Measuring the Power of Learning.®

Research Report
ETS RR–16-34

High-Performance Psychometrics: The Parallel-E Parallel-M Algorithm for Generalized Latent Variable Models

Matthias von Davier

November 2016

ETS Research Report Series

Since its 1947 founding, ETS has conducted and disseminated scientific research to support its products and services, and to advance the measurement and education fields. In keeping with these goals, ETS is committed to making its research freely available to the professional community and to the general public. Published accounts of ETS research, including papers in the ETS Research Report series, undergo a formal peer-review process by ETS staff to ensure that they meet established scientific and professional standards. All such ETS-conducted peer reviews are in addition to any reviews that outside organizations may provide as part of their own publication processes. Peer review notwithstanding, the positions expressed in the ETS Research Report series and other published accounts of ETS research are those of the authors and not necessarily those of the Officers and Trustees of Educational Testing Service.

The Daniel Eignor Editorship is named in honor of Dr. Daniel R. Eignor, who from 2001 until 2011 served the Research and Development division as Editor for the ETS Research Report series. The Eignor Editorship has been created to recognize the pivotal leadership role that Dr. Eignor played in the research publication process at ETS.

ETS Research Report Series ISSN 2330-8516

RESEARCH REPORT

High-Performance Psychometrics: The Parallel-E Parallel-M Algorithm for Generalized Latent Variable Models

Matthias von Davier

Educational Testing Service, Princeton, NJ

This report presents results on a parallel implementation of the expectation-maximization (EM) algorithm for multidimensional latent variable models. The developments presented here are based on code that parallelizes both the E step and the M step of the parallel-E parallel-M algorithm. Examples presented in this report include item response theory, diagnostic classification models, multitrait–multimethod (MTMM) models, and discrete mixture distribution models. These types of models are frequently applied to the analysis of multidimensional responses of test takers to a set of items, for example, in the context of proficiency testing. The algorithm presented here is based on a direct implementation of massive parallelism using a paradigm that allows the distribution of work among a number of processor cores. Modern desktop computers as well as many laptops are using processors that contain 2–4 cores and potentially twice the number of virtual cores. Many servers use 2, 4, or more multicore #central processing units (CPUs), which brings the number of cores to 8, 12, 32, or even 64 or more. The algorithm presented here scales the time reduction in the most calculation-intense part of the program almost linearly for some problems, which means that a server with 32 physical cores executes the parallel-E step algorithm up to 24 times faster than a single-core computer or the equivalent nonparallel algorithm. The overall gain (including parts of the program that cannot be executed in parallel) can reach a reduction in time by a factor of 6 or more for a 12-core machine. The basic approach is to utilize the architecture of modern CPUs, which often involves the design of processors with multiple cores that can run programs simultaneously. The use of this type of architecture for algorithms that produce posterior moments has straightforward appeal: The calculations conducted for each respondent or each distinct response pattern can be split up into simultaneous calculations.

Keywords Parallel programming; EM algorithm; high-performance computation (HPC); efficient estimation; modern psychometric models

doi:10.1002/ets2.12120

This report presents results on a parallel implementation of the expectation-maximization (EM) algorithm for multidimensional latent variable models. The developments presented here are based on code that parallelizes both the E step and the M step and will be referred to as the PEPM algorithm. Examples presented in this report include item response theory, diagnostic classification models, multitrait–multimethod (MTMM) models, and discrete mixture distribution models. Although there are commercial programs, such as LatentGold (Vermunt & Magidson, 2013), flexMIRT (Cai, 2013), and Mplus (Muthén & Muthén, 1998–2011), that provide the ability to specify a number of threads for parallel execution, a systematic comparison of the gain achieved by these appears not yet to be available. Also, note that while general statistical packages such as R may utilize parallel libraries for certain operations, such as matrix manipulations, this is different from a direct implementation of parallel processing in the EM algorithm. In the report, we also provide references to parallel EM algorithms for mixtures of Gaussian distributions used in computer vision.

The current report fills in some of the missing pieces in the domain of complex latent variable models used in psychometrics, describes the parallel extensions of the EM algorithm as they relate to the E step and the M step for multidimensional latent variable models, and compares computational performance on seven unidimensional and multidimensional latent variable, mixture, and multiple-group models, specified in the general diagnostic modeling framework (von Davier, 2005, 2008; von Davier & Rost, 2016). Finally, this report presents data on performance gains as observed on three different multicore computer architectures, ranging from typical laptop/desktop machines with four cores to a 12-core workstation with two central processing units (CPU) to a four-CPU 32(64)-core server.

The types of complex latent variable models utilized in the examples are frequently applied to the analysis of response data of test takers to a set of items that are measuring multidimensional latent variables, for example, in the context of

Corresponding author: M. von Davier, E-mail: mvondavier@ets.org

proficiency testing. Examples are the National Assessment of Educational Progress, the Program for International Student Assessment, the *TOEFL*® test, and the *GRADUATE RECORD EXAMINATIONS*® tests, all of which use item response theory (IRT) or some specific forms of multidimensional item response theory (MIRT). Other examples of models that can be estimated with this improved algorithm are latent class models, latent regression models, multilevel latent class models, and diagnostic classification models.

The algorithm presented here is based on a direct implementation of massive parallelism using a paradigm that allows the distribution of work among a number of processor cores. Modern desktop computers as well as many laptops are using processors that contain two to four cores and potentially twice the number of virtual cores. Many servers use two multicore CPUs, which doubles the number of physical cores to 2×4 or 2×6 or, most recently, 2×18, or more. The algorithm presented here scales the time reduction in the most calculation-intense part of the program almost linearly in the number of physical cores, which means that a server with $2\times6=12$ physical cores executes the parallel-E step algorithm up to 10 times faster than a single-core computer or the equivalent nonparallel algorithm. The overall gain can reach a reduction in time by a factor close to 6 for a dual-CPU 12-core machine, given that there are parts of any algorithm that cannot be parallelized (Amdahl, 1967).

The basic approach is to utilize the architecture of modern CPUs, which often involves the design of processors with multiple cores that can run programs simultaneously. The use of this type of architecture for algorithms that produce posterior moments has straightforward appeal: The calculations conducted for each respondent or each distinct response pattern can be split up into simultaneous calculations.

The remainder of this report is structured as follows: The next section introduces a general class of latent variable models—the general diagnostic model (GDM). A section describing the test cases used in the evaluation of the parallel estimation follows. Then the central results of comparisons between the sequential and the parallel approach to latent variable model estimation are described. Finally, the report closes with a discussion of results and an outlook.

A Class of General Latent Variable Models

The class of general latent variables examined in this report for the purpose of developing a parallel, high-performance estimation approach is the GDM.

The GDM (von Davier, 2005, 2008, 2009, 2013, 2014; von Davier, Xu, & Carstensen, 2011; von Davier & Yamamoto, 2004) provides a framework for developing diagnostic models. As an item response modeling framework, the probability of an item response $x\in\{0,\ \ldots,m_i\}$ by respondents $v=1,\ \ldots,N$ on items $i=1,\ \ldots,I$ can be written as

$$P(X=x\,|i,v)=\frac{\exp\left[f\left(\lambda_{xi},\theta_v\right)\right]}{1+\sum_{y=1}^{m_i}\exp\left[f\left(\lambda_{xi},\theta_v\right)\right]}, \tag{1}$$

with item parameters $\lambda_{xi}=(\beta_{xi},\mathbf{q}_i,\boldsymbol{\gamma}_{xi})$ and a skill vector $\theta_v=(a_{v1},\ \ldots,a_{vK})$ with either continuous, ordinal, or, as in the case of the deterministic input, noisy "and" gate (DINA) model and most other diagnostic models, binary skill variables $a_{\cdot k}\in\{0,1\}$. While the general model given in Equation 1 served as the basis for the formal specification of the log-linear cognitive diagnostic model (L-CDM; Henson, Templin, & Willse, 2009; von Davier, 2014) and other developments for binary skill attributes and data, von Davier (2005, 2008) utilized the general form to derive the linear or partial-credit GDM:

$$P\left(X=x|i,a_{\cdot k},c\right)=\frac{\exp\left(\beta_{ixc}+\sum_{k=1}^{K}\gamma_{ixck}h\left(q_{ik},a_k\right)\right)}{1+\sum_{y=1}^{m_i}\exp\left(\beta_{iyc}+\sum_{k=1}^{K}\gamma_{iyck}h\left(q_{ik},a_k\right)\right)} \tag{2}$$

with discrete skill $a_{\cdot k}\in\left\{a_{k1},a_{k2},\ \ldots\ a_{kL_k}\right\}$, which may be ordinal or binary, and $h(q,a)=qa$ and $\gamma^*_{ixk}=x\gamma_{ik}$ for parsimony. Note that Equation 2 also contains a population indicator, which makes it suitable both for multiple-group and mixture distribution models (von Davier, 2005, 2008; von Davier & Rost, 1995; von Davier & Rost, 2016; von Davier & Yamamoto, 2004). Note that these choices lead to a model that contains located latent class models, multiple classification

latent class models, IRT models, and multidimensional IRT models, as well as a compensatory version of the reparameterized unified model, as special cases (von Davier, 2005). In addition, the linear GDM as well as the general family are suitable for binary, polytomous ordinal, and mixed-format item response data.

The model defined in Equation 2 uses a weighted linear combination of skill components and is therefore a compensatory model by design, whereas the general framework (von Davier & Yamamoto, 2004) given in Equation 1 can be used to define compensatory as well as noncompensatory and conjunctive models. Moreover, it was shown that models with conjunctive (noncompensatory) skill structure can also be subsumed under the GDMs. More specifically, it was shown that the DINA and the L-CDM can be estimated as special cases of the GDM (von Davier, 2013, 2014). Finally, the model as defined in Equation 2 contains multidimensional IRT, mixture IRT, and latent structure as special cases and was extended to multilevel MIRT and diagnostic models (von Davier, 2010).

The Expectation-Maximization Algorithm

The EM algorithm (Dempster, Laird, & Rubin, 1977) is one of the most frequently used approaches for estimating latent variable models (e.g., McLachlan & Krishnan, 1997). The name of the algorithm stems from the alternating, iterative repetition of two steps, the E (expectation) step and the M (maximization) step. The estimation of generalized latent variable models using the EM algorithm requires the estimation of expected values for all required sufficient statistics of the structural parameters of the measurement model as well as the estimation of latent variable distributions in one or more populations. In the M step, the expected values serve as sufficient statistics for the maximization of parameters. At a somewhat abstract level, the EM algorithm for a latent variable model can be characterized as follows:

1. Allocate parameter memory locations and initialize structural parameters with (random) starting values
2. Allocate latent variable distribution memory locations and initialize these with (random) initial distribution
3. Start of loop
   - 3.1. E step: For all respondents $v = 1, \ldots, N$
     - 3.1.1. Calculate posterior distribution of the latent variable given responses of respondent v as well as preliminary prior distribution and structural parameters
     - 3.1.2. Aggregate expected counts of responses in levels of the latent variable space, given posterior distribution from previous step
   - 3.2. M step: For all response variables $i = 1, \ldots, K$
     - 3.2.1. Calculate gradient (and Hessian if required) for parameters $R_{i\cdot} = \rho_{i1}, \ldots, \rho_{iD}$ of response variable i
     - 3.2.2. Determine change amount Δ_i based on gradient method, Newton–Raphson, quasi-Newton, Metropolis–Hastings–Robbins–Monro, or similar)
     - 3.2.3. Calculate new parameter $R_{i\cdot}^{\text{new}} = R_{i\cdot} + \Delta_i$
   - 3.3. Calculate convergence criterion
   - 3.4. Exit loop if criterion smaller than threshold; otherwise go back to Step 3

The basic idea is that the EM algorithm provides expected counts for the unobserved quantities and hence can be used in incomplete data problems. The latent variable(s) are those missing (incomplete) quantities for all respondents, and the distributions of responses in levels of these latent variables are the unknown quantities that have to be supplied by the E step. These are calculated based on observed data (in psychometric applications, the item response variables and preliminary parameter estimates).

The Parallel-E Parallel-M Algorithm

Parallel extensions of the EM algorithm have gained attention in computer vision applications in recent years (Cui, Wei, & Dai, 2010; Cui et al., 2014; Das, Datar, Garg, & Rajaram, 2007; Lopez de Teruel, Garcia, & Acacio, 1999) and have been applied mainly in the context of processing optical data in artificial intelligence, typically in the form of Gaussian clustering. The difference between processing optical data and the current application lies in the types of observables

and in the class of models for which the parallel EM algorithm is specified. In addition, the PEPM algorithm introduces parallelism not only to the E step but also to the M step in the context of multidimensional latent variables and hence provides additional advantages. In this report, an implementation of the PEPM that uses distributed calculations both for the E step and the M step and advantages over sequential calculations are examined. The PEPM algorithm is suitable for a range of generalized linear and nonlinear mixed models for multivariate categorical (binary and polytomous) data:

1. Allocate global parameter space memory and initialize structural parameters with (random) starting values
2. Allocate global latent variable distribution memory and initialize these with (random) initial distribution
3. Start of loop
   - 3.1. Parallelize E step into $C = 1, \ldots, c$ cores
     - 3.1.1. Subdivide respondents into c groups
     - 3.1.2. Allocate c copies of private (new) latent variable distribution memory and initialize: parallel execution in groups for all respondents in group $C = 1, 2, \ldots, c$
       - 3.1.2.1. Calculate posterior distribution of the latent variable given responses of respondent v as well as preliminary prior distribution and structural parameters
       - 3.1.2.2. Aggregate expected counts of responses in levels of the latent variable space, given posterior distribution from previous step
     - 3.1.3. Aggregate private latent variable distribution memory into global latent variable distribution memory
   - 3.2. Parallelize M step into $C = 1, \ldots, c$ cores
     - 3.2.1. Subdivide response variables into c groups
     - 3.2.2. Allocate c copies of private (new) parameter space memory (size $1/c$ is sufficient): parallel execution in c groups for all response variables in group $C = 1, 2, \ldots, c$
       - 3.2.2.1. Calculate gradient (and Hessian, if required) for parameters $R_{i\cdot} = \rho_{i1}, \ldots, \rho_{iD}$ of response variable i
       - 3.2.2.2. Determine change amount Δ_i (based on gradient method, Newton–Raphson, quasi-Newton, Metropolis–Hastings–Robbins–Monro, or similar)
       - 3.2.2.3. Calculate new parameter $R_{i\cdot}^{\text{new}} = R_{i\cdot} + \Delta_i$
     - 3.2.3. Aggregate new parameters into global parameter space memory
   - 3.3. Calculate convergence criterion
   - 3.4. Exit loop if criterion smaller than threshold; otherwise back to Step 3

The PEPM algorithm presented here utilizes parallel processing for both the E step and the M step. Shared memory in symmetric multiprocessor systems (multicore single or dual CPU) is used to distribute the work across C cores or processes. The memory is distributed and allocated as needed. For the E step, multiple copies of the memory arrays needed to calculate expectations are allocated and finally aggregated after all parallel processes are concluded. For the M step, the work is distributed into c processes, each of which generates the quantities needed for execution of a maximization step that updates a subset of structural parameters as assigned to the different processes.

Limitations of Parallel Programming Efficiency Gains: Amdahl's Law and Gustafson's Law

The parallel EM algorithm presented here promises not only to speed up the E step by splitting up and parallelizing computations over many cores for large data sets but also to enable and facilitate efficient estimation of larger multidimensional models while allowing more complex latent structures. This means that, in contrast to previous implementations that did not implement complex latent variable models, the demand for and utilization of parallel algorithms are larger in the algorithm presented here.

However, one can ask how much of a speedup can be expected, because not every calculation can be parallelized and executed simultaneously on a number of cores. First, the response data have to be loaded into the system. Second, the estimation of the latent structure depends on the combination of results from the subsets of the sample that were

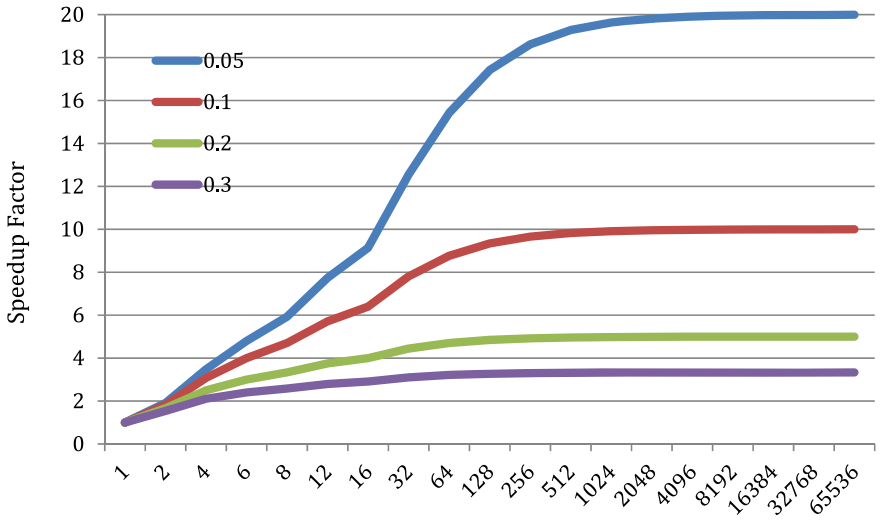

Figure 1 Amdahl's law: execution time speedup as a function of number of processing cores. The lines represent different proportions of the sequential part of the algorithm.

submitted to the different cores. Third, results have to be written to files for use in reporting, subsequent analyses, and, of course, quality control, such as checking convergence of estimation, model fit, and so on.

These considerations imply that parallelism in computation cannot be 100% in most applications and that the PEPM algorithm will not speed up calculations linearly in the number of processing cores applied. Amdahl (1967) described the formal relationship of the percentage of a process or program that can be executed concurrently (parallel on several computers or cores) and the speedup that can be expected as a function of the number of parallel units. This percentage has to be understood as the relative execution time in a sequential version of the software. For a symmetric multicore system with C cores, Amdahl's (1967) law can be expressed as

$$G = \frac{1}{S + P/C} = \frac{C}{SC + P},$$

where $P = 1 - S$ is the proportion of parallel code in the algorithm, and $S = 1 - P$ is the proportion of sequentially executed code. G is the gain, the ratio of execution time of the serial algorithm relative to the parallel implementation of the algorithm. Figure 1 shows the dependency of the gain on the number of processor cores C for four different levels of parallelism.

The equation can also be used to estimate the percentage of time the algorithm is in parallel mode. This requires solving for P in the equation

$$G = \frac{1}{(1 - P) + P/C}.$$

Some algebraic transformations yield

$$\frac{1}{G} = (1 - P) + \frac{P}{C} = 1 + \frac{P}{C} - \frac{PC}{C} = 1 + \frac{P = PC}{C}.$$

Finally, we obtain

$$P = \left(1 - \frac{1}{G}\right)\left(\frac{C}{C - 1}\right)$$

for $C > 1$ and

$$\left(1 - \frac{1}{G}\right) < \left(\frac{C}{C - 1}\right).$$

This allows calculating the approximate percentage of parallelism given a set of trials with measured empirical data on speedup. This approach is used in the following section that compares speedup for a 4-core i7pro laptop, a 12-core dual Xeon X5680 workstation, and a four-CPU 32(64)-core AMD SuperMicro server. The algorithm presented was implemented so it can be used on major operating systems and has been tested successfully on 64-bit systems, including Windows, Mac OSX, and Linux.

Typically, the exact percentage of parallel sections in the algorithm is an estimate at best. In this case, the law may be used to compare the serial and parallel versions and obtain an estimate of what percentage of the code was executed in parallel sections.

Gustafson (1988) challenged the generality of the Amdahl (1967) function by means of practical applications of parallelism that vary the problem size with the number of processor cores available. This approach to parallelism would predict almost linear speedup. The most likely application of latent variable models that would allow for linear speedup would be models for multiple populations in which there are no parameters shared across populations. This would allow running separate models across the C cores of a system. The models presented in this chapter follow more closely the shared parameter problem, in which the percentage of sequential code to "share" information across populations and/or latent variables limits the maximum possible gain.

Databases Used in the Comparisons

To compare the performance of the parallel implementation and the sequential version, the two algorithms were tested on seven different data sets. Each data set was analyzed with a latent variable model, ranging from multiple-group IRT models utilized for large-scale IRT linking (Test Cases A and B) to multidimensional discrete latent trait models—sometimes referred to as diagnostic classification models (Test Case C)—to multidimensional IRT models (Test Cases D, E, and G) and latent class models (Test Case F).

The differences between the test cases are summarized in Table 1 to give an impression of the range of estimation problems for which the software is suitable. von Davier (2008, 2013, 2014) has provided further examples of what types of models can be estimated. It was the main aim of this comparison to examine moderate to very large sample sizes, as the relative gain of a parallel implementation using small data sets can only be marginal. This is because the estimation of model parameters and latent distributions tends to be rather swift if sample sizes are small, so that input and output will take up more time than the actual computations needed to obtain parameter estimates.

Test Cases A and B contain data from 1,614,281 and 1,803,599 respondents, respectively, which are based on data collected across four cycles for Test Case A and five cycles for Test Case B of an international large-scale assessment of student outcomes. Each cycle contains between 50 and 80 samples from different countries, or country-by-language groups for countries with more than one official language, with a total of 312 estimated population distributions for Test Case A and 283 populations for Test Case B. The number of items, sampled in an incomplete block design (Mazzeo & von Davier, 2014), is 293 for Test Case A and 133 for Test Case B. Each distribution of the skill variable in these groups is represented using 21 quadrature points.

Test Case C is based on an innovative assessment that was aimed at assessing process skills and content knowledge simultaneously. Each item taps into multiple (two) skills of a total seven skills and knowledge variables. The model used on this data set is best described as a polytomous diagnostic model (each attribute variable has three levels) with an MTMM structure. The number of items is 214, there is just a single population, and the sample size is 7,377. The size (number of different possible attribute patterns) of the latent structure is 2,187; the structure was modeled using a log-linear approach (von Davier & Rost, 2016; von Davier & Yamamoto, 2004; Xu & von Davier, 2008).

Table 1 Test Case Uses in the Comparison of Sequential and Parallel Algorithms

Test case	Scale	Model	Groups	Items	Sample	QPT	Total QPT	Categories
A	1	IRT-2PL/GPCM	312	293	1,614,281	21	21	2–4
B	1	IRT-2PL/GPCM	283	133	1,803,599	21	21	4
C	7	GDM (MTMM)	1	214	7,377	3	2,187	3
D	2	MIRT (simple)	1	175	5,763	15	225	3
E	2	MIRT (simple)	1	175	5,763	31	961	3
F	NA	LCA	54	54	246,112	1	1	6
G	5	MIRT	1	150	2,026	5	3,125	2

Note. 2PL = two-parameter logistic; GDM = general diagnostic model; GPCM = generalized partial-credit model; IRT = item response theory; LCA = latent class analysis; MIRT = multiple item response theory; MTMM = multitrait–multimethod; QPT = quadrature points.

Test Cases D and E are based on the same data set, analyzed using a multidimensional IRT model, based on data from $N = 5{,}763$ students who were part of a longitudinal extension of a large-scale assessment (von Davier et al., 2011). The only difference between the two test cases is the size of the latent structure. In Test Case D, each dimension is defined based on 15 quadrature points, resulting in 225 nodes, whereas Test Case E uses 31 quadrature points each, giving rise to a latent structure of 961 nodes.

Case F is based on an application of latent class analysis (LCA) for developing a set of conditioning variables (Wetzel, Xu, & von Davier, 2015), not dissimilar to the use of LCA for imputations (Vermunt, van Ginkel, van der Ark, & Sijtsma, 2008). This particular data set is based on publicly available international large-scale assessment data. The number of latent classes is 54, each of which can be represented by a different set of parameters for the response variables. The sample size for Test Case F is 246,112. Test Case G is a five-dimensional MIRT model with simple structure for a data set based on 150 items and 2,026 respondents.

Results

The presentation of results will focus on a comparison of time needed for the most time-consuming part of the runs, the actual estimation of latent distributions and latent expected response frequencies in the levels of the distribution that are calculated as expected values given preliminary item parameters in the E step.

For comparisons, three machines were employed to run all test cases, once with the sequential version and once with the parallel version of the GDM software. More specifically, the runtime advantage of the parallel software was evaluated using a Dell M4700 with 32 GB RAM and a 2.6 GHz i7pro 4(8)-core Intel processor; a 2012 Dell Precision T5500 Workstation with dual X5680 CPU with 2×6 cores and a clock frequency of 3.33 GHz (speed step 3.46 GHz) and 40 GB RAM; and finally, a SuperMicro server with 32(64) (four 8(16)-core processors) AMD Opteron "bulldozer" processors and 128 GB RAM. To preempt any considerations of exceeding the available RAM, all test computers had sufficient amounts of unallocated RAM even when running the largest data sets from Test Cases A and B.

Table 2 shows the results for the i7pro nonparallel and parallel versions of the algorithm for all test cases. The table shows the number of iterations needed, the log-likelihood after convergence, and the time taken in seconds by the two variants of the algorithm. It is important to note that the speedup can reach a factor of 4.77 or more, which would relate to a proportion of parallelism of more than 100% on a four-core machine. However, the i7pro processor supports hyperthreading and vectorization of some instructions. These capabilities improve performance so that the speedup can be expected to be better than a four-threaded algorithm but not quite reaching the performance on eight physical cores, because eight hyperthreads need to share resources. The average degree of parallelism reaches 92%, and the average speedup for this selection of examples equals a factor of 3.68.

Table 3 shows the results for the comparison between parallel and serial EM algorithm using the dual-CPU 12-core Xeon 3.33 GHz workstation. The main result appears to be that the Xeon-based runs are sped up by a factor that ranges from 2.63 to 9.7 when moving to parallel processing. On average across these seven examples, the speedup reaches a factor of 6.6. Closer inspection reveals that the serial version of the algorithm requires more time on the Xeon than on the i7pro in three cases, about the same time in one case, and less time in two cases, while the parallel version appears to be faster on the 12-core Xeon in all instances.

Table 2 Results for the i7pro Four-Core 2.9 GHz Serial Versus Parallel Algorithm

	i7pro single core			i7pro parallel			
Case	Iterations	Likelihood	Seconds	Iterations	Likelihood	Seconds	Speedup
A	126	−14,547,731.24	1,719	126	−14,547,731.24	360	4.77
B	112	−14,639,728.20	1,053	112	−14,639,728.20	287	3.67
C	165	−125,200.51	4,719	165	−125,200.51	1,158	4.07
D	76	−14,468,510.90	70	76	−14,468,510.90	29	2.41
E	86	−14,468,485.63	984	86	−14,468,485.63	242	4.06
F	1,028	−1,234,570.30	5,350	1,028	−1,234,570.30	1,223	4.37
G	277	−130,786.39	3,370	277	−130,786.39	1,382	2.43

Table 3 Results for the Xeon 12-Core 3.46 GHz Serial Versus Parallel Algorithm

	Xeon single core			Xeon parallel			
Case	Iterations	Likelihood	Seconds	Iterations	Likelihood	Seconds	Speedup
A	126	−14,547,731.24	1,356	126	−14,547,731.24	168	8.07
B	112	−14,639,728.20	1,127	112	−14,639,728.20	117	9.63
C	165	−125,200.51	2,465	165	−125,200.51	314	7.85
D	76	−14,468,510.90	44	76	−14,468,510.90	11	4.00
E	86	−14,468,485.63	1,155	86	−14,468,485.63	163	7.08
F	1,028	−1,234,570.30	7,444	1,028	−1,234,570.30	1,039	7.16
G	277	−130,786.39	2,499	277	−130,786.39	949	2.63

Table 4 Results for the AMD 32/64-Core 2.6 GHz Serial Versus Parallel Algorithm

	AMD single core			AMD parallel			
Case	Iterations	Likelihood	Seconds	Iterations	Likelihood	Seconds	Speedup
A	126	−14,547,731.24	2,074	126	−14,547,731.24	256	8.10
B	112	−14,639,728.20	1,553	112	−14,639,728.20	185	8.39
C	165	−125,200.51	6,131	165	−125,200.51	889	6.89
D	76	−14,468,510.90	116	76	−14,468,510.90	21	5.52
E	86	−14,468,485.63	1,945	86	−14,468,485.63	150	12.96
F	1,028	−1,234,570.30	6,427	1,028	−1,234,570.30	377	17.04
G	277	−130,786.39	6,771	277	−130,786.39	287	23.59

The average speedup is 6.63 across the seven test cases. This is equivalent to a level of parallelism of 89% when reversing Amdahl's law. This rate indicates that there may be room for improvement by further reducing the 11% of the algorithm that appears to be processed sequentially.

The last comparison is based on the SuperMicro server with 32(64) cores. The AMD "bulldozer" Opteron 6276 includes eight floating-point cores linked in clusters to two integer cores. Therefore the performance should be somewhere between a 32-core machine and a 64-core machine, because resources are shared, but more hardware support is available than what is present in hyperthreading architectures. Table 4 presents the results for this four-CPU 32(64)-core AMD Opteron server configured with 128 GB RAM.

The average speedup for the AMD server is 11.8, and assuming a full 64-core architecture, this corresponds to an average level of parallelism of more than 92%. When looking at the architecture as a 32-core machine (based on the number of floating-point CPUs), the estimated level of parallelism would be 94% – 95%. Note that in two cases, the speedup reaches 17 and 23.6, respectively. This was validated in several reruns and shows that, depending on the model, different levels of speedup can be expected, while in all cases, significant speedup can be observed.

Discussion

The examples presented in this report provide strong evidence that parallel programming can be valuable for psychometric analyses of large data sets with modern latent variable models. The comparison of execution times shows that the parallel EM algorithm for multidimensional latent variable models can provide much-needed speedup when analyzing large data sets on shared-memory multiprocessor systems. The fact that practically every recent PC or laptop is based on a multicore processor implies that most analyses can be conducted in about 50% of the time required compared to nonparallel versions of the algorithm. In dedicated workstations with multiple CPUs, each providing four, six, or even eight cores, the speedup is even more significant. The speedup reaches a factor of 6 and higher in the examples examined in this report.

This result gains importance once it is understood that all analyses with latent trait models, except for the most basic and naive ones, require extensive quality control and iterative modeling decisions. Each complex data set likely contains response variables or respondents that cannot be appropriately fitted with the initial model.

Examples are classification and clustering algorithms, which are commonly used in unsupervised learning in machine learning applications. Test Case D is a typical data mining or machine learning application where a range of nominal

variables is used to find a latent segmentation of the sample by means of a LCA. In the tryouts reported here, the speedup reaches a factor of 5.09 on the Xeon 12-core workstation hardware. Multiple runs with different numbers of classes are common in these types of applications, and the increase in execution speed certainly helps to arrive at a decision when several classification models have to be estimated and compared.

Another example is with international databases, in which each response variable is based on an item that was presented in a variety of translations, so there is always room for translation issues that may change the meaning, and hence the difficulty, of the items. This may lead to the need to adjust parameterizations and to release equality constraints (Glas & Jehangir, 2014; Oliveri & von Davier, 2014; von Davier et al., 2011). Other sources of model misfit that may require iterative refinements of model constraints are position and context effects (Mazzeo & von Davier, 2014). These issues result in the need not just to run one model once and be done with it but rather to start with a null model that assumes the highest level of constraints and iteratively rerun models with fewer constraints following the evaluation of model data fit (e.g., Molenaar, 1983; Rost & von Davier, 1994; von Davier & Molenaar, 2003). This common analytic requirement to run several estimations underlines the importance of a speedup of analysis that often reduces the time required from 90 minutes or more to 15 minutes or less.

The advantage in execution speed is significant with no consequences to the accuracy of the results. All test cases converge to the same parameters and latent variable distributions. The log-likelihood is identical even in cases with close to 2,000,000 response vectors. Even if there were minute differences in the 6th or 7th decimal, this is not an error introduced by the parallel implementation but rather an effect that is due to numerical calculations and, in particular, summation with finite accuracy. In numerical calculations, addition, strictly speaking, is not commutative: For very long sums, that is, those we encounter when calculating the log-likelihood of the data across individual students, for almost 2,000,000 students, the sequence of additions may make a (very small, not practically relevant) difference. If the Gilula and Haberman (1994) log-penalty would be used instead, the resulting average likelihood per response would be the same for the first 16 or more decimals. In all cases, the same number of iterations was needed for all implementations and hardware combinations, and the estimated latent variable distributions and item parameters were identical.

The advantages of parallel implementation for psychometric modeling and parameter estimation are profound. Instead of relying on subsamples and/or approximations and simplifications based on the model structure (e.g., Cai, 2010a, 2010b; Rijmen, Jeon, Rabe-Hesketh, & von Davier, 2014) or computational approximations (e.g., Jeon & Rijmen, 2014; Jeon, Rijmen, & Rabe-Hesketh, 2013; Rijmen & Jeon, 2013), parallel computing for psychometric modeling with general latent variable models can provide analyses based on the full data without shortcuts.

A move to special-purpose hardware for further speedup of the algorithm presented here appears to be straightforward. Parallel algorithms can utilize special-purpose graphical processing units (GPUs) that provide a much larger number of specialized cores or, alternatively, can make use of multicore coprocessors (such as the Xeon-Phi series) for further speedup. Note that the so-called hyperthreading technology does not provide further speedup, as it does not double the number of physical cores but rather arranges them into virtual cores only. However, even on a customary four-core laptop or notebook computer, significant increases in estimation speed can be gained by applying the PEPM algorithm presented in this report.

References

Amdahl, G. M. (1967). Validity of the single processor approach to achieving large-scale computing capabilities. *AFIPS Conference Proceedings*, *30*, 483 – 485. doi:10.1145/1465482.1465560

Cai, L. (2010a). High-dimensional exploratory item factor analysis by a Metropolis – Hastings Robbins – Monro algorithm. *Psychometrika*, *75*, 33 – 57.

Cai, L. (2010b). Metropolis – Hastings Robbins – Monro algorithm for confirmatory item factor analysis. *Journal of Educational and Behavioral Statistics*, *35*, 307 – 335.

Cai, L. (2013). *flexMIRT: A numerical engine for flexible multilevel multidimensional item analysis and test scoring* (Version 2.0) [Computer software]. Chapel Hill, NC: Vector Psychometric Group.

Cui, H., Tumanov, A., Wei, J., Xu, L., Dai, W., Haber-Kucharsky, J., & Xing, E. (2014). Exploiting iterativeness for parallel ML computations. In *Proceedings of the ACM Symposium on Cloud Computing* (p. 1 – 14). New York, NY: ACM.

Cui, H., Wei, X., & Dai, M. (2010). *Parallel implementation of expectation-maximization for fast convergence*. Retrieved from http://users.ece.cmu.edu/~hengganc/archive/report/final.pdf

Das, A. S., Datar, M., Garg, A., & Rajaram, S. (2007). Google News personalization: Scalable online collaborative filtering. In *Proceedings of the 16th International Conference on World Wide Web, WWW 07* (pp. 271–280). New York, NY: ACM. doi:10.1145/1242572.1242610

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood estimation from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, *39*, 1–38.

Gilula, Z., & Haberman, S. J. (1994). Models for analyzing categorical panel data. *Journal of the American Statistical Association*, *89*, 645–656.

Glas, C. A. W., & Jehangir, K. (2014). Modeling country-specific differential item functioning. In L. Rutkowski, M. von Davier, & D. Rutkowski (Eds.), *Handbook of international large-scale assessment: Background, technical issues, and methods of data analysis* (pp. 97–115). New York, NY: Springer.

Gustafson, J. L. (1988). Reevaluating Amdahl's law. *Communications of the ACM*, *31*, 532–533.

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

Jeon, M., & Rijmen, F. (2014). Recent developments in maximum likelihood estimation of MTMM models for categorical data. *Frontiers in Psychology*, *5*, 269. http://doi.org/10.3389/fpsyg.2014.00269

Jeon, M., Rijmen, F., & Rabe-Hesketh, S. (2013). Modeling differential item functioning using the multiple-group bifactor model. *Journal of Educational and Behavioral Statistics*, *38*, 32–60.

Lopez de Teruel, P. E., Garcia, J. M., & Acacio, M. E. (1999). The parallel EM algorithm and its applications in computer vision. In *Proceedings of the PDPTA, 1999* (pp. 571–578). Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.4.8470&rep=rep1&type=pdf

Mazzeo, J., & von Davier, M. (2014). Linking scales in international large-scale assessments. In L. Rutkowski, M. von Davier, & D. Rutkowski (Eds.), *Handbook of international large-scale assessment: Background, technical issues, and methods of data analysis* (pp. 228–258). New York, NY: Springer.

McLachlan, G., & Krishnan, T. (1997). *The EM algorithm and extensions.* Hoboken, NJ: John Wiley.

Molenaar, I. W. (1983). Some improved diagnostics for failure of the Rasch model. *Psychometrika*, *48*, 49–72.

Muthén, L. K., & Muthén, B. O. (1998–2011). *Mplus user's guide* (6th ed.). Los Angeles, CA: Authors.

Oliveri, M. E., & von Davier, M. (2014). Toward increasing fairness in score scale calibrations employed in international large-scale assessments. *International Journal of Testing*, *14*(1), 1–21.

Rijmen, F., & Jeon, M. (2013). Fitting an item response theory model with random item effects across groups by a variational approximation method. *Annals of Operations Research*, *206*, 647–662.

Rijmen, F., Jeon, M., Rabe-Hesketh, S., & von Davier, M. (2014). A third order item response theory model for modeling the effects of domains and subdomains in large-scale educational assessment surveys. *Journal of Educational and Behavioral Statistics*, *38*, 32–60.

Rost, J., & von Davier, M. (1994). A conditional item fit index for Rasch models. *Applied Psychological Measurement, 18*, 171–182.

Vermunt, J. K., & Magidson, J. (2013). *Technical guide for Latent GOLD 5.0: Basic, advanced, and syntax*. Belmont, MA: Statistical Innovations.

Vermunt, J. K., van Ginkel, J. R., van der Ark, L. A., & Sijtsma, K. (2008). Multiple imputation of incomplete categorical data using latent class analysis. *Sociological Methodology, 38*, 369–397.

von Davier, M. (2005). *A general diagnostic model applied to language testing data* (Research Report No. RR-05-16). Princeton, NJ: Educational Testing Service. 10.1002/j.2333-8504.2005.tb01993.x

von Davier, M. (2008). A general diagnostic model applied to language testing data. *British Journal of Mathematical and Statistical Psychology, 61*, 287–307.

von Davier, M. (2009). Some notes on the reinvention of latent structure models as diagnostic classification models. *Measurement Interdisciplinary Research and Perspectives, 7*(1), 67–74.

von Davier, M. (2010). Hierarchical mixtures of diagnostic models. *Psychological Test and Assessment Modeling, 52*(1), 8–28.

von Davier, M. (2013). The DINA model as a constrained general diagnostic model—two variants of a model equivalency. *British Journal of Mathematical and Statistical Psychology, 67*, 49–71.

von Davier, M. (2014). *The log-linear cognitive diagnostic model (LCDM) as a special case of the general diagnostic model (GDM)* (Research Report No. RR-14-40). Princeton, NJ: Educational Testing Service. 10.1002/ets2.12043

von Davier, M., & Molenaar, I. W. (2003). A person-fit index for polytomous Rasch models, latent class models, and their mixture generalizations. *Psychometrika, 68*, 213–228.

von Davier, M., & Rost, J. (1995). Polytomous mixed Rasch models. In G. H. Fischer & I. W. Molenaar (Eds.), *Rasch models—Foundations, recent developments and applications* (pp. 371–379). New York, NY: Springer.

von Davier, M. & Rost, J. (2016). Logistic mixture-distribution response models. In W. van der Linden (Ed.), *Handbook of item response theory* (2nd ed., Vol. 1, pp. 393–406). Boca Raton, FL: CRC Press.

von Davier, M., Xu, X., & Carstensen, C. H. (2011). Measuring growth in a longitudinal large scale assessment with a general latent variable model. *Psychometrika, 76*, 318–336.

von Davier, M., & Yamamoto, K. (2004, October). *A class of models for cognitive diagnosis*. Paper presented at the 4th Spearman Invitational Conference, ETS, Philadelphia, PA.

Wetzel, E., Xu, X., & von Davier, M. (2015). An alternative way to model population ability distributions in large-scale educational surveys. *Educational and Psychological Measurement, 75*, 739–763.

Xu, X., & von Davier, M. (2008). *Fitting the structured general diagnostic model to NAEP data* (Research Report No. RR-08-27). Princeton, NJ: Educational Testing Service. 10.1002/j.2333-8504.2008.tb02113.x

Suggested citation:

von Davier, M. (2016). *High-performance psychometrics: The parallel-E parallel-M algorithm for generalized latent variable models* (Research Report No. RR-16-34). Princeton, NJ: Educational Testing Service. 10.1002/ets2.12120

Action Editor: Shelby Haberman

Reviewers: Deirdre Kerr and Frederic Robin