Master's Thesis

Specification Search as a Combinatorial Optimization Problem: A Simulation

Department of Statistics
Ludwig-Maximilians-Universität München

Karik Siemund

Munich, March 12th 2023

Under supervision of Prof. Dr. Christian Heumannand in cooperation with Dr. David Goretzko.

Abstract

Structural Equation Modelling (SEM) is a powerful tool for specifying latent variable models, including Confirmatory Factor Analysis (CFA). In CFA, manifest variables are specified to load on latent variables, and the latent variables reflect the shared variance among their predicting observed indicators. However, due to SEMs high flexibility misspecification of such a model can happen easily, and holistic theories are usually needed to understand the relations between measured and latent variables. Traditionally, CFA and SEM are viewed as confirmatory techniques, where misspecified models should be adapted according to theoretical considerations and tested on new data. However, recent research has explored the potential for improving misspecified models using metaheuristic algorithms. In this Master's Thesis, we aimed to contribute to this research by evaluating the performance of metaheuristic algorithms, including Tabu Search, Simulated Annealing, Genetic Algorithms, Particle Swarm Optimization, and hybrid Ant-Colony-Optimization. We used simulated data with varying population models, sample sizes, and misspecified starting models, and found that a simple Genetic Algorithm followed by Particle Swarm Optimization performed the best overall, while Tabu Search outperformed it under certain more complex conditions. We also investigated hybrid algorithms combining Genetic Algorithm and Particle Swarm Optimization with Tabu Search, which performed better under the most complex conditions. Our results indicate the potential of this approach, and we offer researchers our implemented algorithms in the form of an R package called *MetaSS*. However, we advise future studies to analyze possible confounders of our findings, such as the objective function and implementation of algorithms, to further refine this approach. Overall, our study stands as an initial milestone for applied researchers to improve misspecified models using metaheuristic algorithms.

Contents

List of Figures

List of Tables

1 Introduction

Latent variable modelling enables its users to gain a comprehensive understanding of the unobservable structures that may underlie measured data. Therefore, it is particularly important in areas where the quantities of interest are not directly measurable. This is especially true in the social sciences, where the mental and behavioural qualities of humans and groups are at the center of attention. While a behavior may occur at a given time or not, we are often interested in the mental processes or entities that lead to such behavior with a certain frequency. In the psychometric literature, this idea was first described by Spearman (1904), who "advocated a Correlational Psychology" that connects measured responses on a test, such as a school test, based on their similarity or dissimilarity to specific latent variables, named *factors*. His idea was later combined with path analysis, a concept that connects multiple regression equations into a unified model, giving birth to what is now at the core of Structural Equation Modelling (SEM).

In the SEM modeling framework, observed and latent variables are connected in a maximally flexible manner. Both manifest and latent variables can serve as regressors or predictors and regressands or outcomes, possibly at the same time. This flexibility allows for a variety of theories and hypotheses to be modeled, both correlational and including a mean structure. These theories usually form the starting point of a SEM analysis, as they are translated into a set of regression equations that serve as the basis for a SEM model. As such, linear and multiple regression are special cases of SEM in which no latent variables are specified and there is only one regressand.[1]

These theories can incorporate a wide range of prior knowledge. Traditionally, SEM is often used after simpler or more exploratory techniques are applied to the data or to different data from a similar source, such as two samples answering the same questionnaire. Thus, for many applications, it can be assumed that researchers know a part of the modeling space or have a good understanding of certain aspects of it before employing SEM. However, the flexibility of SEM can also come at a cost, as mistakes can easily be made when modeling new aspects of a theory or testing a new hypothesis, leading to a high risk of model misspecification. As of now, there are very few commonly accepted rules or guidelines on what to do in the presence of a misspecified model.

From a classical statistical and psychometric perspective, it would be considered good practice to respecify the initial model based on theoretical considerations and

[1] To fully understand the following chapters, basic SEM terminology should be familiar. For a quick introduction, please refer to the chapter on SEM.

test this respecified model on new data (Mueller, 1997). Using the original data to find a better model would be labeled as data fishing. This strict practice has, however, seen some relaxation, and what we can call *data mining* is now much more common in SEM (K. Marcoulides & Falk, 2018). The statistical community has adopted the idea that learning from data can be an important aspect of modeling and finding a good approximation of reality, as well as giving value to data, which has often been collected through significant effort.

This Master's thesis adopts the aforementioned perspective and aims to contribute to an expanding field of research that addresses the issue of post hoc improving confirmatory SEM models. It has been noted that misspecified models frequently occur in the literature and that data-driven methods for dealing with such models are limited in both quality and quantity (Goretzko et al., in press; K. Marcoulides & Falk, 2018). Therefore, we compare a variety of different methods that allow for searching through possible model alternatives based on a misspecified starting model, a function that evaluates the new proposals and ultimately suggests an improved model. This model should assist the researcher in deciding what to do with the initial model and inform about what structures may better represent reality.

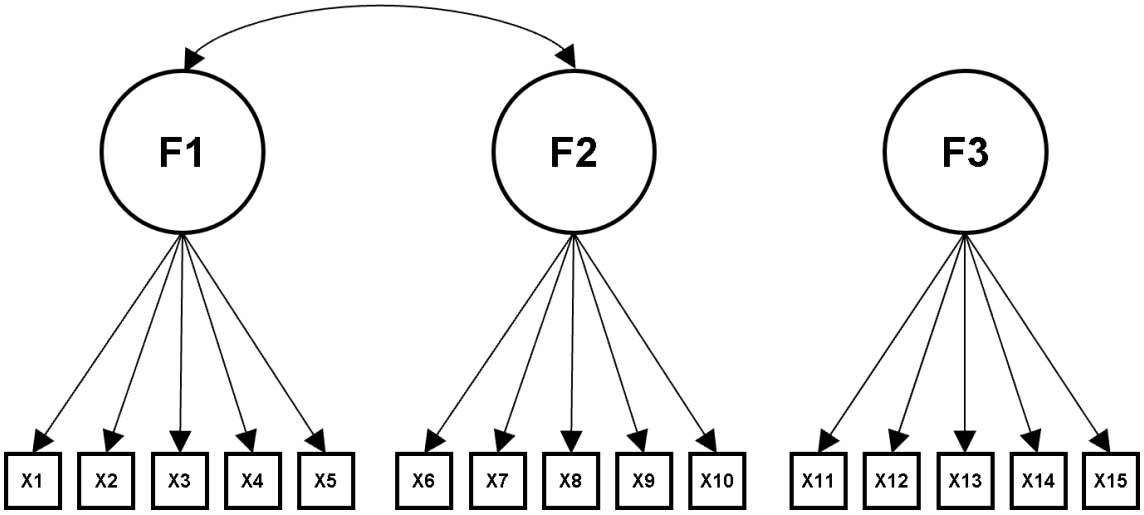
In the following sections, we will first describe the general approach underlying our analysis, followed by a critical summary of current methods and studies in the field. Afterwards, we will outline our method in more detail, introducing the SEM modelling framework, the used algorithms, and our simulation plan. Next, we will provide the results of our analysis, which we will subsequently discuss in detail and in the light of previous and further research.

1.1 Specification Search as a Combinatorial Optimization Problem

The most general perspective we can take when looking at how we want to tackle post hoc improvement of SEM models is through optimization. In optimization, our goal is to find a good set of input values to a function, which is called a *solution*. The quality of a solution is determined by its function value or a transformation thereof. In our case, a solution represents a SEM model that captures our knowledge and ideas about a theory. However, it is common for such models to be misspecified. From an optimization perspective, a misspecified model is just one of many possible solutions to our optimization problem, but it is not the best one. Therefore, we aim to find a better solution starting from this initial, misspecified model. This process is called

specification search in the SEM literature (Long, 1983).

Figure 1
Example of a SEM model similar to the average true population model.



In order to translate specification search into an optimization problem, we need to first define how a model can be viewed as a solution. This is achieved by using a binary indicator vector ϑ, where each entry represents a possible parameter to be estimated: if the entry is 0, we do not estimate the corresponding parameter; if it is 1, we want to estimate it. The resulting vector has a length of k, which is the total number of parameters we could estimate in our model. It is worth noting that we can also conduct our search over only a part of this vector while leaving the rest at pre-specified values. To better understand this approach, let us consider an example SEM model (see Fig. 1). Following standard SEM notation, squares represent manifest variables, while circles represent latent variables. Our model is based on 15 observed variables ($X_1, ..., X_{15}$) and three latent variables (F_1, F_2, and F_3). Looking only at factor loadings ("$=\sim$") and latent variable correlations ("$\sim\sim$"), this results in $k = 48$ possible parameters that can be estimated, ignoring issues of identifiability for now. Therefore, our binary indicator vector would look like this:

$$\vartheta = \begin{pmatrix} F_1 =\sim X_1 \\ \dots \\ F_1 =\sim X_{15} \\ F_2 =\sim X_1 \\ \dots \\ F_1 \sim\sim F_2 \\ F_1 \sim\sim F_3 \\ F_2 \sim\sim F_3 \end{pmatrix} = \begin{pmatrix} 1 \\ \dots \\ 0 \\ 0 \\ \dots \\ 1 \\ 0 \\ 0 \end{pmatrix}.$$

The space of all possible combinations of this vector can be taken as our object set, which fundamentally constitutes our optimization problem. In this case, as we have only binary values in each entry, we would speak of a *combinatorial* optimization problem (COP). A possible solver of this problem could be an algorithm that systematically tries out different solutions, as taking a brute-force approach, i.e., looking at all possible 2^{48} solutions would not be feasible.

To evaluate the quality of each solution and determine the direction of further search, there exist an infinite number of criteria. In optimization, a chosen criterion is called an *objective function*, which is always very problem-specific. In our case, we aim to examine the quality of a fitted SEM model. For this task, a vast variety of research exists (West et al., 2012). Even though significance tests can be employed, they are usually neglected for descriptive measures, as they are strongly related to the sample size[2]. Most of these descriptive measures can be categorized into *fit indices* and *information criteria*, which will be discussed later on in more detail. These measures all allow us to evaluate and compare the respectively fitted models and thus measure and order our modeling space. In this way, we can meaningfully direct our search onto a route that leads to better quality models.

1.2 Current State of the Literature

Looking at previous literature, only five studies have investigated specification search under the lens of combinatorial optimisation at a (close to) realistic level, even though a broader literature exists (Chou & Yang, 2013; Jing et al., 2022; G. Marcoulides & Leite, 2014; Schroeders et al., 2022). Much of this is, however, made up of instructional papers on how different optimisation algorithms could be used in specification search (Drezner & Marcoulides, 1999; G. Marcoulides, 2009; G. Marcoulides & Drezner, 2001, 2003; G. Marcoulides et al., 1998). They describe the use of *Ant-Colony-Optimisation*

[2]For a more theoretical perspective on fit indices, please refer to section 2.1

(ACO; G. Marcoulides & Drezner, 2003), a *Genetic Algorithm* (GA; G. Marcoulides & Drezner, 2001), *Simulated Annealing* (SA; Drezner & Marcoulides, 1999), *Tabu Search* (TS; G. Marcoulides et al., 1998) and *Ruin-and-Recreate* (RaR; G. Marcoulides, 2009) demonstrated on toy examples. As of now, this research is also focused mostly on CFA or CFA-like SEM models (e.g. with cross-loadings or residual correlations).

The first study we want to mention compared ACO and TS in a small simulation, utilising two different true population models with three latent and twelve manifest variables, one of which was a simple CFA model, while the other had shared variance among the manifest variables. Under each of the two true population models, they used a different starting model for the search. For the simple population structure, three wrong loadings were specified while under the other model 16 wrong parameters were initially misspecified, most of them being omitted residual covariances. Such residual covariances are most often investigated in longitudinal or multi-method SEM analysis. They replicated each condition 50 times under a fixed sample size of N=1000, resulting in 100 total datasets. Additionally, they compared transformations of three different quality measures: χ^2 value, Bayesian Information Criterion (BIC), and a combination of the Root Mean Square Error of Approximation (RMSEA), the Comparative Fit Index (CFI), and the Tucker-Lewis Index (TLI), which were run five times on each dataset. In addition to evaluating single runs, they also utilised a batch- or model-averaging-like approach by choosing the intersection of model parameters found in the best three models of each five runs.

In general, TS outperformed ACO by a large margin in each condition. ACO consistently produced overparametrized models, especially when using χ^2 and the fit indices as an objective function and under the complex population model. Under both the simple and complex population model as well as the BIC or fit indices objective functions, over 90% of solutions contained all of the correct parameters. However, *only* the correct model was never obtained more than 50% of the time. In contrast, TS found the correct model in all runs, under all objective functions and under all conditions. They link the significantly worse performance of the ACO to a few reasons, the most important of them being a lack of sensitivity of the objective functions and the high frequency of non-converging SEM models, slowing down and hindering a fast and informed search.

Although the true population models used by G. Marcoulides & Leite (2014) are good representations of actual fitted models in applied research, there are a few limitations worth mentioning. First, the complex population model focuses on a relatively small part of SEM models, namely longitudinal and mixed-methods designs. In con-

trast, bi-factor models, models with cross-loadings, and other SEM types are much more common because they do not require the complex data collection processes associated with longitudinal or mixed-methods designs. Second, although the authors argue that sample size does not influence specification search, it can affect the objective function. For example, when the latter contains the sample size-sensitive RMSEA, the sample size can affect the objective function, and also the difficulty of the optimization problem. When sample sizes are large, the population structure will be more clearly present in the simulated data. In contrast, if sample sizes are small, there is expected to be more (quasi-)random noise in the data. As $N = 1000$ is a large sample size for studies in the social sciences, the modelling scenario is rather restrictive and can be seen as representative of *good* modelling conditions. Despite several other limitations worth mentioning, we question the choice of objective function. Although the χ^2 value provides a good benchmark for more complex measures, and the BIC is a theoretically sound choice for an optimization criterion, the function to combine the RMSEA, CFI, and TLI - max(1-RMSEA, CFI, TLI) - does not take into account that the fit indices, although always practically in the range of $[0, 1]$, are not on the same scale. A 1-RMSEA value of 0.94 would be considered sufficient by Hu & Bentler (1999), while the same value for the CFI or TLI would fall just below the recommended cutoff of 0.95. This hinders a meaningful evaluation of proposed solutions.

In a simulation study, Murohashi & Toyoda (2007) used a GA combined with local search and were the first to test the idea of using metaheuristic algorithms for specification search. Usually, combining local search with GA would involve looking at every neighbor of a proposed child solution and choosing the best one. To reduce computational burden, Murohashi & Toyoda (2007) looked at neighbors sequentially and chose the first model to improve the original. This approach means that the best improvement can very easily be overlooked. Regarding their simulation design, they focused more on the algorithm than the data compared to the other simulation studies. To investigate the sensitivity of their GA to different hyperparameter settings, they examined the following conditions: the population size (5, 10, and 20) and the mutation rate (0.01, 0.05, and 0.1). They also employed three differently complex true models: a simple CFA model with three latent and twelve manifest variables, a second model the same as the first but with one cross-loading, and the third with three cross-loadings. They randomly sampled both factor loadings and factor correlations within a certain range and produced 1000 samples per simulated dataset. This resulted in 27 conditions, and for each of these, 50 iterations were conducted.

Overall, the success rates, i.e. the proportion of correctly found models, ranged

from 0.36 to 0.96. Higher rates could be found for the simpler population models, especially if there is no or only one cross-loading, and when the population size is large. The latter is a known property of the GA and can be easily explained (G. Marcoulides & Drezner, 2001; Murohashi & Toyoda, 2007). A larger population means higher diversity and thus a higher exploratory power. This means that the search space is better investigated and solutions from different parts of the space are selected as possible candidates. This is especially true if elitism is employed, i.e. if the best members of each generation are always protected and can go unchanged into the next generation. Regarding the mutation rate, the results are inconclusive, as the conditions in which different rates are more successful than others didn't follow a clear pattern. This finding also matches expectations, which can be formed from previous studies (Srinivas & Patnaik, 1994), as it is only known that low rates, like the rates chosen by Murohashi & Toyoda (2007), are necessary for a successful GA. This is because higher rates would disrupt the proposed child solutions too much, which inherit high-quality parameter configurations from their parents. Lastly, they looked at the average time for the algorithms to produce a final model in each condition. Results ranged from just below 10 to around thirty minutes.

In addition to their simulation, Murohashi & Toyoda (2007) also investigated their algorithm on a small empirical example using the famous Holzinger and Swineford's (Holzinger & Swineford, 1939) dataset and repeating the analysis 20 times utilizing a GA with a population size of 10 and a mutation rate of 0.05. The best fitting pattern (with respect to the RMSEA) was found 25% of the time with the GA and, in comparison, also by Exploratory Factor Analysis (EFA) with Promax rotation, while other rotation methods produced worse models. The second-best pattern was found 20% of the time with the GA. Additionally, 13 unique models were found by the GA throughout the 20 repetitions. Even though for this smaller example and with a simple structure looking only at the RMSEA might be sufficient, in general, all fit indices are biased by factors like sample size or model complexity. This is why different measures should be taken into account, especially when dealing with very similar models in close proximity. For reasons explained later, information criteria like the BIC are considered the best choice for comparison in this case. However, it is safe to say that a GA can perform as well as an EFA given the right circumstances, and multiple runs of a GA can be conducted to aid with undesired model properties resulting from randomized exploration.

In a project examining project management knowledge and construction engineering performance, a genetic algorithm (GA) was employed to investigate the structural

aspect of an initial unsatisfactory model (Chou & Yang, 2013). The model consisted of nine latent variables, with eight assumed to predict the ninth. The authors found that the GA had limited success in improving model fit and only discovered a superior solution after manually re-specifying the measurement aspect of the model. This outcome may be explained by the restricted search space of the structural model, which hindered exploration of a significant portion of the modelling space, namely the measurement model. This reasoning is reinforced by the authors' success in discovering a far better fitting model when also modifying the measurement model. Although the authors demonstrated that a GA can improve model fit, the limitation of the search space prevents any prediction about the efficacy of the GA or other combinatorial optimization algorithms in specification search.

Another crucial simulation was carried out by Jing et al. (2022). They compared TS and a newly developed hybrid ant colony optimization (hACO) algorithm. The hACO algorithm combines the TS algorithm with search information obtained from an ant colony optimization algorithm during exploration, which is then used to construct the final model[3]. Like G. Marcoulides & Leite (2014), they specified two population models of varying complexity. Both models had two latent variables and ten manifest variables, which are relatively small by standards in psychological research (Goretzko et al., in press). The simpler model exhibited a simple structure and a low correlation (0.15) between the latent variables, while the complex model had three cross-loadings (0.6/0.6, 0.5/0.7, and 0.7/0.5) and a slightly higher correlation (0.25) between the latent variables. In addition to varying population models, they evaluated the algorithms under different conditions, such as sample size (150, 300, 500, 1000, and 1500), misspecification level of the starting model (60%, 40%, and 20% initially correctly specified), and two sets of fit indices for the objective function of the hACO. Each of these 30 conditions (not considering the varying objective functions for the hACO) was assessed with 100 iterations. While TS was used with Bayesian information criterion (BIC) as the objective function, the hACO utilized a logistic transformation of fit indices with a complexity penalty. Compared to G. Marcoulides & Leite (2014), this approach has the advantage of being able to scale a multitude of fit indices correctly and combine them into a consistent and standardized measure. Additionally, by penalizing model complexity, over-parametrization issues, such as those encountered with the ACO, can be reduced.

Regarding the simulation results, the two key findings were the success of the hACO over TS in all conditions and the overall high frequencies of correctly identified

[3]For a detailed description, please refer to section 2.2.5.

models for the hACO. In all 30 conditions and for both sets of fit indices, the proportion of times the hACO was able to find the population model was never below 90%, often even 100%. Very surprisingly, compared to findings by G. Marcoulides & Leite (2014), TS was not able to find the true model for every iteration in any condition. Regarding the conditions, TS was most influenced by sample size, where larger values led to substantially better performance. Especially interesting was that the complexity of the population model, as well as the misspecification level of the starting model, only slightly decreased the success of the hACO and TS. This may be due to the fact that the simple and complex population structures differed only by a few parameters, and no additional latent factors were specified. Also, the framing of the misspecification of the starting models suggests that the models are more wrong and further apart than they actually are. This is because the described correct specification rates are not the final ones, but the resulting rate usually will be higher as the rest of the parameters are sampled. For example, if the rest of the parameters are sampled so that the number of parameters will be the same as the population model (i.e. the probability for each remaining parameter that could be estimated is the number of estimated parameters in the rest of the population model divided by all remaining parameters in the population model), then the resulting correct specification rates for the initial 60%, 40%, and 20% will be approximately 80%, 70%, and 60%, respectively.

Additionally, Jing et al. (2022) looked at the convergence speed of the two algorithms. While the hACO took less than 30 seconds to converge in most conditions, TS took from 42 to 94 seconds. However, even though hACO is a lot faster, the overall range of seconds needed for the algorithms to produce a final model is so low that, in standard research conditions, they shouldn't affect the choice of algorithm.

The last study to investigate metaheuristics in specification search examined another population-style algorithm - the Bee Swarm Optimization (BSO; Schroeders et al., 2022). Focusing on bifactor models, they examined the performance of their BSO algorithm on two real datasets and compared it to EFA. However, as of now, only an incomplete version of the paper has been published, which is why the performance of the BSO cannot be sufficiently evaluated. When choosing the appropriate number of factors, EFA was found to produce a better model than the one proposed by BSO. This was reported for only one of the two datasets, and other results cannot be extracted with certainty, as graphs and tables are missing. Overall, BSO was able to find models that produce a fit around the common cutoffs proposed for the CFI and the RMSEA. Interestingly, their objective function included a minimal factor loading penalization, which penalizes models with a minimum factor loading below 0.3, in addition to a

fit and a penalization part, as used in Jing et al. (2022) study. Although limited in empirical value due to an incomplete report, implementations like these and the theoretical considerations mentioned in the study will become of greater significance when considering the choice, implementation, and evaluation of algorithms used in the present study.

The last study that should be mentioned is focused not on trying out a specific algorithm but rather on laying the foundation to implement them in the statistical programming language R (K. Marcoulides & Falk, 2018; R Core Team, 2016). They introduced a variety of functions that ease the respecification and fitting of models. They based their functions around what they call a *specification table*, which includes all the information necessary to fit SEM models. It also contains the binary indicator vector described before, which is altered according to the candidate model in the search. Although they only provide the basic instruments, such as not checking if a model is identified or suggesting which objective functions could be useful, the functions allow researchers to shorten their work and easily interact with SEM software.

1.3 Current Study

The primary aim of this work is not to address the limitations of previous research, but rather to build on the proof-of-concept studies, empirical studies, and simulation studies in the literature by comparing the most promising algorithms suggested for performing specification search. Seven algorithms (TS, SA, GA, hACO, ACO, BSO, and RaR) have been referenced for this purpose, with five tested on real or simulated data. Based on our considerations, we chose to compare six algorithms: TS, SA, GA, an adaptive GA, a Particle-Swarm-Optimization (PSO) algorithm not yet suggested, and hACO.

TS was included as a benchmark method because of its simplicity. It is a basic hill climbing algorithm with only the addition of a tabu list, making it non-stochastic. As demonstrated by G. Marcoulides & Leite (2014) and Jing et al. (2022), TS can deliver good results and perform better than ACO, which relies heavily on stochasticity. In addition, TS has been implemented as an example by K. Marcoulides & Falk (2018) in their paper on helper functions in R, making it an easy addition to our comparison.

We also chose to include SA in our comparison because it is a very popular method used not only as a standalone algorithm but also as a principle or in combination with other algorithms. Furthermore, it is one of the four algorithms originally suggested by Drezner & Marcoulides (1999). Since SA has not yet been studied in specification

search, there are no expectations on its performance.

GA was an obvious choice for our comparison since it has shown recent success in various areas, including image processing and forecasting (Katoch et al., 2021). It has already been tested in two specification search studies and was suggested and described in detail by G. Marcoulides & Drezner (2001). Moreover, it is a core representative of the population-based metaheuristic algorithms class. Due to its flexibility in implementation, we included two versions of GA in our comparison. The first is a basic GA that closely follows the description of G. Marcoulides & Drezner (2001). The second is a more modern version that focuses on adapting its core hyperparameters based on search progress (Srinivas & Patnaik, 1994) to reduce bias to user specification. We expect the GA (adaptive or not) to perform well in our comparison, especially given its recent success and the findings of Murohashi & Toyoda (2007).

The PSO algorithm is another population-based metaheuristic. Although it has not yet been suggested or implemented to conduct specification search, it was included in the study based on two reasons. The first reason relates to the problems and benefits of ACO, while the second reason is due to the success of GAs. ACO is theoretically appealing because it can search in very different corners of the search space very easily by means of sampling models from a distribution. However, it also tries to estimate many models that are of very bad quality or cannot be estimated properly due to statistical reasons, leading to a misled or uninformed search. Like ACO, PSO also samples its solutions, but this is not done as rigidly. Additionally, in PSO, there are many models in each population, some of which might not change at all. This leads to a more stable search and also shows the relation to GA, which, although based on different operators to update its models, also samples part of its solutions while having a wide variety of population members, some of which might not change at all in an iteration. Especially the best solutions in both algorithms are, depending on the implementation, more or less protected against exploration. Based on this reasoning, we also expect PSO to perform well in specification search.

The last algorithm to include was hACO. Unlike the other algorithms, it is designed specifically for specification search and based on research knowledge in the field. It also showed to perform better than TS with the simplistic population models tested by Jing et al. (2022). This is the only point of reference for hACO's quality so far. However, at its core, it is a TS algorithm that misses the exploratory power of ACO, which the name might suggest. Therefore, we expect hACO to perform similarly to TS under more complex models.

We decided not to include the ACO based on the mentioned and known problems

of the algorithm. The BSO was also not included, as we could not fully investigate the success of the BSO in specification search based on the incomplete publication and deemed our suggested PSO a similar alternative that fit well into the previous research. Lastly, we did not add RaR to the comparison. This is less because we don't think it will perform well, but more due to the fact that we already included six algorithms which we saw as the most logical options based on previous research. Even though RaR has shown success outside of specification search, there is still very little research compared to the GA or SA.

Regarding our simulation design, we oriented ourselves in parts to Jing et al. (2022), however with a few very important changes. This was done because they focused on a variety of different conditions, which we thought to be of high interest, especially for applied researchers. Also, they themselves followed a tradition of research like very important studies by Hu & Bentler (1999) and G. Marcoulides & Leite (2014). Additionally, we decided to focus on models in the realm of CFAs. Even though many other possible subtypes of SEM could be of interest, we deemed CFA the most common specific case, as well as the most interesting starting point when investigating metaheuristics in specification search. This is because SEM was originally developed based on CFA and path analysis and is nowadays still focused in huge parts on latent variables, especially in psychometric research.

Our general goal was to model the research process in SEM as accurately as it can be done in a simulation study. This is why we kept the general conditions Jing et al. (2022) investigated, which were the complexity of the true population model, the misspecification level of the initial model, and sample size. All of these are highly relevant when it comes to applied research, as they are frequently encountered in vastly varying manifestations. Based on our previous research (Goretzko et al., in press), in which we analysed how CFA is conducted in applied research, we can make informed decisions about the exact levels of these conditions. One of our main goals was to use this knowledge to specify quantities like sample sizes or number of latent factors, as they are most often observed in psychological research. For this reason, we followed means and medians of such quantities present in the literature.

With regards to the first condition - complexity of the population model - we followed most of the previous research (Hu & Bentler, 1999; G. Marcoulides & Leite, 2014; Murohashi & Toyoda, 2007) with our simpler model. This was done because Goretzko et al. (in press) have shown that these models closely resemble the average models fitted by researchers. However, we also included a second, more complex model in the comparison. We aimed to investigate the more complex models typically

specified by researchers, which usually have three or more factors and more manifest variables than what can be found in many studies. With this approach, which is not only based on beliefs about what researchers are doing but is empirically analyzed, we were able to model standard and complex research processes more realistically.

The level of misspecification of the starting model was our second condition. Originally, we planned to investigate three levels, like Jing et al. (2022), but due to the fact that we already had six algorithms to compare, which require a lot of computational power and time, we decided to exclude one of the three levels. We called our levels *high* and *low* misspecification, even though with the *low* level we tried to resemble medium to low misspecification. As Goretzko et al. (in press) found, models that lack quality in any fit criteria are frequently (e.g. 64.5% for the Comparative Fit Index) reported. We also argue that the use of metaheuristics is most important if the specified model is at least a certain number of steps away from the correct solution and not, e.g., only three steps away. This is because most of the algorithms compared here are complex and are not designed to improve solutions by just a few steps. On the other hand, the algorithms should obviously be able to not disrupt and protect decent models. Therefore, we chose a level that we believe represents low to medium misspecification, which would frequently occur in research. Additionally, Jing et al. (2022) did not precisely report how they specified the rest of the starting model's parameters. We chose a stochastic approach, which gives a higher probability to underparametrized models. With this, we aimed to model a realistic modeling scenario in which researchers have prior knowledge about some part of the modeling space (e.g., from previous research or a solid theory) and make an educated guess about the rest of the space while attempting to specify a parsimonious model.

Lastly, we examined sample size, a key quantity in all statistical analyses. We decided to investigate four different levels, the first of which we believe to be the minimal size required for the complexity of models used in our study. Next, we included an average level, followed by a large and a very large size. Especially the large size matches several of the studies mentioned, such as G. Marcoulides & Leite (2014) and Murohashi & Toyoda (2007). The average level was also included in Jing et al. (2022). The last very large size was included to rule out the possibility that our findings solely depend on an overly large sample size. Therefore, it was of less practical and more theoretical interest, in order to identify sample size as a potentially confounding factor.

2 Method

2.1 Introduction to SEM

The goal of SEM in most applications is to represent the distribution of observed variables using latent variables in a confirmatory or supervised fashion, using regression equations (Kuechenhoff, 2021/2022). Unlike EFA or CFA, SEM is a framework that allows for building more complex models, such as those with different level latent factors or additional regression relations between variables.

To better understand how SEM works, let us return to the example from Chapter 1.1, depicted in Fig. 1. We looked at a three-factor model with five indicators each. As it stands, the model is a CFA model, which is the most common special case of SEM. Speaking in regression or machine learning terms, the indicators $X_1, ..., X_{15}$ are the dependent variables or outcomes, while the latent factors F_1, F_2, and F_3 in our case are predictors or features. A terminological differentiation that is helpful for understanding SEM is the distinction between the *measurement* and *structural* model. The relations between manifest and manifest/latent variables are specified in the measurement model, while the relations between latent variables are captured in the structural model.

The depiction in Fig. 1 is an example of one common way to represent the structural system of (linear) equations, which gives SEM its name. In formula, the model is written as follows:

Structural model:

$$Empty$$

Measurement model:

$$\begin{aligned} X_1 &= \beta_{11} F_1 + \delta_1 \\ X_2 &= \beta_{12} F_1 + \delta_2 \\ X_3 &= \beta_{13} F_1 + \delta_3 \\ X_4 &= \beta_{14} F_1 + \delta_4 \\ X_5 &= \beta_{15} F_1 + \delta_5 \\ &... \\ X_{15} &= \beta_{35} F_3 + \delta_{15} \end{aligned}$$

$\delta_1, ..., \delta_{15}$ represent the *residuals* or *errors*, and the β coefficients represent the path coefficients, also known as *factor loadings*. As we can see, SEM is nothing more than a system of linear equations. Like in regression, we also have additional assumptions about our modelling components. For instance, we assume that the expected value of the residuals is zero. Additional assumptions include, but are not limited to, zero correlations between latent variables and residuals, and that the expected value of latent variables is also zero. Note that in our structural model, we did not include the correlation between F_1 and F_2. We will explain how it enters the model in the following paragraph.

As shown in the equations above, we have 15 measured variables, $X_1, ..., X_{15}$, and in principle, 33 quantities that we want to *estimate*, i.e., all the quantities on the right-hand side. Obviously, these are too many parameters, and we would speak of a non-identified model. However, since we are mainly interested in the β coefficients and do not need concrete values for our latent variables and residuals, the estimation is done with respect to the covariance matrix of our manifest variables. With our fifteen manifest indicators, the corresponding covariance matrix has 120 unique entries. For estimation, we would thus have 120 *variables*, and thus an over-identified model, which is what we are looking for. To find the best values for our β coefficients, many algorithms and estimators exist. To better understand how this estimation works, let us look at the model from a variance/covariance perspective. With basic statistical formulae and the additional assumptions mentioned, especially the factor correlation between F_1 and F_2, we obtain the following equations:

$$
\begin{aligned}
Var(X_1) &= \beta_{11}^2 Var(F_1) + Var(\delta_1) \\
Var(X_2) &= \beta_{12}^2 Var(F_1) + Var(\delta_2) \\
&... \\
Var(X_6) &= \beta_{21}^2 Var(F_2) + Var(\delta_6) \\
&... \\
Cov(X_1, X_2) &= \beta_{11}\beta_{12} Var(F_1) \\
&... \\
Cov(X_1, X_6) &= \beta_{11}\beta_{21} Cov(F_1, F_2) \\
&... \\
Cov(X_6, X_{11}) &= 0
\end{aligned}
$$

...

$$Cov(X_{14}, X_{15}) = \beta_{34}\beta_{35}Var(F_3)$$

As we can see, assuming no correlation between latent variables, like for F_2 and F_3, leads to the fact, the we expect a covariance of 0 between all indicators loading those factors (X_6 and X_{11} in the above equations). However, assuming a correlation, like for F_1 and F_2, we get another quantity we need to estimate.

Writing this system of equations in matrix form, we get the central two quantities in SEM: the empirical and model-implied covariance matrices, where the letter is compromised of our previous equations. Let's denote this matrix $\Sigma(\theta)$, where θ is a vector of all quantities on the right side we want to estimate, and rewrite the above equations it in the following way

$$\Sigma(\theta) = \Lambda_x \Phi \Lambda_x^\top + E_\delta,$$

where Λ_x (speaking in factor analysis terms) represents our matrix of factor loadings, Φ represents our latent variable variance-covariance matrix, and E_δ is the matrix containing our residuals. With this decomposition, and if we additionally assume no structure between our residuals and standardise variances to 1, i.e. E_δ is a diagonal matrix whose entries are set such that the resulting variances on the diagonal of $\Sigma(\theta)$ are 1, we only need to specify our factor loadings Λ_x and latent variable correlations Φ when simulating our data. This is because we are interested in simulating from a multivariate Gaussian using only $\Sigma(\theta)$ and don't care about the mean structure of our simulated data.

Returning to standard research practice with SEM, the goal when estimating the parameters is to bring the empirical and model-implied variance-covariance matrices close together via a so-called *discrepancy function*. This discrepancy is primarily a theoretical construct that depends on the estimation procedure we wish to use. Common choices are Ordinary Least Squares and Maximum Likelihood (ML) based discrepancy functions. In our case, this choice is straightforward as we know our (simulated) data comes from a multivariate Gaussian distribution and we can directly opt for ML estimation.

The empirical pendant of our discrepancy function, i.e. its value in the case of an observed covariance matrix and a specified model, is a quantity that measures the quality, i.e. the "badness", of our model. The larger the discrepancy, the further away our model-implied and empirical covariance matrices are, and thus the less suited our

model is to represent the covariance structure in the empirical data. Even though the discrepancy function is not the only option to base the quality assessment of our model on, many measures called *fit indices* base their evaluation around it (or, to be more specific, around the χ^2 test-statistic value, which is almost entirely made up of our discrepancy function value). Other options, e.g. rely on low errors as our model should be good at predicting and thus produce low residual values, or the log-likelihood of our model. In the latter case, we speak of *information criteria*.

Table 1
Overview over fit measures.

Measure	Formula	Good-/Badness	Theoretical Range	Cutoff	Sensitive to N	Complexity Penalty
CFI	$\frac{max(\chi_0^2-df_0,0)-max(\chi_k^2-df_k,0)}{max(\chi_0^2-df_0,0)}$	Goodness	0-1	0.95	No	Yes
RMSEA	$\sqrt{\frac{max(\chi^2-df,0)}{df(N-1)}}$	Badness	>0	0.06	Yes	Yes
SRMR	$\sqrt{k^{*-1}(t(\delta)W_s\delta)}$	Badness	>0	0.08	Yes	No
BIC	$-2l(\theta)+p*log(n)$	Badness	$\mathbb{R}$	-	Yes	Yes

Notes: CFI = Comparative Fit Index; RMSEA = Root-Mean-Squared-Error-of-Approximation; SRMR = Standardized Root-Mean-Square-Residual; k^{*-1} = number of non-redundant elements in covariance matrix -1; δ = residuals; W_s = weight matrix for standardization; p = number of estimated model parameters; $l(\theta)$ = log likelihood of the model.

For our study, we will examine three different fit indices and one information criterion, the properties of which are summarised in Table 1 (Hu & Bentler, 1999; West et al., 2012). The Comparative Fit Index (CFI) and Root-Mean-Squared-Error-of-Approximation (RMSEA) are both based on the χ^2 value of our model. This test statistic mainly involves the discrepancy function and is used to test the hypothesis of equality (null) versus inequality (alternative) of the observed and model-implied covariance matrix. A smaller χ^2 value indicates a better fitting model. Additionally, the degrees of freedom serve as a complexity penalty. The CFI not only looks at these two quantities for the actual model, but also compares it to the so-called *null-model*, which assumes complete independence between observed indicators. This results in a diagonal covariance matrix that serves as a baseline for comparison with the actual model-implied covariance matrix. Therefore, the CFI belongs to the class of *comparative* or relative fit indices.

In contrast, other indices like the RMSEA and the Standardized Root-Mean-Square-Residual (SRMR) are also called *absolute* indices as they evaluate the quality of the model without reference to another. The RMSEA additionally takes into account the sample size, making it sensitive to especially small N (West et al., 2012).

The SRMR is built completely differently than the other indices. Its key assumption is that a good model should lead to small residuals, denoted here as δ, because the explaining variables should be able to contain a lot of information of the explained variables. These residuals are standardised with a weight matrix W_s and via k^{*-1}, which accounts for the complexity of the observed data.

The BIC is rooted in a different research tradition compared to other methods, and it can be considered a more theoretically sound measure. It is derived from Bayesian theory, where the aim is to find the model with the highest posterior probability - the most likely true model from a set of models, given that the data have already been observed. However, the posterior probability is not feasible as is, and so Laplace and Taylor approximations are used. When we collect all terms that depend on n (i.e. those that actually change when the data are observed) and multiply this quantity by -2, we obtain the BIC. Therefore, maximising the posterior probability - finding the best model for given data - approximately minimises the BIC. This property makes the BIC advantageous in that it can be used to compare any models on the same dataset, unlike fit indices, which are more sensitive to certain kinds of models (e.g. the SRMR to models with low residuals, which may lack parsimony). Nevertheless, as its derivation shows, the BIC has no absolute meaning, unlike the actual posterior probability. Additionally, theoretically, fit indices can only be used to compare nested models. Nested models have one set of parameters that is a subset of the other's. It is impossible to compare two non-nested models with fit indices because most of them, such as the CFI and RMSEA, are based on the difference between model-implied and empirical covariance matrix. These two will generally be closer when we have more parameters, even if a complexity penalty is added. For non-nested models (and models with different numbers of parameters), we cannot differentiate whether the better fit is simply due to the additional parameters or whether the estimated parameters in one model allow for a better approximation of the empirical covariance matrix. However, this is not the case for the BIC because the log-likelihood is not monotonous in the number of parameters.

As previously mentioned, all these fit measures are descriptive fit measures and do not allow for hypothesis testing directly. Thus, simulation studies were conducted to identify the values at which the indices can identify certain degrees of misspecification at typical α-levels (Hu & Bentler, 1999). These values are known as *cutoffs*. Table 1 shows the direction of the indices (goodness or badness) and the theoretical range of the indices (usually between 0 and 1, except for the BIC). For fit indices, the complexity penalty, even if it is present, is usually not strong enough to select parsimonious models.

This point will become relevant later when we introduce our objective function.

2.2 Introduction to Algorithms

Metaheuristics are design principles that can be used to implement algorithms that solve COPs. These metaheuristics, or their derived algorithms, are designed to efficiently explore a set of potential solutions to a problem. This is necessary because analysing all possible solutions can be computationally expensive or even infeasible for many problems. Metaheuristics often use a combination of deterministic and stochastic rules to select and evaluate candidate solutions, but the specific algorithm chosen for a given problem will depend on the characteristics of the problem itself. It may be necessary to use trial and error to determine the most effective metaheuristic for a particular COP because, in general, it is not possible for any one algorithm to solve all COPs (a concept known as the no-free-lunch theorem). This is why it is necessary to compare the performance of multiple algorithms on realistic, complex datasets to choose the most suitable one for a given problem, which is the aim of our study.

2.2.1 Tabu Search

Figure 2
Graphical representation of Tabu Search.

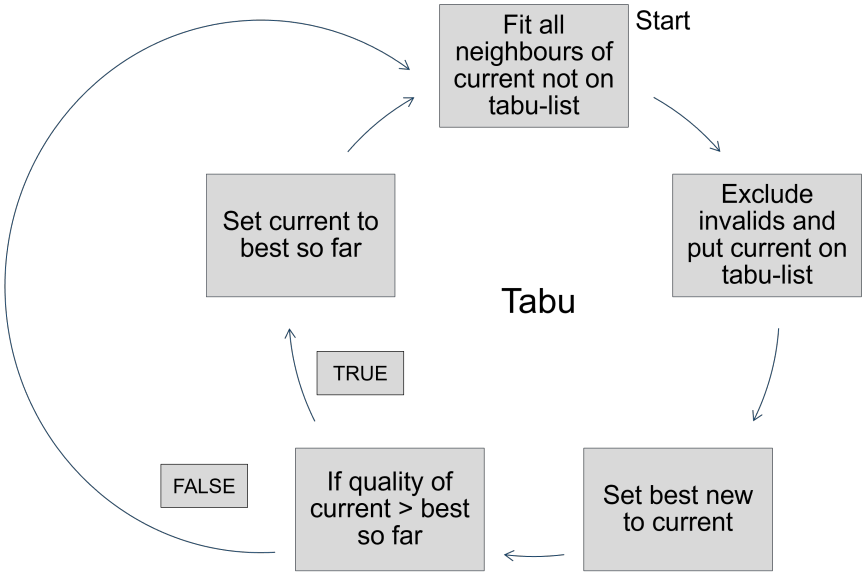


Fig.Fig. 2 illustrates the basic principle of TS (Glover, 1986; G. Marcoulides et al., 1998). TS is a simple greedy algorithm with the additional feature of a tabu-list. Starting from an initial model represented by a binary indicator vector, we evaluate

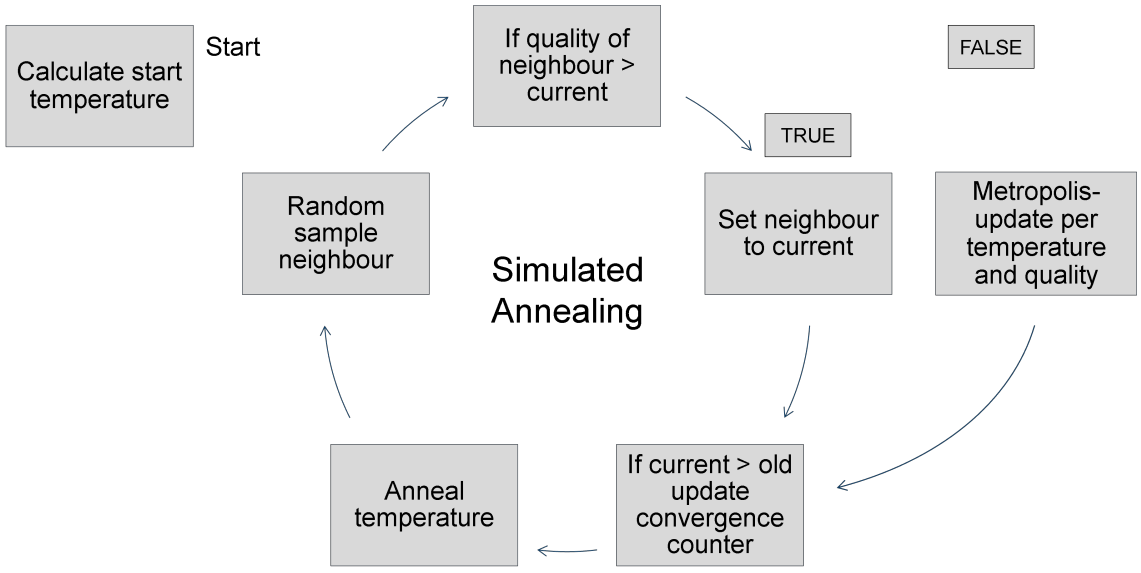
all its neighbours. In the first iteration, there are no models on the tabu-list, so all neighbours are considered. In subsequent iterations, the previously selected current models are added to the tabu-list to avoid cycling through the same models and possibly escaping local solutions by advancing the search *from a different angle*.

Of the neighbours, only those that are valid are considered potential improvements, i.e. those that are identified or do not produce estimation errors common in Structural Equation Modelling (SEM). Among the valid neighbours, the best model is selected and set to the new current model, even if it is worse. This feature allows the algorithm to explore different areas of the search space, while not easily getting stuck in local solutions. The best so far model is also saved. Only if the current model is better than the best so far model, will the latter be updated with the former. This way, the best so far model is protected against the search exploration.

With this technique, it is guaranteed that TS will always perform at least as well as a trivial hill-climbing algorithm. Our TS algorithm stops if the best so far model does not change for k iterations.

2.2.2 Simulated Annealing

Figure 3
Graphical representation of Simulated Annealing.



SA (Fig. 3) is an optimization algorithm that uses a neighbour-to-neighbour random search to find the global minimum or maximum of a function (Drezner & Marcoulides, 1999; Kirkpatrick et al., 1983). It is inspired by the annealing process used in metallurgy to strengthen materials by slowly cooling them. The algorithm starts at

a high temperature and gradually decreases it as the search progresses. The temperature determines the probability of accepting worse solutions, allowing the algorithm to escape local minima and maximise the chance of finding the global minimum or maximum. Near the end of the search, the temperature should have cooled enough such that only better solutions are considered and the global solution is not skipped.

The initial temperature is set to a prefixed value, which can be determined manually or based on the properties of the starting model. It should be high enough to amplify exploration in the beginning of the search, where bad local solutions are more likely to occur. After setting the starting temperature, the cycling algorithm begins. Candidate neighbours are chosen randomly. If the neighbour is better than the current model, the neighbour is accepted and the current model is updated. If the neighbour is worse, a Metropolis-step is performed, in which the worse model is accepted with a certain probability that depends both on the quality of the candidate compared to the current solution and the temperature.

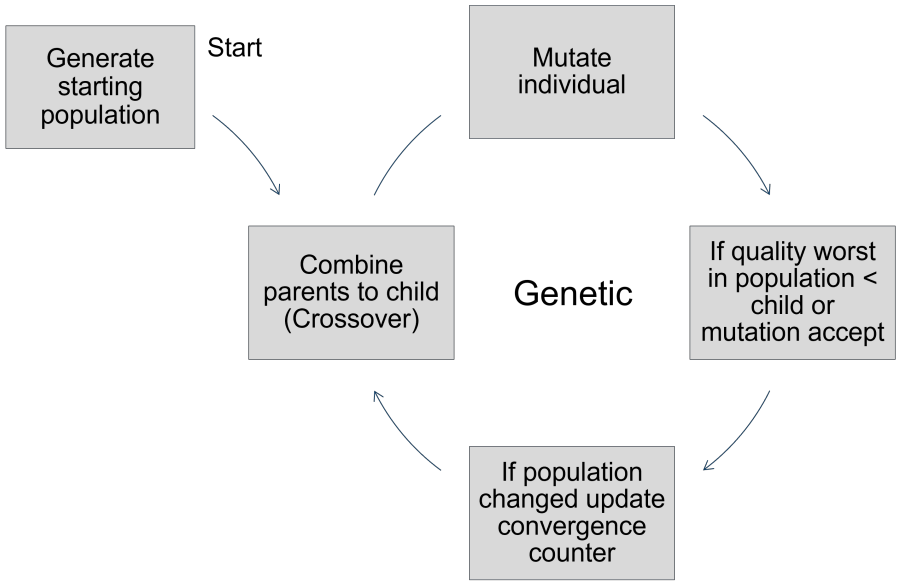
After some iterations, the temperature cools down, i.e. *anneals*, which lowers the acceptance probability of the Metropolis-step. The annealing can be performed in a variety of fashions and has a significant influence on the success of the algorithm. For example, linear or constant cooling rates, but also more complex functions that may depend on the (estimated) search progress, can be applied. At the end of the search, only better models are accepted, and the algorithm converges to a random search hill-climbing algorithm. SA stops if no updates occurred for a certain number of iterations.

2.2.3 Genetic Algorithms

Genetic algorithms (Holland, 1975; G. Marcoulides & Drezner, 2001) are heuristic optimization methods inspired by the process of natural evolution. They use techniques based on genetics and evolution, such as reproduction, mutation, crossover, and selection, to find the optimal solution to a problem. The algorithm represents solutions to a problem as a population of individuals, each with a set of genetic material encoded as a chromosome. Each chromosome contains many genes, which represent binary indicator vectors and their entries, respectively. The algorithm then applies evolutionary operations to the population to generate new generations and improve the overall quality of the solutions over time. The goal is to find the fittest individual in the population, which represents the optimal solution to the problem.

To begin the evolutionary process, a genetically diverse population needs to be generated in order to explore a large part of the search space. After initialization,

Figure 4
Graphical representation of Genetic Algorithm.



individuals are selected from the population to produce offspring through a process called *reproduction* or crossover, in which two models (the parents) are combined into a child based on a certain rule. For example, a cross-point can be chosen such that each half of the resulting model (the child) is taken from one of the two parent models. In the next core genetic operator, mutation, only one model is picked as a candidate, which can be, and usually is, a child. The mutant is constructed based on the candidate by altering one (or more) randomly chosen parameters of the candidate. This is equivalent to looking at a random neighbour of the candidate, as performed in SA. After these processes, the fittest individuals are selected to survive and live into the next generation. However, it is also possible to base the new population only on the children and/or mutants and let only a few (or also none) of the current generation survive. Letting the few fittest solutions survive is called *elitism*.

In the basic GA used in this study (Fig. 4), children and mutants are two completely different proposed solutions that are only accepted into the new generation if they are better than the worst model in the current population (G. Marcoulides & Drezner, 2001). If this is the case, they will replace the worst individual. Candidates for reproduction and mutation are chosen completely randomly. Therefore, in this version of the GA, there is no real selection process. The reproduction process also calls for a split point to be specified, which determines how parents should be combined. For this version of the GA, we decided to always split parents in half. If the population does not change for a specified number of generations/iterations, the

algorithm terminates. Afterwards, the fittest model from the population is selected as the suggested solution.

Because this former GA is very different from how GAs are usually implemented nowadays, we also added an adaptive and more advanced version of the algorithm. In this version, parents are selected by *Roulette Wheel Selection*, which means that better individuals have a higher probability of being selected and have a larger field on the Roulette wheel. For reproduction, parents are combined based on a random split point. This makes it possible for a child to consist almost exclusively of one of its parents. However, reproduction is only performed with a certain probability, which is one of the two core quantities in modern GA algorithms. The other is the mutation probability, which is applied gene-wise. In our adaptive GA, both probabilities are adapted throughout the search and should reflect the progress in convergence. For this reason, we chose to calculate the cross-over probability p_c and the mutation probability p_m according to the following equations (Srinivas & Patnaik, 1994):

$$p_c = k_1 \frac{f_{max} - f'}{f_{max} - \bar{f}}, \quad f' \leq \bar{f}$$
$$p_c = k_3, \quad f' < \bar{f}$$

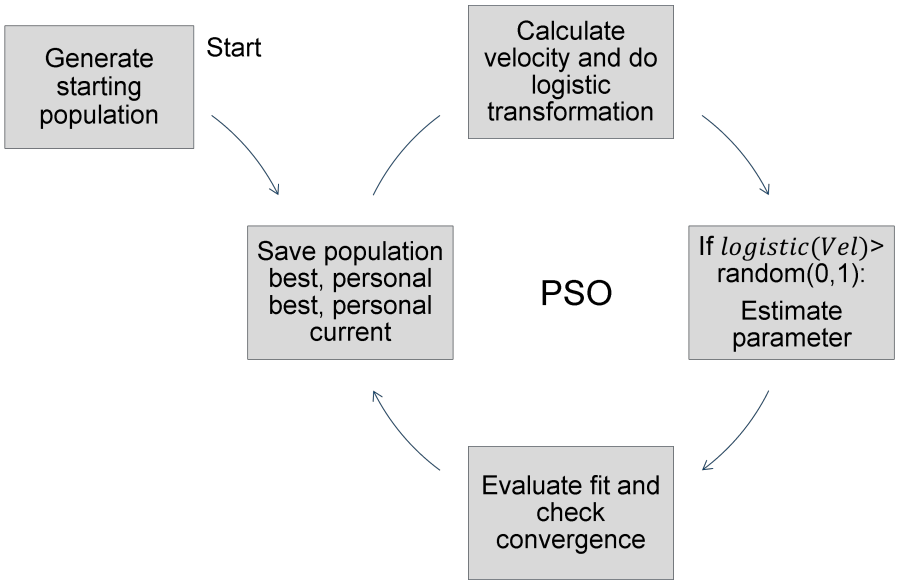
$$p_m = k_2 \frac{f_{max} - f}{f_{max} - \bar{f}}, \quad f' \leq \bar{f}$$
$$p_m = k_4, \quad f < \bar{f},$$

where f_{max} represents the maximum fitness value in the population, f' denotes the larger fitness value of the two selected parents, $\bar{f}$ represents the average fitness in the population, and f indicates the fitness of the child that should undergo mutation. k_1 and k_2 control the overall magnitude of the probabilities, while k_3 and k_4 determine the case when subaverage solutions are selected. The idea is to increase both p_c and p_m as the algorithm gets closer to local optima, which is measured by $f_{max} - \bar{f}$. Initially, we start with very different solutions, which also have varying degrees of fitness. As the algorithm searches through the solution space, it selects more and more similar and higher quality solutions that become closest to the (local) optimal solutions. Thus, as $f_{max} - \bar{f}$ reduces, indicating that our solutions assimilate to the best solution near an optimum, our probability increases. However, this approach may disrupt our global optimal solution. To overcome this problem, we decrease the probability of cross-over and mutation based on the fitness of our potential parent or mutant, where better solutions are protected against disruptive resampling. This is measured by how close

they are to f_{max}. In this way, we aim to overcome local optimal solutions without ignoring or missing high quality, potentially global optimal solutions.

2.2.4 Particle Swarm Optimization

Figure 5
Graphical representation of Particle Swarm Optimization.



PSO represents another population-based optimization algorithm (Kennedy & Eberhart, 1997). In contrast to previous metaheuristics, it more closely resembles a specific algorithm than an abstract principle. As the name suggests, PSO tries to mimic the behaviour of animals (particles) that adapt their search for some objective, such as a food source, based on the behaviour of other members of their swarm. The general idea lies in the mathematical description of the swarm's behaviour in terms of velocity and direction. In addition to random individual movements used for sufficient exploration, a particle tends towards the direction of its personal best and the swarm's best position, with the speed also depending on the quality of the particle's current position.

After the generation of an initial population of particles, the overall best position, $best_{pop}$, the personal best position, $best_{ind}$, and the personal current position, $current_{ind}$, are saved (for the first iteration, the last two are the same; see Fig. 4). These represent certain manifestations of our binary indicator vector. After that, we calculate the central quantity of the algorithm - the velocity, Vel_i, at iteration i -

according to the following rule:

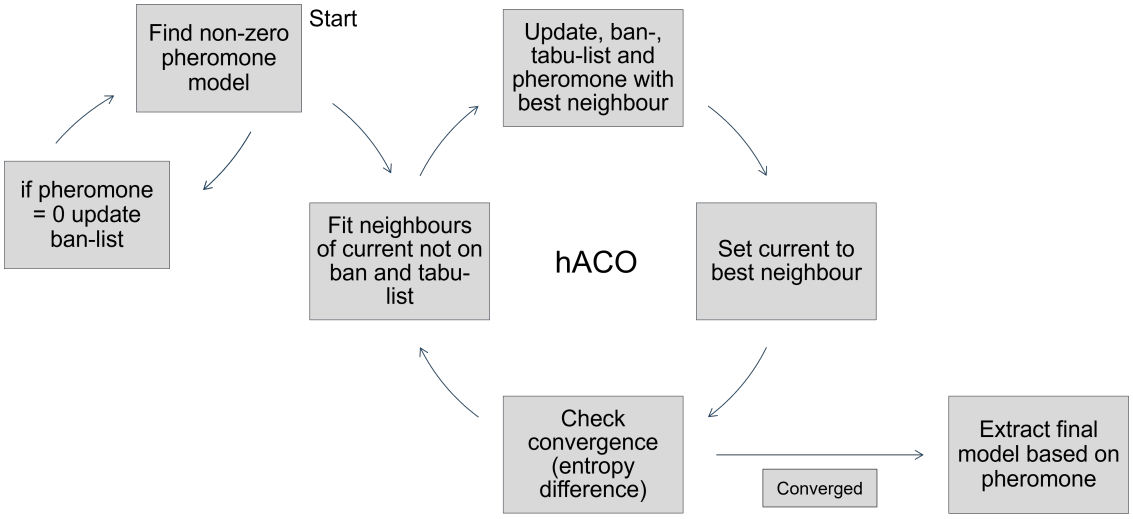
$$Vel_{i+1} = Vel_i + \phi_1(best_{ind} - current_{ind}) + \phi_2(best_{pop} - current_{ind})$$

where ϕ_1 and ϕ_2 are uniform random values in the range $[0, 1]$. The velocity is defined for each potentially to be estimated parameter. After a logistic transformation, Vel is compared to another uniform random value in the range $[0, 1]$. If the logistically transformed velocity is greater, the corresponding parameter will be estimated in the next iteration of the solution/particle.

From the formula, we can easily derive the behaviour of the algorithm. If a parameter is estimated in the $best_{pop}$ or $best_{ind}$ solution, i.e. its value is 1, and not estimated in $current_{ind}$, i.e. its value is 0, the velocity increases by ϕ_1 or ϕ_2 respectively. This increase is kept after logistic transformation, and thus the probability that the random value drawn is smaller than the logistically transformed velocity is also increased. The opposite is true if a parameter is not estimated in the best solutions but is estimated in the current. By this mechanism, the particles will orient towards the best solutions, as the probability that a parameter will be estimated depends on whether it is estimated in the personal or swarm's best position. After we evaluate the quality of the next *generation*, the cycle repeats.

2.2.5 Hybrid Ant-Colony-Optimization

Figure 6
Graphical representation of hybrid Ant-Colony-Optimization.



The hACO (see Fig. 6) represents an attempt to combine the qualities of ACO and

TS (Jing et al., 2022). The idea is to achieve both the speed and simplicity of TS with the amplified exploration of ACO, which suffers from convergence problems. While the core of the algorithm used for exploration is TS, the ACO is used for the generation of the final model. This is done by saving the central quantity of the ACO—the pheromone—along the search of TS. This pheromone also allows for the calculation of a probability distribution, and thus the entropy of iteration over the search space, ultimately giving a probability for each potentially estimated parameter, indicating how probable it is that the parameter is estimated in a high-quality solution.

Looking at the top left of Fig. 6, we can see that the hACO starts with an independent loop, the goal of which is to find a model that produces a non-zero pheromone value/objective function (see Eq. 3). This is only necessary if no non-zero starting model is given. To achieve this, candidate models are completely randomly sampled. Each model that produces zero pheromone is put onto a ban list. Before any model in the following is fit, it is first checked if the model is already on that ban list. When a non-zero pheromone model is found, the actual algorithm starts. It works exactly like the Tabu algorithm; however, instead of the best so-far model, the pheromone value is saved in each iteration. Convergence is checked via the entropy difference between iterations.

When the TS part converges, the final model is extracted from the accumulated pheromone. This is done by using the highest pheromone value in the accumulated pheromone table as a reference and including only parameters above a certain factor of this maximum value. By default, 18 different factors of the maximum pheromone value are tried as a threshold ranging from 0.8 to 0.98. Out of these 18 candidates, the model resulting in the highest pheromone value/objective function is considered the final model. For more details on the algorithm, refer to Jing et al. (2022).

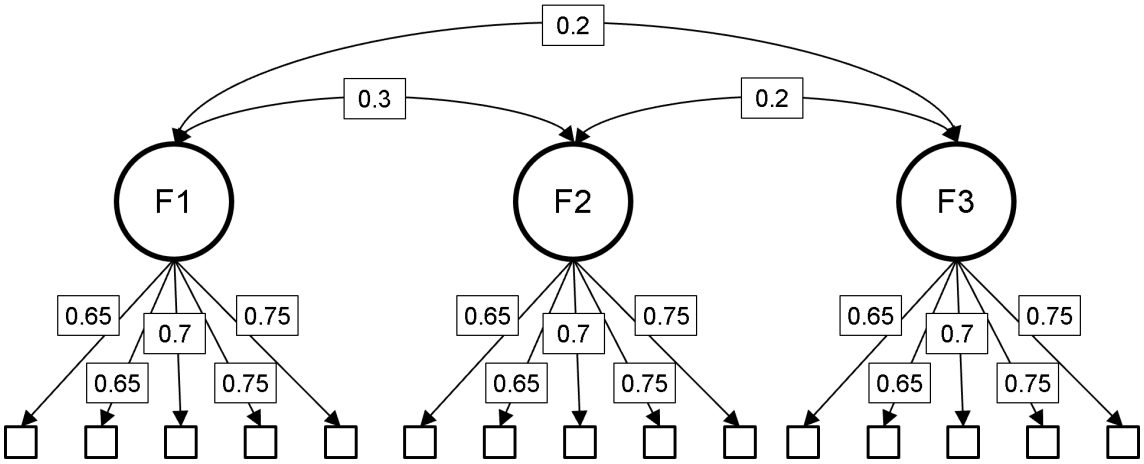
2.3 Simulation Study

As discussed in the introduction, we will evaluate the effectiveness of the algorithms in specification search using simulated data. We achieve this by specifying population models (e.g. Fig. 8), which form the basis of our quasi-random data. In this case, we limit ourselves to CFA models, as they are the most common special case of SEM and provide the foundation for more complex models such as bifactor or hierarchical structures. Before applying this type of specification search to more complex models, the success should first be demonstrated in the simpler here present case.

2.3.1 Population Models

Figure 7

Path-model representation of average population model.

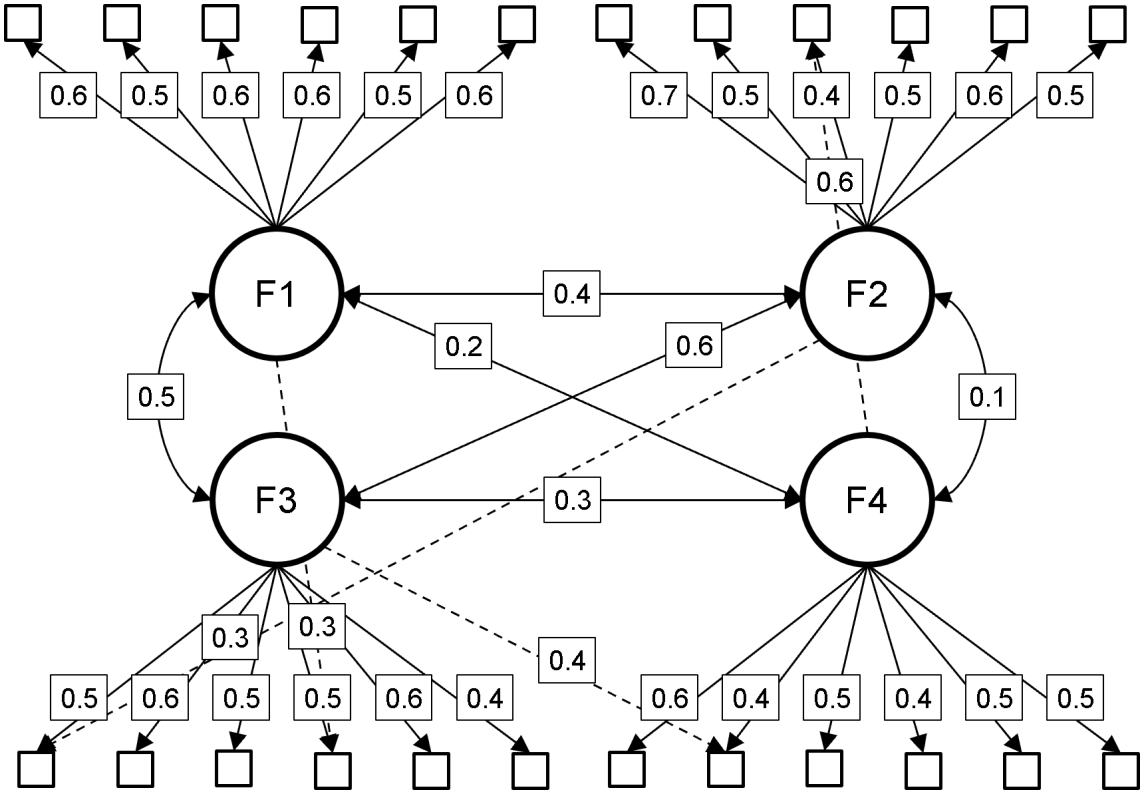


Our first population model aims to represent what an *average* CFA model used in the literature might look like (Fig. 7). This is done by referring to Goretzko et al. (in press), who analysed CFA-based research in over 200 studies containing more than 1000 fitted models. As previously discussed, to represent the search space, it is sufficient to focus only on latent variable correlations and factor loadings to define the population models and thus our simulated data. Our average model contains 15 manifest variables, which closely matches the median (16) and mean (17.65) of the models summarised by Goretzko et al. (in press). While they found a median and mean number of latent variables of 2 and 2.5 respectively, we opted for a three-factor model. This choice better matches the 15 manifest variables chosen, is the same as Hu & Bentler (1999), and expands on Jing et al. (2022), who used 2 factors. Also, the median and average number of indicators per factor were 5 and 7 respectively, which would result in more than two factors. The magnitude of our factor loadings were also decided based on Goretzko et al. (in press), Jing et al. (2022), and Hu & Bentler (1999), where the former found 25%, 50%, and 75% quantiles of median factor loadings of 0.63, 0.7, and 0.77 respectively. Focusing only on the median obviously shrinks the range significantly, but that was the most fine-grained information we could take from Goretzko et al. (in press) and helped to keep the model simpler in contrast to our second population model. The last quantity that needed to be specified was the magnitude of our latent variable correlations. As many models in (Goretzko et al., in press) did not include them in favour of an orthogonal structure, we deemed the reported values of a median and mean of median latent variable correlations of 0.59 inflated from our perspective.

This is because studies usually only adopt an oblique structure if the correlations are high enough. Additionally, very low correlations, like the 0.15 employed by Jing et al. (2022) in their simple population model, can be hard to distinguish from random noise. Based on these considerations and not to mislead our algorithms, we decided on medium values (0.3, 0.2, and 0.2) for our average model. Lastly, we did not allow for any cross-loadings because 95% of the models reported by Goretzko et al. (in press) did not include them. However, this might be due to the fact that a CFA model was originally only an orthogonal model with a simple structure, i.e. did not contain any latent variable correlations and cross-loadings. With our second population structure,

Figure 8

Path-model representation of complex population model.



we wanted to evaluate a more *complex* representative of CFA models. As previous studies only analysed population structures similar to or less complex than our average population model, we relied more on Goretzko et al. (in press) and our own experience to define the model. Additionally, the more complex a model becomes, the more likely it is that rotations of the same yield an equivalent or almost equivalent model. This is why we not only relied on manual specification of the model but also sampled it according to restrictions while optimizing a criterion. These restrictions were: 4 latent

factors, 6 manifest variables per factor, 1 cross-loading per factor between 0.3 and 0.6, factor loadings between 0.4 and 0.8, and latent variable correlations between 0.1 and 0.6. These quantities were again referenced to quantiles found by Goretzko et al. (in press), but not as closely as the previous population model to overcome some of the mentioned limitations. Based on the sampled factor loadings and latent variable correlations, we calculated the variance-covariance matrix for simulation as described before. This process was repeated 100,000 times, resulting in the same number of potential matrices. We decided on the variance-covariance matrix that produced the highest minimal eigenvalue (Fig. 8). This was done because in previous tests, where we manually specified the model, none of the algorithms was able to come close to the population model. We argue that this is due to ambiguity in the sampled data resulting from a covariance matrix that contains (near) redundant information. This can be indicated by its eigenvalues, where, loosely speaking, eigenvalues close to 0 indicate redundancy in our matrix. Choosing the solution out of the 100,000 with the highest minimal eigenvalue guaranteed that our matrix does not suffer from the aforementioned redundancy. This can be understood when considering the geometric interpretation of eigenvalues: they denote the factor by which a corresponding eigenvector is scaled when the linear transformation, represented by the matrix, is applied to it. If an eigenvalue is zero or close to 0, the information contained in that eigenvector is lost after transformation. As our variance-covariance matrix fully depends on the specified factor loadings and latent variable correlations, we thought this to be a symptom of an ambiguous factor structure (the combination of factor loadings and latent variable correlations), which would ultimately mislead our algorithms, as the population model is not clearly represented in the noised data. At least some evidence of this was gathered in our study, as the two previous attempts to specify a complex population structure produced variance-covariance matrices where the last eigenvalues were very close to 0, and none of the algorithms was able to identify the true model in any condition or any run.

2.3.2 Conditions

On top of our simulated data coming from two different population models, we varied two other conditions: the misspecification level of our starting model and the sample size. We chose these conditions and the corresponding levels to closely resemble what researchers might encounter in practice.

The first condition has inherent importance for our task. Only in the presence of a misspecified model is it necessary to conduct a specification search. As described in the

introduction, we aimed to closely mimic how researchers might specify a model in their research: some part of the model might be known a priori from previous exploratory research or a sound theory, and another part might reflect a new hypothesis or untested parts of a theory. We thus specified our initial models in the following way. First, we took a certain percentage (60% for the high and 30% for the low misspecification condition) from the true population model. In contrast to Jing et al. (2022), this was not done once and thus the same for each iteration or run, but was sampled differently in each iteration. To keep the results comparable, each algorithm started with the same set of models. With this approach, we eliminated the possibility that our results depend on a particular misspecified starting model.

Next, the rest of the model had to be specified. We tried to model this process according to an informed and parsimonious guess by a researcher. Despite not knowing exactly how the rest of the model looks like, the researcher might have a rough idea about how many parameters they additionally need in their model, tending towards more simple, meaning less parameterized, models. Based on this thought process, we calculated the probability for each potentially to be estimated parameter j in the rest of the model according to the following definition:

$$P(\vartheta_j = 1) = \frac{n.params_{true} * u}{n.entries},$$

where $n.params_{true}$ and $n.entries$ are the number of parameters to be estimated in the rest of the true model and the overall number of entries, i.e. all potentially to be estimated parameters, in the rest of the true or starting model, respectively, and u is an underparametrization factor. With this formula, we guarantee that most starting models will be underparametrized and misspecified model versions of the true model. It also allows us to calculate the final expected correct specification rates, which are 82.90% for low and 70.19% for high misspecification[4]. This will, on average, require 9 (low misspecification starting model/ average population model), 17 (low/complex), 15 (high/average), and 29 (high/complex) steps to get to the true model.

Our second condition was the sample size. This is not only important due to the direct consequences of how well the true model is reflected in the noised-up data but also because of its dependence on our later-described objective functions. Last but not least, the sample size is a key factor in all statistical analysis and thus of high importance for practitioners. Based on Goretzko et al. (in press), we specified three levels and one additional of N=3000 that we added to, at least to some extent, rule out

[4]These are also averaged across the true and complex population models, as the rates slightly vary based on their complexity.

the possibility that our results solely depend on the overall low sample sizes frequently occurring in applied social and life science research. They found the 25%, 50%, and 75% quantiles of the sample sizes to be 299, 528, and 998. We almost precisely stuck to these values; however, we aimed for a slightly lower low sample size to diverge the levels more and better test the behavior in low sample sizes. Thus, we specified a low (N=250), medium (N=500), and large sample size (N=1000).

2.3.3 Model Estimation

As discussed before, ML estimation is the logical choice in our case, because we know our data follows a normal distribution. We believe ML estimation to be familiar to most readers and in order not to overload with unnecessary notation, we want to very briefly introduce how we derive our before mentioned discrepancy function on the grounds of ML (Bollen, 1989; Preacher, 2016).

First, we summarize all our random variables, manifest and latent, into random vectors z_i of length $p+q$, where $i \in [1, N]$ and p and q are the numbers of our manifest and latent variables respectively. With a multivariate normal distribution, independently distributed z_i and, like above, putting all our to be estimated parameters, i.e. the factor-loadings and latent variable correlations, in a vector θ we get our standard log-likelihood

$$l(\theta) = \frac{-N(p+q)}{2}\log(2\pi) - \frac{-N}{2}\log(|\Sigma(\theta)|) - \frac{1}{2}\sum_{i=1}^{N} z_i^\top \Sigma(\theta)^{-1} z_i.$$

As we are, in our case, not interested in the mean structure of our data, we can assume that the z_i are centred around 0. This also implies that the mean of our Gaussian is (a vector of) 0 and $z_i z_i^\top$ yields our variance-covariance matrix when divided by $n-1$. With this, the fact that a scalar is equal to the trace of a scalar, and by introducing a factor of 1 in the form of $\frac{N}{N}$, we can rewrite the last subtrahend as

$$-\frac{N}{2} tr\left(S^* \Sigma(\theta)^{-1}\right)$$

where S^* is the ML estimated covariance matrix, which uses N instead of $N-1$ in the denominator, making it slightly biased. Ignoring constant terms and substituting the just derived quantity into our log-likelihood we arrive at:

$$-\frac{N}{2}\left(\log(|\Sigma(\theta)|) + tr\left(S^* \Sigma(\theta)^{-1}\right)\right)$$

which is slightly different than the real ML discrepancy function

$$F_{ML} = \log(|\Sigma(\theta)|) + tr\left(S\Sigma(\theta)^{-1}\right) - \log(|S|) - (p+q).$$

The two last terms result, because the F_{ML} is actually derived not only from the likelihood but from the likelihood-ratio, which compares our likelihood to that of the saturated model, i.e. the model which perfectly reproduces the (slightly biased) empirical variance-covariance matrix. For its derivation we can essentially follow the same steps as before. $\log(|S|)$ corresponds to $\log(|\Sigma(\theta)|)$ with only the matrices exchanged, and $(p+q)$ results because $SS^{-1} = I$ (same for S^*) which is the identity matrix and its trace thus the number of parameters in our saturated model. We subtract the terms as we take the log of the likelihood-ratio, which produces the difference between the two log-likelihoods. As the additional terms, however, don't depend on θ, the estimated parameters will be the same as looking only at the models likelihood.

As F_{ML} is analytically infeasible we resort to non-linear optimization. The optimizer we will use in our study is called *nlminb* and is a variant of a Newton-Raphson or Quasi-Newton algorithm (R Core Team, 2016). In contrast to the latter, it implements additional constraints on the parameters, which include varying lower and upper bounds on the parameter estimates. The algorithm approximates the to be optimized non-linear with a linear function, which in the scalar case is a Taylor expansion around the current iteration, at each iteration. It starts with an initial guess of the parameter values, and then iteratively improves the guess by the multivariate analogons used by the Taylor expansion - the gradient and the Hessian - of the function at the current guess. In our case, these are very close to the Fisher Information Matrix and the score function, as our F_{ML} is mostly build around the likelihood of our model.

2.4 Analysis

As previously mentioned, the aim of our study is to compare various algorithms on simulated data under different conditions. To achieve this, we implemented the algorithms in R (version 4.2.2; R Core Team, 2016) using the lavaan package (version 0.6-12; Rosseel, 2012) and R functions published by K. Marcoulides & Falk (2018). The latter facilitated communication with lavaan by handling the conversion of our binary indicator vector to a fitted SEM model. We built the algorithms based on the papers cited in the relevant chapters, modifying them as necessary.[5]

For computation, we used a Linux-based High-Performance-Cluster provided by

[5] All implementation details not described in this work can be found on this git repository.

the Leibniz-Rechenzentrum. We used the R package batchtools (version 0.9.15; Lang et al., 2017) to communicate with the cluster. This package provided a range of helper functions that set individual seeds for each generated data instance, enabling the same data to be analysed by each algorithm. We allowed each algorithm a maximum of 12 hours on average and 24 hours for the complex population model to converge. We provide the algorithms also in the form of an R package called *MetaSS*, which is made available through GitLab. After completing the computations, we assessed performance using various indicators, which we will discuss later. Our objective is not only to identify the best algorithm(s), which we report as *study 1*, but also to test potential hybrid versions of the best algorithms, which we refer to as *study 2*.

Regarding the search space of our algorithms, we decided to explore only factor loadings and latent variable correlations. Although different types of parameters, such as regression coefficients for the latent variables or residual correlations, are possible, we focused solely on the two types mentioned. In psychometric research, residual correlations are the next most common type after factor loadings and latent variable correlations. Although residual correlations have some justification in cross-sectional study designs (Landis et al., 2009), they are more commonly used in longitudinal or multi-method settings to account for unique variance in repeated measures of the same variable or method. Although these cases are interesting, they are rather specific, and we opted to stay closer to simpler CFA-like models.

2.4.1 Specification of Objective Functions

The most crucial choice when implementing metaheuristic algorithms in a specific setting is likely to be that of the objective function. Often, it is not only highly problem-dependent but also needs to match the corresponding optimization technique. For our study, we utilized two generally differing bases to defining the objective functions: fit indices and information criteria.

We decided to use information criterion-based objective functions for our algorithms that are highly dependent on neighbour-to-neighbour movements: TS and SA. Both algorithms, in our case, define a neighbour as a solution that differs only in one parameter, e.g. 101 would be considered a neighbour of 001 but not of 011. With this choice, the objective function needs to be sensitive to very small changes in the current iteration model and to the solutions around the starting model. Of less importance is the scale of the objective function values. With these criteria, we see the BIC as a reasonable choice. In addition to it also being utilized by G. Marcoulides & Leite (2014) and Jing et al. (2022), the BIC's derivation shows that it is highly sound with

Bayesian theory and developed on solid statistical grounds.

The GA, AGA, and PSO are less dependent on the mentioned limitations, as they explore the search space more rigorously. Also, for the calculation of the AGAs adaptive parameters, the scale of our solutions is more important. With this in mind, and by reference to Jing et al. (2022), G. Marcoulides & Leite (2014) and Murohashi & Toyoda (2007), we make use of fit indices. Fit indices have the benefit that their scale allows us to meaningfully define bounds and search for specific values we want to achieve. Also, different fit indices have varying qualities regarding their sensitivity to special kinds of misspecification (West et al., 2012). By combining multiple indices, we can thus avoid some of the danger associated with this problem.

To combine multiple fit indices into one objective function, we need to first bring them onto the same scale. Most indices are in $[0, 1]$, but not on the same scale, i.e. a specific value has a different meaning for each index (even when inverting badness-of-fit to goodness-of-fit indices or vice versa). This can be achieved by utilizing a three-parameter logistic transformation

$$f(x) = \frac{a}{1 + \exp(-b * (c - x))} \tag{1}$$

where a sets the maximum value of the function, b defines the slope of the function and thus the range of sensitivity, and c is used for centering around a specific value. In our case, we want to limit our function to the range $[0, 1]$ and have b defined such that models with a low enough x value, i.e. that of our fit index, are squashed to practically 0. With this, we tackle some danger of running into very bad local solutions, as our objective function will not be sensitive to values outside a specific range. c is especially important to bring multiple fit indices onto the same scale. In our case, we will use the cutoff for acceptable model fit described in Table 1.

The choice of fit indices to include in the objective function was based on recommendations by Hu & Bentler (1999) and experience from our own previous research. Hu & Bentler (1999) advised using the CFI and SRMR together, as they were best able to identify misspecification in their simulation. From this study, we also took the cutoff values of 0.95 and 0.08, which correspond to the typical α level of 5% in falsely classifying a model as correctly specified. We decided to also include the RMSEA, as it is nowadays one of the most common fit indices and fits in the picture of overcoming problems of the other indices (Hu & Bentler, 1999; West et al., 2012): CFI is not affected by especially small N, but measures fit compared to a null-model, which may be inappropriate in some cases; SRMR is sensitive to many manifest variables having

small residual variance, but has good overall performance and is an absolute measure; RMSEA is sensitive to small N, but overcomes the problem of sensitivity to small residuals and is also an absolute fit index.

Even though two of the above indices include explicit complexity penalization in the form of degrees of freedom, it is known that this penalization is usually too weak and overparameterized models will yield better fit. Due to this fact, we added an additional complexity penalty to the objective function, in the form of a linear function of the model parameters used in exploration. With this, we arrived at the following objective function for the GA, PSO, and its inverse for the AGA:

$$\frac{1-\lambda}{3}\left(\left(1-\frac{1}{(1+\exp(-55*(0.08-SRMR))}\right)+\left(\frac{1}{1+\exp(-55*(0.95-CFI))}\right)+\left(1-\frac{1}{(1+\exp(-55*(0.06-RMSEA))}\right)\right)\lambda\frac{n_{m.params}}{n_{t.params}} \tag{2}$$

where $n_{m.params}$ and $n_{t.params}$ are the models estimated parameters within the search space and the search spaces' total potentially estimated parameters respectively. Our objective function is just the mean of our logistically transformed fit indices weighted with a linear penalty for model complexity.

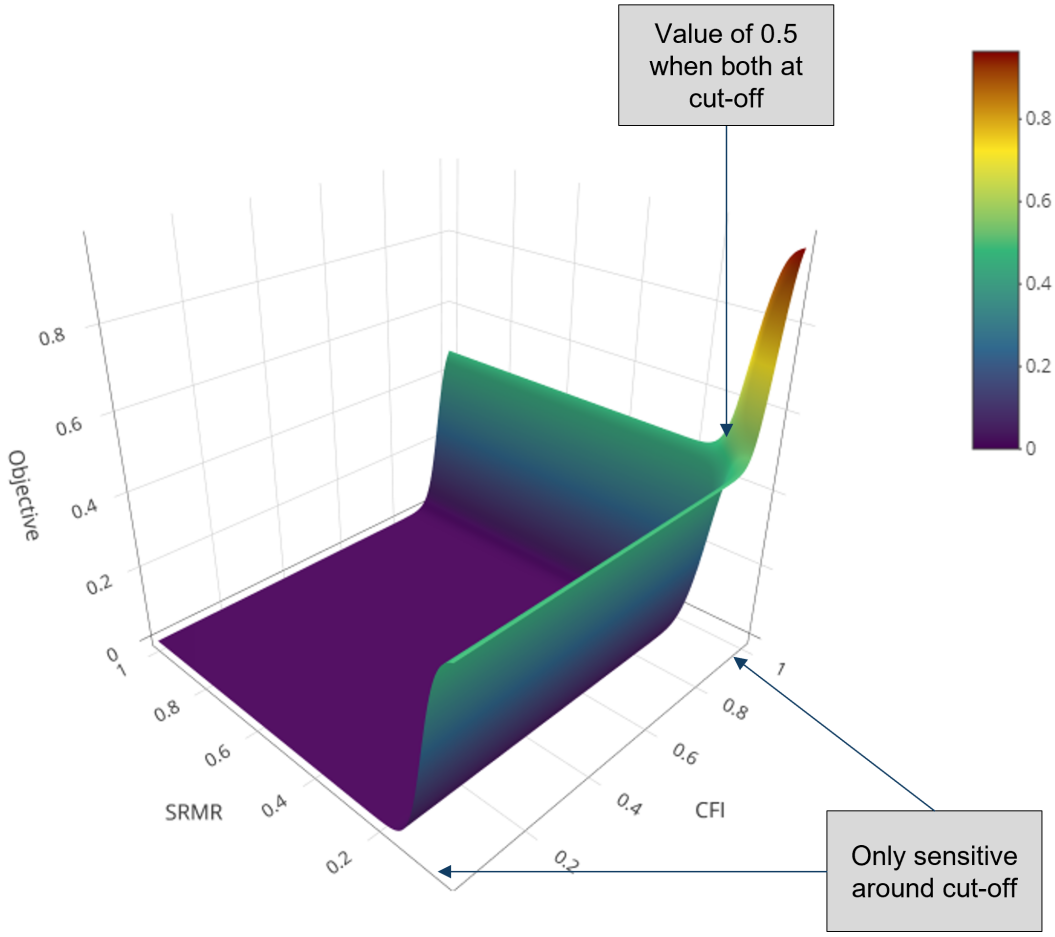
The last objective function used in our study is only used for the hACO and very similar to Eq.2. It is defined as

$$\max\left[0,\frac{1}{3}\left(\left(\frac{1}{(1+\exp(-55*(0.08-SRMR))}\right)+\left(1-\frac{1}{1+\exp(-55*(0.95-CFI))}\right)\left(\frac{1}{(1+\exp(-55*(0.06-RMSEA))}\right)\right)-\lambda\log_2\left(1+\frac{n_{m.params}}{n_{t.params}}\right)\right] \tag{3}$$

where both the combination of fit indices and the penalty are handled differently. This is in order to easily exclude zero objective function models, which are thought to mislead ACO (and thus hACO; Jing et al., 2022). Remember that in hACO, only a model with a non-zero value will be used as a starting model for the Tabu part of the algorithm. The purpose of the penalty is thus not only to punish bad models to a certain degree but to exclude them completely from the search. In addition, contrasting the original hACO objective function, we used the mean of the fit indices and not the product of the same. The latter resulted in too small values, and no convergence could be achieved.

To get a better idea of how the fit index-based objective functions work, we represented part of it in Fig. 9. We can easily see that the objective function is only sensitive to certain values of the fit indices, which are located around the cutoff. For large parts of the input space, the resulting value will be very close to 0, which matches exactly what we aimed for.

Figure 9
Fit Part of Objective Functions.



2.4.2 Algorithm Specific Choices and Hyperparameter Settings

When implementing the algorithms, a few additional choices, including hyperparameter settings, needed to be made. In the following, we want to describe and justify the most important ones.

TS was built on an existing function, implemented by K. Marcoulides & Falk (2018). However, to be able to compare the results across the metaheuristics, a different termination criterion was chosen. Instead of the number of iterations, the number of iterations for which no update occurred for the best so far model was decided upon as a criterion. This also removed some of the subjectivity, as knowing the number of iterations it takes the algorithm to converge is largely based on speculation. In order to assume convergence, based on previous research and manual testing, the number of iterations not resulting in an improvement was set to 40. The length of the Tabu list

was set to 8 based on previous studies (Jing et al., 2022; G. Marcoulides et al., 1998) and manual tuning.

In SA, the calculation of the starting temperature is a known problem. We oriented ourselves towards a guideline by Ben-Ameur (2004), in which the initial temperature is based on the desired acceptance rate of the Metropolis step (here 0.8) and the mean increase in the objective function over all worse neighbours of the starting model. This should ensure that enough exploration is done with respect to the starting condition the initial model provides, i.e. that the temperature is actually high enough that we would accept the average worse model compared to the starting model with the desired initial acceptance rate. Another important choice is that of the annealing function. We decided on the adaptive rule:

$$T_{t+1} = T_t \exp\left(\frac{-0.7T_t}{SD_{obj,t}}\right)$$

where T_t is the temperature at time t and $SD_{obj,t}$ is the standard deviation of objectives at time t (MD et al., 1986; Rozenberg et al., 2012). This ensures that our temperature decreases faster when the standard deviation of objectives is lower, i.e. when the algorithm doesn't find models that considerably improve the current model, which is more likely to happen as the search is further progressed. The factor of 0.7 is usually another hyperparameter, which is an estimate of the difference in the expected cost at t and $t+1$ divided by $SD_{obj,t}$. It would be possible to estimate this property retrospectively; however, MD et al. (1986) suggest a value of 0.7 to which we adhered, as the computational burden would otherwise increase dramatically. Additional hyperparameters are the convergence criterion (the number of iterations not resulting in an update) and the number of iterations before the temperature is annealed. We set the values to 100 and 10, respectively. Like for TS, a neighbouring model was a solution that differed only in one parameter. This was chosen because the algorithm is simpler compared to the others, and a broader definition could increase the risk of skipping potentially good and important solutions.

In the GA, the choice of hyperparameters is simpler compared to previous algorithms. Larger values ensure more exploration without the danger of accepting too many bad models, as seen in SA. Based on our previous testing, we set the population size, number of parents, number of mutants, and convergence criterion (number of generations for which objectives in the population did not change) to 30, 8, 6, and 100, respectively. We used a penalization value of 0.5 and only changed one randomly chosen parameter for mutation.

For the AGA, we first adapted the hyperparameters suggested by Srinivas & Patnaik (1994). However, this resulted in excessively high adaptive probabilities, causing the algorithm to suggest completely random models too frequently. To rectify this, we manually tuned the algorithm by printing the calculated adaptive probabilities in each iteration and strongly decreasing the range of accepted probabilities for mutation. We started with $k_2 = k_4 = 0.5$, which resulted in the complete disruption of subaverage solutions, but we then decreased these values to 0.05 and 0.1 respectively. This ensured that models resulting from crossover, which possibly inherited good qualities from their parents, underwent less change. The hyperparameters k_1 and k_3 were both set to 1, ensuring that low fitness parents always underwent crossover, consistent with Srinivas & Patnaik (1994). We made this decision based on the success of the GA in Murohashi & Toyoda (2007), where the probability of crossover was implicitly set to 1[6]. The remaining hyperparameters were set like those in the simple version of the GA.

We were not able to reference any previous research on specification search when defining the hyperparameters for PSO. However, as GAs and PSO are closely related (Kennedy & Eberhart, 1997), we used its settings as a guide. We initially tried values similar to those used in our simple GA for the population size and the number of generations without change (convergence criterion), but we found that convergence times were too long. In the end, we decided on values of 30 and 50, respectively, which relaxed the convergence criterion. We observed that larger values did not result in better models. Another hyperparameter in PSO is the maximum absolute velocity that is allowed. This is specified to retain new changes in the solutions, i.e., changes that are not only in the direction of the swarm's best solution, even when they reach their personal best position. We set the maximum velocity to 6, as recommended by Kennedy & Eberhart (1997). After logistic transformation, this results in probabilities between 0.9975 and 0.0025. As with the other algorithms, we set the penalization to 0.5.

For the hACO, we used Jing et al. (2022) as a reference[7]. In order to better compare our algorithms and because we found it to be more justifiable (as mentioned above), we decided to change the fit indices used in the objective function to those mentioned previously. Regarding the hyperparameters, we set the size of the tabu-list

[6]Note that the same logic does not apply to the probability of mutation. In the GA employed in our study and G. Marcoulides & Drezner (2001), exactly one parameter is changed, whereas the mutation probability is applied to each potentially estimated parameter in the AGA.

[7]We also tested the original implementation in a previous project, but it did not converge within the given time frame.

to 3 and the cutoffs for the fit indices that put a solution on the ban-list to SRMR ¿ 0.2, CFI ¡ 0.75, and RMSEA ¿ 0.2. The change in entropy between iterations, which defined the convergence criterion, was set to 0.001, as in Jing et al. (2022), and the penalty was set to 0.5, as before.

2.4.3 Performance Evaluation

The evaluation and comparison of the algorithm was conducted based on the following criteria:

- Proportion of runs in which the true model was obtained
- Hamming-Distance to the true model
- Stability of performance across conditions
- Convergence Speed

The first metric we wanted to look at is how often the correct model was found. This was of central importance in our study as the true model sets the goal the algorithms ought to achieve. However, it is not only important whether the algorithms found the true models, but also how close they got. We additionally looked at the Hamming distance to the true model. The Hamming distance is a measure of the difference of bit strings, where the value is the number of different bits in equal length strings (in our case, the binary indicator vectors representing our models). For example, the distance between 101 and 100 is 1, while the distance between 101 and 010 is 3. In our case, however, there is a problem with this metric (or any other metric) - our final models, i.e. the population models, are not unique. If, for example, the first five indicators in our average population model load on factor F_1 and the second five load on factor F_2, like in Fig. 7, or vice versa, the first five load on F_2 and the second five on F_1, it produces the exact same model. They only differ in how the factors are indexed.

This problem is especially severe for highly exploratory algorithms like the GAs and PSO, where it is more likely that (close to) equivalent models are suggested. For an algorithm like TS, which looks at neighbours that differ only in one parameter, it will only happen that the search redirects to a different but equivalent model if the algorithm starts *on the edge* of two such models. Regarding the Hamming distance, the question remains as to which of these final models ($3! = 6$ in the average population and $4! = 24$ in the complex population model) we evaluate the distance. In our study, we decided to take the minimum of these distances, assuming that the algorithms

are on track to one of those population models and have achieved the most progress regarding this version. From a statistical perspective, we can view this as an estimator for our Hamming distance, which will obviously be biased downwards. However, we believe that there is no justifiable alternative as we simply cannot know which version the algorithm will tend towards. It might also be the case that the algorithm is completely misled and not on track to any of the population models, in which case one version might produce a low value by chance, thereby again making our estimator biased downwards.

Another important criterion that was used in conjunction with the previous two is the stability of our metrics across the conditions. Even though it might be acceptable if an algorithm only performs well in a few conditions, and these conditions are known, we desire to find an algorithm that performs well across them. For this reason, we both inspected our results graphically as well as used the runs as observations in regression analysis, where the regressands were the respective metrics. In our case, this resulted in a binomial regression for the runs in which the true model was obtained and a negative binomial or Poisson regression for the Hamming-Distance. While the former is an obvious choice, the latter might be confusing, as we typically use these types of regression for count data. Taking the definition of our Hamming-Distance, we can, however, immediately realize that it is actually more accurately described as a count rather than a distance, because we simply count the number of different entries in our solution vectors. Regarding the choice between Poisson and negative binomial distributions for our outcome, we checked for overdispersion and decided on the more appropriate model. Overdispersion refers to the assumption of the Poisson distribution of equal mean and variance, which is violated if our data are more dispersed than our estimator suggests. In this case, we want to entangle mean and variance, which can be achieved with a negative binomial distribution.

Lastly, we want to look at the speed of our algorithms. It is obviously important that an algorithm should not only be able to achieve good results but also do so in a reasonable amount of time. One example would be the brute-force method, which tries every solution and picks the best one, which will always be the correct solution. The whole reason we conducted our study and why metaheuristic algorithms were developed is that brute-force is infeasible in many situations like ours because it does not converge in a reasonable amount of time. As practical researchers need to follow restrictions, deadlines, and want orientation on how long a procedure needs, we see it as additionally important to find out how long they take to suggest an improved model.

In SEM and factor analysis, equivalent models cannot only result like described above but also by means of rotation. In EFA, this is a common issue, known as the *rotation problem*. The problem, put simply, states that rotations of our latent-variable and factor loading matrices also lead to acceptable models, which produce the same expected values and variance-covariance matrices as their unrotated counterparts. In our case, the population models are already fairly well-rotated, which might be hard to see from only their graphical path-model representation. Writing them down in matrix format, however (which can be looked at in the accommodating code on GitLab), reveals that they already show a simple structure, which is what most rotation methods aim at. To be more sure regarding this topic, we also rotated our models using the approaches we deemed most common: Oblimin, Geomin, Varimax, Promax, and Quartimax. We found that especially the orthogonal methods, i.e. methods that produce uncorrelated factors, produced models that significantly less clearly represented a simple structure as they needed to compensate for the factor correlations present in the population models with increased factor loadings. However, even though this hurt the simple structure, the original models' loadings still showed the highest values after rotation by large margins. Our original idea was to use the alternative rotated models in the evaluation with respect to our metrics like we use the equivalent models from re-indexing. However, as no obvious alternative patterns resulted from rotation, we decided to neglect this problem in our study.

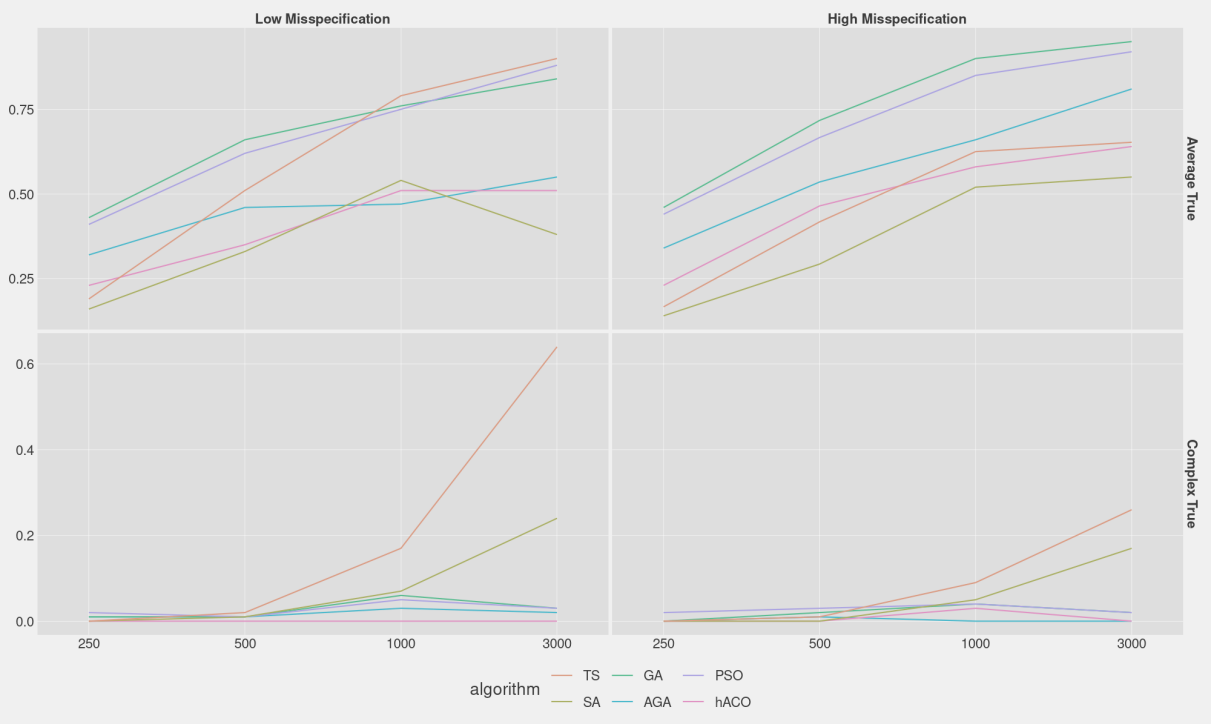
3 Results

3.1 Study 1

In our first and primary study, we investigated six different algorithms to perform specification search as described above. We evaluated the proportion of runs that produced the correct model, the Hamming distance to the true model, and convergence times.

3.1.1 Proportion of Found Population Models

Figure 10
Proportion of runs producing the true population model.



In Fig. 10, we summarise one of the main results of our study. On the y-axis, we see the proportion of runs finding the true model, while the x-axis displays the sample sizes, and the grid disentangles the additional conditions of the starting model (columns) and population models (rows).

For the average true model, we can see that the best two algorithms were GA and PSO, which behaved very similarly in their overall performance. The best result across all conditions was achieved by the GA under the worst starting model and with a sample size of 3000 - 95% of runs resulted in the true model being found. While

all algorithms in general profited from a larger sample size, the same is not true for a better starting model: while TS greatly suffered from being far from the true model in the beginning, the opposite holds for most of the other algorithms, especially GA, AGA, and PSO. For example, with a sample size of 1000, they found the correct model in 90%, 66%, and 85% of the runs under the worst starting model, respectively, while only 76%, 47%, and 75% of the time under the better starting solution. SA and hACO had more difficulty in reaching their goal. They were only able to achieve rates of just above 50% under sample sizes of 1000 and 3000.

The results under the complex true model were very different. Excluding TS and the largest sample size, the population model was never found in more than 7% of the runs, which was achieved by SA. While GA and PSO again performed better under low sample sizes compared to the other algorithms, the magnitudes are far from that under the average population model. This time, SA performed second best, while it was the worst under the average model. By far the best algorithm, given sufficiently large sample sizes of 1000 or 3000, was TS, which found the true model up to 64% of the time under its best condition. Like before, TS was largely hindered by a worse starting model.

Table 2
Binomial regression results of found true models.

Predictor	B_{N3000}	B_{N1000}
(Intercept)	1.50 (0.10)***	0.71 (0.10)***
Complex True	−3.58 (0.08)***	−4.03 (0.12)***
Algorithm: SA	−1.03 (0.10)***	−0.66 (0.12)***
Algorithm: GA	0.19 (0.10)	0.71 (0.12)***
Algorithm: AGA	−0.64 (0.10)***	−0.12 (0.12)
Algorithm: PSO	0.12 (0.10)	0.60 (0.12)***
Algorithm: hACO	−0.99 (0.10)***	−0.44 (0.12)***
N: 250	−2.11 (0.09)***	−1.72 (0.09)***
N: 500	−1.21 (0.08)***	−0.80 (0.08)***
N: 1000	−0.45 (0.08)***	
High Miss. Start	0.07 (0.06)	0.10 (0.07)

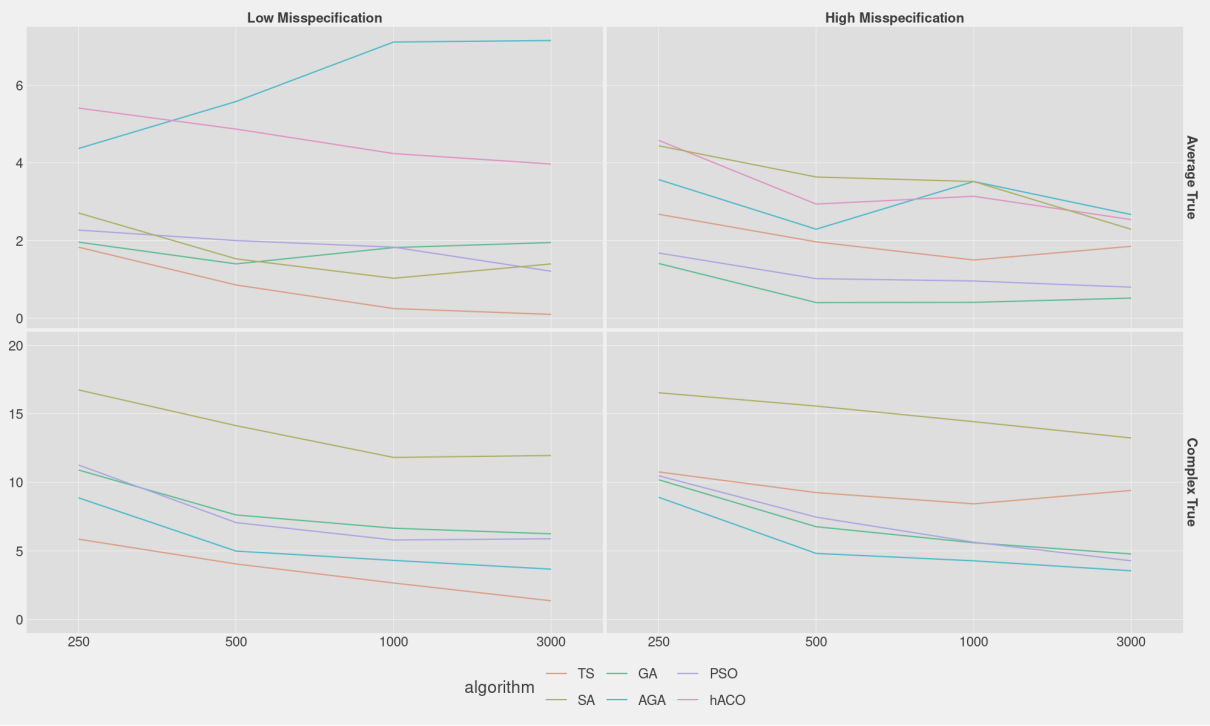
Notes. Standard errors in brackets. $^{***}p < 0.001$; $^{**}p < 0.01$; $^{*}p < 0.05$

To gain a better understanding of how the algorithms performed across conditions, we conducted a binomial regression using the algorithms and conditions as predictors, essentially controlling for them, and the outcome was whether the true model was found in a run or not (Tab. 2). As a sample size of 3000 is rare in psychological research, we performed the regression with and without this level. Our findings were consistent

with our visual observations. Both GA and PSO increased the odds of finding the true model compared to TS, but this increase was not statistically significant when we included a sample size of 3000. Overall, GA performed slightly better than PSO. The other algorithms were worse at finding the true model than TS, regardless of whether the maximum sample size allowed was 3000 or 1000. Regarding our controlling variables, i.e., our conditions, we found that the complex true model was the strongest predictor overall, which was also confirmed visually. Decreasing sample sizes led to decreasing odds overall, while a bad starting model did not have the same effect when controlling for the algorithms.

3.1.2 Hamming-Distances to Found Population Models

Figure 11
Hamming-Distances to the true population model.



To gain a better quantitative understanding of the performance of our algorithms, we examined the estimated Hamming distance to the true models (Fig. 11). Because the distances for hACO were so large (40-60 on average), we decided to exclude them from the graphic to maintain resolution on the majority of the findings. Under the average population model, the best-performing algorithm depended on the misspecification level of the starting model. When we started closer to the true structure,

TS performed the best, never producing models further than two parameters away on average from the true model. When we started further away, the results were almost the same for GA and PSO, with GA producing slightly better models. We also observed SA performing as well as GA and PSO under low misspecification of the starting model, but it was the worst algorithm overall under high misspecification. AGA and hACO performed the worst overall under the average true model, producing models that were on average 7 parameters away from the true model.

When looking at the Hamming distance to the complex population model, the overall magnitude of values was much larger. While we previously looked at ranges from 0 to 7 parameters, we observed differences of 2 to 29 parameters (hACO result not shown) on average to the complex population model. We found TS to perform the best across all sample sizes for better starting solutions, but it performed much worse when the latter was further away, resulting in the overall second-largest Hamming distances. The other algorithms did not show this dependence. Compared to before, AGA now performed the best or second best depending on the starting model, while GA and PSO produced very similar, comparatively good results. They were mostly within 10 parameters of the true model. The hACO was most influenced by the complexity of the true model. While its performance was within the range of the other algorithms under average complexity, the complex structures produced run averages that were further away from the true model than the average starting model (17 for the better, 29 for the worse starting model). e again checked the visual inspection of our

Table 3
Negative binomial regression results of Hamming-Distances.

Predictor	B_{N3000}	B_{N1000}
(Intercept)	0.21 (0.05)***	0.26 (0.05)***
Complex True	1.53 (0.03)***	1.51 (0.03)***
Algorithm: SA	0.73 (0.05)***	0.73 (0.05)***
Algorithm: GA	0.03 (0.05)	0.02 (0.05)
Algorithm: AGA	0.69 (0.05)***	0.61 (0.05)***
Algorithm: PSO	0.09 (0.05)	0.12 (0.05)*
Algorithm: hACO	1.52 (0.05)***	1.41 (0.05)***
N: 250	0.36 (0.04)***	0.33 (0.03)***
N: 500	0.08 (0.04)*	0.06 (0.04)
N: 1000	0.02 (0.04)	
High Miss. Start	−0.04 (0.03)	−0.02 (0.03)

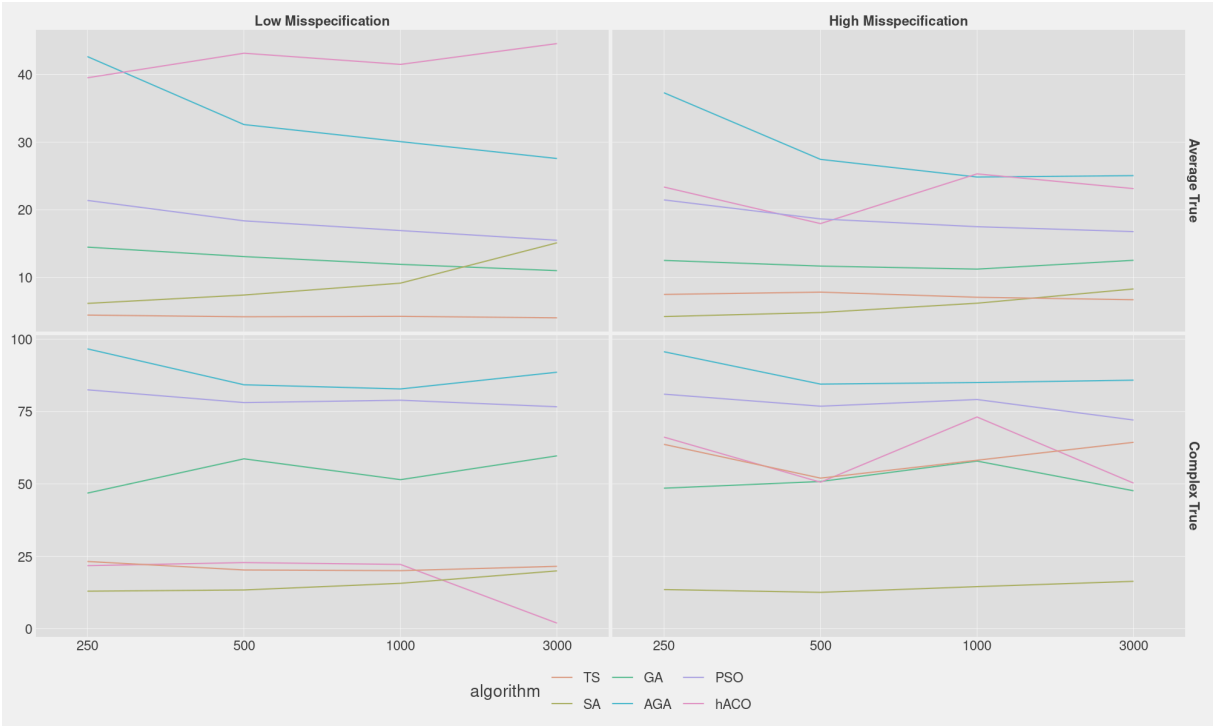
Notes. Standard errors in brackets. $^{***}p < 0.001$; $^{**}p < 0.01$; $^{*}p < 0.05$

findings with regression analysis (Tab. 3) to gain a better quantitative understanding

of algorithm performance. As the Hamming-Distance can be interpreted as a count of how many parameters differ between two solutions, we relied on negative binomial regression. With an overdispersion-parameter of 8.7 and a positive overdispersion-test, we chose the negative binomial over Poisson-regression, which estimates the variance separately from the expected value. Compared to TS in the intercept, all other algorithms led to an increased expected Hamming-Distance. However, the increase for GA and PSO was minor and non-significant. Although AGA produced close models under the more complex population model, it did not perform well when controlling for our conditions. The worst results were found for hACO and SA. Regarding our conditions, we found the same pattern as for the resulting true models: the more complex the true model, the worse the final solutions, increasing sample sizes led to overall better performance, and the misspecification level of the starting model did not show an overall effect. However, the latter varied strongly between algorithms, as we have seen in our graphical analysis.

3.1.3 Convergence Times

Figure 12
Average convergence times in minutes.



Lastly, we investigated the convergence times of our algorithms (Fig. 12). Overall,

the results ranged from only a few minutes to over one and a half hours. SA was shown to be the fastest algorithm and suggested final models in less than 20 minutes on average. The population-based algorithms (GA, AGA, PSO) took longer on average than those which evolve single solutions (TS, SA, hACO). While the former needed the same time regardless of how far our starting solution was from the true model, the latter were highly influenced by the initial solution. By this, a very simple algorithm like TS can actually take longer than a more complex algorithm like GA. While SA is the fastest algorithm overall, GA is by far the fastest with regard to population-based algorithms, followed by PSO and then AGA. hACO took longer than most other algorithms; however, it displayed a very similar performance to TS under the complex true model. As with our previous results, the complexity of the true model was the overall most influential factor.

3.2 Study 2

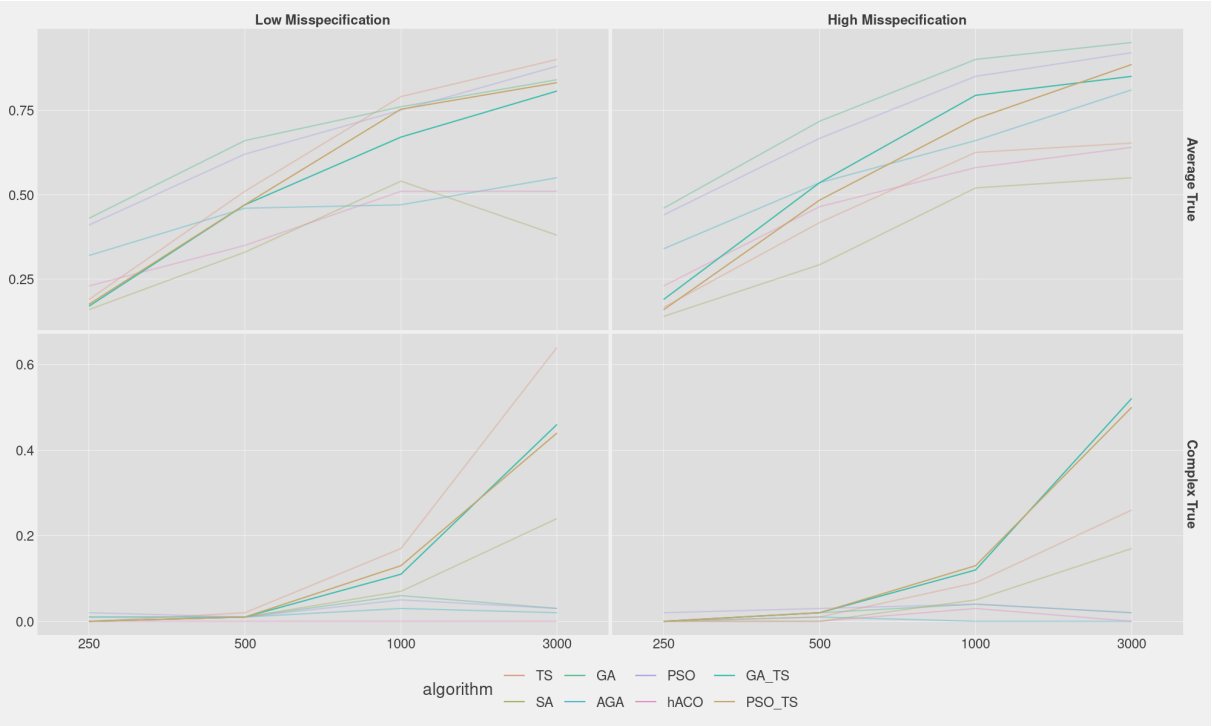
In Study 2 of our work, we aimed to combine the best algorithms from the previous study. Based on the quality of models found and convergence times, we selected TS, GA, and PSO (in no specific order) to be the best out of the given selection. These algorithms performed well in some conditions (TS) or delivered overall good results (GA and PSO). Although there are many possibilities for combining these algorithms, we decided on a simple approach based on knowledge of GA's exploratory strategy and the interpretation of our results. It is known that GA excels in finding very good, near-optimal solutions because it can easily escape local solutions but has difficulty making small model improvements due to its strong exploration, which we believe is also the case for PSO. Conversely, TS is better when starting with a good model, which we can provide from the results of GA and PSO. As being able to make these small improvements is more important when close to the optimal solution, we thought a sequential approach would be the best way to combine these algorithms. We thus used the models produced by GA and PSO as starting models for TS, resulting in GA-TS and PSO-TS algorithms. As mentioned before, this approach is very common with GAs.

3.2.1 Proportion of Found Population Models

Like our non-hybrid algorithms, we first examined the runs in which the hybrid algorithms (GA-TS and PSO-TS) found the correct model. Under the average population model, the hybrid algorithms overall found the correct model less frequently than their

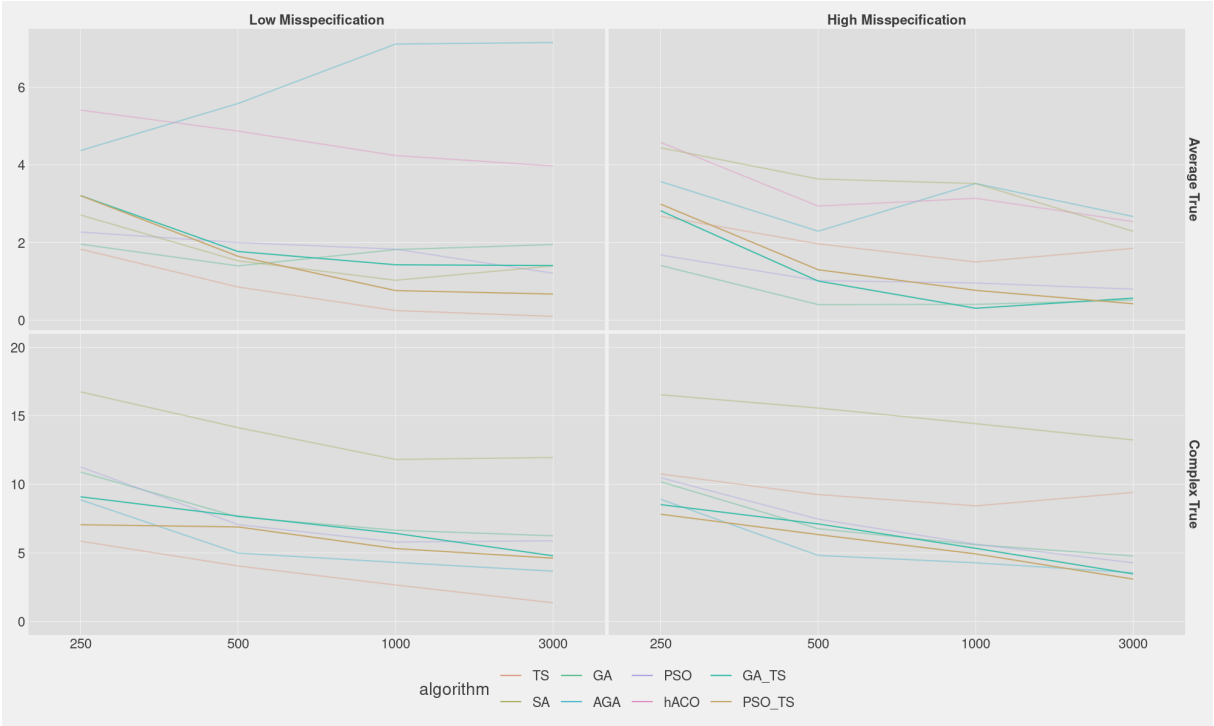
Figure 13

Proportion of runs producing the true population model (hybrid Algorithms).



non-hybrid counterparts GA and PSO. For TS, the hybrids only improved the results under worse starting models (Fig. 13). Similar to TS, the hybrid algorithms showed a strong dependence on sample size. The best result they achieved was slightly above 75% under larger sample sizes. Under the complex true model, the results were more promising. While the hybrids performed worse than TS and better than GA and PSO under low misspecification, they significantly improved performance compared to all three algorithms when starting with a less accurate model. Like GA and PSO, the hybrids also demonstrated the quality of not being negatively influenced by the starting solution, producing almost identical results across these conditions. Lastly, the two hybrid versions performed almost identically, with GA-TS being slightly better given a worse starting model and an average population model, and PSO-TS being slightly better given a better starting model and an average population model. When compared to the remaining algorithms, both hybrids generally performed better, especially under the complex true model.

Figure 14
Hamming-Distances to the true population model (hybrid Algorithms).



3.2.2 Hamming-Distances to Found Population Models

We again excluded the distances for the hACO, as they were simply too large (Fig. 14). Given sufficient sample sizes, the hybrid algorithms on average result in similar distances to their non-hybrid counterparts. However, they are more influenced by sample size, showing a similar pattern to TS. Compared to the latter, we found that they generally improve over TS alone when the starting model is bad. They also performed better than all other algorithms in the most complex condition (low sample size, high misspecification starting model, and complex population model) and were only surpassed by AGA when sample sizes were medium to large.

4 General Discussion

n our simulation study, we compared six different metaheuristic algorithms to perform specification search in SEM. We analysed conditions of complexity of the true model, misspecification level of the starting model, and sample size. Out of the given collection, GA performed the best across those conditions, followed by PSO and TS, which was confirmed both by visual inspection of the plotted metrics as well as regression results. GA found the true model most often, consistently produced small Hamming distances and achieved the fastest convergence out of the population-based algorithms. Even though GA performed best overall, it did not do so in specific conditions. When the underlying true model is complex, PSO and TS generally produced better results, where especially in large sample sizes, TS outperforms the other algorithms. TS, on the other hand, strongly suffered from a weak starting model. Under the complex population model, AGA was also able to produce good results, however, only with respect to the Hamming distance. Taking the three best algorithms - GA, PSO, and TS - we also combined the former two with the latter sequentially, which is a common approach with GAs. This approach only aided performance in the most complex conditions and significantly weakened the correctness of the found models in the average conditions.

4.1 Comparison to Previous Studies

We find that our results match the overall tone of previous research, with a few exceptions. Like G. Marcoulides & Leite (2014) or Jing et al. (2022), we show that specification search can successfully be conducted with metaheuristic algorithms on artificial data, even though the success strongly depends on the algorithms and conditions.

For TS, G. Marcoulides & Leite (2014) showed that it found the correct model in all of their runs and under all conditions, which is not in line with Jing et al. (2022), who used much simpler population models. Our findings are more in line with the latter, as TS was highly influenced by similar conditions and did not always find the correct model. This finding is surprising, as the population models used by G. Marcoulides & Leite (2014) seem to be much closer to our average population model regarding the number of indicators, factors and magnitude of correlations. One reason for this might be that the true population models in G. Marcoulides & Leite (2014) did not include any cross-loadings, but only residual correlations (for the more complex of the two), and correlations between latent variables were not explored. As

factor loadings and factor correlations can to some extent compensate for each other, it might be easier for an algorithm if it only explores one of the two, in this case, factor loadings. This is because different models can show the same fit, as they produce the same model-implied and estimated variance-covariance matrix (see the description of the rotation problem in section 2.1). This problem does not arise with residual correlations. Additionally, G. Marcoulides & Leite (2014) only looked at a sample size of 1000. Our study and Jing et al. (2022) found that lower sample sizes highly hinder the accuracy of TS, and $N = 1000$ was a minimum for TS to find the true models at reasonable rates.

The point where our results most differ from previous studies is the performance of hACO compared to that of Jing et al. (2022). In our study, hACO was by far the worst algorithm, producing significantly overfitted models in most runs (see additional figures in electronic appendix). Even under the average population model, for which the results of TS were similar as described above, hACO performed nowhere near as well as in the original study. One reason for this might be the different objective function or implementation. However, we implemented hACO as described in the original study and also tried out the original objective function and implementation in previous testing, which resulted in worse performance compared to our own hACO. We thus argue that the difference in findings is due to the more complex population models and the fact that hACO generates a random starting model if the initial model does not fit the data well enough. In our case, 36% of the starting models with weak misspecification and 75.5% of the starting models with strong misspecification produced too low objective function values and thus resulted in a randomly sampled starting model. As we know that in most runs much of the model will be correctly specified, this is a strong disadvantage in finding the true model, as the probability that a randomly sampled and barely acceptable starting model will be close to our population model is likely very low. Our reasoning is backed up by the performance of TS, which is what hACO is built around. TS is only able to perform under weak misspecification, i.e. if we are already quite close to the correct model. However, if we start with a randomly sampled starting model in many runs, we will most likely be far away from the true model and not be able to advance our model only by the exploration of neighbour-to-neighbour moves utilized by TS. This may also highlight a flaw in the theory underlying hACO. It is meant to combine the best of ACO and TS; however, the largest benefit of ACO lies in its very extensive exploration by means of stochastic resampling of models. In hACO, on the other hand, the part used to explore the search space is TS. TS itself is, however, a greedy neighbour-to-neighbour-based algorithm

that excels at the end of a search, i.e. if we need to advance our model in small steps to the already near optimum. Here, hACO utilizes the concepts from ACO, which is, as discussed, better suited for a broad exploration. Our proposed hybrid algorithms GA-TS and PSO-TS were based on the exact opposite logic of hACO, which we thus also argue to be more in line with the underlying optimization theory. The better performance of our hybrids over hACO also backs up this reasoning.

The last algorithm to be investigated under realistic settings was the GA by Murohashi & Toyoda (2007). Looking at the most comparable setting of their and our study (a population size of 20 and a simple structure three factor model), our GA found the true model 90% of the time, while theirs found it 94%. Thus, the results overlap well. The slightly better performance can most likely be explained by the fact that Murohashi & Toyoda (2007) did not explore factor correlations, but always estimated them in all models, resulting in fewer different models producing the same data fit. However, it is also possible that their GA, which is a mixture of our GA and AGA, performs slightly better overall as it combines the benefits of our two more extreme GAs. Sadly, we are unable to compare findings regarding the other population structures. This is because, for their more complex population models, the reported metric values a run as successful if the estimated factor pattern is a subset of the true factor pattern. We find this to be a strong limitation as, for example, a final model in which only one parameter is estimated, which is also present in the true model, would be counted as a successful run. Essentially, all underparametrized versions of the true model would achieve this. This is very contradictory to our design, in which we try to advance possibly underparametrized solutions towards better ones. Lastly, our GA also took less time to converge, which is likely due to the absence of mutation and cross-over probabilities that slow down convergence (our AGA also utilized these probabilities and took about the same time).

All in all, there are very few studies that allow a realistic comparison with our findings. Although we give explanations for diverging findings, it is especially surprising that the hACO was by far the worst algorithm in our study and performed so well in Jing et al. (2022). For the TS and GA, our findings fit right into previous studies when taking different exploration strategies into account. Regarding SA and PSO, we were not able to compare our results to earlier literature.

4.2 Interpretation of Results

For TS, we found that its results are highly dependent on all conditions. It performed well under low misspecification of the starting model, large sample sizes, and, when sample sizes were sufficient, under the complex population model. Based on its design, it is no surprise that a bad starting model highly hinders its capability to find a good or the true model. Relying only on neighbour-to-neighbour movement translates to the fact that many more steps are needed to reach good solutions, and the probability of getting stuck in a local solution on this longer path is increased. With the latter consideration, we can also evaluate the influence of sample size: for low sample sizes, there is more noise in the data, and the true structure is less present. As TS has only the tabu-list as a (very rudimentary) mechanism to overcome local solutions, it is more likely to be misled by random noise in the data compared to more exploratory algorithms, which can better look ahead and farther away from the current possibly locally optimal solution. Like all algorithms, TS was not able to produce good results when sample sizes were below 1000 and the complex true model was supposed to be found. We address this as less of a problem of the algorithm but more of the complexity of the true model, for which low sample sizes are too low to clearly represent the true model as well as the objective function. We also argue that the latter gives one very important reason why TS performed significantly better than the other algorithms for sample sizes of 1000 and upwards. The BIC is better suited to compare small changes in models, especially non-nested ones. The fit indices, which were used for most of the other algorithms, are known to be dependent on various factors of a model and its data. Even if we tried to combine indices that mitigate each other's problems, these biases are still present. For example, the RMSEA is highly influenced by sample size, selecting less complex models when sample size increases. Even though the other indices do not show this property, the objective function will still push the solution into this direction. We see the most important backup of our reasoning in the performance of SA (which also used the BIC): SA was the worst-performing algorithm across conditions, and it even produced the second-largest Hamming-Distances under the complex true model. However, it completely outperformed the remaining algorithms for large sample sizes and under the complex population model with respect to the times it found the correct model. Comparing TS and SA, there is another concept they have in common in addition to the objective function: they both rely on neighbour-to-neighbour movement. We, however, do not see a reasoning that could explain why this simpler approach is better suited for finding the complex population model. A last argument adding to our reasoning is that BIC is known to be bad at

recovering the true model under small sample sizes (West et al., 2012). This can be clearly observed when looking at TS and SA's findings, as the proportion of runs in which the true model is found highly increases with N, also compared to the other algorithms.

SA was one of the two worst-performing algorithms across all conditions. It only identified the correct model in 50% of runs under the easiest conditions and produced large Hamming distances overall. It only produced good models under low misspecification of the average population model. However, it is difficult for neighbour-to-neighbour-based algorithms like SA to produce bad models when starting with a model close to the optimum. We see one reason for its overall poor performance in the possibly too low hyperparameter values for the number of runs before the algorithm cools down and the possibly too conservative convergence criterion. The former point can be easily understood by contrasting SA and TS: while TS evaluates all neighbours of a solution, SA typically only looks at a few of those, which hinders the exploratory power of the algorithm. As TS got stuck in many local solutions, SA is even more likely to do so. SA was the fastest algorithm overall, which serves as another indicator that the algorithm did not evaluate as many models as its competitors.

We consider GA the winner in our comparison because it found the correct model most often under the average population model and was also the most stable algorithm across sample sizes and misspecification levels of the starting model. Additionally, its convergence time was by far the shortest out of the population-based algorithms. We attribute its success to its two mechanisms, which represent both simple (stochastic) neighbour-to-neighbour hill-climbing (mutation) and a broad highly randomized exploration (crossover). In this way, our GA essentially combines aspects of algorithms like TS and PSO: while mutation is very similar to TS, where instead of evaluating all neighbours and choosing the best one, we stochastically propose neighbours and select only those that improve our solution, crossover (like implemented in our GA) essentially performs simplified PSO updating, where we do not explicitly push our solution into the direction of the swarm's best position but implicitly let children inherit qualities of their parents, which survived through previous generations due to their improved fitness. To better understand this, we can look at our AGA, where we again explicitly choose parents to produce offspring based on their current fitness value with respect to the population. By utilizing these two mechanisms, our GA can easily escape local solutions and explore the object set sufficiently as well as push especially late solutions in small distances to the correct model. Based on the arguments mentioned for TS, we also explain the lacking success of GA under the complex true model. We

argue that our fit index-based objective function is not suitable for model comparison in such a difficult scenario of the complex population model. Additionally, the cutoffs of the fit indices, which constitute important parameters of our objective function, were derived based on three-factor models, which closely resemble our average but not complex population model.

AGA did in our study not perform as well as its simpler counterpart, especially when considering the rate of found correct models and convergence times. It did however produce the best Hamming-Distances under the complex population structure. To explain these results, we resort to its theoretical derivation, which we described in section 2.2.3. AGA is designed to overcome locally optimal solutions, which, however, also hinders it in finding the global optimal solution. This is because we expect individuals in the population to become more similar near optimal solutions, whether they are local or global, as individuals close to any optimum are less likely to change or to leave the population and inherit their configuration to other individuals. Even though there exists a counter mechanism to overcome this problem for the global optimum, the algorithm will still have a tendency to come out of any optimal solution. The algorithm should thus be good at finding good local rather than the optimal solution, which is exactly what we saw under the complex population model, i.e. small Hamming-Distances. When observing mutation and crossover probabilities manually for a few selected runs, we found especially the former to be very high. The consequence of this is, that proposed child solutions were very frequently resampled to a high degree, resulting in a close to completely random search. We tried to minder this effect by drastically reducing the maximum mutation probability from the originally proposed 0.5 to 0.1, which however still resulted in very large model changes. We also see this to be the reason for its very long convergence times and why AGA was not able to perform under the average population model. The algorithms' flexibility induced by this mechanism, which allowed it to perform very well under the complex population model, was too high for our simpler population model. E.g. for the simplest condition only 8 steps are needed on average to reach the correct solution and with 48 parameters to be explored under the average population model and a mutation rate of 0.1, which was very often the case, almost 5 parameters are changed from the originally proposed child solution which should inherit qualities from its parents. Like this, it is no surprise that under the exact same simplest condition the average Hamming-Distance was 6.01. Its simpler counterpart was much better able to make fine tuned moves based on its fixed one parameter that is changed when mutating.

We considered PSO, like GA and TS, to be one of the overall more successful algo-

rithms. The reasons were the same as for the GA: high proportions of correctly found models under the average population model and low Hamming-Distances. However, compared to GA, PSO took much longer to converge. We explain this by the fact that the swarm particles fluctuate more when compared to individuals in GA. In the latter, individuals in the population do not change; they just get replaced by fitter individuals. In every generation of a GA, even if all the models produced by crossover and mutation were better than all individuals in the current population (which is highly unlikely), only a fraction of them would change. This is because the number of individuals undergoing these mechanisms is typically smaller than the population size. In our study, 14 solutions underwent crossover and mutation, while our sample size was 30. In PSO, this change is much more rapid. In every generation, every *gene* of every individual changes with a certain probability. Unlike GA, where only better proposed solutions are accepted, this change is always accepted, regardless of whether the solution is better or worse than any solution in the population. Despite the convergence time, we ascribe the similarity in the GAs and PSOs performance to the basis that allows the reasoning we just gave. PSO is seen in close proximity to GA (Kennedy & Eberhart, 1997), as, in both algorithms, solutions act dependent on other (better) solutions and inherit their qualities to some degree. This becomes even more apparent when comparing PSO to our AGA, which, ignoring the adaptivity of our AGA, shows striking similarities. Both algorithms modify solutions based on the fittest/ fitter individuals in the population (crossover in AGA and the influence by the fittest solution in PSO), parameter-wise stochasticity (mutation in AGA and multiplication by ϕ in PSO) and protect individual solutions based on their fitness (only replace if another solution is better in GA and orientation towards personal best as well as stochastic modification based on calculated fitness after velocity update in PSO). Thus, like GA, PSO also has the capability to both explore the search space thoroughly and make very minor changes to individuals, which we argue is the reason for its success.

The last non-hybrid algorithm we looked at was hACO. As we explained its results in the previous section, we will just refer to our explanation here.

Next, we propose two hybrid algorithms, based on the previous findings: GA-TS and PSO-TS. Overall, the algorithms showed a clear pattern: they performed better under the complex population model and worse under the average. Also, the findings of both algorithms are almost identical. We ascribe this pattern, again, to the objective functions. It is known that BIC is bad at finding the true model under smaller sample sizes (West et al., 2012), which is exactly what we observe for sample

sizes under 1000. Under the average population, the same is only found approximately half the time compared to that of the standalone GA and PSO for low sample sizes. As sample size increases, the hybrid versions still recover the true model less often but come closer to the standalone algorithms. Interestingly, this finding is not the same with respect to the Hamming-Distance. Here, the hybrid algorithms perform better for both high sample sizes and the average population model, as well as across almost all sample sizes under the complex true structure. This pattern matches our reasoning from above: BIC is not good at smaller sample sizes and is a lot better than fit indices under the more complex population structure, where fit indices are misled by their biased tendencies. The real advantage of the hybrids can be seen under the most complex settings, where initial misspecification is high and the true model is complex. In this condition, TS alone is hindered by the bad starting model, and GA/PSO are held back by their objective function, while the better starting models given to TS by GA and PSO allow the benefits of GAs and PSOs exploratory power in conjunction with the soundness of the BIC to really shine through.

4.3 Generalizability

Even though our study was designed to represent the practical research process, taking into account previous findings as realistically as possible, there are limitations we want and need to mention.

One of the largest pain points from a methodological point of view is the dependence of our findings on the objective functions. Even though we decided on what we believed to be the best choices for the respective algorithms based on our own and other previous research, the question remains what other functions could propose good alternatives. Especially interesting inside the scope of our study is the question of whether GA and PSO would be able to perform better if used with BIC. This would possibly also aid the hybrid algorithms as the search is conducted with the same criteria throughout its entire length. Also, we are not limited to BIC, which does not show the best low sample size performance, but can resort to other information criteria like Akaike-Information-Criterion (AIC) or its sample size adapted counterpart cAIC. With this, we could potentially aid low sample size performance as well as increase performance under more complex models. Overall, this limitation leads to the fact that our results should be seen through the lens of each combination of algorithm and its objective function.

The second-largest limitation of our study is most likely the rudimentary approach

to setting the algorithms' hyperparameters. As all algorithms are highly dependent on their hyperparameter values, it would have been good practice to introduce more objectivity into specifying them. Our approach relied on manual inspection of the algorithms' performance during the search (e.g. how often solutions were completely resampled in our AGA or how the temperature anneals as we move closer to our population's model's objective function in SA) for a few selected runs. A better alternative that could be used in conjunction with our approach would be cross-validation. For each hyperparameter, some values could be tried for a few runs in each condition and for each algorithm and checked, which leads to the best performance on unseen data from the same distribution. This is common practice in machine learning and is also done in psychometrics, however to a far lesser extent. We see one reason for this in the relatively low sample size frequently observed in social science studies, which limits the number of observations that can be held back from training and utilized for testing. Nonetheless, in our simulation, we had the possibility to overcome this limitation and look at the algorithms under their *best settings*. We decided not to utilize cross-validation due to limited temporal and computational resources.

We see the next factor hindering the generalizability of our findings in the choice of levels of our conditions. While the two population models are built around a solid understanding of the models that are actually fitted in practice, it would be interesting to see the algorithm's performance under simpler, especially unidimensional models, as these make up a large proportion of the models utilized by researchers. We decided not to include this level based on two reasons: we already had a lot of computations to conduct, as well as the known very good performance of TS and hACO under simpler models (Jing et al., 2022). However, to really call an algorithm superior to others, like, for example, the GA in our case, it should also be able to perform under simpler conditions. Additionally, it would have been interesting to see how the algorithms perform when used as purely exploratory techniques, i.e., when starting with completely randomized starting models. We see this, however, less as a limitation of our study, but more as an additional area that would be interesting to investigate. Our study should aid researchers who want to utilize the data that they have when their initial theory didn't match the same so that they can learn it.

Even though we are talking about SEM most of the time, in our study, we look at a rather restrictive type of SEMs. As described earlier, the framework SEM comprises a lot of different models like (multiple) regression, bi-factor and hierarchical models, or CFAs. In our study, we mostly relate to the latter type. Even though cross-loadings are not a part of classic CFA theory, their addition leaves our models farther away

from the mentioned instances. In the same manner, we focused only on cross-loadings and latent variable correlations, not recognizing residual correlations. Even though CFA is probably the most common special case and one of the key methods that led to the development of SEM, it would also be appropriate to term the models we evaluate *CFA-like* instead of SEM. We decided to stick with the SEM framework based on two reasons. First, SEM is really one of the drivers that empowered the use of cross-loadings and residual correlations. Second, we see our study as one of the starting points of research in this specific field and find CFAs the best starting point within SEM to explore techniques for specification search. This is mainly because CFA is both likely the most common special of SEM as well as a rather simple one. Trying out these algorithms on more complex models should be done when feasibility is demonstrated in the present simpler scenario.

The last point of criticism we want to mention is with respect to the algorithms and their specific implementation. Even though TS is straightforward and hACO was described in all detail by Jing et al. (2022), for the other algorithms, many possibilities exist on how to specify their mechanisms. For example, in SA, we could specify varying annealing functions, starting temperatures and neighbourhood definitions. However, especially in GAs, there exists a lot of variety (consider also that GA refers to a class rather than a specific algorithm). We tried to account for this diversity by comparing two versions of a GA. However, it is possible that we chose two ends of the spectrum that were too extreme: one being very simple with only the least amount of stochasticity in selection, crossover, and mutation and the other being very complex, where this stochasticity is not only present but also varies adaptively throughout the search. Looking at Murohashi & Toyoda (2007) implementation, it proposes a much better midway and standard version of the GA. As stated before, our approach of trying to compare as many algorithms as possible hindered the same to be specified in their *best version*. We decided on this approach because we believe it to be more fruitful to compare a lot of algorithms in relatively basic forms and work on further developing the most promising ones, like we did in combining GA and PSO with TS. Nonetheless, our findings are limited to the extent that differently implemented mechanisms in the algorithms allowed them to perform significantly differently.

4.4 Future Research

Both our findings and their limitations allow us to directly formulate implications for following research in the field. First and foremost, we advise further investigation

of objective functions. Our study indicated a need to further develop and try the algorithms under different functions. Specifically, we propose two additions. The first is to base the cut-offs of the fit indices used in our objective function not on fixed values from singular simulation studies, but on *dynamic cut-offs* (McNeish & Wolf, 2021). These are values based on a similar simulation to that of Hu & Bentler (1999), however, the model used to calculate them is the actual fitted model by the researcher. This takes into account that the model as well as the sample size influences how fit indices behave, e.g. an RMSEA cut-off of 0.06 might be a good choice for an orthogonal three-factor model but too liberal for a unidimensional model. With this approach, our objective function will be better tailored to the model at hand. The second addition we would like to propose comes from Schroeders et al. (2022). They not only penalize model complexity based on the number of parameters but also on the magnitude of factor loadings. In their objective function, factor loadings that become smaller than 0.3 lead to an increasing penalization. Even though we think this is too high of a value, especially for cross-loadings, the general approach allows us to better lead the search in a direction of models that are more likely to be deemed acceptable for researchers. Obviously, many more ways to refine objective functions exist, and we think that exactly this refinement is needed for the algorithms to perform better in practice and leave less room for error for researchers who want to utilize these algorithms to improve their models.

Another direction we deem to be of high importance to make these algorithms applicable for applied research lies in comparing different versions of the algorithms and giving specific guidelines on how to specify hyperparameters. We think that the GA is the overall most promising algorithm, and trying different variants of the same should be a vital part of further research in the field. Also, in a study focusing more on one algorithm, the correct hyperparameters under different true models could be investigated with intensive cross-validation. For this, we advise using our two population models, as well as a more simple, unidimensional one. With this, a broad range of models is covered, and applied researchers would be able to choose the hyperparameters best suited for their specific model.

Of high interest at this point is also the question for which types of SEM specification search via metaheuristic algorithms are feasible. Like others (e.g. Murohashi & Toyoda, 2007 or G. Marcoulides & Leite, 2014), we demonstrated its feasibility in the area of and around CFA models. To recommend the method for SEM as a whole, however, other types of models like the ones previously mentioned should be considered. This will allow researchers to evaluate if this rather time-intensive approach is

useful for their specific scenario.

One last direction we would like to point to is the implementation of the algorithms. We relied on the rather superficial approach of fitting all models separately, meaning that each model was fit like a completely new model using the helper functions from K. Marcoulides & Falk (2018) and lavaan. A much more sophisticated and, more importantly, time-saving approach would be to reuse computations from previous iterations. For example, if one of two orthogonal factors, which don't share any indicators, changes (e.g. gains or loses an indicator), there is no need to estimate the factor loadings and residuals again for the other factor and its indicators. Although investing work into this implementation should only be done when the feasibility of the approach has been analysed in more detail, we consider a good implementation to be of high relevance in order to make this approach more attractive for researchers to utilize.

4.5 Conclusion

Before specific recommendations can be made to researchers on how and when to utilise metaheuristic algorithms to improve their models, we think that a few additional but small steps are in order. This is to truly maximise the potential of the idea and to determine in which scenarios which algorithm with what hyperparameters is the best choice. On the path to this goal, our study and implemented algorithms available in the R package MetaSS stand as an important milestone, which should inform and guide future research in the field. If these few extra steps are taken, metaheuristics will be a fruitful addition to researchers' toolkits, helping them to extract more information from their data.

A Electronic appendix

Data, code and figures are provided in electronic form under `https://gitlab.lrz.de/KarikSiemund/specification-search-as-cop-simulation` and `https://gitlab.lrz.de/KarikSiemund/MetaSS`.

References

Ben-Ameur, W. (2004). Computing the initial temperature of simulated annealing. *Computational Optimization and Applications*, *29*, 369-385. doi: https://doi.org/10.1023/B:COAP.0000044187.23143.bd

Bollen, K. A. (1989). *Structural equations with latent variables*. John Wiley & Sons. doi: https://doi.org/10.1002/9781118619179

Chou, J.-S., & Yang, J.-G. (2013). Evolutionary optimization of model specification searches between project management knowledge and construction engineering performance. *Expert Systems with Applications*, *40*(11), 4414-4426. doi: https://doi.org/10.1016/j.eswa.2013.01.049

Drezner, Z., & Marcoulides, G. (1999). Using simulated annealing for model selection in multipleregression analysis. *Multiple Linear Regression Viewpoints*, *25*(2), 1-4.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, *13*(5), 533-549. doi: https://doi.org/10.1016/0305-0548(86)90048-1

Goretzko, D., Siemund, K., & Sterner, P. (in press). Misspecified measurement models: A review of the current practice. *Educational and Psychological Measurement*.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. University of Michigan Press. doi: https://doi.org/10.7551/mitpress/1090.001.0001

Holzinger, K. J., & Swineford, F. (1939). A study in factor analysis: The stability of a bi-factor solution. *Supplementary Educational Monographs*, *48*, xi+91.

Hu, L., & Bentler, P. M. (1999). Cutoff criteria for fit indexes in covariance structure analysis: Conventional criteria versus new alternatives. *Structural Equation Modeling: A Multidisciplinary Journal*, *6*(1), 1-55. doi: https://doi.org/10.1080/10705519909540118

Jing, Z., Kuang, H., Leite, W. L., Marcoulides, K., & Fisk, C. L. (2022). Model specification searches in structural equation modeling with a hybrid ant colony optimization algorithm. *Structural Equation Modeling: A Multidisciplinary Journal*, *29*(5), 655-666. doi: https://doi.org/10.1080/10705511.2021.2020119

Katoch, S., Chauhan, S. S., & Kumar, V. (2021). A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications*, *80*, 8091-8126. doi: https://doi.org/10.1007/s11042-020-10139-6

Kennedy, J., & Eberhart, R. (1997, October 12-15). A discrete binary version of the particle swarm algorithm [Paper Presentation]. In *Ieee international conference on systems, man, and cybernetics. computational cybernetics and simulation.* Orlando, FL, United States. doi: https://doi.org/10.1109/ICSMC.1997.637339

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671-680. doi: https://doi.org/d10.1126/science.220.4598.671

Kuechenhoff, H. (2021/2022). [Lecture notes on statistical modelling]. Department of Statistics, Ludwig-Maximilians-University Munich.

Landis, R., Edwards, B., & Cortina, J. (2009). On the practice of allowing correlated residuals among indicators in structural equation models. In C. E. Lance & R. J. Vandenberg (Eds.), *Statistical and methodological myths and urban legends* (p. 193-214). Routledge/Taylor & Francis Group. doi: https://doi.org/10.4324/9780203867266

Lang, M., Bischl, B., & Surmann, D. (2017). batchtools: Tools for r to work on batch systems. *Journal of Open Source Software*, *2*(10), 135.

Long, J. S. (1983). *Confirmatory factor analysis: A preface to lisrel.* Sage publications.

Marcoulides, G. (2009). *Conducting specification searchers in sem using a ruin-and-recreate principle.* San Francisco, CA, United States.

Marcoulides, G., & Drezner, Z. (2001). Specification searches in structural equation modeling with a genetic algorithm. In C. E. Lance & R. J. Vandenberg (Eds.), *New developments and techniques in structural equation modeling* (p. 267-288). Lawrence Erlbaum Associates Publishers.

Marcoulides, G., & Drezner, Z. (2003). Model specification searches using ant colony optimization algorithms. *Structural Equation Modeling: A Multidisciplinary Journal*, *10*(1), 154-164. doi: https://doi.org/10.1207/S15328007SEM1001_8

Marcoulides, G., Drezner, Z., & Schumacker, R. E. (1998). Model specification searches in structural equation modeling using tabu search. *Structural Equation Modeling: A Multidisciplinary Journal*, *5*(4), 365-376. doi: https://doi.org/10.1080/10705519809540112

Marcoulides, G., & Leite, W. (2014). Exploratory data mining algorithms for conducting searches in structural equation modeling: A comparison of some fit criteria. In J. J. McArdle & G. Ritschard (Eds.), *Contemporary issues in exploratory data mining in the behavioral sciences* (p. 150–171). Routledge/Taylor & Francis Group.

Marcoulides, K., & Falk, C. F. (2018). Model specification searches in structural equation modeling with r. *Structural Equation Modeling: A Multidisciplinary Journal*, *25*(3), 484-491.

McNeish, D., & Wolf, M. G. (2021). Dynamic fit index cutoffs for confirmatory factor analysis models. *Psychological Methods*. doi: https://doi.org/10.1037/met0000425

MD, H., Romeo, F., & Sangiovanni-Vincentelli, A. (1986, November). An efficient general cooling schedule for simulated annealing [Paper Presentation]. In *Proceedings of ieee international conference on computer aided design.* Santa Clara, CA, United States.

Mueller, R. O. (1997). Structural equation modeling: Back to basics. *Structural Equation Modeling: A Multidisciplinary Journal*, *4*(4), 353-369.

Murohashi, H., & Toyoda, H. (2007). Model specification search using a genetic algorithm with factor reordering for a simple structure factor analysis model. *Japanese Psychological Research*, *49*(3), 179-191.

Preacher, K. (2016). *Derivation of the ml discrepancy function from likelihoods.* `http://quantpsy.org/misc/discrepancy_derivation_022316.pdf`.

R Core Team. (2016). *R: A language and environment for statistical computing* [Programming language]. Vienna, Austria.

Rosseel, Y. (2012). lavaan: An R package for structural equation modeling. *Journal of Statistical Softwacoare*, *48*(2), 1-36. doi: https://doi.org/10.18637/jss.v048.i02

Rozenberg, G., Bäck, T., & Kok, J. N. (2012). *Handbook of natural computing*. Springer.

Schroeders, U., Scharf, F., & Olaru, G. (2022). *Model specification searches in structural equation modeling using bee swarm optimization*. PsyArXiv. doi: https://doi.org/10.31234/osf.io/sm5zt

Spearman, C. (1904). General intelligence objectively determined and measured. *American Journal of Psychology*, *15*(2), 201–293. doi: https://doi.org/10.2307/1412107

Srinivas, M., & Patnaik, L. M. (1994). Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, *24*(4), 656-667.

West, S., Taylor, A., & Wu, W. (2012). Model fit and model selection in structural equation modeling. In H. Hoyle (Ed.), *Handbook of structural equation modeling* (p. 209-231). The Guilford Press.

Declaration of authorship

I hereby declare that the report submitted is my own unaided work. All direct or indirect sources used are acknowledged as references. I am aware that the student paper in digital form can be examined for the use of unauthorized aid and in order to determine whether the report as a whole or parts incorporated in it may be deemed as plagiarism. For the comparison of my work with existing sources I agree that it can be entered in a database where it shall also remain after examination, to enable comparison with future works submitted. Further rights of reproduction and usage, however, are not granted here. This paper was not previously presented to another examination board and has not been published.

Location, date

Name