APPLYING PARTICLE SWARM OPTIMIZATION TO ESTIMATE PSYCHOMETRIC MODELS WITH CATEGORICAL RESPONSES

BY

Zhehan Jiang

Submitted to the graduate degree program in the Department of Educational Psychology and the Graduate Faculty of the University of Kansas in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Education.

John Poggio, Committee Chairman

William Skorupski, Committee Member

Lisa Wolf-Wendel, Committee Member

Jonathan Templin, Committee Member

Changming Duan, Committee Member

Date defended: Dec 11th , 2017

The Doctoral Committee for Zhehan Jiang
certifies that this is the approved version of the following thesis:

APPLYING PARTICLE SWARM OPTIMIZATION TO ESTIMATE PSYCHOMETRIC MODELS WITH CATEGORICAL RESPONSES

John Poggio, Committee Chairman

Date approved: Dec 11th , 2017

Abstract

Current psychometrics tend to model response data hypothesized to arise from multiple attributes. As a result, the estimation complexity has been greatly increased so that traditional approaches such as the expected-maximization algorithm would fail to produce accurate results. To improve the estimation quality, high-dimensional models are estimated via a global optimization approach- particle swarm optimization (PSO), which is an efficient stochastic method of handling the complexity difficulties. The PSO has been widely used in machine learning fields but remains less-known in the psychometrics community. Details on the integration of the proposed approach to current psychometric model estimation practices are provided. The algorithm tuning process and the accuracy of the proposed approach are demonstrated with simulations. As an illustration, the proposed approach is applied to log-linear cognitive diagnosis models and multi-dimensional item response theory models. These two model families are fairly popular yet challenging frameworks used in assessment and evaluation research to explain how participants respond to item level stimuli. The aim of this dissertation is to fill the gap between the field of psychometric modeling and machine learning estimation techniques.

Acknowledgements

This work would not have been possible without the support of my family, faculty members, and friends. I am grateful to all of those with whom I have had the pleasure to work during my graduate study. This dissertation is a milestone from which I am to work hard and delicately to become a better researcher.

Table of Contents

Chapter 1: Introduction

Psychometrics is the field of study connecting statistical analyses with the theory of psychological measurement. In general, works of psychometrics can be categorized into (1) the instrument construction and measurement process and (2) the improvement of measurement theory. Broadly, most social science subjects, such as sociology, psychology, and education require psychometrics to conduct analyses. Among others, areas, such as the measurement of intelligence, personality, learning paths, and psychological diseases, deploy psychometrics more frequently and therefore tremendous contributions on psychometrics development have been stemmed from these fields. For instance, foundation psychometrics works are attributed to intelligence assessment scientists such as Charles Spearman, L.L. Thurstone, Karl Pearson, Georg Rasch, and Arthur Jensen (see Lyle, 2007 for the history of psychometrics).

Modern psychometrics has devoted more to models with latent structures, for instance, exploratory factor analysis (Cudeck & MacCallum, 2007 ;Thurstone,1947), confirmatory factor analysis (Joreskog, 1969), covariance structure analysis (Bock & Bargmann, 1966; Bollen, 1989; Joreskog, 1970), item response theory (IRT; Lord & Novick, 1968; Thissen & Wainer, 2001), and finally diagnosis classification modeling (DCM; DiBello, Roussos, & Stout, 2007; Henson, Templin, & Willse, 2009). Among those, the latter two -DCMs and IRT- have gained substantive attention and contributions more recently for the reason that they have provided advanced modeling frameworks for research designs with categorical item responses.

DCMs have been developed to identify whether a student masters each attribute required for solving corresponding items. For instance, addition, subtraction, multiplication, and division are four common attributes defined in math ability assessment practice, where test items such as "2+4-1" measure the first two attributes and "4x2/3" measure the last two attributes. In addition to educational testing, DCMs are useful in psychological measurement. For example, literature indicates that neuro-vegetative symptoms (NS) are a general construct that contains three attributes: depression (DEP), fatigue (FAT), and sleeplessness (SLE) (Rabinowitz, Fisher, & Arnett, 2011). Using DCMs allows researchers/practitioners to investigate the attributes for a given patient. Applied works can be found in more topics, for example, Stefanutti, Anselmi, and Robusto (2011) uses the DCM framework to construct leaning map, and Svetina, Dai, and Wang (2017) study differential item functioning in accommodations via the DCM.

IRT, on the other hand, has already become the preeminent modeling paradigm in educational and psychological measurement due to its longer development history. In large-scale testing, IRT has played a dominant role in operational calibration and scoring. The development and application of IRT models has been well-studied; for example, historical overviews can be found in van der Linden and Hambleton (1997), Embretson and Reise (2000), Thissen and Wainer (2001), among others. IRT models posit the probabilistic relationship between a person's latent ability and the probability of an item response. The modeling process links the theory underlying the test, the administrative practices for distributing the test, and statistical modeling so that a test can be constructed fairly and scientifically.

What distinguishes DCMs from IRT models is the latent variable assumption; IRT models are able to provide scores for ordering students along latent a continuum, where DCMs assume that the latent attributes are multiple categories (could be also binary). To be concrete, if an math item, 10/4+5, is created to measure respondents' fraction and subtraction knowledge, IRT would produce numeric values based upon an artificial scale for each respondent, where DCMs could provide the information about mastery or non-mastery on each attribute (i.e., subtraction and faction in this example). This discrepancy reflects on the specification of the statistical models. That is, DCMs are essentially mixture models and IRT possesses integral part in its likelihood functions.

Within each of the families, multi-dimensional item response theory (MIRT) models and log-linear cognitive diagnosis models (LCDMs) are known to be more flexible and informative than other variants of their kinds. However, as a trade-off, estimating these models tend to be more difficult due to a complicated latent structure and a large number of parameters of interest. These models are estimated in a number of ways. Perhaps the most often-used method is marginal maximum likelihood (MML) estimation using the expectation maximization (EM; Dempster, Laird, & Rubin, 1977) algorithm and some variants of this kind (e.g., Baker & Kim, 2004, Bock & Aitkin, 1981). For consistency purpose, this type of algorithms are all named as the EM algorithm. The EM algorithm has been proved insufficient in multi-dimensional settings such as MIRT models and LCDMs. For example, to estimate MIRT models, the EM algorithm relies upon numerical integration to marginalize the likelihood function across the space of the latent attributes. The integration process requires a set of discrete quadrature to approximate the integral, so the number of quadrature points increases exponentially as the number of latent attributes increases linearly. As a result, models with numerous quadrature points take tremendous amounts of calculations to estimate yet often yield inaccurate results. Adaptive quadrature has been developed to handle the computational deficiency by using fewer points (see Schilling & Bock, 2005), but does not solve the problem completely. In terms of LCDM, the EM algorithm is likely to encounter (1) local maxima and (2) label switching problems (Lao, 2016). To be concrete, there will be multiple local maxima of the log-likelihood function that trap the algorithms. Particularly, the EM algorithm is known to converge at local maxima instead of global maxima, where only the latter provides legitimate estimates. Label switching, on the other hand, leads to unreasonable interpretations of item parameters as well as disruption of the converging process. Following an estimation method similar to the EM algorithm, the Quasi-Monte Carlo integration (QMCEM) algorithm replaces quadrature points with pseudorandom numbers (e.g., Niederreiter, 1978). Although the QMCEM algorithm is better suited to high-dimensional integration, it is relatively slow in estimation time when compared with some fully Bayesian estimation algorithms. In addition, the QMCEM does not fit the LCDM framework, as the QMCEM is not devised to handle mixture models.

Different from the frequentist methods mentioned above, Bayesian algorithms are based upon Markov Chain Monte Carlo (MCMC) process. The most frequently used Bayesian algorithms are based upon two fundamental mechanisms: Gibbs sampling and the Metropolis-Hastings (MH) algorithm. Gibbs sampling is used in situations where full conditional posterior distributions of parameters can be derived in closed-form expressions, whereas the MH algorithm uses a proposal distribution substituting the real conditional distribution to enable the MCMC process (e.g., Lynch, 2010). In the current context, both Gibbs and the MH are not effective solutions for a few reasons: (1) logistic link functions which are the used to model the categorical responses given certain attribute(s) are difficult for constructing Gibbs samplers, (2) he MH algorithm requires a rejection/acceptance decision for each parameter at each step of the Markov chain and therefore the converging could be slow or nearly impossible at some situations, and (3) particularly in the field of DCMs, the Bayesian approaches are not as widely-adopted as those in the field of IRT.

Newer algorithms have combined Bayesian and maximum likelihood estimation with stochastic approximation methods such as the Metropolis–Hastings Robbins–Monroe (MHRM) Algorithm (e.g., Cai, 2010). Similarly, Hamiltonian Monte Carlo, a hybrid of the MH algorithm and Hamiltonian dynamics stochastic process, has gained researchers' attention in recent years (HMC; see Brooks, Gelman, Jones, & Meng, 2011; Hoffman & Gelman, 2014). In each iteration of the algorithm, the values of parameters are said to "leap frog" to states closer to their posterior densities, short-cutting the time the MH algorithm takes by avoiding proposal values that are ultimately rejected. Once new values are proposed, the HMC algorithm uses MH to accept/reject proposals. Both MHRM and HMC, compared with general MH algorithms, leads to a more efficient Monte Carlo sampler. The drawback is that these methods are mathematically difficult and therefore not as approachable as the EM algorithm. The estimation times for both approaches will be much larger for lower dimensional problems when compared to the EM algorithm. In addition, the log-likelihood calculation requires extra steps and can be unstable. Finally, the parameter estimates will not be identical between different estimations (Chalmers, 2012).

This dissertation proposes a global optimization approach- particle swarm optimization (PSO; Eberhart & Kennedy, 1995)- to handle the psychometric model estimations. The PSO is an efficient stochastic method that has been widely used in machine learning field but remains less-known in the psychometrics community. The proposed technique is a "derive-free" mean that can be embedded to other algorithms such as the EM algorithms and the MH sampling. Overall, the hypotheses are 1) this novel estimation technique can be used in psychometric models, 2) the estimation results would be equally and/or more accurate the some traditional approaches. 3) parallel computing facilities can be applied to the proposed estimation, and 4) tunings of the proposed estimation approach would yield results differently in various ways such that customized guidelines can be provided in this particular case.

Chapter 2: Literature Review

Psychometric models of intelligence are generally concerned with the structure and organization of attributes of interest; they focus on conceptions of attributes that depend exclusively on the basis of designed tests as measures of individual differences, and the models are derived from statistical manipulations of scores obtained within and across the tests. In the past five decades, classical test theory has been rapidly expanded in various directions (Crocker & Algina, 1986). Specifically, as the focus in data analysis is moving from univariate to multivariate procedures, the statistical modeling of test data is becoming more complex involving structural equation modeling (SEM), or modeling with modern test theories such as item response theory (IRT) and diagnosis classification modeling (DCM). Fitting a complex psychometric model relies on the ability to accurately estimate the model parameters, which can be realized with the availability of enhanced computational technology and the emergence of advanced statistical estimation methods. The two psychometric models- log-linear cognitive diagnostic model and multidimensional item response theory model- are reviewed. In addition, particle swarm optimization is introduced in details. The concepts and mathematical expressions are presented along with the models and estimations.

Log-linear Cognitive Diagnostic Model

Recent advances in model development have produced general diagnostic models, for instance, generalized Deterministic Input; Noisy "And" gate model (G-DINA; de la Torre 2011), General Diagnostic Model (GDM; von Davier, 2005), and Log-linear Cognitive Diagnosis Model (LCDM; Henson, Templin, & Willse, 2009). A LCDM (G-DINA or GDM) provides great flexibility such as 1) subsuming most latent attributes, 2) enabling both additive and non additive relationships between attributes and items simultaneously, and 3) syncing with other psychometric models, increasing insightfulness. Rupp, Templin, and Henson (2010, p.163) proved that LCDM can be converted to core DCMs such as Deterministic Input; Noisy "And" gate (DINA; Junker & Sijtsma, 2001), Noisy Input; Deterministic "And" gate model (NIDA, Junker & Sijtsma, 2001), and the Reparameterized Unified Model (RUM, Hartz, 2002), whereas examples of disjunctive models include the Deterministic Input; Noisy "Or" gate model (DINO, Templin & Henson, 2006). Throughout the article, the general diagnostic model is referred as a LCDM for consistency purpose.

As a member of latent class models, a LCDM is mathematically defined as:

$$P\left(\boldsymbol{Y}_p = \boldsymbol{y_p}\right) = \sum_{c=1}^{N_c}\left(v_c \prod_{i=1}^{N_i} \pi_{ci}^{y_{pi}}(1-\pi_{ci})^{1-y_{pi}}\right), \tag{1}$$

where $\boldsymbol{y}_p = (y_{p1}, y_{p2}, \dots, y_{pI})$ is the correct/incorrect response vector of respondent *p* on a test comprised of N_i items, and element y_{pi} is the corresponding response on item *i*. v_c is the probability of membership in latent class *c,* and π_{pi} is the probability of correct response to item *i* by respondent *p* in the class. Extended from Equation 1, the log-likelihood function for a random sample of size N_p can be expressed:

$$LogL = \sum_{p=1}^{N_\mathrm{p}} log\left\{\sum_{c=1}^{N_c}\left(v_c \prod_{i=1}^{N_i} \pi_{ci}^{y_{pi}}(1-\pi_{ci})^{1-y_{pi}}\right)\right\}. \tag{2}$$

To simplfy computational efforts, Equation 2 is often re-written as:

$$LogL = \sum_{p=1}^{N_\mathrm{p}} log\left\{\sum_{c=1}^{N_c}\left(exp\left(log(v_c) + log(\prod_{i=1}^{N_i} \pi_{ci}^{y_{pi}}(1-\pi_{ci})^{1-y_{pi}})\right)\right)\right\}, \tag{3}$$

where $log(\prod_{i=1}^{N_i} \pi_{ci}^{y_{pi}}(1-\pi_{ci})^{1-y_{pi}})$ can be further converted to $\sum_{i=1}^{N_i} log(\pi_{ci}^{y_{pi}}(1-\pi_{ci})^{1-y_{pi}})$.

Suppose there are $N_\mathrm{\alpha}$ attributes. The cognitive state of a respondent is denoted by attribute vector $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \dots, \alpha_{N_\alpha})$, where each element in $\boldsymbol{\alpha}$ is a 1/0 binary variable indicating whether a respondent has mastered ath attribute α_a. There are a total number of 2^{N_α} possible attribute patterns (i.e., classes). To illustrate, a respondent p with a pattern $\boldsymbol{\alpha} = (0, 1, 1, 0\ )$ has mastered the second and the third attributes, but not the first and the forth ones. Similarly, if the pattern becomes $\boldsymbol{\alpha} = (1, 1, 1, 1\ )$, it means the respondent has mastered all attributes. To identify attributes that are required to solve each item, content experts provide a Q-matrix of size $N_i * N_\mathrm{\alpha}$, where N_i and $N_\mathrm{\alpha}$ are the numbers of items and attributes in a test respectively. The (i, a) entry of the Q-matrix q_{ia} is 1 when item *i* is associated with attribute a, and otherwise $q_{ia} = 0$. Given respondent *p*'s attribute pattern is $\boldsymbol{\alpha_c}$, the conditional probability of item *i* can be stated as:

$$\pi_{ci} = P(y_{\mathrm{p}i}|\boldsymbol{\alpha_c}) = \frac{\exp\left(\lambda_{i,0} + \boldsymbol{\lambda}_i^T \boldsymbol{h}(\boldsymbol{\alpha_c}, \boldsymbol{q_i})\right)}{1 + \exp\left(\lambda_{i,0} + \boldsymbol{\lambda}_i^T \boldsymbol{h}(\boldsymbol{\alpha_c}, \boldsymbol{q_i})\right)}, \tag{4}$$

Where $\boldsymbol{q_i}$ is the set of Q-matrix entries for item *i*, $\lambda_{i,0}$ is the intercept parameter, where $\boldsymbol{\lambda_i}$ represents a vector of size $(2^{N_\alpha} - 1) * 1$ that contains main effect and interaction effect parameters of item *i*, and $\boldsymbol{h}(\boldsymbol{\alpha_c}, \boldsymbol{q_i})$ is a vector of size $(2^{N_\alpha} - 1) * 1$ with linear combinations of the $\boldsymbol{\alpha_c}$ and $\boldsymbol{q_i}$. Particularly, $\boldsymbol{\lambda}_i^T \boldsymbol{h}(\boldsymbol{\alpha_c}, \boldsymbol{q_i})$ inside the exponent function can be expressed as:

$$\boldsymbol{\lambda}_i^T \boldsymbol{h}(\boldsymbol{\alpha_c}, \boldsymbol{q_i}) = \sum_{a=1}^{N_\alpha} \lambda_{i,1,(a)} \mathrm{\alpha}_{ca} q_{ia} + \sum_{a=1}^{N_\alpha} \sum_{a'>1} \lambda_{i,2,(a,a')} \mathrm{\alpha}_{ca} \mathrm{\alpha}_{ca'} q_{ia}\, q_{ia'} + \cdots, \tag{5}$$

Where $\lambda_{i,1,(a)}$ and $\lambda_{i,2,(a,a')}$ are the main effect for ath attribute α_a and a two-way interaction effect for α_a and $\alpha_{a'}$. Since elements of $\boldsymbol{\alpha_c}$ and $\boldsymbol{q_i}$ are binary, $\boldsymbol{h}(\boldsymbol{\alpha_c}, \boldsymbol{q_i})$ contains binary elements, which indicate effects needed to be estimated. For an item measuring *A* attributes, *A*-way interaction effects should be specified in $\boldsymbol{h}(\boldsymbol{\alpha_c}, \boldsymbol{q_i})$. Table 1 shows a concrete example of a measure with three attributes: Item 1 that measures α_1 only has two estimates, where Item 3 measuring all three attributes has 8 estimates in total.

The item parameters, however, do require monotonicity constraints; otherwise the LCDM estimation is likely to encounter (1) local maxima and (2) label switching problems (Lao & Templin, 2016). To be concrete, without the constraints, there will be multiple local maxima of the log-likelihood function that trap the estimation process. Particularly, the EM algorithm- a dominant method in DCM estimations- is known to converge at local maxima instead of global maxima, where only the latter provides legitimate estimates. Label switching, on the other hand, leads to unreasonable interpretations of item parameters as well as disruption of the converging process. Rupp, Templin, and Henson (2010) outlined the parameter constraint approach, for example, ensuring the positive-ness of $\boldsymbol{\lambda_{i,1}}$in Equation 5 and forcing the 2-way interaction effect $\lambda_{i,2,(a,a')}$ to be bigger than the corresponding negative main effects -$\lambda_{i,1,(a)}$ and -$\lambda_{i,1,(a')}$. Evidence suggests that the parameter constraint approach would decrease of risk of reaching local maxima and keeping label consistency (Lao & Templin, 2016), however, the constrained true sampling space remains unknown due to mathematical complexity.

Table 1. Formula Expression Example of a Log-linear Cognitive Diagnosis Model

Item	α_1	α_2	α_3	Complete $\lambda_{i,0} + \boldsymbol{\lambda}_i^T \boldsymbol{h}(\boldsymbol{\alpha_c}, \boldsymbol{q_i})$ Expression	Simplified Expression
1	1	0	0	$\lambda_{1,0} + \lambda_{1,1}(1) + \lambda_{1,2}(0) + \lambda_{1,3}(0) + \lambda_{1,12}(1 * 0) + \lambda_{1,13}(1 * 0) + \lambda_{1,23}(0 * 0) + \lambda_{1,123}(1 * 0 * 0)$	$\lambda_{1,0} + \lambda_{1,1}(1)$
2	0	1	1	$\lambda_{2,0} + \lambda_{2,1}(0) + \lambda_{2,2}(1) + \lambda_{2,3}(1) + \lambda_{2,12}(0 * 1) + \lambda_{2,13}(0 * 1) + \lambda_{2,23}(1 * 1) + \lambda_{2,123}(0 * 1 * 1)$	$\lambda_{2,0} + \lambda_{2,2}(1) + \lambda_{2,3}(1) + \lambda_{2,23}(1)$
3	1	1	1	$\lambda_{3,0} + \lambda_{3,1}(1) + \lambda_{3,2}(1) + \lambda_{3,3}(1) + \lambda_{3,12}(1 * 1) + \lambda_{3,13}(1 * 1) + \lambda_{3,23}(1 * 1) + \lambda_{3,123}(1 * 1 * 1)$	$\lambda_{3,0} + \lambda_{3,1}(1) + \lambda_{3,2}(1) + \lambda_{3,3}(1) + \lambda_{3,12}(1) + \lambda_{3,13}(1) + \lambda_{3,23}(1) + \lambda_{3,123}(1)$

Multi-dimensional Item Response Theory

Item Response Theory (IRT; Lord & Novick, 1968; Thissen & Wainer, 2001) has several variants in both unidimensional and multi-dimensional contexts: they are the Rasch model (i.e., 1-PL), 2-PL, 3-PL, and finally 4-PL. What differentiates these variants is the number of parameters for each item. For example, 2-PL requires difficulty (intercept) and discrimination (main effect) parameters to be estimated for each item, where 3-PL has an extra parameter-guessing-in addition to a 2-PL model. Practically, 2-PL has been a reasonable choice, and therefore, throughout the dissertation, the IRT model is referred as a 2-PL model, which is akin to the expressions defined in the LCDM context. In a MIRT model, since

According to the IRT definition that the latent attributes are assumed to follow a multivariate normal distribution, the equation for the probability of the score response for a respondent is defined as:

$$P(\boldsymbol{Y}_p = \boldsymbol{y_p}) = \int_{-\infty}^{+\infty} \prod_{i=1}^{N_i} \pi_{pi}^{y_{pi}} (1 - \pi_{pi})^{1-y_{pi}} \, G(\boldsymbol{\theta}_\mathbf{p}) \, \mathrm{d}\boldsymbol{\theta}_\mathbf{p}, \tag{6}$$

where $G(\boldsymbol{\theta})$ is the probability density function for a vector $\boldsymbol{\theta}$, which is the latent attributes for respondent p. Other than $\boldsymbol{\theta}$ and its related terms, $\boldsymbol{y}_p = (y_{p1}, y_{p2}, \dots, y_{pI})$ is again the correct/incorrect response vector of respondent p on a test comprised of N_i items, element y_{pi} is the corresponding response on item i, and finally π_{pi} is the probability of correct response to item i by respondent p. It can be seen that, compared with Equation 1, the measurement part $\prod_{i=1}^{N_i} \pi_{ci}^{y_{pi}} (1 - \pi_{ci})^{1-y_{pi}}$ remains identical where the structure part- $g(\boldsymbol{\theta})$ and v_c-are presented differently. In order to marginalize the likelihood function across the space of the latent attributes, the integral should be evaluated through an approximation procedure, as it has no closed-form solutions. The Q-matrix applies to MIRT models in an identical way to that of DCMs. However, to distinguish the LCDM whose attributes are in binary scale, the attributes in MIRT models are represented by $\boldsymbol{\theta} = (\theta_1, \theta_2, \dots, \theta_{N_\alpha})$ where the number of attributes is N_α. For the simplicity purpose, let $\boldsymbol{\theta} \sim MVN(\mathbf{0}, \boldsymbol{\Sigma}_\mathbf{p})$ where *MVN* represents a multivariate normal distribution, $\mathbf{0}$ means that the latent attribute means are all zeros, and $\boldsymbol{\Sigma}_\mathbf{p}$ is a correlation matrix. When the number of attributes is small, for example, lower than five, Gauss-Hermite quadrature can be used to integrate $\boldsymbol{\theta}$**.** The expression for the quadrature integral is:

$$P(\boldsymbol{Y}_p = \boldsymbol{y_p}) \approx \sum_{qN_a=1}^{N_{qd}} \dots \sum_{q2=1}^{N_{qd}} \sum_{q1=1}^{N_{qd}} \prod_{i=1}^{N_i} \pi_{pi}^{y_{pi}} (1 - \pi_{pi})^{1-y_{pi}} A(X_{q1}) A(X_{q2}) \dots A(X_{qN_a}), \quad (7)$$

where N_{qd} is the number of quadrature points, X_{qa} is the value of a quadrature point for attribute a, $A(X_{qa})$ is a weight that is related to the height of the normal density function at the attribute a's quadrature point value and the distance between the quadrature points.

Given respondent p's attribute vector is $\boldsymbol{\theta_p}$, the conditional probability of item i can be stated as:

$$\pi_{pi} = P(y_{pi} | \boldsymbol{\theta_p}) = \frac{\exp\left(\lambda_{i,0} + \boldsymbol{\lambda}_i^T \boldsymbol{h}(\boldsymbol{\theta_p}, \boldsymbol{q_i})\right)}{1 + \exp\left(\lambda_{i,0} + \boldsymbol{\lambda}_i^T \boldsymbol{h}(\boldsymbol{\theta_p}, \boldsymbol{q_i})\right)}, \quad (8)$$

where $\boldsymbol{q_i}$, $\lambda_{i,0}$, and $\boldsymbol{\lambda_i}$ are identical to what are defined in Equation 3. Similarly, $\boldsymbol{\lambda}_i^T \boldsymbol{h}(\boldsymbol{\theta_p}, \boldsymbol{q_i})$ can be expressed as:

$$\boldsymbol{\lambda}_i^T \boldsymbol{h}(\boldsymbol{\theta_p}, \boldsymbol{q_i}) = \sum_{\boldsymbol{a}=1}^{N_\alpha} \lambda_{i,1,(a)} \theta_a q_{ia} + \sum_{a=1} \sum_{a'>1}^{N_\alpha} \lambda_{i,2,(a,a')} \theta_a \theta_{ca'} q_{ia} \, q_{ia'} + \cdots, \quad (9)$$

where $\lambda_{i,1,(a)}$ and $\lambda_{i,2,(a,a')}$ are the main effect for ath attribute θ_a and a two-way interaction effect for θ_a and $\theta_{a'}$.

Particle Swarm Optimization

The Particle Swarm Optimization (PSO) is a stochastic algorithm that belongs to the Swarm Intelligence methods family. Inspired by the social behavior of bird flocking and fish schooling, Eberhart and Kennedy (1995) proposed PSO to find solutions to optimization problems, such as numerical integration and the travelling salesman problem (see Dorigo & Gambardella, 1997; Djerou, Khelil, & Batouche, 2011 for details). The term ‘particle’ represents a natural agent that possesses swarm behaviors (i.e., the ability of performing social interaction). Examples of swarm behaviors include (1) improving the estimation accuracy of particle themselves to expected levels and (2) interacting with their neighborhood. In the sense of estimation, each particle stochastically explores permissible space to yield the optimal solution. The PSO is particularly useful when solutions do not exist analytically or specifically have been proven to be theoretically intractable.

Table 2. Reference Labels for PSO Terminology

PSO Terminology	Meaning	Reference
Particle	A vector containing parameters estimates- a candidate solution vector	Vector of the estimates ($\boldsymbol{\lambda_l}$)
Velocity	A vector updating the parameter estimates	Vector of update steps ($Update_l$)
Inertia Weight	Direction of the vector of update steps	Update direction parameter (w)
Learning/Acceleration Factor	Coefficient for avoiding the premature convergence	Update correction parameter (c)

The concept of the PSO algorithm, although straightforward, is confusing when it is addressed with psychometric models. In particular, the terms and meanings of the PSO components are not familiar to researchers in the field of measurement if not all social sciences. To keep the reading flow consistent, Table 2 provides the references for the PSO components and therefore, for the rest of the dissertation, the components are called in accordance with the forth column of Table 2; for instance, the term “particle” would be called as “vector of the estimates". And similarly, the parameters used to alter the (1) direction and the (2) correction of the update steps for next iteration are called (1) update direction parameter and (2) update correction parameter, respectively. In addition to the reference names, mathematical symbols are often listed in Table 2 such that expressions in the next sections match Equations 1 to 9.

Throughout the dissertation, given the parameters of interest is $\boldsymbol{\lambda}$, the same symbol is used represent a vector of the estimates for the sake of consistency. The strategy of the PSO algorithm is outlined as follows: each vector of the estimates *l* represents a candidate solution to the optimization problem in a D-dimensional space, where the current solution and the vector of update steps of the vector of the estimates are presented by $\boldsymbol{\lambda}_l = (\lambda_{l1}, \lambda_{l2}, \dots, \lambda_{lD})$ and $\boldsymbol{Update}_l = (Update_{l1}, Update_{l2}, \dots, Update_{lD})$. To be concrete, let's assume there are three vectors of the estimates (i.e., three candidate solutions) and follow the similar fashion to Table 1, if at a certain step the computation is about estimating the second item's parameters $\lambda_{2,0}$ and $\lambda_{2,11}$, three vectors of the estimates $\boldsymbol{\lambda}_{l=1}$, $\boldsymbol{\lambda}_{l=2}$, and $\boldsymbol{\lambda}_{l=3}$ would produce three sets of the estimates for $\boldsymbol{\lambda} = (\lambda_{2,0}, \lambda_{2,11})$. Similarly, the vectors of update steps ($Update_l$) have the same vector format as the vectors of the estimates. Note that in parallel computing framework, each particle can be allocated to a processing unit such that multiple particles can be executed simultaneously. The vector of update steps ($Update_l$) is the changing step of $\boldsymbol{\lambda}_l$ from its current solution to a future one. In addition, as the PSO algorithm stores information from its iteration history, the optimal solution of particle *l* (local best; $\boldsymbol{\lambda}_{best_{lo}}$) and the optimal solution across all particles (global best; $\boldsymbol{\lambda}_{best_g}$) are used to guide velocity updates. Mathematically the iterative updating of velocity and solution can be expressed as:

$$Update_{lj}^{t} = w^{t} * Update_{lj}^{t-1} + c_1 r_1^{t}\left(\lambda_{best_{lj}}^{t-1} - \boldsymbol{\lambda}_{lj}^{t-1}\right) + c_2 r_2^{t}\left(\boldsymbol{\lambda}_{best_{gj}}^{t-1} - \boldsymbol{\lambda}_{lj}^{t-1}\right), \quad (10)$$

$$\lambda_{lj}^{t} = Update_{lj}^{t} + \lambda_{lj}^{t-1},$$

where $\lambda_{lj}^{t}$and $Update_{lj}^{t}$ are the vector *l* of the estimates and its corresponding vector of update steps at iteration *t*. Parameters c_1 and c_2 are learning/acceleration factors for the local and the global best solution vectors; these $\boldsymbol{c}$ exclusively situated in the range of 2 to 4. Parameter w^t is called inertia weight $(0 \leq w \leq 1)$ that can be adaptively changed along iterations. The function of *w* is, again, changing the direction of t $Update$. Finally, $r_1$and $r_2$are random numbers sampled from 0 to 1 independently. Tremendous variants of the PSO (hybrid PSO) have been proposed to improve the algorithm performance, for example, manipulating parameters $\boldsymbol{c}$ and w, mutation of the vector of the estimates, and adaptively tuning the vector of the estimates (Guedria, 2016; Lee & Ko, 2009; Maitra & Chatterjee, 2008).

In this dissertation, the proposed estimation is based upon the PSO, which has been widely used in machine learning fields but remains less-known in the psychometrics community. As discussed in this chapter, psychometric models such as the two that are referred to here having complex model specifications, which result in certain estimation difficulties. In addition, traditional approaches always involved deriving processes for obtaining the first- and the second-derivatives of the parameters of interest, which are mathematically demanding per se; if there are constraints adding onto the models, the difficulty of deriving processes becomes substantive. In practice, although many researchers and psychometricians have solid statistics backgrounds, they are not necessarily skilled enough to derive the mathematical formulas needed for the estimation. The PSO can be used to circumvent the requirement and provide precise results. The objective is integrating the PSO into psychometric model estimations. As addressed previously, psychometric models, such as LCDMs and MIRT models, have difficulties in their item parameter estimates due to the dimensionalities. The PSO is a stochastic derive-free technique that can be a feasible solution for the aforementioned problems.

Chapter 3: Method

The Particle swarm optimization (PSO) is also an important soft computing algorithm, which models the behavior of a flock of birds. It utilizes a population of particles to represent candidate solutions in a search space, and optimizes the problem by iteration to move these particles to the best solutions with regard to a given measure of quality. The PSO is customized to estimate the aforementioned models. Particularly, the details about embedding the PSO into the EM algorithm is provided with pseudo code and plain-text explanations. This customized PSO was constructed, tested, and compared via simulation studies in the R environment, in which multiple conditions were created through Monte Carlo approach and therefore the proposed algorithm is examined comprehensively such that instructional recommendations can be present.

Hybrid PSO-EM Algorithm for LCDM Estimation

The proposed algorithm is called the hybrid PSO-EM (HPSOEM) algorithm. As the name indicates, it integrates the properties of the hybrid PSO into the EM algorithm. That is, the hybrid PSO is used to replace the item parameter updates within the aforementioned M step. That is, not the entire M step is replaced by the PSO. Pseudocode of the HPSOEM algorithm for the LCDM is outlined in Figure 1. Explanation about the steps is present in the following paragraphs.

Step 2 outlines user-defined configurations of the HPSOEM algorithm: Meeting either condition- (1) maximum iteration number or (2) minimum variance of log-likelihood of all solution vectors' local optimum- would stop the estimation. Like any algorithm, setting the maximum iteration number is necessary in real estimation practice. The unique part of the proposed algorithm is using the minimum variance of log-likelihood of all solution vectors' local optimum to investigate converging status. The swarm effect brings particles to the space of the optimal solution such that eventually they all end up being identical.

Providing appropriate initial values, as mentioned in Steps 3 and 4, is helpful for starting the HPSOEM algorithm. The results obtained from the EM algorithm can be used to serve as starting values of one vector of the estimates. This vector of the estimates allows update steps to start from numerical space better than that of arbitrary. Note $\boldsymbol{Update}$ in Step 4 is essentially a matrix; each $\boldsymbol{Update}_l$ for $l = 1, \ldots, N_l$ within the matrix is a column vector contains a set of update step values. After obtaining item parameter estimates from PSO, $\hat{\boldsymbol{\lambda}}$, with Equations 4 and 8 one can calculate $\widehat{\pi_{pi}}$. Then the computation switches to E-step, that is, conditioning on $\boldsymbol{\pi}$, the posterior class probability for a respondent $H(c \mid \boldsymbol{y_p})$ is updated as:

$$H(c \mid \boldsymbol{y_p}) = \frac{v_c^{t-1} \prod_{i=1}^{I} \pi_{pi}(1 - \pi_{pi})^{1-y_{pi}}}{\sum_{c=1}^{N_c} v_c^{t-1} \prod_{i=1}^{I} \pi_{pi}(1 - \pi_{pi})^{1-y_{pi}}},$$

where again, subscripts *t, i, c,* and *p* represent the iteration, item, latent class, and person respectively; this is recorded in Step 6. Step 7 calculates the probability of membership v_c based upon $H\widehat{(c \mid \boldsymbol{y_p})}$ from the previous step via:

$$\hat{v}_c^t = \sum_{p=1}^{P} \frac{H(c \mid \boldsymbol{y_p})}{P}.$$

The marginal probability of class membership v_c is obtained by aggregating distribution on individual level.

```
1.  Begin with Iteration $t$=1
2.  Define algorithm parameters: (1) the number of iteration $Iter_{stop}$, (2) the number of vectors of the estimates (i.e., solution vectors) $N_l$, (3) the minimum variance of log-likelihood of solution vectors' local optimum $Var_{stop}$, (4) the list of model parameter constraints $List_{cons}$, (5) the penalty of violating constraints $LL_{penalty}$, (6) the maximum and minimum of the update direction parameter ($w_{max}$, $w_{min}$), (7) the constraint violation tolerance $Mutate_{count}$, and (8) the update correction parameters $(c_1, c_2)$
3.  Initialize $N_l$ $D$-dimensional PSOs and assign to all solution vectors' local optimum solutions $\boldsymbol{\lambda}_{best_l}$ to start M-step
4.  Initialize $N_l$ vectors of update steps $\boldsymbol{Update} = (\boldsymbol{Update}_1, \boldsymbol{Update}_2, \dots, \boldsymbol{Update}_{N_l})$
5.  Set all PSOs' constraint violation counts $\boldsymbol{Vio}_{count} = \left(Vio_{l1}, Vio_{l2}, \dots, Vio_{lN_{par}}\right)$ to 0
6.  Calculate the posterior class probability $H\left(c \mid y_p\right)$ for each respondent and there for to execute E-step
7.  Finish M-step calculate the probability of membership $v_c$ for each class
8.  Calculate the log-likelihood of the initialization and assign to candidate solutions' local optimum log-likelihood $\boldsymbol{LL}_{best_L} =$ $(LL_{best_{l1}}, LL_{best_{l2}}, \dots, LL_{best_{lN_{par}}})$
9.  Select global maximum $LL_{best_g}$ from $\boldsymbol{LL}_{best_l}$ and its estiamtes are saved to $\lambda_{best_g}$
10. While current iteration $t < Iter_{stop}$ or $Var\,(\boldsymbol{LL}_{best\_l}) > Var_{stop}$ do
11.   Calculate inertia weight $w^t = w_{max} - \frac{w_{max} - w_{min}}{Iter_{stop}} * t$
12.   For each vector of the estimates $l$ do
13.     If $Vio_l > Mutate_{count}$ do
14.       Current solution $\lambda_l^t$ is obtained by mutating from $\lambda_{best_l}$ and $\lambda_{best_g}$
15.       Current velocity $\boldsymbol{Update}_l^t$ is re-initialized
16.       $Vio_l$=0
17.     End If
18.     If $Vio_l \leq Mutate_{count}$ do
19.       Current vector of update steps $\boldsymbol{Update}_l^t$ and vector of the estimates $\boldsymbol{\lambda}_l^t$ are updated
20.     End If
21.     Update the probability of membership in latent classes $v$
22.     Calculate Log-likelihood of the current iteration $LL_l^t$
23.     If $x_l$ violates $List_{cons}$ do
24.       $Vio_l = Vio_l + 1$ and $LL_l = LL_l + LL_{penalty}$
25.     End If
26.     If $LL_l^t > LL_{best_l}$ do
27.       $\boldsymbol{\lambda}_{best_l} = \boldsymbol{\lambda}_l^t$
28.     End If
29.     If $LL_l^t > LL_{best_g}$ do
30.       $\boldsymbol{\lambda}_{best_g} = \boldsymbol{\lambda}_l^t$
31.     End If
32.   End For
33.   $t$= $t$+1
34. End While
```

Figure 1. Hybrid PSO-EM Algorithm Pseudo Code

In addition, when a solution violates the model constraint, its corresponding log-likelihood will be penalized to a certain degree. The larger the penalty is set, the less frequently the algorithm explores the solution vector's neighbor space. Update direction parameter (w) being set to be adaptive as Step 11 shows could "control the impact of the previous history of velocities on the current velocity and to influence the trade-off between global and local exploration abilities" of the updating particles (Kim & Li, 2011). In other words, balancing the explorations between global and local space, the adaptive strategy can effectively shorten converging time. The key element of the "hybrid" aspect is integrating the mutation idea, which is borrowed from evolutionary theory: mutation takes place when an organism needs to survive and have more offspring in a changing environment. In fact, this algorithm is named the evolutionary algorithm (EA). The essence is, if a vector of the estimates has violated the model constraint for a pre-defined count consecutively, as Step 14 shows, this solution vector will be replaced by mutating from its local optimum and global optimum estimates, while its vector of update steps will be reinitialized by random generation. The means of mutation can be found in EA literature (Zhang, Sun,& Tsang, 2005; Shukla, Hazela, Shukla, & Mishar, 2017). In this dissertation, mutation of a solution vector is created by randomly selecting a half of the solution vector from the local optimum and the other from the global optimum.

To better understand how the algorithm works, a tutorial-based but also simplifed example is provided here. Let the situation to be simple as five items ($N_i = 5$), two attributes ($N_\alpha = 4$), and four respondents ($N_p = 4$). The first two items measure the first attribute only, the third and the forth items measure the second attribute, and the last item measures both attributes. Given two attributes leading to three classes, the formula expressions can be seen from Table 3 and the parameters of interest $\boldsymbol{\lambda} = (\lambda_{1,0}, \lambda_{1,1}, \lambda_{2,0}, \lambda_{2,1}, \lambda_{3,0}, \lambda_{3,2}, \lambda_{4,0}, \lambda_{4,2}, \lambda_{5,0}, \lambda_{5,1}, \lambda_{5,2}, \lambda_{5,12}\ )$. If the number of candidate solution vector is 3 ($N_l = 3$), each of the solution vector contains estimates for $\boldsymbol{\lambda}$. The algorithm starts from the first iteration (t=1) by assigning some random values to (1) $\boldsymbol{\lambda}_{l=1}^{t=1}$, $\boldsymbol{\lambda}_{l=2}^{t=1}$, and $\boldsymbol{\lambda}_{l=3}^{t=1}$, (2) their vectors of update steps $\boldsymbol{Update}_{l=1}^{t=1}$, $\boldsymbol{Update}_{l=2}^{t=1}$, and $\boldsymbol{Update}_{l=3}^{t=1}$, (3) $v_c^{t=1}$ for c=1,…, 3. Assume the log-likelihood values of three vectors of the estimates at t=1 were -80,-90, and -70, then the global optimal solution was $\boldsymbol{\lambda}_{l=3}^{t=1}$, where the local optimal solutions were simply $\boldsymbol{\lambda}_{l=1}^{t=1}$, $\boldsymbol{\lambda}_{l=2}^{t=1}$, and $\boldsymbol{\lambda}_{l=3}^{t=1}$, given there was only one record in each iteration history. The wining solution vector $\boldsymbol{\lambda}_{l=3}^{t=1}$ was proceeded to execute the E-step and M-step at t=2; that is, $v_c^{t=2}$ and $H(c \mid \boldsymbol{y_p})^{t=2}$ for c=1,…, 3. The ***Update*** vectors were altered using the local and global optimal solution vectors, for example, $\boldsymbol{Update}_{l=1}^{t=2}$ was changed by $\boldsymbol{\lambda}_{l=1}^{t=1}$ and $\boldsymbol{\lambda}_{l=3}^{t=1}$, in addition to c, r and w parameters. With the functionalities of $\boldsymbol{Update}_{l=1}^{t=2}$, $\boldsymbol{\lambda}_{l=1}^{t=2}$ was updated and similar idea applies to other two solution vectors. If the log-likelihood values of three vectors of the estimates at t=2 were -88,-60, and -65, the global optimal solution vector became $\boldsymbol{\lambda}_{l=2}^{t=2}$, where the local optimal solution vectors for l=1,2,3 became $\boldsymbol{\lambda}_{l=1}^{t=1}$, $\boldsymbol{\lambda}_{l=2}^{t=2}$, and $\boldsymbol{\lambda}_{l=3}^{t=2}$. Assume at iterations 10 to 15 that $\boldsymbol{\lambda}_{l=1}$ had failed to yield a larger log-likelihood value than its local optimal solution in the iterating history, a new $\boldsymbol{\lambda}_{l=1}^{t=16}$ would be constructed via the aforementioned EA procedure. To emphasize, this paragraph skips several steps in Figure 1 for illustration purpose.

Table 3. Formula Expression Example of a Log-linear Cognitive Diagnosis Model

Item	α_1	α_2	Simplified Expression
1	1	0	$\lambda_{1,0} + \lambda_{1,1}$
2	1	0	$\lambda_{2,0} + \lambda_{2,1}$
3	0	1	$\lambda_{3,0} + \lambda_{3,2}$
4	0	1	$\lambda_{4,0} + \lambda_{4,2}$
5	1	1	$\lambda_{5,0} + \lambda_{5,1} + \lambda_{5,2} + \lambda_{5,12}$

Hybrid PSO-EM Algorithm for MIRT Estimation

As mentioned earlier, MIRT models contains an integral over $\boldsymbol{\theta}_{\boldsymbol{p}}$ where closed-forms do not exist. To handle the issue, numerical approximation approach- generating and evaluating quadrature points-is adopted. In the uni-dimensional IRT framework, the quadrature points can be selected simply from -4 to 4 in increments of 0.2 such that 99.9% of the probability mass is covered. In other words, in the uni-dimensional case, $P(\boldsymbol{Y}_p = \boldsymbol{y}_{\boldsymbol{p}})$ in Equation 7 can be re-written as $\sum_{q1=1}^{N_{qd}} \prod_{i=1}^{N_i} \pi_{pi}^{y_{pi}} (1 - \pi_{pi})^{1-y_{pi}} A(X_{q1})$, where N_{qd} is 40. Similarly, in MIRT models, the quadrature points from multidimensional space should be generated and evaluated. However, instead of taking the values from a continuum, the quadrature points in MIRT should be sampled from a grid constructed by all attributes. If $N_a$is two, the grid becomes a plane where x-axis holds the points of the first attribute and y-axis holds those of the second attribute. When N_a is larger than three, the grid becomes a hyper-plane. As $\boldsymbol{\theta}$ is assumed to follow $MVN(\mathbf{0}, \boldsymbol{\Sigma}_{\mathbf{p}})$ which allows attributes across dimensions to be sampled simultaneously, the approximation of $P(\boldsymbol{Y}_p = \boldsymbol{y}_{\boldsymbol{p}})$ can be simplified to $\sum_{q=1}^{N_{qd}} \prod_{i=1}^{N_i} \pi_{pi}^{y_{pi}} (1 - \pi_{pi})^{1-y_{pi}} \boldsymbol{A}(\boldsymbol{X}_{\boldsymbol{q}})$, where $\boldsymbol{A}(\boldsymbol{X}_{\boldsymbol{q}})$, the corresponding weights as a set of normalized ordinates of the quadrature points from the population distribution $G(\boldsymbol{\theta}_{\mathbf{q}})$, can be defined as $G(\boldsymbol{X}_{\boldsymbol{q}}) / \sum_{q=1}^{N_{qd}} G(\boldsymbol{X}_{\boldsymbol{q}})$.

Bock and Aitkin (1981) further derived the height of the posterior distribution at quadrature point $\boldsymbol{X}_{\boldsymbol{q}}$ for a given respondent *p* at an item *i* can be approximated via:

$$P(\boldsymbol{X}_q | \boldsymbol{y}_{\boldsymbol{p}}) \approx \frac{\pi_{pi}^{y_{pi}} (1 - \pi_{pi})^{1-y_{pi}} \boldsymbol{A}(\boldsymbol{X}_{\boldsymbol{q}})}{P(\boldsymbol{Y}_p = \boldsymbol{y}_{\boldsymbol{p}})}. \tag{8}$$

On the other hand, give the complete data log-likelihood for the item parameters $\boldsymbol{\lambda}$ can be expressed as:

$$LogL(\boldsymbol{\lambda}) \approx \sum_{p=1}^{N_\mathrm{p}} \sum_{i=1}^{N_i} y_{pi} \log(\pi_{pi}) + \sum_{p=1}^{N_\mathrm{p}} \sum_{i=1}^{N_i} (1 - y_{pi}) \log(1 - \pi_{pi}), \tag{9}$$

The conditional expected complete data likelihood given item parameters can be approximated by:

$$L \approx \sum_{p=1}^{N_\mathrm{p}} \sum_{q=1}^{N_{qd}} \sum_{i=1}^{N_i} y_{pi} \log(\pi_{qi}) P(\boldsymbol{X}_q | \boldsymbol{y}_{\boldsymbol{p}}) + \sum_{p=1}^{N_\mathrm{p}} \sum_{i=1}^{N_i} \sum_{q=1}^{N_{qd}} (1 - y_{pi}) \log(1 - \pi_{qi}) P(\boldsymbol{X}_q | \boldsymbol{y}_{\boldsymbol{p}}). \tag{10}$$

Note that in the MIRT context, the number of parameter constraints is less than that of the LCDMs. In particular, MIRT models merely require main effects to be non-negative, where the LCDMs also set dependencies on interaction terms, if there is any. Given there is no class membership in MIRT models, Step 20 in Figure 1becomes inappropriate; this line should, instead, be placed by generating and evaluating quadrature points as Equations 8 and 9 illustrate.

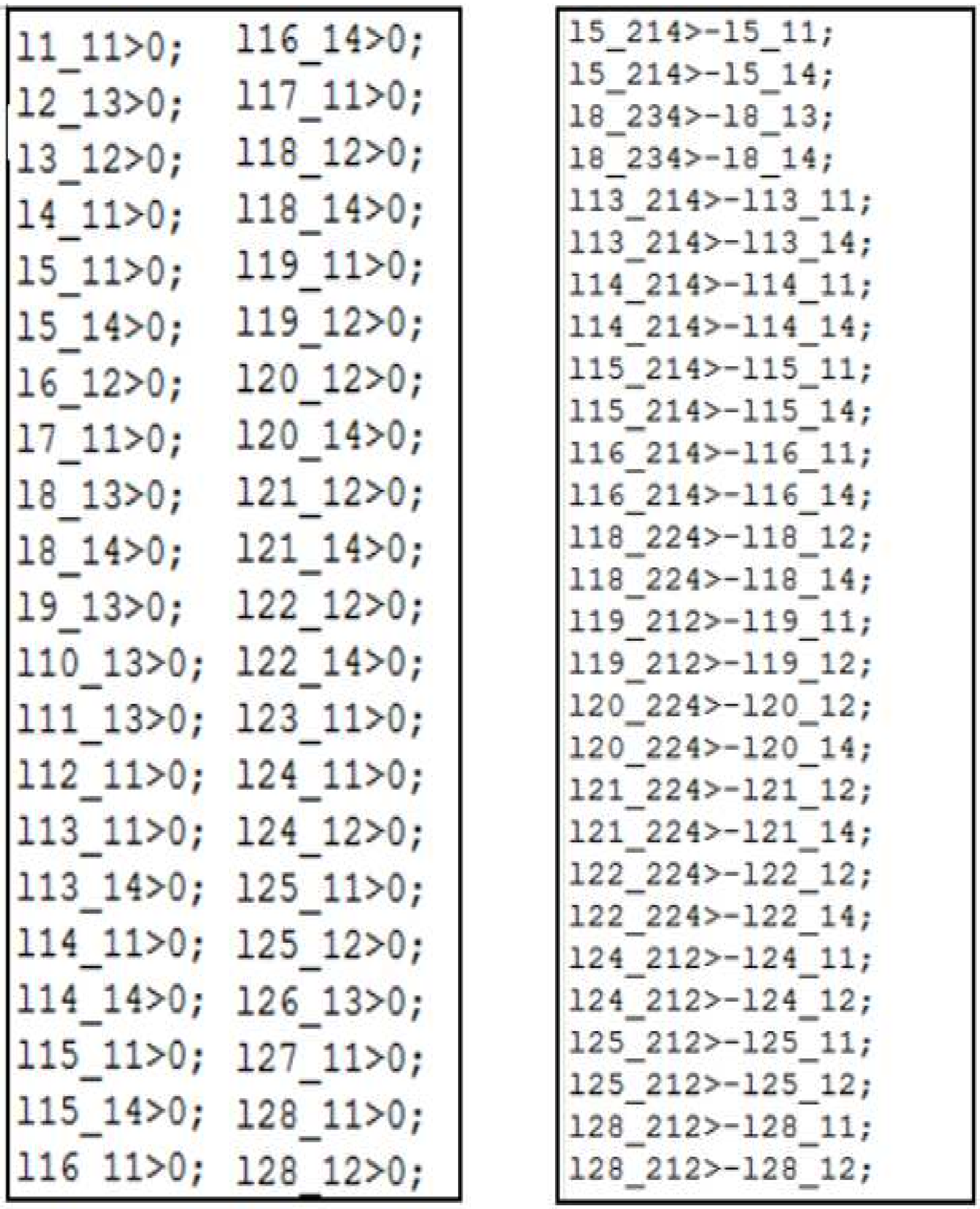
Data Generation

Simulation studies were conducted to examine the application of the HPSOEM to psychometric model estimation. The simulations are based upon the Q-matrix provided in Templin and Bradshaw (2014; reproduced in Table 4). As one finds, there are four attributes and 28 items in total. Each item measures one or more attributes; that is, indicators can be cross-loaded in multiple latent traits simultaneously, for example, Item 1 measures the first attribute only, while Item 22 measures the second and the forth attributes. Provided the Q-matrix, LCDMs and MIRTs were selected to be the simulation frameworks.

In the first study, responses (i.e., simulated datasets) were generated via LCDMs. Particularly, item intercepts were randomly generated from [-1, 1], main effects were drew uniformly from [1.5, 3], and interaction terms were sampled from a uniform distribution of which range is [-1, 1.5]. The situations where item parameters violate the aforementioned constraint rules, the generation would re-start until the values produced are in permissible numeric space. The constraint rules for the Q-matrix can be found in Appendix 1. Given the number of attribute is 4 and the sum of membership probabilities is 1, there are 16 classes in total and the probability of membership was set equal (i.e., [1/16, 1/16, …, 1/16]).

The second study used MIRT models to generate responses. Item parameters were produced in an identical way to those of the first study. Different from LCDMs, MIRT models assume that attributes follow a multivariate normal distribution. Therefore, in the second simulation study, the latent attributes were generated from multivariate normal distributions. For simplicity purposes, the means of the distribution were all set to 0, the variance components were all set to 1, and all covariance components (i.e., correlations) were set to 0.6.

Note that the given Q-matrix in the LCDM context implies parameter constraints listed in Figure 2. The expression rules follow the conventions proposed by Rupp, Templin, and Hensen (2010, p. 206). That is, (1) *l* simply represents λ, (2) the number before symbol _ indicates item number, (3) the first number after symbol _ represents item effect name, where 0,1,*x* are the labels of intercept, main effect, and *x-way* interaction effects respectively, (4) the remaining numbers identify items that contain attribute interaction, if there is any. To illustrate, *l9_0* represents the intercept of Item 9 and *l9_213* represents the 2-way interaction effects between the first and the third attributes. According to Rupp, Templin, and Henson (2010), in addition to ensuring the non-negativity of the main effects that are shown in the left panel of Figure 2, the interaction constraints are also set in the right panel. For the MIRT models, only left panel applies.



```
l1_11>0;   l16_14>0;
l2_13>0;   l17_11>0;
l3_12>0;   l18_12>0;
l4_11>0;   l18_14>0;
l5_11>0;   l19_11>0;
l5_14>0;   l19_12>0;
l6_12>0;   l20_12>0;
l7_11>0;   l20_14>0;
l8_13>0;   l21_12>0;
l8_14>0;   l21_14>0;
l9_13>0;   l22_12>0;
l10_13>0;  l22_14>0;
l11_13>0;  l23_11>0;
l12_11>0;  l24_11>0;
l13_11>0;  l24_12>0;
l13_14>0;  l25_11>0;
l14_11>0;  l25_12>0;
l14_14>0;  l26_13>0;
l15_11>0;  l27_11>0;
l15_14>0;  l28_11>0;
l16 11>0;  l28_12>0;
```

```
l5_214>-l5_11;
l5_214>-l5_14;
l8_234>-l8_13;
l8_234>-l8_14;
l13_214>-l13_11;
l13_214>-l13_14;
l14_214>-l14_11;
l14_214>-l14_14;
l15_214>-l15_11;
l15_214>-l15_14;
l16_214>-l16_11;
l16_214>-l16_14;
l18_224>-l18_12;
l18_224>-l18_14;
l19_212>-l19_11;
l19_212>-l19_12;
l20_224>-l20_12;
l20_224>-l20_14;
l21_224>-l21_12;
l21_224>-l21_14;
l22_224>-l22_12;
l22_224>-l22_14;
l24_212>-l24_11;
l24_212>-l24_12;
l25_212>-l25_11;
l25_212>-l25_12;
l28_212>-l28_11;
l28_212>-l28_12;
```

Figure 2. Parameter Constraints of the Q-matrix

Table 4. Q-matrix used for the Simulation Study

Item No.	Attribute1	Attribute2	Attribute3	Attribute4
1	1	0	0	0
2	0	0	1	0
3	0	1	0	0
4	1	0	0	0
5	1	0	0	1
6	0	1	0	0
7	1	0	0	0
8	0	0	1	1
9	0	0	1	0
10	0	0	1	0
11	0	0	1	0
12	1	0	0	0
13	1	0	0	1
14	1	0	0	1
15	1	0	0	1
16	1	0	0	1
17	1	0	0	0
18	0	1	0	1
19	1	1	0	0
20	0	1	0	1
21	0	1	0	1
22	0	1	0	1
23	1	0	0	0
24	1	1	0	0
25	1	1	0	0
26	0	0	1	0
27	1	0	0	0
28	1	1	0	0

Independent Variables

The parameters consisted in the HPSOEM were controlled in the simulation studies: (1) the number of particles N_{par}, (2) the penalty of violating constraints $LL_{penalty}$, and (3) the particle updating parameters (c_1, c_2). Note that inertia weight IS NOT an independent variable in the simulation studies, because it has been proved that setting to adaptive inertia weight works more efficient than other combinations across various computational tasks (Kessentini & Barchiesi, 2015; Rezaee & Jasni, 2013; Kim & Li, 2011). Let w^t=$w_{max} - \frac{\mathrm{Current_{iter}}}{\mathrm{N_{iter}}}(w_{max}, w_{min})$, while (w_{max}, w_{min}) was set to (0.9, 0.4). To make simulation studies manageable as well as meaningful practically, the conditions of independent variables, listed in Table 5, were selected based upon configurations suggested by published works (Clarke, Al-Abdeli, & Kothapalli, 2014; Malekpour, & Seifi, 2010). In particular, $LL_{penalty}$ was presented as multipliers: the actual penalty was calculated by multiplying the pre-defined values to log-likelihood of current iteration. Note that setting the multiplier to zero is equivalent to no penalty. In total, there are 3x3x3=27 simulation conditions.

Table 5. Independent Variables of the Simulation Studies

Variable	Pre-defined Values
N_{par}	[50, 100, 200]
$LL_{penalty}$ Multiplier	[0, 0.5, 1]
$c(c_1 = c_2)$	[0.5, 1, 2]

Software and Hardware

R environment (R Core Team, 2017) was used to conduct the simulation study. Currently R is one of the world's most popular programming languages due to its cost-free property, flexible extensions, rapid package updates, and active community supports (Robers, Best, Dunn, Treml, & Halpin, 2010). Throughout the paper, data generation and the algorithm comparisons were executed in R. The HPSOEM stopping criteria was set to either (1) the variance of particles becomes less than 0.01 or (2) the number of iterations reaches to 10000. On the other hand, M*plus*, known as a toolkit for numerous statistical estimations, has been widely cited in a large body of published social and psychological research works (see DeMars, 2016; Eckes& Baghaei, 2015; Matlock, Turner, & Gitchel, 2016). Appendix 2 shows the R code for the proposed algorithm. To verify the precisions and utility of the proposed algorithm, M*plus* was used to estimate both LCDMs and MIRTs on the simulated data. The M*plus* stopping criteria was set to either (1) the log-likelihood change from last iteration becomes less than 0.001 or (2) the number of iterations reaches to 10000. In order to execute M*plus* in R environment, a packaged called *MplusAutomation* (Hallquist, & Wiley, 2011) was implemented. *MplusAutomation* enables R to communicate with M*plus* such as streamlining Monte Carlo simulation studies and the comparisons of many models can become plausible. Specifically, *MplusAutomation* provides routines to 1) create and manage syntax for groups of related models, 2) automate the estimation of many models; and 3) provide tools to extract and compare model fit statistics, parameter estimates, and present model outputs.

In terms of hardware, the machine used in the simulation tasks was a Lenovo IdeaPad with 16GB RAM and a 2.6 GHz i7 6th Gen 4-core Intel processor as well as NVIDIA GeForce GTX 960M GPU. Given the availability of multiple cores, parallel computing was set default in both R and M*plus*. Note that in R, the Graphic Processing Units (GPUs) were implemented to replace the Central Processing Units (CPUs) as recent studies have shown that the GPUs are more efficient when basic math calculation is executing in a parallel computing facility

Dependent Variables

To understand the estimation accuracy of the proposed estimation method and constrained EM algorithm in M*plus*, (1) model parameter bias and relative mean squared error (RMSE), (2) log-likelihood values, (3) the number of iteration to convergence and computational time were recorded for each replication. As each given condition was replicated 500 times, the results were represented by the means of the replications. In particular, the bias is calculated as:

$$Bias_{\beta} = \frac{\sum_{r=1}^{R}\sum_{i=1}^{Ne}\left(\hat{\beta}_{ir} - \beta_i\right)}{RN} = \bar{\hat{\beta}}_{ir} - \beta_i.$$

and RMSE is obtained by

$$RMSE_{\beta} = \sqrt[2]{\frac{\sum_{r=1}^{R}\sum_{i=1}^{Ne}\left(\hat{\beta}_{ir} - \beta_i\right)^2}{RN}}$$

where *Ne* is the number of elements in the set of β and R is the number of replication. Note that the number of iteration to convergence and computational time are essentially measuring the same quality- the speed of the proposed algorithm; these values would expected to have substantive differences on different machines. The results are present in the following chapter.

Chapter 4: Results

Accordingly Table 5, there were 3x3x3=27 simulation conditions . This chapter begins with the LCDM simulation followed by those of the MIRT models. Using Monte Carlo simulation, a computerized mathematical technique that allows people to account for unknown qualities of an estimation or approach in quantitative analysis and decision-making, was implemented to examine the features of the proposed estimation. The outcomes are presented with the following order: (1) relative item parameter bias and relative mean squared error (RMSE), (2) log-likelihood, (3) the number of iteration to convergence as well as computational time. For reference purpose, M*plus* results were demonstrated along with those of the proposed algorithm. The complete results can be found in Table 6. However, Table 6 is complex as the outcomes were listed in a multidimensional setting. To understand the results better, in this chapter, the results are be addressed case by case. Along with the results, recommendations are provided such that the instructional values of the current simulation could be emphasized.

Table 6. Complete Simulation Across 27 Conditions and 3 Outcomes.

Updating Parameter	Penalty Multiplier	Particle Number	Iteration Number		Convergence Time (mins)	
			LCDM	MIRT	LCDM	MIRT
c=0.5	$LL_{penalty} = 0.0$	$N_{par} = 50$	382	469	87.94	147.83
c=0.5	$LL_{penalty} = 0.0$	$N_{par} = 100$	343	414	80.79	135.3
c=0.5	$LL_{penalty} = 0.0$	$N_{par} = 200$	338	400	78.4	131.74
c=0.5	$LL_{penalty} = 0.5$	$N_{par} = 50$	309	376	72.85	129.6
c=0.5	$LL_{penalty} = 0.5$	$N_{par} = 100$	292	376	65.11	126.44
c=0.5	$LL_{penalty} = 0.5$	$N_{par} = 200$	285	339	65.40	121.03
c=0.5	$LL_{penalty} = 1.0$	$N_{par} = 50$	331	407	79.19	135.1
c=0.5	$LL_{penalty} = 1.0$	$N_{par} = 100$	299	354	70.34	126.58
c=0.5	$LL_{penalty} = 1.0$	$N_{par} = 200$	294	346	68.88	122.66
c=1.0	$LL_{penalty} = 0.0$	$N_{par} = 50$	356	396	81.88	132.14
c=1.0	$LL_{penalty} = 0.0$	$N_{par} = 100$	324	385	74.62	126.58
c=1.0	$LL_{penalty} = 0.0$	$N_{par} = 200$	315	353	71.92	126.43

c=1.0	$LL_{penalty} = 0.5$	$N_{par} = 50$	289	354	65.52	123.52
c=1.0	$LL_{penalty} = 0.5$	$N_{par} = 100$	255	309	58.00	115.97
c=1.0	$LL_{penalty} = 0.5$	$N_{par} = 200$	252	308	59.32	112.2
c=1.0	$LL_{penalty} = 1.0$	$N_{par} = 50$	305	357	67.84	123.88
c=1.0	$LL_{penalty} = 1.0$	$N_{par} = 100$	271	356	62.14	124.59
c=1.0	$LL_{penalty} = 1.0$	$N_{par} = 200$	268	316	62.04	118.41
c=2.0	$LL_{penalty} = 0.0$	$N_{par} = 50$	362	451	84.53	143.14
c=2.0	$LL_{penalty} = 0.0$	$N_{par} = 100$	324	383	75.41	127.84
c=2.0	$LL_{penalty} = 0.0$	$N_{par} = 200$	319	356	72.26	121.8
c=2.0	$LL_{penalty} = 0.5$	$N_{par} = 50$	273	316	63.07	115.73
c=2.0	$LL_{penalty} = 0.5$	$N_{par} = 100$	253	291	57.19	108.06
c=2.0	$LL_{penalty} = 0.5$	$N_{par} = 200$	244	301	61.51	112.29
c=2.0	$LL_{penalty} = 1.0$	$N_{par} = 50$	312	350	74.59	124.53
c=2.0	$LL_{penalty} = 1.0$	$N_{par} = 100$	273	357	64.93	122.43
c=2.0	$LL_{penalty} = 1.0$	$N_{par} = 200$	270	311	67.77	118.01

LCDM Results

Across all 27 conditions, the biases and the RMSEs do not have systematic differences and therefore the results were collapsed into one set- HPSOEM as seen in Table 7. Overall both M*plus* and the HPSOEM yielded similar item parameter estimates, while in some situations one is better than the other; to be concrete, the differences of the absolute values of biases for intercepts, main effects, and interactions effects are 0.001 (0.01-0.011), 0.002 (0.007-0.005), and 0.027 (0.057-0.030). It can be seen that, although not substantively, the HPSOEM seems to handle the interaction effects slightly better. For both algorithms, all biases are below 0.06 and all RMSEs are lower than 0.2 As Table 7 shows. It can be seen that both intercept and main effect estimates have smaller biases than interaction effect ones; In particular, slightly unsatisfactory results were found in the interaction effects that led the bias to above 0.05. These findings are consistent with Templin and Bradshaw (2014). The biases of class membership probability estimates are even more negligible as the values are below 0.005. Unsurprisingly the corresponding RMSE is 0.009. From Table 5, one can claim that the HPSOEM can produce results as accurate as M*plus* does.

Table 7. Independent Variables of the LCDM Simulation Study

	Intercepts		Main Effects		Interaction Effects	
	M*plus*	HPSOEM	M*plus*	HPSOEM	M*plus*	HPSOEM
Bias	-0.010	0.011	0.005	-0.007	0.057	0.030
RMSE	0.151	0.184	0.212	0.155	0.178	0.209

In addition to the investigation on parameters, log-likelihood difference between two algorithms was also monitored. Similar to the biases and the RMSEs, the log-likelihood values across 27 conditions only showed ignorable differences and therefore were collapsed. Listed in Figure 3, within 95% confidence interval, the difference ranges from -1.47 to 1.65. That said, at 5% α-level, the HPSOEM log-likelihood is not statistically different from that of M*plus*.

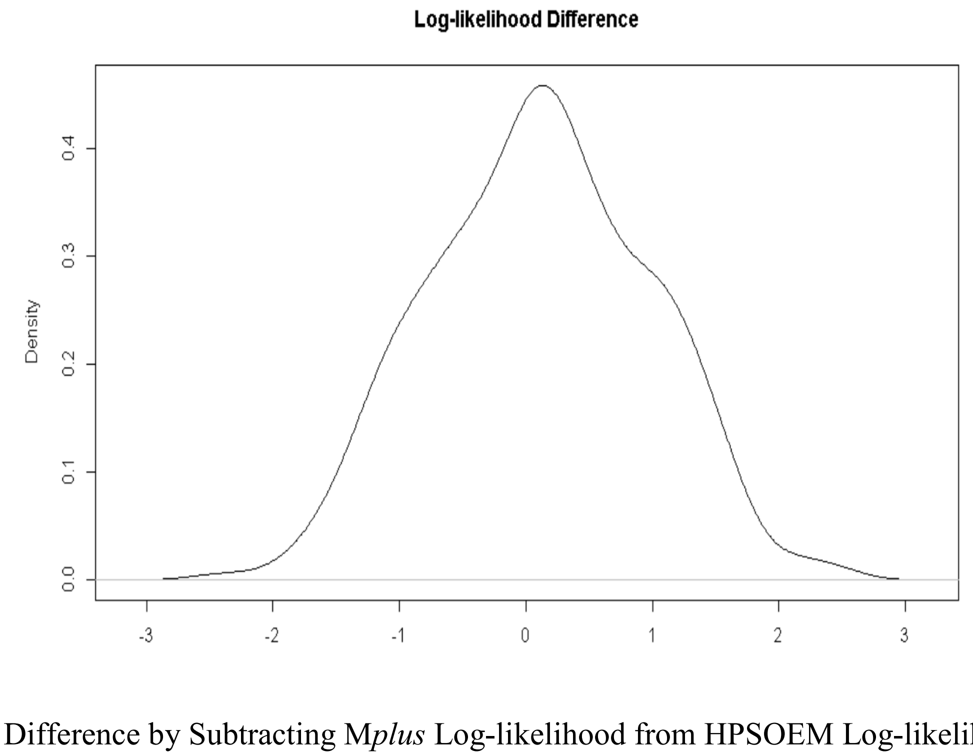


Figure 3. Difference by Subtracting M*plus* Log-likelihood from HPSOEM Log-likelihood for LCDM Simulation Study

Compared with the previous two outcomes that do not have much variability across different conditions, the number of iteration to convergence and the computational time do show discrepancies from condition to condition. The upper, middle, and lower panel of Table 8 show the main effects of N_{par}, $LL_{penalty}$ Multiplier, and $\boldsymbol{c}$ on both simulation dependent variables -the number of iterations to convergence and the computational time respectively. Start from the upper panel, one can find that $N_{par} = 50$ requires more iterations and therefore longer time to converge than the other two conditions. In particular, to reach convergence, $N_{par} = 100$ and $N_{par} = 200$ need 31 less iterations and costs 7 less minutes than those of $N_{par} = 50$. However, $N_{par} = 100$ and $N_{par} = 200$ do not differ significantly as their numbers of iterations to convergence and the computational time are nearly identical. It can be concluded that a larger N_{par} leads to a faster convergence until it reaches a certain sufficient level (i.e., 100 at the current example).

The second main effect of the independent variable is $LL_{penalty}$ Multiplier. It can be found that putting no penalties causes extra computational power and time in estimating models; it may due to the reason that an un-ignorable proportion of computation was spent on impermissible numeric space. On the other hand, setting the multiplier to 1 seems to be less efficient than 0.5 as the differences in the iteration number and convergence time are 20 and 6 minutes respectively. That said, among three penalty choices, setting the multiplier to 0.5 yields the fastest speed. The impact of particle updating parameters $\boldsymbol{c}$ on iteration number follows a monotonic order: as $\boldsymbol{c}$ increases from 0.5 to 2.0, the iteration number decreases from 319 to 292. The convergence time, however, doesn't show the same monotonicity consistency as the iteration number does.

Table 8. Computation Speed Results of the LCDM Simulation Study

	$N_{par} = 50$	$N_{par} = 100$	$N_{par} = 200$
Iteration Number	324	293	287
Convergence Time(mins)	75	68	68
	$LL_{penalty}$ Multiplier =0	$LL_{penalty}$ Multiplier =0.5	$LL_{penalty}$ Multiplier =1.0
Iteration Number	340	272	291
Convergence Time(mins)	79	63	69
	$\boldsymbol{C = 0.5}$	$\boldsymbol{C = 1.0}$	$\boldsymbol{C = 2.0}$
Iteration Number	319	293	292
Convergence Time(mins)	74	67	69

MIRT Results

Similar to LCDM results, the biases and the RMSEs were collapsed into one set named as HPSOEM as seen in Table 9, due to no systematic differences across all 27 conditions. Again, overall both M*plus* and HPSOEM yielded similar item parameter estimates, while in some situations one was better than the other. For both algorithms, all biases are below 0.062 and RMSEs are lower than 0.281. The pattern that both intercept and main effect estimates have smaller biases than interaction effect ones is found again in the MIRT simulation. However, compared with those of the LCDM simulation, the values of both biases and RMSEs are larger, despite the discrepancies are relatively ignorable. The potential reason for the differences is the numerical approach for approximating the integral part. The biases of the correlation matrix

$\mathbf{\Sigma_p}$ range from 0.007 to 0.012 and the maximum of RMSEs is 0.022. Overall, the HPSOEM can yield accurate and efficient estimates for both item parameters and latent structure parameters.

Table 9. Independent Variables of the LCDM Simulation Study

	Intercepts		Main Effects		Interaction Effects	
	M*plus*	HPSOEM	M*plus*	HPSOEM	M*plus*	HPSOEM
Bias	-0.020	0.032	-0.012	0.047	0.062	0.054
RMSE	0.191	0.281	0.260	0.275	0.192	0.133

MIRT log-likelihood results were recorded as the LCDM simulation study did. Due to the same reason that the log-likelihood values across 27 conditions only show ignorable differences, these results were collapsed. Figure 4 shows the log-likelihood differences between two estimation approaches. Compared with that of the LCDM simulation, the distribution in Figure 4 does not have a smooth bell-curve shape. Within 95% confidence interval, the HPSOEM log-likelihood is not statistically different from that of M*plus* because the difference ranges from -5.16 to 1.44. That said, at 5% α-level, the HPSOEM log-likelihood is not statistically different from that of M*plus*. However, compared with that of the LCDM simulation, the log-likelihood gap in the current simulation is larger. Besides, a large proportion of HPSOEM log-likelihood ends up being lower than that of M*plus*.

It is not surprisingly that the independent variables have impact on the convergence and the computational time, similar to what was demonstrated in the LCDM simulation. Table 10 lists the impacts on the two outcomes of interests; the upper panel shows the main effect of N_{par}, the middle panel is about $LL_{penalty}$ Multiplier, finally the lower panel indicates the impact of particle updating parameters-$\boldsymbol{c}$. The main effect of N_{par} on iteration number has a monotonic trend: the iteration number decreases from 386 to 337 when $N_{par} = 50$ boosts to $N_{par} = 200$. On the other hand, the convergence time does not differ substantively; particularly the time for N_{par}=100 is nearly identical to N_{par}=200. This phenomena is reasonable because, even though N_{par}=200 takes only 337 iterations to converge averagely, the time of each iteration for a larger size of particles tend to be longer. The second main effect of the independent variable, $LL_{penalty}$ Multiplier, shows the identical pattern as seen in the LCDM simulation. That is, both setting no penalties and oversized penalties could lead to extra computational power and time in estimating models. The iteration number and the convergence time for $LL_{penalty}$ Multiplier =0 are 401 and 133 minutes which are 49 more iterations and 9 more minutes than $LL_{penalty}$ Multiplier=1.0, and 69 more iterations and 11 more minutes than $LL_{penalty}$ Multiplier=0.5. The effect of particle updating parameters $\boldsymbol{C}$ on the iteration number follows a monotonic order: as $\boldsymbol{C}$ increases from 0.5 to 2.0, the iteration number decreases from 387 to 346. Nevertheless, the convergence time again shows a different pattern from that of the iteration number as the time had nearly no changes between $\boldsymbol{C = 1.0}$ and $\boldsymbol{C = 2}$.

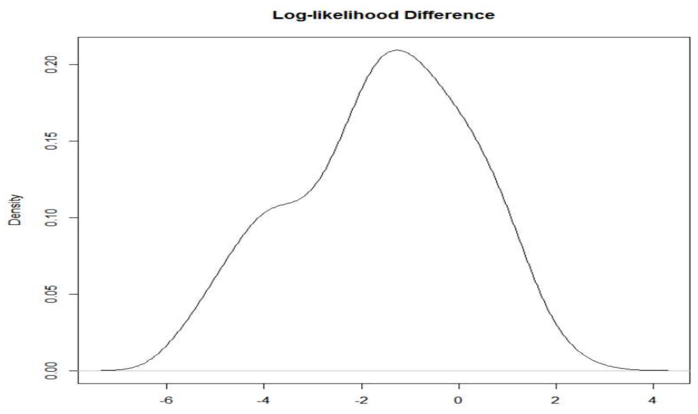


Figure 4. Difference by Subtracting M*plus* Log-likelihood from HPSOEM Log-likelihood for MIRT Simulation Study

Table 10. Computation Speed Results of the MIRT Simulation Study

	$N_{par} = 50$	$N_{par} = 100$	$N_{par} = 200$
Iteration Number	386	358	337
Convergence Time(mins)	131	124	121

	$LL_{penalty}$ Multiplier =0	$LL_{penalty}$ Multiplier =0.5	$LL_{penalty}$ Multiplier =1.0
Iteration Number	401	330	350
Convergence Time(mins)	133	118	124

	$\boldsymbol{C = 0.5}$	$\boldsymbol{C = 1.0}$	$\boldsymbol{C = 2.0}$
Iteration Number	387	348	346
Convergence Time(mins)	131	123	122

Chapter 5: Discussion

In particle swarm optimization (PSO) the set of candidate solutions to the optimization problem is defined as a swarm of particles, which may flow through the parameter space defining trajectories that are driven by their own and neighbors' best performances. Integrating the PSO to the EM algorithm, the proposed estimation was shown to be an accurate approach for estimating both LCDMs and MIRT models through simulation studies. Taking stochastic process and swarm behavior into consideration, the HPSOEM is able to overcome the problems of local maxima and label switching that the EM algorithm (without constraints) encounters. Based upon the simulation results, recommendations about tuning the proposed algorithm and conclusions about the algorithmic utility are given below:

- Increasing the number of particles N_{par} doesn't necessarily yields neither faster convergence nor more accurate estimations; it has a ceiling effect such that when N_{par} reaches to a certain sufficient level, the computational speed becomes stable.
- Setting no penalties for the parameter constraints of a model would waste computational efforts in exploring impermissible numeric space, where overwhelming penalties would also cause stochastic search jumps unexpectedly farer such that the optimal solutions could be skipped frequently.
- Updating parameters work similar to N_{par}: that a larger updating parameter set is able to improve the estimation speed, while the ceiling effect does occur when the updating parameters become too large.
- Iteration number doesn't necessarily reflect convergence time as one can find from Table 6 and Table 8. For an estimation with a large size of particles, it takes longer time to complete an iteration.

- Estimating MIRT models takes longer time than estimating LCDMs because the integral approximation consumes computational power to evaluate.
- The HPSOEM seems to produce more accurate results for LCDMs than for MIRT models. A primary reason is that the integral approximation implemented in the MIRT simulation study is a naïve version of the approximation technique.
- In the current simulation studies, the combination that ***c*=1 or 2**, $LL_{penalty}$ Multiplier =0.5, and N_{par}=100 is more appropriate than other configuration combinations. However, it is not necessarily the standard for all other models. With a less complicated model, the optimal combination may alter.

Meanwhile, as mentioned previously, the naïve version of approximation technique was used in constructing the proposed algorithm. This practice is sufficiently useful for lower-dimensional latent space, but often fails to produce satisfactory results for those with a larger dimension number. The reason is that, by evenly pining quadrature points from the latent space, the naïve approximation doesn't take the importance of each quadrature point into consideration. To improve the approximation accuracy while maintain the number of quadrature points, Schilling and Bock (2005) demonstrated how adaptive quadrature could be used in a high-dimensional model. Essentially, the adaptive quadrature points are produced with mean and covariance adjustments at each iteration of the EM algorithm such that latent space of more important area can be emphasized and that of less important area releases more efforts. As a result, fewer quadrature points are needed to yield an accurate fast-converging solution. More recently, stochastic estimation approaches have been deployed to replace the practice of using quadrature. In addition to aforementioned Bayesian approaches in Chapter 1, Delyon, Lavielle, and Moulines (1999) used a stochastic averaging procedure to replace the integration.

In addition to parameter recovery, the probability of Type I error was also calculated at 0.05 nominal α level: using the standard errors of the estimates, one can construct confidence intervals for the estimated variance and covariance components. For example, multiplying the standard error of the estimate with 1.96, one can obtain a 95 % confidence interval. A criterion for examining the standard error is assuring that the true parameters are located within the confidence intervals. Both LCDMs and MIRT models are based upon logistic regression model whose standard errors of the coefficients are the square roots of the diagonal entries of the covariance matrix. For all simulation conditions via M*plus* and the HPSOEM, the Type I error rates ranged from 0.059 to 0.042 which are fairly close to 0.05, although the interaction effects tend to have lower Type I error due to the larger standard errors they have. Expectedly, the standard errors do not differ between two estimation approaches as they are both maximizing the aforementioned likelihood values.

As all other studies, this dissertation has several limitations. The simulation designs, although containing 27 conditions, still have large room to explore. Above all, the effect sizes that were used to generate responses were pre-defined. These effect sizes match the values demonstrated in literature (Harwell, Stone, Hsu, &Kirisci, 1996), but in certain situations such as extremely large and/or small effect size of the item parameters would still occur. In addition, there was only one Q-matrix being used for specifying item parameters. It is known that specifying different Q-matrices can dramatically change the estimation process: in practice, the Q-matrix for most tests should be estimated to specify the associations between items and attributes, otherwise, incorrect classification of examinees will occur (Köhn, Chiu, & Brusco, 2015). In the present design, neither varying Q-matrix nor the effect of mis-specifying Q-matrix is taken into consideration. Missing data problem is not addressed in this dissertation either, but it is a common problem in practice and therefore, being able to handle the missingness while estimating models sheds the lights on the future research direction. A potential solution to deal with the missingness is modifying the HPSOEM to maximize the full information likelihood function that is known as FIML.

M*plus* outperformed the proposed algorithm in all conditions in the LCDM simulation, but fell behind the HPSOEM in the MIRT simulation. The average computational time for LCDMs and MIRT models are 35 minutes and 452 minutes. Having difficulties in MIRT estimation, M*plus* is not designed to fit IRT models and therefore IRT estimations tend to exhaust M*plus*. On the other hand, there is large space for the improvement of the proposed algorithm. Although the HPSOEM was outperformed by M*plus* in the LCDM simulation, this result could be due to how the HPSOEM algorithm was coded. Theoretically, HPSOEM can be many times faster than what it is now if the entire function is constructed in *C++* or *Fortran*; currently the HPSOEM algorithm is written in base *R* software scripting language. Research has shown that using compiler package with *R* often takes less than half of time executing the same function than that of without packages (e.g., Aruoba & Fernández, 2014).

As a variant of PSO, the proposed algorithm lends itself better to rapidly developing computing resources related to parallel multiple processing, for example, multi-core processors, parallel graphics processing units (GPUs), and computing clusters (McNabb & Seppi, 2014). Algorithms designed on multiple processing framework could utilize parallel computing technique and therefore improve the convergence speed. In particular, particles updating at each iteration can be assigned to different computational units such that the inefficiency caused by sequential updating design is avoided. As one can find, when (1) the number of attributes and/or (2) the sample size increase, the number of parameters in both LCDMs and MIRT models would exponentially grow. With the assistance of multiple processing, theoretically the HPSOEM would maximize the benefits of strong computational facility to estimate the psychometric models on large scale data sets and/or complex Q-matrices. Earlier works had focused on utilizing multi-core processors (e.g., MapReduce; Aljarah & Ludwig, 2012) to update particles. Recent studies have shed light on using the GPU architecture as a parallel computing framework in PSO algorithms (Dali & Bouamama, 2015). Compared with CPUs, GPUs are known for (1) lower cost (2) more cores, and (3) faster in multiple matrix multiplications. In fact, estimating aforementioned LCDMs or tasks of this kind, a strong CPU with 16 cores tend to perform worse than a low-end GPU that contains 700 cores. In the present dissertation, the proposed algorithm was executed in a desktop because the simulation design is not overwhelmingly demanding, meaning the computation cannot be handled in a personal computer. However, if the estimation raises to a substantive situation, for example, a 500x50 Q-matrix with more than 1000 item parameters, the HPSOEM can be implemented in a cloud computing facility with strong GPUs and/or multi-core CPUs.

To sum, the purpose of this dissertation is to propose a machine-learning based algorithm for the estimation of psychometric models. In particular, the proposed estimator is a combination of the EM algorithm and the PSO techniques, which have been popular in neural networks and other similar fields. The performance of the proposed algorithm is evaluated through a straightforward simulation study of which the results indicate that it is an appropriate option to handle psychometric models estimation task. To handle many psychometric models with a few thousands of respondents and 20 to 40 items, which are frequently seen in pratice, setting ***c*=1 or 2**, $LL_{penalty}$ Multiplier =0.5, and N_{par}=100 can yield accurate and faster estimation than other configuration combination. The result cantions users that, setting parameters in the HPSOEM or other similar algorithm frameworks should be tuned according to the datasets and model complexity. Although penality can be used to handle parameter constraint requirement, the penalty sizes need to be chosen via careful literature review or simulaiton studies. A powerful hardware environment, although mostly useful in estimation, is not always helping gain computational speed; from the simulation results, there are margin effects in utilizing the computational capacity for the implementation of the HPSOEM. The primary research direction in the future is integrating more advanced PSO techniques and other similar machine learning approaches into the field of measurement. The proposed estimation is still based upon the EM algorithm, which may lead to inconsistency in the updating process (i.e., the pure EM is all definitive). What is more, even though GPUs were implemented in the propsed estimation, users may not be satisfactory with the performace: fast calculation is partially cancelled-off by writing/reading via graphical memories; it will be useful to study how to balance the arrangement of GPUs and CPUs such that optimal estimation can be configured. The HPSOEM is a frequentist approach, despite that it involves stochastic components. There are other fast stochastic-based algorithms are not discussed here, for example, Hamiltonian dynamics stochastic process that is implemented in Stan program. Simulation studies for comparing different algorithms may be valuable for offering practitioners selection guidelines.

References

Aljarah, I., & Ludwig, S. A. (2012, November). Parallel particle swarm optimization clustering algorithm based on mapreduce methodology. In *Nature and biologically inspired computing (NaBIC), 2012 fourth world congress on*(pp. 104-111). IEEE.

Clarke, D. P., Al-Abdeli, Y. M., & Kothapalli, G. (2014). The impact of using Particle Swarm Optimisation on the operational characteristics of a stand-alone hydrogen system with on-site water production. *International Journal of Hydrogen Energy*, *39*(28), 15307-15319.

Dali, N., & Bouamama, S. (2015). GPU-PSO: parallel particle swarm optimization approaches on graphical processing unit for constraint reasoning: case of Max-CSPs. *Procedia Computer Science*, *60*, 1070-1080.

Davier, M. (2005). A general diagnostic model applied to language testing data. *ETS Research Report Series*, *2005*(2).

Delyon, B., Lavielle, M., & Moulines, E. (1999). Convergence of a stochastic approximation version of the EM algorithm. *Annals of statistics*, 94-128.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B (methodological)*, 1-38.

DiBello, L. V., & Stout, W. (2003). Student profile scoring for formative assessment. In *New developments in psychometrics* (pp. 81-92). Springer, Tokyo.

Djerou, L., Khelil, N., & Batouche, M. (2011). Numerical integration method based on particle swarm optimization. *Advances in Swarm Intelligence*, 221-226.

Dorigo, M., & Gambardella, L. M. (1997). Ant colonies for the travelling salesman problem. *biosystems*, *43*(2), 73-81.

Guedria, N. B. (2016). Improved accelerated PSO algorithm for mechanical engineering optimization problems. *Applied Soft Computing*, *40*, 455-467.

Hallquist, M., & Wiley, J. (2011). MplusAutomation: Automating Mplus model estimation and interpretation. *R package version 0.6. Available online at: http://cran. r-project. org/web/packages/MplusAutomation/index. html*.

Hartz, S. M. (2002). *A Bayesian framework for the unified model for assessing cognitive abilities: Blending theory with practicality* (Doctoral dissertation, University of Illinois at Urbana-Champaign).

Harwell, M., Stone, C. A., Hsu, T. C., & Kirisci, L. (1996). Monte Carlo studies in item response theory. *Applied psychological measurement*, *20*(2), 101-125.

Henson, R. A., Templin, J. L., & Willse, J. T. (2009). Defining a family of cognitive diagnosis models using log-linear models with latent variables. *Psychometrika*, *74*(2), 191-210.

Junker, B. W., & Sijtsma, K. (2001). Cognitive assessment models with few assumptions, and connections with nonparametric item response theory. *Applied Psychological Measurement*, *25*(3), 258-272.

Kennedy, J., & Eberhart, R. C. (1997, October). A discrete binary version of the particle swarm algorithm. In *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on* (Vol. 5, pp. 4104-4108). IEEE.

Kessentini, S., & Barchiesi, D. (2015). Particle Swarm Optimization with Adaptive Inertia Weight. *International Journal of Machine Learning and Computing*, *5*(5), 368.

Kim, S., & Li, L. (2011). A novel global search algorithm for nonlinear mixed-effects models using particle swarm optimization. *Journal of pharmacokinetics and pharmacodynamics*, *38*(4), 471-495.

Köhn, H. F., Chiu, C. Y., & Brusco, M. J. (2015). Heuristic cognitive diagnosis when the Q□matrix is unknown. *British Journal of Mathematical and Statistical Psychology*, *68*(2), 268-291.

Lao, H., and Templin, J. (2016). *Estimation of Diagnostic Classification Models without Constraints: Issues with Class Label Switching*. Paper presented at Annual Meeting of the National Council on Measurement in Education,, Washington, District of Columbia.

Lee, C. M., & Ko, C. N. (2009). Time series prediction using RBF neural networks with a nonlinear time-varying evolution PSO algorithm. *Neurocomputing*, *73*(1), 449-460.

Malekpour, A. R., & Seifi, A. R. (2010). Application of constriction factor particle swarm optimization to optimum load shedding in power system. *Modern Applied Science*, *4*(7), 188.

Maitra, M., & Chatterjee, A. (2008). A hybrid cooperative–comprehensive learning based PSO algorithm for image segmentation using multilevel thresholding. *Expert Systems with Applications*, *34*(2), 1341-1350.

McNabb, A., & Seppi, K. (2014, July). Serial PSO results are irrelevant in a multi-core parallel world. In *Evolutionary Computation (CEC), 2014 IEEE Congress on* (pp. 3143-3150). IEEE.

Muthén, L. K., & Muthén, B. O. (2007). Mplus. *Statistical analysis with latent variables. Version*, *3*.

R Core Team. (2017). *R: a language and environment for statistical computing*. R Foundation for Statistical Computing. Vienna, Austria. Retrieved from http://www. R-project.org/

Rezaee Jordehi, A., & Jasni, J. (2013). Parameter selection in particle swarm optimisation: a survey. *Journal of Experimental & Theoretical Artificial Intelligence,* 25(4), 527-542.

Schilling, S., & Bock, R. D. (2005). High-dimensional maximum marginal likelihood item factor analysis by adaptive quadrature. *Psychometrika*, *70*(3), 533-555

Shukla, R., Hazela, B., Shukla, S., Prakash, R., & Mishra, K. K. (2017). Variant of Differential Evolution Algorithm. In *Advances in Computer and Computational Sciences* (pp. 601-608). Springer, Singapore.

Templin, J. L., & Henson, R. A. (2006). Measurement of psychological disorders using cognitive diagnosis models. *Psychological methods*, *11*(3), 287.

Templin, J., & Bradshaw, L. (2014). Hierarchical diagnostic classification models: A family of models for estimating and testing attribute hierarchies. *Psychometrika*, 79(2), 317-339.

Zhang, Q., Sun, J., & Tsang, E. (2005). An evolutionary algorithm with guided mutation for the maximum clique problem. *IEEE Transactions on Evolutionary Computation*, *9*(2), 192-200.

.

Appendix I

M*plus* Model Specification Syntax

```
MODEL:
%OVERALL%
[c#1] (m1); ! Latent variable mean for class 1
[c#2] (m2); ! Latent variable mean for class 2
[c#3] (m3); ! Latent variable mean for class 3
[c#4] (m4); ! Latent variable mean for class 4
[c#5] (m5); ! Latent variable mean for class 5
[c#6] (m6); ! Latent variable mean for class 6
[c#7] (m7); ! Latent variable mean for class 7
[c#8] (m8); ! Latent variable mean for class 8
[c#9] (m9); ! Latent variable mean for class 9
[c#10] (m10); ! Latent variable mean for class 10
[c#11] (m11); ! Latent variable mean for class 11
[c#12] (m12); ! Latent variable mean for class 12
[c#13] (m13); ! Latent variable mean for class 13
[c#14] (m14); ! Latent variable mean for class 14
[c#15] (m15); ! Latent variable mean for class 15
%c#1%
[x1$1] (t1_1);
[x2$1] (t2_1);
[x3$1] (t3_1);
[x4$1] (t4_1);
[x5$1] (t5_1);
[x6$1] (t6_1);
[x7$1] (t7_1);
[x8$1] (t8_1);
[x9$1] (t9_1);
[x10$1] (t10_1);
[x11$1] (t11_1);
[x12$1] (t12_1);
[x13$1] (t13_1);
[x14$1] (t14_1);
[x15$1] (t15_1);
[x16$1] (t16_1);
[x17$1] (t17_1);
[x18$1] (t18_1);
[x19$1] (t19_1);
[x20$1] (t20_1);
[x21$1] (t21_1);
[x22$1] (t22_1);
[x23$1] (t23_1);
[x24$1] (t24_1);
[x25$1] (t25_1);
```

```
[x26$1] (t26_1);
[x27$1] (t27_1);
[x28$1] (t28_1);

%c#2%
[x1$1] (t1_1);
[x2$1] (t2_1);
[x3$1] (t3_1);
[x4$1] (t4_1);
[x5$1] (t5_2);
[x6$1] (t6_1);
[x7$1] (t7_1);
[x8$1] (t8_2);
[x9$1] (t9_1);
[x10$1] (t10_1);
[x11$1] (t11_1);
[x12$1] (t12_1);
[x13$1] (t13_2);
[x14$1] (t14_2);
[x15$1] (t15_2);
[x16$1] (t16_2);
[x17$1] (t17_1);
[x18$1] (t18_2);
[x19$1] (t19_1);
[x20$1] (t20_2);
[x21$1] (t21_2);
[x22$1] (t22_2);
[x23$1] (t23_1);
[x24$1] (t24_1);
[x25$1] (t25_1);
[x26$1] (t26_1);
[x27$1] (t27_1);
[x28$1] (t28_1);

%c#3%
[x1$1] (t1_1);
[x2$1] (t2_2);
[x3$1] (t3_1);
[x4$1] (t4_1);
[x5$1] (t5_1);
[x6$1] (t6_1);
[x7$1] (t7_1);
[x8$1] (t8_3);
[x9$1] (t9_2);
[x10$1] (t10_2);
[x11$1] (t11_2);
```

```
[x12$1] (t12_1);
[x13$1] (t13_1);
[x14$1] (t14_1);
[x15$1] (t15_1);
[x16$1] (t16_1);
[x17$1] (t17_1);
[x18$1] (t18_1);
[x19$1] (t19_1);
[x20$1] (t20_1);
[x21$1] (t21_1);
[x22$1] (t22_1);
[x23$1] (t23_1);
[x24$1] (t24_1);
[x25$1] (t25_1);
[x26$1] (t26_2);
[x27$1] (t27_1);
[x28$1] (t28_1);

%c#4%
[x1$1] (t1_1);
[x2$1] (t2_2);
[x3$1] (t3_1);
[x4$1] (t4_1);
[x5$1] (t5_2);
[x6$1] (t6_1);
[x7$1] (t7_1);
[x8$1] (t8_4);
[x9$1] (t9_2);
[x10$1] (t10_2);
[x11$1] (t11_2);
[x12$1] (t12_1);
[x13$1] (t13_2);
[x14$1] (t14_2);
[x15$1] (t15_2);
[x16$1] (t16_2);
[x17$1] (t17_1);
[x18$1] (t18_2);
[x19$1] (t19_1);
[x20$1] (t20_2);
[x21$1] (t21_2);
[x22$1] (t22_2);
[x23$1] (t23_1);
[x24$1] (t24_1);
[x25$1] (t25_1);
[x26$1] (t26_2);
[x27$1] (t27_1);
```

```
[x28$1] (t28_1);

%c#5%
[x1$1] (t1_1);
[x2$1] (t2_1);
[x3$1] (t3_2);
[x4$1] (t4_1);
[x5$1] (t5_1);
[x6$1] (t6_2);
[x7$1] (t7_1);
[x8$1] (t8_1);
[x9$1] (t9_1);
[x10$1] (t10_1);
[x11$1] (t11_1);
[x12$1] (t12_1);
[x13$1] (t13_1);
[x14$1] (t14_1);
[x15$1] (t15_1);
[x16$1] (t16_1);
[x17$1] (t17_1);
[x18$1] (t18_3);
[x19$1] (t19_2);
[x20$1] (t20_3);
[x21$1] (t21_3);
[x22$1] (t22_3);
[x23$1] (t23_1);
[x24$1] (t24_2);
[x25$1] (t25_2);
[x26$1] (t26_1);
[x27$1] (t27_1);
[x28$1] (t28_2);

%c#6%
[x1$1] (t1_1);
[x2$1] (t2_1);
[x3$1] (t3_2);
[x4$1] (t4_1);
[x5$1] (t5_2);
[x6$1] (t6_2);
[x7$1] (t7_1);
[x8$1] (t8_2);
[x9$1] (t9_1);
[x10$1] (t10_1);
[x11$1] (t11_1);
[x12$1] (t12_1);
[x13$1] (t13_2);
```

```
[x14$1] (t14_2);
[x15$1] (t15_2);
[x16$1] (t16_2);
[x17$1] (t17_1);
[x18$1] (t18_4);
[x19$1] (t19_2);
[x20$1] (t20_4);
[x21$1] (t21_4);
[x22$1] (t22_4);
[x23$1] (t23_1);
[x24$1] (t24_2);
[x25$1] (t25_2);
[x26$1] (t26_1);
[x27$1] (t27_1);
[x28$1] (t28_2);

%c#7%
[x1$1] (t1_1);
[x2$1] (t2_2);
[x3$1] (t3_2);
[x4$1] (t4_1);
[x5$1] (t5_1);
[x6$1] (t6_2);
[x7$1] (t7_1);
[x8$1] (t8_3);
[x9$1] (t9_2);
[x10$1] (t10_2);
[x11$1] (t11_2);
[x12$1] (t12_1);
[x13$1] (t13_1);
[x14$1] (t14_1);
[x15$1] (t15_1);
[x16$1] (t16_1);
[x17$1] (t17_1);
[x18$1] (t18_3);
[x19$1] (t19_2);
[x20$1] (t20_3);
[x21$1] (t21_3);
[x22$1] (t22_3);
[x23$1] (t23_1);
[x24$1] (t24_2);
[x25$1] (t25_2);
[x26$1] (t26_2);
[x27$1] (t27_1);
[x28$1] (t28_2);
```

```
%c#8%
[x1$1] (t1_1);
[x2$1] (t2_2);
[x3$1] (t3_2);
[x4$1] (t4_1);
[x5$1] (t5_2);
[x6$1] (t6_2);
[x7$1] (t7_1);
[x8$1] (t8_4);
[x9$1] (t9_2);
[x10$1] (t10_2);
[x11$1] (t11_2);
[x12$1] (t12_1);
[x13$1] (t13_2);
[x14$1] (t14_2);
[x15$1] (t15_2);
[x16$1] (t16_2);
[x17$1] (t17_1);
[x18$1] (t18_4);
[x19$1] (t19_2);
[x20$1] (t20_4);
[x21$1] (t21_4);
[x22$1] (t22_4);
[x23$1] (t23_1);
[x24$1] (t24_2);
[x25$1] (t25_2);
[x26$1] (t26_2);
[x27$1] (t27_1);
[x28$1] (t28_2);

%c#9%
[x1$1] (t1_2);
[x2$1] (t2_1);
[x3$1] (t3_1);
[x4$1] (t4_2);
[x5$1] (t5_3);
[x6$1] (t6_1);
[x7$1] (t7_2);
[x8$1] (t8_1);
[x9$1] (t9_1);
[x10$1] (t10_1);
[x11$1] (t11_1);
[x12$1] (t12_2);
[x13$1] (t13_3);
[x14$1] (t14_3);
[x15$1] (t15_3);
```

```
[x16$1] (t16_3);
[x17$1] (t17_2);
[x18$1] (t18_1);
[x19$1] (t19_3);
[x20$1] (t20_1);
[x21$1] (t21_1);
[x22$1] (t22_1);
[x23$1] (t23_2);
[x24$1] (t24_3);
[x25$1] (t25_3);
[x26$1] (t26_1);
[x27$1] (t27_2);
[x28$1] (t28_3);

%c#10%
[x1$1] (t1_2);
[x2$1] (t2_1);
[x3$1] (t3_1);
[x4$1] (t4_2);
[x5$1] (t5_4);
[x6$1] (t6_1);
[x7$1] (t7_2);
[x8$1] (t8_2);
[x9$1] (t9_1);
[x10$1] (t10_1);
[x11$1] (t11_1);
[x12$1] (t12_2);
[x13$1] (t13_4);
[x14$1] (t14_4);
[x15$1] (t15_4);
[x16$1] (t16_4);
[x17$1] (t17_2);
[x18$1] (t18_2);
[x19$1] (t19_3);
[x20$1] (t20_2);
[x21$1] (t21_2);
[x22$1] (t22_2);
[x23$1] (t23_2);
[x24$1] (t24_3);
[x25$1] (t25_3);
[x26$1] (t26_1);
[x27$1] (t27_2);
[x28$1] (t28_3);

%c#11%
[x1$1] (t1_2);
```

```
[x2$1] (t2_2);
[x3$1] (t3_1);
[x4$1] (t4_2);
[x5$1] (t5_3);
[x6$1] (t6_1);
[x7$1] (t7_2);
[x8$1] (t8_3);
[x9$1] (t9_2);
[x10$1] (t10_2);
[x11$1] (t11_2);
[x12$1] (t12_2);
[x13$1] (t13_3);
[x14$1] (t14_3);
[x15$1] (t15_3);
[x16$1] (t16_3);
[x17$1] (t17_2);
[x18$1] (t18_1);
[x19$1] (t19_3);
[x20$1] (t20_1);
[x21$1] (t21_1);
[x22$1] (t22_1);
[x23$1] (t23_2);
[x24$1] (t24_3);
[x25$1] (t25_3);
[x26$1] (t26_2);
[x27$1] (t27_2);
[x28$1] (t28_3);

%c#12%
[x1$1] (t1_2);
[x2$1] (t2_2);
[x3$1] (t3_1);
[x4$1] (t4_2);
[x5$1] (t5_4);
[x6$1] (t6_1);
[x7$1] (t7_2);
[x8$1] (t8_4);
[x9$1] (t9_2);
[x10$1] (t10_2);
[x11$1] (t11_2);
[x12$1] (t12_2);
[x13$1] (t13_4);
[x14$1] (t14_4);
[x15$1] (t15_4);
[x16$1] (t16_4);
[x17$1] (t17_2);
```

```
[x18$1] (t18_2);
[x19$1] (t19_3);
[x20$1] (t20_2);
[x21$1] (t21_2);
[x22$1] (t22_2);
[x23$1] (t23_2);
[x24$1] (t24_3);
[x25$1] (t25_3);
[x26$1] (t26_2);
[x27$1] (t27_2);
[x28$1] (t28_3);

%c#13%
[x1$1] (t1_2);
[x2$1] (t2_1);
[x3$1] (t3_2);
[x4$1] (t4_2);
[x5$1] (t5_3);
[x6$1] (t6_2);
[x7$1] (t7_2);
[x8$1] (t8_1);
[x9$1] (t9_1);
[x10$1] (t10_1);
[x11$1] (t11_1);
[x12$1] (t12_2);
[x13$1] (t13_3);
[x14$1] (t14_3);
[x15$1] (t15_3);
[x16$1] (t16_3);
[x17$1] (t17_2);
[x18$1] (t18_3);
[x19$1] (t19_4);
[x20$1] (t20_3);
[x21$1] (t21_3);
[x22$1] (t22_3);
[x23$1] (t23_2);
[x24$1] (t24_4);
[x25$1] (t25_4);
[x26$1] (t26_1);
[x27$1] (t27_2);
[x28$1] (t28_4);

%c#14%
[x1$1] (t1_2);
[x2$1] (t2_1);
[x3$1] (t3_2);
```

```
[x4$1] (t4_2);
[x5$1] (t5_4);
[x6$1] (t6_2);
[x7$1] (t7_2);
[x8$1] (t8_2);
[x9$1] (t9_1);
[x10$1] (t10_1);
[x11$1] (t11_1);
[x12$1] (t12_2);
[x13$1] (t13_4);
[x14$1] (t14_4);
[x15$1] (t15_4);
[x16$1] (t16_4);
[x17$1] (t17_2);
[x18$1] (t18_4);
[x19$1] (t19_4);
[x20$1] (t20_4);
[x21$1] (t21_4);
[x22$1] (t22_4);
[x23$1] (t23_2);
[x24$1] (t24_4);
[x25$1] (t25_4);
[x26$1] (t26_1);
[x27$1] (t27_2);
[x28$1] (t28_4);

%c#15%
[x1$1] (t1_2);
[x2$1] (t2_2);
[x3$1] (t3_2);
[x4$1] (t4_2);
[x5$1] (t5_3);
[x6$1] (t6_2);
[x7$1] (t7_2);
[x8$1] (t8_3);
[x9$1] (t9_2);
[x10$1] (t10_2);
[x11$1] (t11_2);
[x12$1] (t12_2);
[x13$1] (t13_3);
[x14$1] (t14_3);
[x15$1] (t15_3);
[x16$1] (t16_3);
[x17$1] (t17_2);
[x18$1] (t18_3);
[x19$1] (t19_4);
```

```
[x20$1] (t20_3);
[x21$1] (t21_3);
[x22$1] (t22_3);
[x23$1] (t23_2);
[x24$1] (t24_4);
[x25$1] (t25_4);
[x26$1] (t26_2);
[x27$1] (t27_2);
[x28$1] (t28_4);

%c#16%
[x1$1] (t1_2);
[x2$1] (t2_2);
[x3$1] (t3_2);
[x4$1] (t4_2);
[x5$1] (t5_4);
[x6$1] (t6_2);
[x7$1] (t7_2);
[x8$1] (t8_4);
[x9$1] (t9_2);
[x10$1] (t10_2);
[x11$1] (t11_2);
[x12$1] (t12_2);
[x13$1] (t13_4);
[x14$1] (t14_4);
[x15$1] (t15_4);
[x16$1] (t16_4);
[x17$1] (t17_2);
[x18$1] (t18_4);
[x19$1] (t19_4);
[x20$1] (t20_4);
[x21$1] (t21_4);
[x22$1] (t22_4);
[x23$1] (t23_2);
[x24$1] (t24_4);
[x25$1] (t25_4);
[x26$1] (t26_2);
[x27$1] (t27_2);
[x28$1] (t28_4);

MODEL CONSTRAINT:
m1>-15;
m2>-15;
m3>-15;
m4>-15;
m5>-15;
```

```
m6>-15;
m7>-15;
m8>-15;
m9>-15;
m10>-15;
m11>-15;
m12>-15;
m13>-15;
m14>-15;
m15>-15;
NEW( l1_11 );
NEW( l2_13 );
NEW( l3_12 );
NEW( l4_11 );
NEW( l5_11 );
NEW( l5_14 );
NEW( l5_214 );
NEW( l6_12 );
NEW( l7_11 );
NEW( l8_13 );
NEW( l8_14 );
NEW( l8_234 );
NEW( l9_13 );
NEW( l10_13 );
NEW( l11_13 );
NEW( l12_11 );
NEW( l13_11 );
NEW( l13_14 );
NEW( l13_214 );
NEW( l14_11 );
NEW( l14_14 );
NEW( l14_214 );
NEW( l15_11 );
NEW( l15_14 );
NEW( l15_214 );
NEW( l16_11 );
NEW( l16_14 );
NEW( l16_214 );
NEW( l17_11 );
NEW( l18_12 );
NEW( l18_14 );
NEW( l18_224 );
NEW( l19_11 );
NEW( l19_12 );
NEW( l19_212 );
NEW( l20_12 );
NEW( l20_14 );
NEW( l20_224 );
NEW( l21_12 );
NEW( l21_14 );
NEW( l21_224 );
NEW( l22_12 );
NEW( l22_14 );
NEW( l22_224 );
NEW( l23_11 );
NEW( l24_11 );
NEW( l24_12 );
NEW( l24_212 );
NEW( l25_11 );
NEW( l25_12 );
NEW( l25_212 );
NEW( l26_13 );
NEW( l27_11 );
NEW( l28_11 );
NEW( l28_12 );
NEW( l28_212 );
NEW( l1_0 );
NEW( l2_0 );
NEW( l3_0 );
NEW( l4_0 );
NEW( l5_0 );
NEW( l6_0 );
NEW( l7_0 );
NEW( l8_0 );
NEW( l9_0 );
NEW( l10_0 );
NEW( l11_0 );
NEW( l12_0 );
NEW( l13_0 );
NEW( l14_0 );
NEW( l15_0 );
NEW( l16_0 );
NEW( l17_0 );
NEW( l18_0 );
NEW( l19_0 );
NEW( l20_0 );
NEW( l21_0 );
NEW( l22_0 );
NEW( l23_0 );
NEW( l24_0 );
NEW( l25_0 );
NEW( l26_0 );
NEW( l27_0 );
NEW( l28_0 );
t1_1=-(l1_0);
 t1_2=-(l1_0+l1_11);
 t2_1=-(l2_0);
 t2_2=-(l2_0+l2_13);
 t3_1=-(l3_0);
 t3_2=-(l3_0+l3_12);
 t4_1=-(l4_0);
 t4_2=-(l4_0+l4_11);
 t5_1=-(l5_0);
 t5_2=-(l5_0+l5_14);
 t5_3=-(l5_0+l5_11);
 t5_4=-(l5_0+l5_11+l5_14+l5_214);
 t6_1=-(l6_0);
 t6_2=-(l6_0+l6_12);
 t7_1=-(l7_0);
 t7_2=-(l7_0+l7_11);
 t8_1=-(l8_0);
 t8_2=-(l8_0+l8_14);
 t8_3=-(l8_0+l8_13);
 t8_4=-(l8_0+l8_13+l8_14+l8_234);
 t9_1=-(l9_0);
 t9_2=-(l9_0+l9_13);
 t10_1=-(l10_0);
 t10_2=-(l10_0+l10_13);
 t11_1=-(l11_0);
 t11_2=-(l11_0+l11_13);
 t12_1=-(l12_0);
 t12_2=-(l12_0+l12_11);
 t13_1=-(l13_0);
 t13_2=-(l13_0+l13_14);
 t13_3=-(l13_0+l13_11);
 t13_4=-(l13_0+l13_11+l13_14+l13_214);
 t14_1=-(l14_0);
 t14_2=-(l14_0+l14_14);
 t14_3=-(l14_0+l14_11);
 t14_4=-(l14_0+l14_11+l14_14+l14_214);
 t15_1=-(l15_0);
 t15_2=-(l15_0+l15_14);
 t15_3=-(l15_0+l15_11);
 t15_4=-(l15_0+l15_11+l15_14+l15_214);
 t16_1=-(l16_0);
 t16_2=-(l16_0+l16_14);
 t16_3=-(l16_0+l16_11);
 t16_4=-(l16_0+l16_11+l16_14+l16_214);
```

```
t17_1=-(l17_0);
t17_2=-(l17_0+l17_11);
t18_1=-(l18_0);
t18_2=-(l18_0+l18_14);
t18_3=-(l18_0+l18_12);
t18_4=-(l18_0+l18_12+l18_14+l18_224);
t19_1=-(l19_0);
t19_2=-(l19_0+l19_12);
t19_3=-(l19_0+l19_11);
t19_4=-(l19_0+l19_11+l19_12+l19_212);
t20_1=-(l20_0);
t20_2=-(l20_0+l20_14);
t20_3=-(l20_0+l20_12);
t20_4=-(l20_0+l20_12+l20_14+l20_224);
t21_1=-(l21_0);
t21_2=-(l21_0+l21_14);
t21_3=-(l21_0+l21_12);
t21_4=-(l21_0+l21_12+l21_14+l21_224);
t22_1=-(l22_0);
t22_2=-(l22_0+l22_14);
t22_3=-(l22_0+l22_12);
t22_4=-(l22_0+l22_12+l22_14+l22_224);
t23_1=-(l23_0);
t23_2=-(l23_0+l23_11);
t24_1=-(l24_0);
t24_2=-(l24_0+l24_12);
t24_3=-(l24_0+l24_11);
t24_4=-(l24_0+l24_11+l24_12+l24_212);
t25_1=-(l25_0);
t25_2=-(l25_0+l25_12);
t25_3=-(l25_0+l25_11);
t25_4=-(l25_0+l25_11+l25_12+l25_212);
t26_1=-(l26_0);
t26_2=-(l26_0+l26_13);
t27_1=-(l27_0);
t27_2=-(l27_0+l27_11);
t28_1=-(l28_0);
t28_2=-(l28_0+l28_12);
t28_3=-(l28_0+l28_11);
t28_4=-(l28_0+l28_11+l28_12+l28_212);

l1_11>0;
l2_13>0;
l3_12>0;
l4_11>0;
l5_11>0;
```

```
 l5_14>0;
 l6_12>0;
 l7_11>0;
 l8_13>0;
 l8_14>0;
 l9_13>0;
 l10_13>0;
 l11_13>0;
 l12_11>0;
 l13_11>0;
 l13_14>0;
 l14_11>0;
 l14_14>0;
 l15_11>0;
 l15_14>0;
 l16_11>0;

 l16_14>0;
 l17_11>0;
 l18_12>0;
 l18_14>0;
 l19_11>0;
 l19_12>0;
 l20_12>0;
 l20_14>0;
 l21_12>0;
 l21_14>0;
 l22_12>0;
 l22_14>0;
 l23_11>0;
 l24_11>0;
 l24_12>0;
 l25_11>0;
 l25_12>0;
 l26_13>0;
 l27_11>0;
 l28_11>0;
 l28_12>0;

l5_214>-l5_11;
l5_214>-l5_14;
l8_234>-l8_13;
l8_234>-l8_14;
l13_214>-l13_11;
l13_214>-l13_14;
l14_214>-l14_11;
```

```
l14_214>-l14_14;
l15_214>-l15_11;
l15_214>-l15_14;
l16_214>-l16_11;
l16_214>-l16_14;
l18_224>-l18_12;
l18_224>-l18_14;
l19_212>-l19_11;
l19_212>-l19_12;
l20_224>-l20_12;
l20_224>-l20_14;
l21_224>-l21_12;
l21_224>-l21_14;
l22_224>-l22_12;
l22_224>-l22_14;
l24_212>-l24_11;
l24_212>-l24_12;
l25_212>-l25_11;
l25_212>-l25_12;
l28_212>-l28_11;
l28_212>-l28_12;
```

Appendix II

The HPSOEM R code

```
##################################################################################
############################PSOEM#########################################
##################################################################################

Model.log.likelihood<-
function(loopParticle,Class.Probability.vec=Class.Probability.vec,Classp.exp1=Classp.exp1,Clas
sp.exp2=Classp.exp2){
  for(loopEffect in (1:num_Parm)){
    assign(itemParmName[loopEffect],(globalSolution[loopParticle,loopEffect]))
  }
  Kernel.vec<-matrix(0,nrow(Kernel.exp),ncol(Kernel.exp))
  for(j in 1:length(Classp.exp1)){
    for(i in 1:nrow(Kernel.exp)){
      Kernel.vec[i,j]<-eval(parse(text=Kernel.exp[i,j]))
    }
  }

  z <- 1/(1+exp(-Kernel.vec))
  Z<-z[]
  Np<-nrow(respMatrix)
  Allperson.likelihood<-rep(0,Np)
  for (i in 1:Np){
    Zprime<-Z
    Zprime[respMatrix[i,]==0,]<-1-Z[respMatrix[i,]==0,]
    Allperson.likelihood[i]<-apply(Zprime,2,prod)%*%Class.Probability.vec
  }

  sum(log(Allperson.likelihood)[])
}


##################################################################################
###############GPU computation methods#########################################
##################################################################################

#for a certain profile and a certain response vector test.resp<-rbinom(28,1,0.8)
Class.Probability.vec<-NULL
Kernel.vec<-matrix(0,nrow(Kernel.exp),ncol(Kernel.exp))
for(j in 1:length(Classp.exp1)){
  Class.Probability.vec<<-c(Class.Probability.vec,eval(parse(text=Classp.exp2[j])))
  for(i in 1:nrow(Kernel.exp)){
    Kernel.vec[i,j]<-eval(parse(text=Kernel.exp[i,j]))
  }
}

z <- 1/(1+exp(-Kernel.vec))
Z<-z
Np<-nrow(respMatrix)
Allperson.likelihood<-rep(0,Np)
for (i in 1:Np){
  Zprime<-Z
  Zprime[respMatrix[i,]==0,]<-1-Z[respMatrix[i,]==0,]
  Allperson.likelihood[i]<-apply(Zprime,2,prod)%*%Class.Probability.vec
}

#Algorithm Specifications
num_Iteration=1000;num_Particle=198;stop_Criterion=0.01;cVector=c(1.5,1.5)
inertiaVector=c(0.95,0.4);dVector=c(0.2,7)
num_Parm<-length(itemParmName)
Np<-nrow(respMatrix)
FAILparticlecount<-rep(0,num_Particle)
globalSolution<-matrix(0,num_Particle,num_Parm)
globalClassProb<-matrix(0,num_Particle,nclass)
localOptimum<-matrix(0,num_Particle,num_Parm)
optimSolution<-rep(0,num_Parm)
globalVelocity<-matrix(0,num_Particle,num_Parm)
globalLL<-rep(0,num_Particle)
iterateLL<-rep(0,num_Iteration)
prematureCount<-rep(0,num_Particle)
w.max<-1
w.min<-0
#Initial Values
for (loopParticle in 1:num_Particle){
  globalSolution[loopParticle,]<-runif(num_Parm,-2,2)
  globalSolution[loopParticle,1:numMainEffect]<-runif((numMainEffect),0.1,2)
}
globalSolution[1,]<-parm.ini
localOptimum<-globalSolution
for (loopParticle in 1:num_Particle){
  globalVelocity[loopParticle,1:(num_Parm)]<-runif((num_Parm),-0.1,0.1)
}

globalLL<-foreach(exponent=1:num_Particle,
                  .combine=c,
                  .export=c(ls())) %dopar%

Model.log.likelihood(exponent,Class.Probability.vec=Class.Probability.vec,Classp.exp1=Classp.
exp1,Classp.exp2=Classp.exp2)
```

```
ref.LL<-mean(globalLL)
optimSolution<-localOptimum[which.max(globalLL),]

##################################################################################
#############################Updating function##################################
##################################################################################

particle.update<-
function(w,loopParticle,FAILparticlecount=FAILparticlecount,globalLL=globalLL,globalSoluti
on=globalSolution,localOptimum=localOptimum,globalVelocity=globalVelocity,cVector=cVect
or,Class.Probability.vec=Class.Probability.vec){
  reshuffle<-FAILparticlecount[loopParticle]
  if(FAILparticlecount[loopParticle]>0){
    firsthalf<-sample(1:length(optimSolution),round(length(optimSolution)/2,0),replace=F)
    secondhalf<-c(1:length(optimSolution))[c(1:length(optimSolution))%in%firsthalf]
    #stocahstically take a half genes from the global best solution
    globalSolution[loopParticle,firsthalf]<-optimSolution[firsthalf]
    #stocahstically take a half genes from the global 2nd best solution
    globalSolution[loopParticle,secondhalf]<-localOptimum[order(globalLL*-1)[2],secondhalf]
    globalVelocity[loopParticle,]<-runif(num_Parm,-1,1)
    reshuffle<-0}

  if(FAILparticlecount[loopParticle]==0){
    globalVelocity[loopParticle,]<-w*globalVelocity[loopParticle,]+
      cVector[1]*runif(1,0.001,0.999)*(localOptimum[loopParticle,]-
globalSolution[loopParticle,])+
      cVector[2]*runif(1,0.001,0.999) *(optimSolution-globalSolution[loopParticle,])
    #Use the new velocity to update the particle' selected variable
    globalSolution[loopParticle,]<-globalSolution[loopParticle,]+globalVelocity[loopParticle,]
  }
  #Assign the particle's solution to model: MainEffect and OtherEffect for items
  for(loopEffect in (1:num_Parm)){
    assign(itemParmName[loopEffect],(globalSolution[loopParticle,loopEffect]))
  }
  #See if the Item parameter constrans violation happens
  LLpenalty<-Cons.Vio.Penalty(constrain.List=Constrain.List)

  if(LLpenalty==0){ItemParm.constrainViolation<-0
  #Take the last Class.Probability.vec, this step will update the new Class.Probability.vec
  #1. since all Kernels were PSO updated, they should be assigned to their names
  Kernel.vec<-matrix(0,nrow(Kernel.exp),ncol(Kernel.exp))
  for(j in 1:length(Classp.exp1)){
    for(i in 1:nrow(Kernel.exp)){
      Kernel.vec[i,j]<-eval(parse(text=Kernel.exp[i,j]))
    }
  }
  z <- 1/(1+exp(-Kernel.vec))
  Z<-z[]
  Np<-nrow(respMatrix)
  Allperson.likelihood<-rep(0,Np)
  Allperson.conditional.class.probability<-matrix(0,Np,nclass)
  for (i in 1:Np){
    Zprime<-Z
    Zprime[respMatrix[i,]==0,]<-1-Z[respMatrix[i,]==0,]
    each.class.likelihood<-apply(Zprime,2,prod)
    Allperson.likelihood[i]<-t(each.class.likelihood)%*%Class.Probability.vec
    Allperson.conditional.class.probability[i,]<-each.class.likelihood*Class.Probability.vec
    Allperson.conditional.class.probability[i,]<-
Allperson.conditional.class.probability[i,]/Allperson.likelihood[i]
  }
  Update.Class.Probability.vec<-apply(Allperson.conditional.class.probability,2,sum)/Np
  for (i in 1:Np){
    Zprime<-Z
    Zprime[respMatrix[i,]==0,]<-1-Z[respMatrix[i,]==0,]
    Allperson.likelihood[i]<-apply(Zprime,2,prod)%*%Update.Class.Probability.vec
  }

  currentLL<-sum(log(Allperson.likelihood)[])
  }
  if(LLpenalty!=0){print('Item Parameter updates are unsuccessful')
    currentLL<-LLpenalty+globalLL[loopParticle]
  }
  if(currentLL=='NaN'){currentLL<-1*ref.LL;reshuffle<-FAILparticlecount[loopParticle]+1}
  if (currentLL>=globalLL[loopParticle]){
    localOptimum[loopParticle,]<-globalSolution[loopParticle,]
    Class.Probability.vec<<-Update.Class.Probability.vec
    globalLL[loopParticle]<-currentLL
  }#else{globalSolution[loopParticle,]<-localOptimum[loopParticle,]}


  particle.output<-list()
  particle.output[[1]]<-globalLL[loopParticle]
  particle.output[[2]]<-localOptimum[loopParticle,]
  particle.output[[3]]<-globalSolution[loopParticle,]
  particle.output[[4]]<-globalVelocity[loopParticle,]
  particle.output[[5]]<-Class.Probability.vec
  particle.output[[6]]<-reshuffle
  particle.output[[7]]<-currentLL
  particle.output
}
```

```
for(loopIteration in 1:num_Iteration){
  if(var(globalLL)>stop_Criterion){
    w=w.max-((w.max-w.min)/num_Iteration)*loopIteration
    Iter.result = foreach(exponent=1:num_Particle,
                        .combine = list,
                        .multicombine = TRUE,
                        .export=ls(.GlobalEnv)) %dopar%

particle.update(w,exponent,globalLL=globalLL,FAILparticlecount=FAILparticlecount,globalSol
ution=globalSolution,localOptimum=localOptimum,globalVelocity=globalVelocity,cVector=cV
ector,Class.Probability.vec=Class.Probability.vec)


    if(num_Particle<=398&num_Particle>301){#num_particle:301-398
      Iter.Result<-Iter.result[[1]][[1]][[1]][[1]]
      for(loop.2nd in 2:(100)){Iter.Result[[loop.2nd+99]]<-Iter.result[[1]][[1]][[1]][[loop.2nd]]}
      for(loop.2nd in 2:(100)){Iter.Result[[loop.2nd+199-1]]<-Iter.result[[1]][[1]][[loop.2nd]]}
      for(loop.2nd in 2:(100)){Iter.Result[[loop.2nd+299-1]]<-Iter.result[[1]][[loop.2nd]]}
      Iter.Result[[399]]<-Iter.result[[2]]
      Iter.result<-Iter.Result}

    if(num_Particle<=198&num_Particle>101){#num_particle:101-198
      Iter.Result<-Iter.result[[1]]
      for(loop.2nd in 2:(num_Particle-100+1)){Iter.Result[[loop.2nd+99]]<-
Iter.result[[loop.2nd]]}
      Iter.result<-Iter.Result}

    CURRENTLL<-unlist(lapply(Iter.result,function(x){unlist(x[[7]])}))
    globalLL<<-unlist(lapply(Iter.result,function(x){unlist(x[[1]])}))
    localOptimum<<-
t(matrix(unlist(lapply(Iter.result,function(x){unlist(x[[2]])})),num_Parm,num_Particle))
    globalSolution<<-
t(matrix(unlist(lapply(Iter.result,function(x){unlist(x[[3]])})),num_Parm,num_Particle))
    globalVelocity<<-
t(matrix(unlist(lapply(Iter.result,function(x){unlist(x[[4]])})),num_Parm,num_Particle))
    optimSolution<<-localOptimum[which.max(globalLL),]
    Class.Probability.vec<<-
t(matrix(unlist(lapply(Iter.result,function(x){unlist(x[[5]])})),nclass,num_Particle))[which.max(g
lobalLL),]
    FAILparticlecount<<-
FAILparticlecount+as.numeric(unlist(lapply(Iter.result,function(x){unlist(x[[6]])})))
    FAILparticlecount[FAILparticlecount>1]<-0
    particle.partition<-round(num_Particle/2,0)
    if(sum(FAILparticlecount[1:particle.partition])>=(num_Particle/2-
num_Particle/4)){FAILparticlecount[1:particle.partition]<-rep(0,num_Parm/2)}
```

```
    if(sum(FAILparticlecount[particle.partition:num_Particle])>=(num_Particle/2-
num_Particle/4)){FAILparticlecount[particle.partition:num_Particle]<-rep(0,num_Parm/2)}
    iterateLL[loopIteration]<-max(globalLL)
    print(c(max(globalLL),mod1[8]$loglike))
  }
}
if(max(iterateLL)<mod1[8]$loglike){
  optimSolution<-parm.ini
  Class.Probability.vec<-prop.ini
  PSOEM.loglik[rep.loop]<-mod1[8]$loglike
}
```