. E. (1986). Decoding, Reading, and Reading Disability. *Remedial and Special Education*, 7(1):6–10.

Guasch, M., Haro, J., and Boada, R. (2017). Clustering Words to Match Conditions: An Algorithm for Stimuli Selection in Factorial Designs. *Psicologica: International Journal of Methodology and Experimental Psychology*, 38(1):111–131.

Göbel, N., Cazzoli, D., Gutbrod, C., Müri, R. M., and Eberhard-Moscicka, A. K. (2023). An item sorting heuristic to derive equivalent parallel test versions from multivariate items. *PLOS ONE*, 18(4):e0284768.

Harkio, N. and Pietilä, P. (2016). The Role of Vocabulary Breadth and Depth in Reading Comprehension: A Quantitative Study of Finnish EFL Learners. *Journal of Language Teaching and Research*, 7:1079.

Herbers, J. E., Cutuli, J. J., Supkoff, L. M., Heistad, D., Chan, C.-K., Hinz, E., and Masten, A. S. (2012). Early Reading Skills and Academic Achievement Trajectories of Students Facing Poverty, Homelessness, and High Residential Mobility. *Educational Researcher*, 41(9):366–374. Publisher: American Educational Research Association.

Hooke, R. and Jeeves, T. A. (1961). “ Direct Search” Solution of Numerical and Statistical Problems. *Journal of the ACM*, 8(2):212–229.

Hsu, L. M. (1999). A comparison of three methods of identifying reliable and clinically significant client changes: Commentary on Hageman and Arrindell. *Behaviour Research and Therapy*, 37(12):1195–1202.

Hyvärinen, A. (1997). New Approximations of Differential Entropy for Independent Component Analysis and Projection Pursuit. In *Advances in Neural Information Processing Systems*, volume 10. MIT Press.

Iverson, G. L. (2011). Reliable Change Index. In Kreutzer, J. S., DeLuca, J., and Caplan, B., editors, *Encyclopedia of Clinical Neuropsychology*, pages 2150–2153. Springer, New York, NY.

Ivnik, R. J., Smith, G. E., Lucas, J. A., Petersen, R. C., Boeve, B. F., Kokmen, E., and Tangalos, E. G. (1999). Testing normal older people three or four times at 1- to 2-year intervals: Defining normal variance. *Neuropsychology*, 13(1):121–127.

Jacobson, N. S. and Truax, P. (1991). Clinical significance : A statistical approach to defining meaningful change in psychotherapy research. *Journal of Consulting and Clinical Psychology*, 59(1):12–19.

Joanes, D. N. and Gill, C. A. (1998). Comparing measures of sample skewness and kurtosis. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 47(1):183–189.

Jones, M. C. and Sibson, R. (1987). What is Projection Pursuit? *Journal of the Royal Statistical Society. Series A (General)*, 150(1):1–37.

Karaboga, D. (2005). An Idea Based on Honey Bee Swarm for Numerical Optimization. *Technical Report, Erciyes University.*

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4.

Kim, Y.-S. G. (2017). Why the Simple View of Reading Is Not Simplistic: Unpacking Component Skills of Reading Using a Direct and Indirect Effect Model of Reading (DIER). *Scientific Studies of Reading*, 21(4):310–333.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, 220(4598):671–680.

Kolesar, P. J. (1967). A Branch and Bound Algorithm for the Knapsack Problem. *Management Science*, 13(9):723–735.

Kučera, H. and Francis, W. (1967). *Computational analysis of present-day American English.* Brown University Press, Providence, RI.

Lahl, O. and Pietrowsky, R. (2006). EQUIWORD: A software application for the automatic creation of truly equivalent word lists. *Behavior Research Methods*, 38(1):146–152.

Laine, M. and Virtanen, P. (1999). WordMill lexical search program.

Laufer, B. (1989). What Percentage of Text-Lexis is Essential for Comprehension? *Special language: From humans thinking to thinking machines*, pages 316–326.

Lemhöfer, K. and Broersma, M. (2012). Introducing LexTALE: A quick and valid Lexical Test for Advanced Learners of English. *Behavior Research Methods*, 44(2):325–343.

Lintz, E. N., Lim, P. C., and Johnson, M. R. (2021). A new tool for equating lexical stimuli across experimental conditions. *MethodsX*, 8:101545.

Lonigan, C. J., Burgess, S. R., and Schatschneider, C. (2018). Examining the Simple View of Reading with elementary school children: Still simple after all these years. *Remedial and Special Education*, 39(5):260–273.

Lord, F. M. (1977). Practical Applications of Item Characteristic Curve Theory. *Journal of Educational Measurement*, 14(2):117–138.

Lund, K. and Burgess, C. (1996). Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers*, 28(2):203–208.

Lyytinen, H., Richardson, U., and Aro, M. (2019). Developmental Dyslexia in Finnish. In Perfetti, C., Pugh, K., and Verhoeven, L., editors, *Developmental Dyslexia across Languages and Writing Systems*, pages 118–132. Cambridge University Press, Cambridge.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, volume 5.1, pages 281–298. University of California Press.

Martin, O., Otto, S. W., and Felten, E. W. (1992). Large-step markov chains for the TSP incorporating local search heuristics. *Operations Research Letters*, 11(4):219–224.

McDonald, R. P. (1999). *Test Theory: A Unified Treatment*. Psychology Press, New York.

Michael, G. C. (1972). A Review of Heuristic Programming. *Decision Sciences*, 3(3):74–100.

Mitchell, K. M. (2024). Does reading ability predict student success? A scoping review. *Nurse Education Today*, 136:106150.

Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100.

Morrison, C. and Ellis, A. (1995). Roles of Word Frequency and Age of Acquisition in Word Naming and Lexical Decision. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 21:116–133.

Niemensivu, T. (2023). Timi Niemensivu / Master thesis · GitLab.

OECD (2019). PISA 2018 Results (Volume I). In *How PISA results are reported: What is a PISA score?* OECD Publishing, Paris.

Oneil, H., Sugrue, B., and Baker, E. (1995). Effects of Motivational Interventions on the National Assessment of Educational Progress Mathematics Performance. *Educational Assessment*, 3:135–157.

Paivio, A., Yuille, J., and Madigan, S. (1968). Concreteness, imagery, and meaningfulness values for 925 nouns. *Journal of experimental psychology*, 76:Suppl:1–25.

Papenberg, M. and Klau, G. (2021). Using anticlustering to partition data sets into equivalent parts. *Psychological Methods*, 26(2):161–174.

Pham, D. and Castellani, M. (2009). The Bees Algorithm: Modelling foraging behaviour to solve continuous optimization problems. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 223:2919–2938.

R Core Team (2023). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria.

Ruff, R. M., Light, R. H., Parker, S. B., and Levin, H. S. (1996). Benton controlled oral word association test: Reliability and updated norms. *Archives of Clinical Neuropsychology*, 11(4):329–338.

Räsänen, P., Aunio, P., Laine, A., Hakkarainen, A., Väisänen, E., Finell, J., Rajala, T., Laakso, M.-J., and Korhonen, J. (2021). Effects of Gender on Basic Numerical and Arithmetic Skills: Pilot Data From Third to Ninth Grade for a Large-Scale Online Dyscalculia Screener. *Frontiers in Education*, 6.

Salmela, R., Lehtonen, M., Garusi, S., and Bertram, R. (2021). Lexize: A test to quickly assess vocabulary knowledge in Finnish. *Scandinavian Journal of Psychology*, 62(6):806–819.

Scarborough, H. (2001). Connecting early language and literacy to later reading disabilities: Evidence, theory, and practice. In Neuman, S. and David, D., editors, *Handbook for research in early literacy, Volume 1*. Guildford Press, New York, NY.

Sergienko, I. V. and Shylo, V. P. (2006). Problems of discrete optimization: Challenges and main approaches to solve them. *Cybernetics and Systems Analysis*, 42(4):465–482.

Shi, Y. and Eberhart, R. (1998). A modified particle swarm optimizer. In *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*, pages 69–73.

Speer, D. C. and Greenbaum, P. E. (1995). Five methods for computing significant individual client change and improvement rates: Support for an individual growth curve approach. *Journal of Consulting and Clinical Psychology*, 63(6):1044–1048.

Sundre, D. L. (1999). Does examinee motivation moderate the relationship between test consequences and test performance. In *Annual Meeting of the American Educational Research Association*, Montreal, Quebec, Canada.

Swanson, L. and Stocking, M. L. (1993). A Model and Heuristic For Solving Very Large Item Selection Problems. *Applied Psychological Measurement*, 17(2):151–166.

Sörensen, K. (2015). Metaheuristics—the metaphor exposed. *International Transactions in Operational Research*, 22(1):3–18.

The British Psychological Society, editor (2017). *Psychological Testing: A Test User's Guide*. The British Psychological Society, Leicester, UK.

Theisen, M. E., Rapport, L. J., Axelrod, B. N., and Brines, D. B. (1998). Effects of Practice in Repeated Administrations of the Wechsler Memory Scale-Revised in Normal Adults. *Assessment*, 5(1):85–92.

Theunissen, T. (1986). Some Applications of Optimization Algorithms in Test Design and Adaptive Testing. *Applied Psychological Measurement*, 10(4):381–389.

Theunissen, T. J. J. M. (1985). Binary programming and test design. *Psychometrika*, 50(4):411–420.

Toglia, M. P. (1978). *Handbook of semantic word norms*. Lawrence Erlbaum Associates, Hillsdale, N.J, New York.

Toste, J. R., Didion, L., Peng, P., Filderman, M. J., and McClelland, A. M. (2020). A Meta-Analytic Review of the Relations Between Motivation and Reading Achievement for K–12 Students. *Review of Educational Research*, 90(3):420–456. Publisher: American Educational Research Association.

Tsang, E. and Voudouris, C. (1997). Fast local search and guided local search and their application to British Telecom's workforce scheduling problem. *Operations Research Letters*, 20(3):119–127.

Tunmer, W. E. and Chapman, J. W. (2012). The Simple View of Reading Redux: Vocabulary Knowledge and the Independent Components Hypothesis. *Journal of Learning Disabilities*, 45(5):453–466.

Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, LIX(236):433–460.

Turku Research Institute for Learning Analytics (2022). FUNA – Oppimisanalytiikka.

Uchiyama, C. L., D'Elia, L. F., Dellinger, A. M., Becker, J. T., Selnes, O. A., Wesch, J. E., Chen, B. B., Satz, P., van Gorp, W., and Miller, E. N. (1995). Alternate forms of the auditory-verbal learning test: issues of test comparability, longitudinal reliability, and moderating demographic variables. *Archives of Clinical Neuropsychology*, 10(2):133–145.

Valev, V. (1998). Set partition principles revisited. In Amin, A., Dori, D., Pudil, P., and Freeman, H., editors, *Advances in Pattern Recognition*, pages 875–881, Berlin, Heidelberg. Springer.

van Casteren, M. and Davis, M. H. (2007). Match: A program to assist in matching the conditions of factorial experiments. *Behavior Research Methods*, 39(4):973–978.

Van Der Linden, W. J. (1987). Automated Test Construction Using Minimax Programming. In Van Der Linden, W. J., editor, *IRT-based Test Construction*, number 15 in Project Psychometric Aspects of Item Banking, pages 44–60. University of Twente, Faculty of Educational Science and Technology, Enschede, the Netherlands.

van der Linden, W. J. (1996). Assembling Tests for the Measurement of Multiple Traits. *Applied Psychological Measurement*, 20(4):373–388.

van der Linden, W. J. (1998). Optimal Assembly of Psychological and Educational Tests. *Applied Psychological Measurement*, 22(3):195–211.

Welch, B. L. (1947). The Generalization of 'Student's' Problem when Several Different Population Variances are Involved. *Biometrika*, 34(1/2):28–35.

Wilcoxon, F. (1945). Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6):80–83.

Wilson, M. (1988). MRC psycholinguistic database: Machine-usable dictionary, version 2.00. *Behavior Research Methods, Instruments, & Computers*, 20(1):6–10.

Wise, S. L. and DeMars, C. E. (2005). Low Examinee Effort in Low-Stakes Assessment: Problems and Potential Solutions. *Educational Assessment*, 10(1):1–17.

Woodhouse, B. and Jackson, P. H. (1977). Lower bounds for the reliability of the total score on a test composed of non-homogeneous items: II: A search procedure to locate the greatest lower bound. *Psychometrika*, 42(4):579–591.

Wright, T. S. and Cervetti, G. N. (2017). A Systematic Review of the Research on Vocabulary Instruction That Impacts Text Comprehension. *Reading Research Quarterly*, 52(2):203–226.

Yang, X.-S. (2009). Firefly Algorithms for Multimodal Optimization. In Watanabe, O. and Zeugmann, T., editors, *Stochastic Algorithms: Foundations and Applications*, pages 169–178, Berlin, Heidelberg. Springer.

Yang, X.-S. (2010). A New Metaheuristic Bat-Inspired Algorithm. In González, J. R., Pelta, D. A., Cruz, C., Terrazas, G., and Krasnogor, N., editors, *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, pages 65–74. Springer, Berlin, Heidelberg.

Zhang, Y., Wang, S., and Ji, G. (2013). A Rule-Based Model for Bankruptcy Prediction Based on an Improved Genetic Ant Colony Algorithm. *Mathematical Problems in Engineering*, 2013:e753251.

Zimmermann, T. and Salamon, P. (1992). The demon algorithm. *International Journal of Computer Mathematics*, 42:21–31.

A Additional results

Definition 7. Estimator $\hat{\theta}$ is an unbiased estimator of parameter θ, if

$$E(\hat{\theta}) = \theta.$$

Theorem 3. *Sample mean $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ is unbiased estimator of the population-level mean μ.*

Proof. As the $E(x_i) = \mu$, following the linearity of the expected value we get

$$E(\bar{x}) = E\left(\frac{1}{n}\sum_{i=1}^{n} x_i\right) = \frac{1}{n}\sum_{i=1}^{n} E(x_i) = \frac{1}{n} n\mu = \mu.$$

□

Theorem 4. *Sample variance $s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(\bar{x} - x_i)^2$ is unbiased estimator of the population-level variance $\sigma^2 = E[(x-\mu)^2]$.*

Proof. As the $E(x_i) = \mu$, we can rewrite

$$\sum_{i=1}^{n}(\bar{x} - x_i)^2 = \sum_{i=1}^{n}\left((x_i - \mu)^2 - (\bar{x} - \mu)^2\right) = \sum_{i=1}^{n}\left((x_i - \mu)^2\right) - n(\bar{x} - \mu)^2.$$

With the previous and following the linearity of the expected value we get

$$E(s^2) = E\left(\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2\right) = \frac{1}{n-1}\left(\sum_{i=1}^{n} E((x_i - \mu)^2) - nE((\bar{x} - \mu)^2)\right).$$

Based on the definition of the variance

$$\sum_{i=1}^{n} E((x_i - \mu)^2) = n\sigma^2.$$

As the sample mean is an unbiased estimator and variance is additive with property $var(\alpha x) = \alpha^2 var(x)$, we get

$$E((\bar{x}-\mu)^2) = var(\bar{x}) = var\left(\frac{1}{n}\sum_{i=1}^{n} x_i\right) = \frac{1}{n}\sum_{i=1}^{n} var\,(x_i) = \frac{1}{n}n\sigma^2 = \sigma^2.$$

Combining previous equations gives us

$$E(s^2) = \frac{1}{n-1}\left(\sum_{i=1}^{n} E((x_i-\mu)^2) - nE((\bar{x}-\mu)^2)\right) = \frac{1}{n-1}\left(n\sigma^2 - \sigma^2\right) = \sigma^2.$$

□

From now on $A \subseteq \mathbb{R}^n$, $n \in \mathbb{N}$ and $x, y \in A$.

Theorem 5. *Let $f : A \to \mathbb{R}$ be some arbitrary function, $A \subseteq \mathbb{R}^n$. Now define the function $-f$ by $-f(\boldsymbol{x}) = -(f(\boldsymbol{x}))$ for all $x \in A$. If the $\boldsymbol{x}^*$ is the global minimum of f, it is also the global maximum of $-f$.*

Proof. As the x^* is the global minimum of the f, $f(\boldsymbol{x}^*) \leq f(\boldsymbol{y})$, for all $\boldsymbol{y} \in A$. By multiplying both sides of the inequality by -1 we get

$$-f(\boldsymbol{x}^*) \geq -f(\boldsymbol{y}), \text{ for all } \boldsymbol{y} \in A,$$

which makes $\boldsymbol{x}^*$ the global maximum of $-f$. □

Theorem 6. *Let $f : A \to \mathbb{R}$ be some arbitrary function, $A \subseteq \mathbb{R}^n$. If f is convex, the maximum of f in some discrete set $S_1 \subseteq \mathbb{N}$ is the same as the maximum in the set $S_2 = conv(S_1)$.*

Example 1. Consider minimising a function $f : \mathbb{R} \leftarrow \mathbb{R}, f(x) = 5x^3 - 5x^2 + x$. In the case of binary constraint set $\{0, 1\}$, minimum of f is at $x = 0$, as $f(0) = 0$ and $f(1) = 0$. However, when considering the relaxed constraint set $[0, 1]$, the minimum is far from the minimum of the original binary set. As the f is continuous and smooth, it gains its minimum in a closed interval either at the roots of the derivative or the endpoints of the interval. Roots of the derivative are solutions to equation $15x^2 - 10x + 1 = 0$ or $x = \frac{5\pm\sqrt{10}}{15}$. Using the second derivative $f''(x) = 30x - 10$, it can be seen that the derivative is increasing before $x = \frac{1}{3}$ and decreasing after. This means that the $x = \frac{5-\sqrt{10}}{15}$ is local maximum and $x = \frac{5+\sqrt{10}}{15}$ is the local minimum. $f\left(\frac{5+\sqrt{10}}{15}\right) = \frac{-5-4\sqrt{10}}{135} < 0$, which means that it is also the minimum in interval $[0, 1]$. It is to be noted that the minimum of the relaxed case is closer to $x = 1$ than to $x = 0$, which means that this would lead to a false conclusion of $x = 1$ being the better solution for the minimization problem if only information gained from the relaxed problem. In this case, it is easy to see what the actual optimum is, but in practical cases, the constraints and objective function are more complex making similar analysis similar to an exhaustive search.