PSYCHOMETRIKA—VOL. 72, NO. 3, 377–391
SEPTEMBER 2007
DOI: 10.1007/S11336-007-9000-9

# THE LOCAL MINIMA PROBLEM IN HIERARCHICAL CLASSES ANALYSIS: AN EVALUATION OF A SIMULATED ANNEALING ALGORITHM AND VARIOUS MULTISTART PROCEDURES

EVA CEULEMANS AND IVEN VAN MECHELEN

KATHOLIEKE UNIVERSITEIT LEUVEN

IWIN LEENEN

UNIVERSIDAD COMPLUTENSE DE MADRID

Hierarchical classes models are quasi-order retaining Boolean decomposition models for $N$-way $N$-mode binary data. To fit these models to data, rationally started alternating least squares (or, equivalently, alternating least absolute deviations) algorithms have been proposed. Extensive simulation studies showed that these algorithms succeed quite well in recovering the underlying truth but frequently end in a local minimum. In this paper we evaluate whether or not this local minimum problem can be mitigated by means of two common strategies for avoiding local minima in combinatorial data analysis: simulated annealing (SA) and use of a multistart procedure. In particular, we propose a generic SA algorithm for hierarchical classes analysis and three different types of random starts. The effectiveness of the SA algorithm and the random starts is evaluated by reanalyzing data sets of previous simulation studies. The reported results support the use of the proposed SA algorithm in combination with a random multistart procedure, regardless of the properties of the data set under study.

Key words: hierarchical classes, HICLAS, Tucker3-HICLAS, multiway clustering, simulated annealing.

## 1. Introduction

The family of hierarchical classes models—HICLAS (De Boeck & Rosenberg, 1988; Van Mechelen, De Boeck, & Rosenberg, 1995), INDCLAS (Leenen, Van Mechelen, De Boeck, & Rosenberg, 1999), Tucker3-HICLAS (Ceulemans, Van Mechelen, & Leenen, 2003), and Tucker2-HICLAS (Ceulemans & Van Mechelen, 2004)—is a collection of structural models for $N$-way $N$-mode binary data. This type of data often occurs in psychology. For an example of two-way two-mode binary data, one may think of person by item solve/not solve data. Consumer byproduct by time point select/not select data constitute an example of three-way three-mode binary data. All hierarchical classes models include a Boolean decomposition of the $N$-way $N$-mode binary data set into up to $n$ ($n \leq N$) binary component matrices, which each reduce one of the $N$ modes to a few binary components, and a linking structure among the $n$ sets of components (and, if applicable, the elements of the unreduced modes). As such, hierarchical classes analysis of an $N$-way $N$-mode binary data set may uncover the structural mechanism underlying the data. For instance, hierarchical classes analysis may reveal the latent choice requisites that underlie consumer byproduct select/not select data, with a consumer selecting those products that satisfy all of his requisites (Van Mechelen & Van Damme, 1994). Furthermore, all hierarchical classes models include hierarchically organized classifications of the elements of the $n$ reduced modes, implying the representation of if–then type of relations among the elements of

Eva Ceulemans is a post-doctoral fellow of the Fund for Scientific Research Flanders (Belgium). Iwin Leenen is a post-doctoral researcher of the Spanish Ministerio de Educación y Ciencia (programa Ramón y Cajal). The research reported in this paper was partially supported by the Research Council of K.U. Leuven (GOA/05/04).

Requests for reprints should be sent to Eva Ceulemans, Department of Psychology, Tiensestraat 102, B-3000 Leuven, Belgium. E-mail: eva.ceulemans@psy.kuleuven.be



© 2007 The Psychometric Society

these modes. Whereas the simultaneous classifications meet the substantive need of personality psychologists searching for triple typologies of persons, situations, and behaviors in binary person by situation by behavior display/not display data (Vansteelandt & Van Mechelen, 1998, 2006), the if–then type relations are of key relevance in the study of person perception (Gara & Rosenberg, 1990), in psychiatric diagnosis research (Van Mechelen & De Boeck, 1989), and in differential emotion psychology (Kuppens, Van Mechelen, Smits, De Boeck, & Ceulemans, 2007).

Because of the Boolean nature of the hierarchical classes models, fitting these models to data implies solving a complex combinatorial optimization problem. To solve this problem, each of the models has been introduced along with an associated alternating least squares (or, equivalently, alternating least absolute deviations) algorithm. The performances of these algorithms have been evaluated in extensive simulation studies (Leenen & Van Mechelen, 2001; Leenen et al., 1999; Ceulemans et al., 2003; Ceulemans & Van Mechelen, 2004). The results of these studies showed that the algorithms in question, in conjunction with a rational start, succeed quite well in recovering the underlying truth but frequently end in a local minimum. The latter is problematic because it implies that when applying hierarchical classes analysis in practice, one may often obtain and interpret a good but suboptimal solution.

Local minima constitute an ubiquitous challenge for combinatorial data analysis techniques (see, e.g., Selim & Ismail, 1984; Hubert, Arabie, & Hesson-McInnis, 1992). Not surprisingly, a great deal of research effort has therefore been put into the search for strategies for avoiding them. In particular, many authors advocate the following two strategies: using local search techniques such as simulated annealing, genetic algorithms, or tabu search (see, e.g., Al-Sultan & Khan, 1996; Brusco, 2001; Murillo, Vera, & Heiser, 2005) and/or implementing multistart procedures with a number of good starting values or a very large number of random starting values (see, e.g., Hand & Krzanowski, 2005; Milligan, 1980; Steinley, 2003). In this paper we will therefore investigate to which extent the local minima problem in hierarchical classes analysis can be mitigated by means of simulated annealing (SA) and/or by using various multistart procedures.

The remainder of this paper is organized as follows: In section 2 the HICLAS and Tucker3-HICLAS models are briefly recapitulated. In section 3 the general scheme of the original alternating least squares algorithms for hierarchical classes analysis is described, together with the new, generic SA algorithm for hierarchical classes analysis. In section 4 we discuss four types of starting procedures for hierarchical classes analysis. In section 5 we evaluate the effectiveness of SA and multistart procedures in hierarchical classes analysis by reanalyzing simulated data sets reported by Leenen and Van Mechelen (2001) and Ceulemans et al. (2003). Section 6 contains some concluding remarks.

## 2. The HICLAS and Tucker3-HICLAS Models

### *2.1. The HICLAS Model*

The HICLAS model (De Boeck & Rosenberg, 1988) approximates an $I$ (objects) $\times$ $J$ (attributes) binary data matrix $\mathbf{D}$ by an $I \times J$ binary reconstructed data matrix $\mathbf{M}$, which can be further decomposed into an $I \times R$ binary matrix $\mathbf{A}$ and a $J \times R$ binary matrix $\mathbf{B}$, where $R$ denotes the rank of the model. $\mathbf{A}$ and $\mathbf{B}$ define $R$ binary object and attribute components and are therefore called object component matrix and attribute component matrix, respectively.

The HICLAS model represents two types of structural relations in $\mathbf{M}$.

*Association.* The association relation is the binary relation between the objects and attributes as defined by the 1-entries in $\mathbf{M}$. The HICLAS model represents the association relation

by the following rule:

$$\mathbf{M} = \mathbf{A} \otimes \mathbf{B}', \tag{1}$$

or, equivalently,

$$m_{ij} = \bigoplus_{r=1}^{R} a_{ir} b_{jr}, \tag{2}$$

where $\otimes$ and $\oplus$, respectively, denote the Boolean matrix product and the Boolean sum.

*Quasi-order.* A quasi-order $\preceq$ is defined on each mode of $\mathbf{M}$. More specifically, object $i \preceq$ object $i'$ iff $S^i \subseteq S^{i'}$, where $S^i$ and $S^{i'}$ denote the sets of attributes to which $i$ and $i'$ are associated in $\mathbf{M}$. Similarly, attribute $j \preceq$ attribute $j'$ iff $S^j \subseteq S^{j'}$, where $S^j$ and $S^{j'}$ denote the sets of objects to which $j$ and $j'$ are associated in $\mathbf{M}$. The HICLAS model represents the object and attribute quasi-order relations in terms of subset–superset relations among the object and attribute component patterns, respectively: object $i \preceq$ object $i'$ iff $\mathbf{a}_{i.} \leq \mathbf{a}_{i'.}$ and attribute $j \preceq$ attribute $j'$ iff $\mathbf{b}_{j.} \leq \mathbf{b}_{j'.}$. Note that the representation of the object and attribute quasi-order relation yields hierarchically organized classifications of the two modes. In particular, two elements belong to the same class iff they have identical component patterns. Furthermore, an element (class) is hierarchically below another element (class) iff the component pattern of the first is a proper subset of the component pattern of the latter.

### 2.2. *The Tucker3-HICLAS Model*

The Tucker3-HICLAS model (Ceulemans et al., 2003) approximates an $I$ (objects) $\times$ $J$ (attributes) $\times$ $K$ (sources) binary data array $\underline{\mathbf{D}}$ by an $I \times J \times K$ binary reconstructed data array $\underline{\mathbf{M}}$, which can be further decomposed into an $I \times P$ binary object component matrix $\mathbf{A}$, a $J \times Q$ binary attribute component matrix $\mathbf{B}$, a $K \times R$ binary source component matrix $\mathbf{C}$, and a $P \times Q \times R$ binary array $\underline{\mathbf{G}}$, where $(P, Q, R)$ denotes the rank of the model. $\underline{\mathbf{G}}$ is called core array because it defines a ternary linking structure among the three sets of components.

Like the HICLAS model, the Tucker3-HICLAS model represents the association and quasi-order relations in $\underline{\mathbf{M}}$.

*Association.* The Tucker3-HICLAS model represents the association relation among the objects, attributes, and sources in $\underline{\mathbf{M}}$ by the following rule:

$$m_{ijk} = \bigoplus_{p=1}^{P} \bigoplus_{q=1}^{Q} \bigoplus_{r=1}^{R} a_{ip} b_{jq} c_{kr} g_{pqr}. \tag{3}$$

This rule implies that an object $i$, an attribute $j$, and a source $k$ are associated in $\underline{\mathbf{M}}$ iff an object component $p$, an attribute component $q$, and a source component $r$ exist to which $i$, $j$, and $k$, respectively, belong and that are linked in $\underline{\mathbf{G}}$ (i.e., the corresponding core entry $g_{pqr}$ equals 1).

*Quasi-order.* A quasi-order relation $\preceq$ is defined on each mode of $\underline{\mathbf{M}}$. In particular, element $x \preceq$ element $y$ iff $S^x \subseteq S^y$, where $S^x$ (resp., $S^y$) denotes the set of pairs of elements of the other two modes $x$ (resp., $y$), is associated with in $\underline{\mathbf{M}}$. The Tucker3-HICLAS model represents the quasi-order relations among the objects, attributes, and sources in terms of subset–superset relations among the component patterns in $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$, respectively.

## 3. Algorithms for Hierarchical Classes Analysis

In this section we first discuss the general scheme of the original alternating least squares hierarchical classes algorithms. Subsequently, we propose a generic simulated annealing algorithm for hierarchical classes analysis. Finally, we comment on runtime differences between both types of algorithms.

### *3.1. Scheme of Original Alternating Least Squares Hierarchical Classes Algorithms*

Given an $I_1 \times I_2 \times \cdots \times I_N$ binary data array $\underline{\mathbf{D}}$, a specific type of hierarchical classes model, and a rank, the corresponding alternating least squares (ALS) hierarchical classes algorithm searches, by means of two routines, a same-sized binary reconstructed data array $\underline{\mathbf{M}}$ that has a minimal value on the least squares (or, equivalently, least absolute deviations) loss function

$$L = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} (d_{i_1 i_2 \ldots i_N} - m_{i_1 i_2 \ldots i_N})^2 \tag{4}$$

and that can be further decomposed into a hierarchical classes model of the specified type and rank. The first routine is an alternating least squares routine, in which, given a starting configuration for all but one of the parameter sets requiring estimation, each of the component matrices and, if applicable, the core array is reestimated conditionally upon all the others by means of Boolean regression (Leenen & Van Mechelen, 1998); this routine continues until no updating of a component matrix or core array further decreases the loss function (4). The second routine applies a closure operation to the output of the first routine; in particular, the closure operation consists of changing each zero-entry in the obtained component matrices to one if this change does not alter $\underline{\mathbf{M}}$. It has been shown that this closure operation is a sufficient condition for a correct representation of the quasi-order relations (Van Mechelen et al., 1995).

### *3.2. A Generic Simulated Annealing Algorithm for Hierarchical Classes Analysis*

Simulated annealing (SA), based on an analogy to a metallurgical cooling process, is a local search technique that is often used to solve problems of combinatorial data analysis such as $K$-means clustering (see, e.g., Al-Sultan & Khan, 1996; Klein & Dubes, 1989) and unidimensional scaling (see, e.g., Brusco & Stahl, 2000; Murillo et al., 2005). Given a possible solution to a combinatorial optimization problem—the current solution—an SA algorithm generates a new solution—the trial solution—by randomly changing one or more parameter values of the current solution. If the trial solution has a better loss function value than the current solution, the trial solution is accepted, that is, the current solution is replaced by the trial solution. However, a more important feature of the SA algorithms, which makes them less prone to getting stuck in local minima, is that trial solutions with a worse loss function value are accepted with a probability that is gradually decreased throughout the algorithm. In particular, the probability $p_{\text{acc}}$ of accepting a trial solution that is inferior to the current solution is given by

$$p_{\text{acc}} = \exp\left(\frac{L_{\text{current}} - L_{\text{trial}}}{T_{\text{current}}}\right), \tag{5}$$

where $L_{\text{current}}$ and $L_{\text{trial}}$ are the loss function values of the current and trial solutions, respectively, and $T_{\text{current}}$ indicates the 'temperature' of the algorithmic process, with $T_{\text{current}} > 0$. As this temperature $T_{\text{current}}$ is slowly decreased during the algorithm, it can indeed be derived from (5) that in the early stages of the algorithm the probability of accepting a trial solution that is worse than the current solution is much higher than in the final stages.

Initialize($S_{\text{current}}$, $L_{\text{current}}$, $T_{\text{initial}}$, $CL$, $L_{\text{best}}$, $L_{\text{previous}}$);
$T_{\text{current}} := T_{\text{initial}}$;
**repeat**
  $i_{\text{gen}} := 0$;
  $i_{\text{acc}} := 0$;
  **while** ($i_{\text{gen}} < CL$) and ($i_{\text{acc}} < (.1 * CL)$) **do**
    $i_{\text{gen}} := i_{\text{gen}} + 1$;
    generate $S_{\text{trial}}$ and associated $L_{\text{trial}}$;
    draw $h$ from $U(0, 1)$
    **if** ($L_{\text{trial}} < L_{\text{current}}$) or ($h < \exp(\frac{L_{\text{current}} - L_{\text{trial}}}{T_{\text{current}}})$) **then**
      **if** $L_{\text{trial}} < L_{\text{best}}$ **then**
        $S_{\text{best}} := S_{\text{trial}}$;
        $L_{\text{best}} := L_{\text{trial}}$
      **end if**
      $S_{\text{current}} := S_{\text{trial}}$;
      $L_{\text{current}} := L_{\text{trial}}$;
      $i_{\text{acc}} := i_{\text{acc}} + 1$
    **end if**
  **end while**
  $T_{\text{current}} := \alpha * T_{\text{current}}$;
  **if** $L_{\text{current}} = L_{\text{previous}}$ **then**
    $i_{\text{id}} := i_{\text{id}} + 1$
  **else**
    $i_{\text{id}} := 1$;
    $L_{\text{previous}} := L_{\text{current}}$
  **end if**
**until** ($T_{\text{current}} \leq T_{\text{stop}}$) or ($i_{\text{id}} = \max_{i_{\text{id}}}$);
apply ALS routine to $S_{\text{best}}$;
return $S_{\text{best}}$;

ALGORITHM 1.
The SA algorithm for hierarchical classes analysis.

As such, the core of the generic SA algorithm for hierarchical classes analysis, of which the pseudo code and associated notation is presented in Algorithm 1 and Table 1, respectively, consists of generating chains of trial solutions. In particular, given the current hierarchical classes solution $S_{\text{current}}$, the generation mechanism of the trial hierarchical classes solution $S_{\text{trial}}$ consists of altering the value of one randomly chosen parameter of $S_{\text{current}}$, where each parameter has an equal probability of being chosen. Subsequently, $S_{\text{current}}$ is replaced by $S_{\text{trial}}$ if $L_{\text{trial}}$ is lower, and thus better, than $L_{\text{current}}$; otherwise, $S_{\text{trial}}$ is accepted with probability $p_{\text{acc}}$, as defined by (5). Moreover, it is also checked whether the best encountered hierarchical classes solution $S_{\text{best}}$ is to be replaced by $S_{\text{trial}}$, which is the case if $L_{\text{trial}} < L_{\text{best}}$; note that $S_{\text{best}}$ has to be updated during the chain, because the final $S_{\text{current}}$ of a chain may be worse than an intermediate $S_{\text{current}}$. A chain is considered complete if at the current temperature $T_{\text{current}}$ $CL$ trial solutions have been generated or $.1 * CL$ trial solutions have been accepted, the latter implying that at the current temperature the solution space has been explored sufficiently (see, e.g., Brusco & Stahl, 2000); subsequently, the temperature $T_{\text{current}}$ is decreased by multiplying it by the cooling factor $\alpha$. The generation of new chains of trial solutions stops if $T_{\text{current}}$ is smaller than the stop temperature $T_{\text{stop}}$ or if $\max_{i_{\text{id}}}$ subsequent chains had a final current solution with an identical loss function value, the latter suggesting that on the basis of that current solution $S_{\text{current}}$ it is (almost) impossible to

TABLE 1.
Notation for the generic SA algorithm for hierarchical classes analysis.

| Label | Indicates |
|---|---|
| $S_{\text{current}}$, $S_{\text{trial}}$, $S_{\text{best}}$ | The bundle matrices and, if applicable, the core array of the current, trial, and best encountered hierarchical classes solution, respectively. |
| $L_{\text{current}}$, $L_{\text{trial}}$, $L_{\text{best}}$ | The loss function value of the current, trial, and best encountered hierarchical classes solution, respectively. |
| $T_{\text{current}}$, $T_{\text{initial}}$, $T_{\text{stop}}$ | The current, initial, and stop temperature, respectively. |
| $\alpha$ | The cooling factor by which $T_{\text{current}}$ is multiplied to reduce the temperature, $0 < \alpha < 1$. |
| $CL$ | The chain length, that is, the maximum number of trial solutions generated at each temperature. |
| $i_{\text{gen}}$ | The number of trial solutions that have already been generated in the current chain. |
| $i_{\text{acc}}$ | The number of trial solutions that have already been accepted in the current chain. |
| $L_{\text{previous}}$ | The loss function value of the final current solution of the previous chain. |
| $i_{\text{id}}$ | The number of subsequent chains of which the final current solutions had an identical loss function value. |
| $\max_{i_{\text{id}}}$ | The maximum number of subsequent chains in which the loss function value of the final current solution remains unchanged. |

arrive at better trial solutions. Finally, the SA algorithm executes a post-processing routine, which consists of running the ALS algorithm with $S_{\text{best}}$ as starting configuration. This post-processing routine is included to ensure that the resulting $S_{\text{best}}$ is at least a local minimum (see, e.g., Aarts & Lenstra, 1997; Murillo et al., 2005). For instance, in our simulation studies reported in section 5, this routine further improved the retained $S_{\text{best}}$ in 7% of the pseudo random HICLAS cases and in 3.2% of the pseudo random Tucker3-HICLAS cases. Once the post-processing routine is completed, $S_{\text{best}}$ is returned.

From the above description it is clear that the generic SA algorithm for hierarchical classes analysis requires the specification of $T_{\text{stop}}$, $\alpha$, and $\max_{i_{\text{id}}}$, and the initialization of $S_{\text{current}}$, $L_{\text{current}}$, $T_{\text{initial}}$, $CL$, $L_{\text{best}}$, and $L_{\text{previous}}$. On the basis of a few pilot studies, the stop temperature $T_{\text{stop}}$, the cooling factor $\alpha$, and the maximum number $\max_{i_{\text{id}}}$ of subsequent chains with equal final $L_{\text{current}}$ were set to .000001, .975, and 50, respectively. With respect to the initialization of $S_{\text{current}}$, $L_{\text{current}}$, $T_{\text{initial}}$, $CL$, $L_{\text{best}}$, and $L_{\text{previous}}$, the initial current hierarchical classes solution $S_{\text{current}}$ with associated loss function value $L_{\text{current}}$ can be chosen in many different ways. In this paper we will consider four different initialization procedures that are described in the next section. It is important to note that the SA hierarchical classes algorithm differs from the ALS algorithm in that a starting configuration is needed for all the parameter sets requiring estimation. With respect to the initial temperature $T_{\text{initial}}$, many authors claim that a good initial temperature results in an average acceptance probability of worse trial solutions of .8. A common method for estimating $T_{\text{initial}}$ (see, e.g., Murillo et al., 2005) consists of generating one chain of trial solutions in which all trial solutions are accepted irrespective of their loss function values. While generating

this chain, one records how many trial solutions had a worse loss function value than the respective current solution and how much worse these trial solutions were, that is, $L_{\text{current}} - L_{\text{trial}}$. Subsequently, $T_{\text{initial}}$ is computed as

$$T_{\text{initial}} = \frac{\text{average } (L_{\text{current}} - L_{\text{trial}})\text{-value of the worse trial solutions}}{\ln(.8)}. \tag{6}$$

Regarding the chain length *CL*, like many authors (see, e.g., Brusco & Stahl, 2000), we decided to let this value depend on the complexity of the problem at hand. More specifically, for HICLAS analyses, *CL* was set to $(I + J) \cdot 2^R$, whereas for Tucker3-HICLAS analyses, *CL* was set to $I \cdot 2^P + J \cdot 2^Q + K \cdot 2^R$. Finally, the initial values of the loss function value $L_{\text{best}}$ of the best encountered hierarchical classes solution and the loss function value $L_{\text{previous}}$ of the final current solution of the previous chain consisted of the number of data cells.

### *3.3. Runtime Differences Between the Generic ALS and SA Algorithms for Hierarchical Classes Analysis*

Based on the above descriptions of the generic ALS and SA algorithms for hierarchical classes analysis, one can formulate the hypothesis that the computational load implied by an SA analysis of a specific data set will probably be much larger than that of an ALS analysis of the same data set. This hypothesis is supported by time results for other techniques. For instance, Al-Sultan and Khan (1996) report that the computation time of their SA algorithm for *K*-means cluster analysis is up to 5000 times longer than that of their ALS algorithm. Of course, one should not interpret the exact time difference too strictly, as one can never be entirely sure that the implementation of an SA and an ALS algorithm for solving a specific data analytic problem is equally efficient.

## 4. Starting Procedures for Hierarchical Classes Analysis

In this paper we wish to evaluate whether the local minima problem in hierarchical classes analysis can be mitigated by means of a multistart procedure. In general, a multistart procedure implies rerunning an algorithm from a user-specified number of different starting configurations and retaining the best solution only. For such a multistart procedure, one has to decide how the different starting configurations are generated. In this paper we will consider three possibilities: truly random, pseudo random, and smart random. Before describing these three types of random starts in detail, we will first recapitulate the rational starting configuration for HICLAS and Tucker3-HICLAS analysis as proposed in the original papers.

*Rational.* In HICLAS analysis, a rational rank *R* starting configuration for the component matrix **A** is obtained by constructing *J* candidates $\mathbf{A}^{\tilde{j}}$ $(\tilde{j} = 1, \ldots, J)$, where the first $R - 1$ components of $\mathbf{A}^{\tilde{j}}$ are the components of the component matrix **A** of rank $R - 1$—as obtained via the rationally started ALS algorithm—and the *R*th component is the data vector $\mathbf{d}_{\cdot\tilde{j}}$ (Leenen & Van Mechelen, 2001). Out of these *J* candidates, the candidate is selected for which

$$\sum_{i=1}^{I}\sum_{j=1}^{J}\left(d_{ij} - \bigoplus_{r=1}^{R} a_{ir}^{\tilde{j}} b_{jr}^{\tilde{j}}\right)^2 \tag{7}$$

is minimal, $\mathbf{B}^{\tilde{j}}$ being the conditional estimate for **B** given $\mathbf{A}^{\tilde{j}}$. A rational rank *R* starting configuration for the component matrix **B** can be obtained similarly. In Tucker3-HICLAS analysis, given that the Tucker3-HICLAS model is a constrained HICLAS model (see Ceulemans & Van

Mechelen, 2005), a rational starting configuration for the component matrices **A**, **B**, and **C** is obtained by applying rationally started ALS HICLAS analyses to the matricized $I \times JK$, $J \times IK$, and $K \times IJ$ data array in ranks $P$, $Q$, $R$, respectively (Ceulemans et al., 2003); conditional upon the rational starts for **A**, **B**, and **C**, a rational start for $\underline{\mathbf{G}}$ can further be calculated by means of Boolean regression.

*Truly random.* A truly random starting configuration for the ALS and SA hierarchical classes algorithms is generated by letting the entries of the respective matrices be independent realizations of a Bernoulli variable with some parameter $\pi$.

*Pseudo random.* A pseudo random starting configuration for a hierarchical classes component matrix is generated by setting each component $r$ of this matrix to a randomly selected data vector. For instance, to generate a pseudo random start for the HICLAS object component matrix **A**, one randomly selects an attribute $j$ for each component $r$ of **A** and sets $\mathbf{a}_{.r} = \mathbf{d}_{.j}$. As another example, a pseudo random start for the Tucker3-HICLAS attribute component matrix **B** is generated by randomly selecting an object $i$ and a source $k$ for each component $q$ of **B** and setting $\mathbf{b}_{.q} = \mathbf{d}_{i.k}$. Because a pseudo random core array counterpart of such pseudo random component matrices is not readily available, for the simulations reported in the present paper we generate the initial core array in a truly random way, setting the Bernoulli parameter $\pi$ to .5.

*Smart random.* When using an ALS hierarchical classes algorithm, given the huge number of possible starting configurations (e.g., $2^{IP} \times 2^{JQ} \times 2^{KR}$ possible Tucker3-HICLAS starting configurations for a single data array) and the generally low number of iterations performed before the ALS routine converges (e.g., in the Tucker3-HICLAS simulation study, on average only 4.91 iterations were performed, Ceulemans et al., 2003), it is reasonable to assume that truly or pseudo random multistart procedures may not solve the local minima problem, unless one uses a huge number of starts, which is not computationally feasible. Therefore, in this paper, we also consider a smart random multistart procedure in which the random starts are selected in such a way that they can be expected to belong to the subset of most promising starting configurations. In particular, we propose to use as a smart random start the rational initial configuration perturbed with 10% of error, that is, with 10% of the parameters changed in value. Note that a pilot study in which various error percentages were evaluated, showed that 10% error leads to the best results.

## 5. Simulation Studies

In this section we investigate to what extent the HICLAS and Tucker3-HICLAS local minima problems can be mitigated by means of the above introduced SA algorithm and/or multistart procedures. To this end, we reanalyzed artificial data sets from the HICLAS and Tucker3-HICLAS simulation studies reported by Leenen and Van Mechelen (2001) and Ceulemans et al. (2003).

### *5.1. Reanalysis of the Simulated HICLAS Data Sets*

*5.1.1. Design and Procedure.* In hierarchical classes simulation studies, three different types of binary matrices must be distinguished: true matrices **T**, which can be perfectly represented by a low-rank hierarchical classes model as they are constructed by the simulation researcher; data matrices **D**, which are **T** perturbed with error; and reconstructed data matrices **M**, which are obtained by applying a hierarchical classes algorithm to the data matrices **D** and can therefore also be perfectly represented by a low-rank hierarchical classes model.

The design of the original HICLAS simulation study consisted of three between-block independent variables:

(a) the *Size*, $I \times J$, of **T**, **D**, and **M**, at four levels: $15 \times 25$, $20 \times 20$, $80 \times 20$, $40 \times 40$. Note that Leenen & Van Mechelen (2001) also included two large sizes, $500 \times 50$ and $150 \times 150$, which we omit to decrease the computational load;
(b) the *True rank*, $r$, of the HICLAS model for **T**, at three levels: 3, 5, 8;
(c) the *Error level*, $\varepsilon$, which is the proportion of cells $d_{ij}$ differing from $t_{ij}$, at six levels: .00, .05, .10, .15, .20, .25.

For each cell of this design 25 replicates were generated, yielding $25 \times 4$ (size) $\times 3$ (true rank) $\times 6$ (error level) $= 1800$ simulated data sets.

In the present simulation study, these 1800 data arrays were reanalyzed, adding two within-block independent variables to the design:

(a) the *Start procedure*, at four levels: rational, truly random, pseudo random, smart random;
(b) the *Algorithm*, at two levels: ALS, SA.

The rational ALS and SA analyses implied two runs of the ALS and SA algorithms, respectively. In particular, the first run of the rational ALS analyses implied a rational start for **A**, the second a rational start for **B**. With respect to the SA analyses, the two rational runs both started from the same rational start for **A** and **B**. As such, the retained rational ALS and SA solutions are the best solutions across these two ALS and SA runs, respectively. Similarly, the truly random, pseudo random, and smart random ALS and SA analyses each consisted of 100 runs of the ALS and SA algorithms, respectively, with the first 50 runs of the truly random, pseudo random, and smart random ALS analyses starting from a randomly generated configuration for **A**, the last 50 from a random start for **B**. For each of the 100 truly random, pseudo random, and smart random SA runs a new random start for both **A** and **B** was generated. The retained truly random, pseudo random, and smart random ALS and SA solutions are thus the best solutions across 100 ALS and SA runs, respectively.

*5.1.2. Results.* Except for simulated data sets with $\varepsilon = 0$, the global minimum for the HICLAS analysis of a simulated data set is unknown. To examine the ability of the ALS HICLAS algorithm to minimize the loss function (4), Leenen and Van Mechelen (2001) noted that the badness-of-data-value *BOD*,

$$BOD = \sum_{i=1}^{I} \sum_{j=1}^{J} (d_{ij} - t_{ij})^2, \tag{8}$$

which computes the number of cells $d_{ij}$ differing from $t_{ij}$, constitutes an upper bound for the loss function value of the global minimum. Therefore, these authors proposed to use this *BOD* value as a proxy for the loss function value of the global minimum. In this simulation study we can further refine this proxy because one or more of the eight types of analyses (i.e., four start procedures $\times$ two algorithms) for each data set may yield a loss function value that is lower than the *BOD* value of the data set; in particular, note that the lowest loss function value $loss_8$ across the eight types of analyses for the data set was better, equal, and worse than *BOD* in 1368, 431, and 1 cases, respectively. As such, we derived for each of the 1800 data sets a proxy of the loss function value of the global minimum by recording the minimum of the badness-of-data-value *BOD* and $loss_8$,

$$proxy = \min(BOD, loss_8). \tag{9}$$

Subsequently, for each (start, algorithm) combination, we computed the percentage of data sets for which the (start, algorithm) combination ended in this proxy of the global minimum; these

TABLE 2.
Percentage of HICLAS analyses that ended in the proxy of the global minimum as a function of start procedure and algorithm.

| | Algorithm | | |
|---|---|---|---|
| Start procedure | ALS | SA | Overall |
| Rational | 35.4 | 50.1 | 42.8 |
| Truly random | 48.2 | 86.3 | 67.3 |
| Pseudo random | 50.7 | 87.6 | 69.1 |
| Smart random | 56.6 | 86.5 | 71.6 |
| Overall | 47.7 | 77.6 | |

percentages are shown in Table 2. From Table 2 it can be derived that, irrespective of the actual type of random start, the (random, SA) combinations succeed best in minimizing the HICLAS loss function: The (random, SA) analyses end in about 87% of the data sets in the proxy of the global minimum. The worse performance of the (rational, SA) combination is due to the fact that the (rational, SA) analyses implied only two runs of the SA algorithm, whereas the (random, SA) analyses consisted of 100 runs.

To check whether or not the overall better performance of the (random, SA) combinations is qualified by the manipulated data characteristics, we conducted a split plot factorial analysis of variance with three between-block factors (size, rank, error) and two within-block factors (start procedure, algorithm), and with a variable coding whether or not the loss function value equals the proxy of the global minimum as the dependent variable. Only considering effects that account for at least 5% of the explained variance, the analysis revealed (apart from main effects of start procedure (9.2%) and algorithm (15.4%)) main effects of rank (27.4%) and error (18%), implying that the higher the complexity of the underlying truth and the more error on the data, the harder it becomes to minimize the HICLAS loss function (the percentage of analyses that ended in the proxy equals 87.5, 61.9, 38.6 for rank 3, 5, 8, and 87, 76.3, 67.5, 57.2, 48.6, 39.5 for 0, 5, 10, 15, 20, 25% error). Given the absence of sizeable interactions between start and algorithm on the one hand and data characteristics on the other hand, it can be concluded that the use of a (random, SA) type of analysis is recommended irrespective of the data characteristics at hand.

Subsequently, we investigated whether the performance of a (start, algorithm) combination with respect to minimizing the loss function is indicative of its performance with respect to recovery of the underlying truth. To this end, we computed for each of the 1800 data sets and each of the eight (start, algorithm) combinations the following badness-of-recovery-value *BOR*,

$$BOR = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J}(m_{ij} - t_{ij})^2}{I\,J}, \tag{10}$$

which denotes the proportion of reconstructed data matrix entries that differ from the corresponding true matrix entries. For each of the eight (start, algorithm) combinations, the mean *BOR* across the 1800 data sets is shown in Table 3. Table 3 reveals that the average *BOR* values of the eight (start, algorithm) combinations hardly differ, implying that suboptimal solutions may still yield a good reconstruction of the underlying truth.

Finally, regarding difference in computation time, the runtime of the SA analyses is about 50 times longer than that of the ALS analyses. For instance, on a PC with a Pentium IV processor (3 GHz) and 1 GB RAM, the pseudo random ALS analysis of the 1800 data sets took 1975 seconds in total, whereas the pseudo random SA analysis of these data sets finished in 105,409 seconds. As such, one may wonder whether the differences in performance between pseudo random SA and ALS analyses are only due to the fact that the SA analyses were allowed more computation time than the ALS analyses. To check this, we computed for each combination

TABLE 3.
Mean badness-of-recovery for the HICLAS data sets as a function of start procedure and algorithm.

| | Algorithm | |
|---|---|---|
| Start procedure | ALS | SA |
| Rational | .088 | .083 |
| Truly random | .086 | .082 |
| Pseudo random | .086 | .082 |
| Smart random | .084 | .082 |

of size, true rank, and error the average SA and ALS runtime. Subsequently, we reanalyzed each data set with the pseudo random SA algorithm with the number of random starts equal to ceil(average ALS runtime $*$ 100/average SA runtime). Comparing the loss function values of the thus obtained pseudo random SA solutions with the loss function values of the pseudo random ALS solutions revealed that SA yielded better, equal, and worse results than ALS in 628, 887, and 285 cases, respectively. This result suggests that the differences in performance between pseudo random SA and ALS analyses are only partly due to differences in computation time.

### 5.2. *Reanalysis of the Simulated Tucker3-HICLAS Data Sets*

*5.2.1. Design and Procedure.* Like in the HICLAS simulation study, a distinction has to be made between true arrays $\underline{\mathbf{T}}$, data arrays $\underline{\mathbf{D}}$, and reconstructed data arrays $\underline{\mathbf{M}}$. When generating the true arrays and the data arrays, Ceulemans et al. (2003) used a design with three between-block independent variables:

(a) the *Size*, $I \times J \times K$, of $\underline{\mathbf{T}}$, $\underline{\mathbf{D}}$, and $\underline{\mathbf{M}}$, at three levels: $15 \times 15 \times 15$, $30 \times 20 \times 10$, $30 \times 20 \times 60$;
(b) the *True rank*, $r$, of the Tucker3-HICLAS model for $\underline{\mathbf{T}}$, at three levels: $(2, 3, 3)$, $(4, 3, 2)$, $(4, 4, 4)$. Note that Ceulemans et al. (2003) also included true rank $(2, 2, 2)$, which we omit to decrease the computational load;
(c) the *Error level*, $\varepsilon$, which is the proportion of cells $d_{ijk}$ differing from $t_{ijk}$, at five levels: .00, .05, .10, .20, .30.

For each cell of this design 20 replicates were generated, yielding $20 \times 3$ (size) $\times$ 3 (true rank) $\times$ 5 (error level) $=$ 900 simulated data sets.

In the present simulation study we reanalyzed these 900 data arrays, adding two within-block independent variables to the design:

(a) the *Start procedure*, at three levels: rational, pseudo random, smart random—as an ALS (resp., SA) pilot study revealed that a pseudo random start leads to better (resp., similar) results than a truly random start, the truly random start was omitted in order to decrease the computational load;
(b) the *Algorithm*, at two levels: ALS, SA.

The rational, pseudo random, and smart random analyses implied 6, 100, and 100 runs, respectively, from which the best solution was retained. More specifically, the six rational ALS runs all used the same starting configuration but implied a different updating order for the component matrices (see, Ceulemans et al., 2003), whereas the 100 pseudo or smart random ALS runs used a different starting configuration but implied the same $\mathbf{A} \rightarrow \mathbf{B} \rightarrow \mathbf{C}$ updating order for the component matrices. Likewise, the six rational SA runs all started from the same rational starting configuration, whereas for each of the 100 pseudo or smart random runs a new start was generated.

TABLE 4.
Percentage of Tucker3-HICLAS analyses that ended in the proxy of the global minimum as a function of start procedure and algorithm.

| | Algorithm | | |
|---|---|---|---|
| Start procedure | ALS | SA | Overall |
| Rational | 37.2 | 94.0 | 65.6 |
| Pseudo random | 12.0 | 99.0 | 55.5 |
| Smart random | 68.8 | 98.8 | 83.8 |
| Overall | 39.3 | 97.3 | |

*5.2.2. Results.* As was the case for the HICLAS simulation, the global minimum for the Tucker3-HICLAS simulated data sets is unknown. Therefore, we obtained for each of the 900 data sets a proxy of the global minimum by taking the minimum of the badness-of-data-value *BOD*,

$$BOD = \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K}(d_{ijk} - t_{ijk})^2 \tag{11}$$

and the lowest loss function value $loss_6$ across the six groups of runs (i.e., three start procedures $\times$ two algorithms) for the data set:

$$proxy = \min(BOD, loss_6). \tag{12}$$

Note that $loss_6$ was better, equal, and worse than *BOD* in 152, 746, and 2 cases, respectively. Table 4 shows for each (start, algorithm) combination the percentage of data sets for which the (start, algorithm) combination ended in the proxy of the global minimum. As hypothesized, for the ALS algorithm, using smart random starts improves the goodness-of-fit performance, whereas pseudo random starts yield worse results than the rational start. However, the main conclusion is that, irrespective of the actual type of random start, (random, SA) combinations have the best overall performance and end in about 99% of the data sets in the proxy of the global minimum. Again, the worse performance of the (rational, SA) combination is due to the fact that the (rational, SA) analyses implied less runs of the SA algorithm than the (random, SA) analyses (i.e., six runs versus 100 runs). Finally, comparing the Tucker3-HICLAS results in Table 4 with the HICLAS results in Table 2, one might be tended to conclude that Tucker3-HICLAS problems are easier to solve than HICLAS problems. However, one should take into account that the HICLAS simulation study included higher true ranks than the Tucker3-HICLAS study: Indeed, only considering the simulation results for the HICLAS data sets of true rank 3, the (random, SA) HICLAS analyses ended in more than 99% of the cases in the proxy of the global minimum.

To check whether the conclusion that (random, SA) analyses perform best holds for all considered types of data, we conducted a split plot factorial analysis of variance with as dependent variable whether or not the obtained loss function value equals the proxy of the global minimum, with size, true rank, and error level as between-block independent variables, and with start procedure and algorithm as within-block independent variables. As can be expected from the above-mentioned results, the analysis revealed main effects of start procedure (9.4%) and algorithm (57.8%), qualified by their interaction effect (9.3%). Regarding the effect of the manipulated data characteristics, a main effect of error (8.3%) was obtained, implying that the more error on the data, the harder it becomes to minimize the Tucker3-HICLAS loss function: the percentage of analyses that ended in the proxy equals 71.7, 78.8, 77.1, 65.6, 48.2, for 0, 5, 10, 20, 30% error. With respect to the latter, note that the result that problems without error are more

TABLE 5.
Mean badness-of-recovery for the Tucker3-HICLAS data sets as a function of start procedure and algorithm.

| | Algorithm | |
|---|---|---|
| Start procedure | ALS | SA |
| Rational | .030 | .006 |
| Pseudo random | .041 | .004 |
| Smart random | .014 | .004 |

difficult to solve than problems with a small amount of error, is due to the ALS analyses: all (random, SA) analyses of the data sets without error ended in the proxy of the global minimum. As no sizeable interactions were found between start and algorithm on the one hand and data characteristics on the other hand, we conclude that the use of a (random, SA) type of analysis is recommended irrespective of the data characteristics at hand.

To investigate the relationship between ability to minimize the loss function and the recovery of the underlying truth, we computed for each of the six (start, algorithm) combinations the mean badness-of-recovery-value *BOR*,

$$BOR = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K}(m_{ijk} - t_{ijk})^2}{I\,J\,K} \tag{13}$$

across the 900 data sets; the results are shown in Table 5. Based on a comparison of Tables 4 and 5, we conclude that there is a clear positive relationship between ability to minimize the loss function and ability to recover the underlying truth.

Finally, to investigate differences in computation time, a pseudo random ALS and SA analysis of three data sets for each cell of the design was run on a PC with a Pentium IV processor (3 GHz) and 1 GB RAM. The total runtime for the ALS and SA analyses amounted to 51,135 and 291,776 seconds, respectively, implying that the runtime of the SA analyses is about six times longer than that of the ALS analyses. Again, one may wonder whether this difference in runtime is the main cause of the difference in performance between the two types of analyses. To investigate this, we applied the same procedure as in the HICLAS simulation study, that is, repeating the pseudo random SA analysis with the number of starts about equal to the average runtime of a pseudo random ALS analysis with 100 starts divided by the average runtime of a pseudo random SA analysis with one start only (the average runtimes were computed for each combination of size, true rank, and error). The loss function values of the thus obtained pseudo random SA solutions were lower than or equal to those of the pseudo random ALS solutions, which suggests that the differences in performance between both types of analyses are not or at most only partly due to differences in computation time.

## 6. Conclusion and Discussion

All hierarchical classes simulation studies showed that ALS hierarchical classes algorithms, in conjunction with a rational start, frequently end in a local minimum. In this paper it was evaluated whether or not this local minima problem can be mitigated by means of two common strategies for avoiding local minima in combinatorial data analysis: SA and use of a multistart procedure. To this end, we proposed a generic SA algorithm for hierarchical classes analysis and three different types of random starts. In line with results for other combinatorial data analytic techniques (see, e.g., Al-Sultan & Khan, 1996; Hand & Krzanowski, 2005; Steinley, 2003; Murillo et al., 2005), a reanalysis of artificial HICLAS and Tucker3-HICLAS data sets revealed

that both SA and a random multistart procedure indeed mitigate the hierarchical classes local minima problem regardless of the properties of the data set under study. Moreover, the biggest reduction of the local minima problem was obtained when both strategies were combined. Therefore, when applying hierarchical classes analysis in practice, it is recommended to use the proposed SA algorithm in combination with a random multistart procedure.

Possible future work in this area includes research on the number of random starts needed in the function of (1) data characteristics and (2) the specification of the parameters of the SA algorithm for hierarchical classes analysis. Regarding the first issue, in the simulation studies reported in this paper, we always used 100 random starts. However, for many data sets, this relatively large amount of random starts and hence computational effort was not really necessary. For instance, in 90% of the (random, SA) analyses of the artificial HICLAS data sets the retained solution was already obtained in the first 50 runs. Moreover, the average number of runs needed to arrive at this solution was clearly related to the size and rank of the considered data sets. Therefore, the derivation of the 'needed number of starts'-rules of thumb would be very helpful for practical users of hierarchical classes analysis. Regarding the second issue, it can be hypothesized that there is a trade-off relation between the number of random starts needed and the specification of the values of the parameters of the SA algorithm: the more strict the parameters are specified, the less random starts are needed. In this paper the number of random starts was kept fixed at 100. Therefore, it would be worthwhile to investigate which combination of number of random starts and SA parameter values yields the smallest computational load while performing equally well as the combination used in this paper.

## References


Aarts, E.H., & Lenstra, J.K. (1997). *Local search in combinatorial optimization*. Chichester, UK: Wiley.

Al-Sultan, K.S., & Khan, M.M. (1996). Computational experience on four algorithms for the hard clustering problem. *Pattern Recognition Letters, 17*, 295–308.

Brusco, M.J. (2001). A simulated annealing heuristic for unidimensional and multidimensional (city-block) scaling of symmetric proximity matrices. *Journal of Classification, 18*, 3–33.

Brusco, M.J., & Stahl, S. (2000). Using quadratic assignment methods to generate initial permutations for least-squares unidimensional scaling of symmetric proximity matrices. *Journal of Classification, 17*, 197–223.

Ceulemans, E., & Van Mechelen, I. (2004). Tucker2 hierarchical classes analysis. *Psychometrika, 69*, 375–399.

Ceulemans, E., & Van Mechelen, I. (2005). Hierarchical classes models for three-way three-mode binary data: Interrelations and model selection. *Psychometrika, 70*, 1–20.

Ceulemans, E., Van Mechelen, I., & Leenen, I. (2003). Tucker3 hierarchical classes analysis. *Psychometrika, 68*, 413–433.

De Boeck, P., & Rosenberg, S. (1988). Hierarchical classes: Model and data analysis. *Psychometrika, 53*, 361–381.

Gara, M., & Rosenberg, S. (1990). A set-theoretical model of person perception. *Behavioral Research, 25*, 275–293.

Hand, D.J., & Krzanowski, W.J. (2005). Optimising *k*-means clustering results with standard software packages. *Computational Statistics and Data Analysis, 49*, 969–973.

Hubert, L., Arabie, P., & Hesson-McInnis, M. (1992). Multidimensional-scaling in the city-block metric—a combinatorial approach. *Journal of Classification*, *9*, 211–236.

Klein, R.W., & Dubes, R.C. (1989). Experiments in projection and clustering by simulated annealing. *Pattern Recognition*, *22*, 213–220.

Kuppens, P., Van Mechelen, I., Smits, D.J.M., De Boeck, P., & Ceulemans, E. (2007). Individual differences in patterns of appraisal and anger experience. *Cognition & Emotion*, *21*, 689–713.

Leenen, I., & Van Mechelen, I. (1998). A branch-and-bound algorithm for Boolean regression. In I. Balderjahn, R. Mathar, & M. Schader, *Data highways and information flooding, A challenge for classification and data analysis* (pp. 164–171). Berlin: Springer.

Leenen, I., & Van Mechelen, I. (2001). An evaluation of two algorithms for hierarchical classes analysis. *Journal of Classification, 18*, 57–80.

Leenen, I., Van Mechelen, I., De Boeck, P., & Rosenberg, S. (1999). INDCLAS: A three-way hierarchical classes model. *Psychometrika, 64*, 9–24.

Milligan, G.W. (1980). An examination of the effect of six types of error perturbation on fifteen clustering algorithms. *Psychometrika, 45*, 325–342.

Murillo, A., Vera, J.F., & Heiser, W.J. (2005). A permutation-translation simulated annealing algorithm for $l_1$ and $l_2$ unidimensional scaling. *Journal of Classification, 22*, 119–138.

Selim, S.Z., & Ismail, M.A. (1984). K-means-type algorithms—A generalized convergence theorem and characterization of local optimality. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 6*, 81–87.

Steinley, D. (2003). Local optima in $k$-means clustering: What you don't know may hurt you. *Psychological Methods, 8*, 294–304.

Van Mechelen, I., & De Boeck, P. (1989). Implicit taxonomy in psychiatric diagnosis: A case study. *Journal of Social and Clinical Psychology, 8*, 276–287.

Van Mechelen, I., & Van Damme, G. (1994). A latent criteria model for choice data. *Acta Psychologica, 87*, 85–94.

Van Mechelen, I., De Boeck, P., & Rosenberg, S. (1995). The conjunctive model of hierarchical classes. *Psychometrika, 60*, 505–521.

Vansteelandt, K., & Van Mechelen, I. (1998). Individual differences in situation-behavior profiles: A triple typology model. *Journal of Personality and Social Psychology, 75*, 751–765.

Vansteelandt, K., & Van Mechelen, I. (2006). Individual differences in anger and sadness: In pursuit of active situational features and psychological processes. *Journal of Personality, 74*, 871–909.

*Manuscript received 5 MAR 2004*
*Final version received 23 OCT 2006*
*Published Online Date: 13 JUN 2007*