

Model Evaluation

ACT-R, IMPRINT, and Matlab

Comparisons and Parameter Optimizations

- ACT-R, IMPRINT, and Matlab
- First 'unified test problem' Keystroke entry task
- Second 'unified test problem' RADAR (Matlab conversion in progress)
- Computational speed summary
- Parameter optimization Genetic algorithms / Simulated annealing
- Conclusions
- Real-time demonstration of Genetic Algorithm optimization

Bengt Fornberg

University of Colorado, Boulder
Department of Applied Mathematics

Presentation at annual MURI meeting, held at University of Colorado, Boulder, September 12, 2008.

<u>Modeling System:</u>	<u>Designed for:</u>
ACT-R	cognitive modeling
IMPRINT	human and system performances in military tasks
Matlab	science and engineering applications

We are **NOT** comparing the systems' overall capabilities in a broad sense, only their effectiveness on very precisely stated 'unified test problems'

- The issue is **ACT-R and IMPRINT and additional possibilities (notably Matlab)**.
- Long-term goal: Integrate platforms in a way that combines best features of the different systems.

First 'unified test problem'

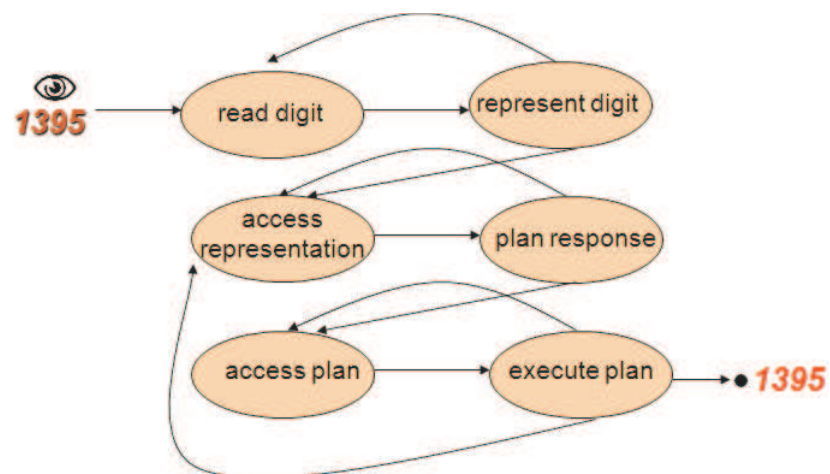
Keystroke data entry task: Healy, Kole, Buck-Gengler and Bourne (2004)

Type on a number key pad a series of 4-digit numbers (e.g. 1395) as quickly and accurately as possible when they are presented; terminate with pressing the "Enter" key

In more detail:

	Experiment 1 32 subjects	Experiment 2 32 subjects
Stimuli	4-digit numbers presented on screen No repetitions of digits; 0 not used Numeral format only for presentation No inter-trial pause	
Practice schedule	64 numbers per block, each repeated 5 times in 1 st half, with new set in 2 nd half.	640 unique numbers
Hand used for typing (only right-handed subjects)	Left hand only	Both hands used (R-R,R-L,L-R,L-L)
Response	Digits typed on number key pad only No visual feedback of digits typed	

Cognitive model:



Model implementations and performances on this task have earlier been reported in detail:

ACT-R

Gonzales, Fu, Healy, Kole and Bourne (2006)

IMPRINT

Buck-Gengler, Raymond, Healy and Bourne (2007)

Matlab, with ACT-R & IMPRINT comparisons

Fornberg, Raymond and Best (2007)

- The ACT-R and the IMPRINT codes were developed separately;
- The Matlab code is a mathematically equivalent translation from the IMPRINT code.

Reported last year:

Goodness of fit between models and experiment

- The ACT-R, IMPRINT and Matlab models all gave comparable (and satisfactory) agreement with experimental results for both speed and accuracy of digit data entry.
- Usually the smallest variations were seen when the same model was run repeatedly (or when IMPRINT was compared with its direct translation to Matlab).
- With the Matlab model, it is easy and fast to 'tune' parameters to fit still better with experiment - **a critical capability both for improving the model's goodness of fit, and for obtaining new insights from the model.**

Also reported last year:

Comparisons of computational speeds between models

ACT-R

Typical timing report for one simulation

```
; cpu time (non-gc) 605,045 msec (00:10:05.045) user, 392 msec system
; cpu time (gc)      251,940 msec (00:04:11.940) user, 77 msec system
; cpu time (total)  856,985 msec (00:14:16.985) user, 469 msec system
; real time 859,952 msec (00:14:19.952)
```

- non-gc **Actual program execution**
 - gc (garbage collection) **Lisp system activity**
 - total **around 14 minutes**







IMPRINT

Wall-clock timing (only method available) - about **24 minutes**

Matlab

Extensive profiling options available. Sample output:

Lines where the most time was spent

Line Number	Code	Calls	Total Time	% Time	Time Plot
105	<code>g_dist = (0.5*sum(randn(nr_c,8...</code>	320	0.025 s	29.1%	
116	<code>total_times = ...</code>	320	0.017 s	20.3%	
113	<code>CC_PH_Actual = (Counts(:,1)-Co...</code>	320	0.010 s	11.7%	
73	<code>Counts = cumsum([[ItemCt,Corre...</code>	320	0.006 s	7.1%	
71	<code>ab = 3-s1-s2;</code>	320	0.003 s	3.5%	
All other lines			0.024 s	28.3%	
Totals			0.085 s	100%	

Total Matlab time: **0.085 seconds.**

Parameter optimizations

Complete results reported in Raymond, Fornberg, Buck-Gengler, Healy and Bourne (2008)

The Matlab translation of the IMPRINT model has about a dozen parameters describing cognitive factors.

- Select a subset of key parameters related to the typing speed.
- Vary the parameters in order to improve agreement between model and experiment.
- Use the obtained values to get insights into the underlying cognitive and motoric processes.

Parameters selected:

Cognitive learning	(cognitive)
Motoric learning	(physical)
Repetition priming	(rep learn)
Left-hand penalty	(left penalty)
Cognitive slowdown	(fatigue)

'Reasonable ranges':

[-0.20, 0.00]
[-0.10, 0.00]
[0.00, 0.30]
[1.00, 1.50]
[0.00, 0.10]

'Hand derived values':

-0.045
-0.015
0.050
1.125
0.0125

Measure selected:

RMSE (Root Mean Square Error; Averaged over Experiments 1 and 2)

First approach - Exhaustive Search

First illustration: Holding 3 out of the 5 variables at their 'Hand derived values' and letting the remaining two vary over their full 'Reasonable ranges': → → → →

Every dot represents two full simulations (Experiments 1 and 2):

IMPRINT: 48 minutes Matlab: 0.17 seconds

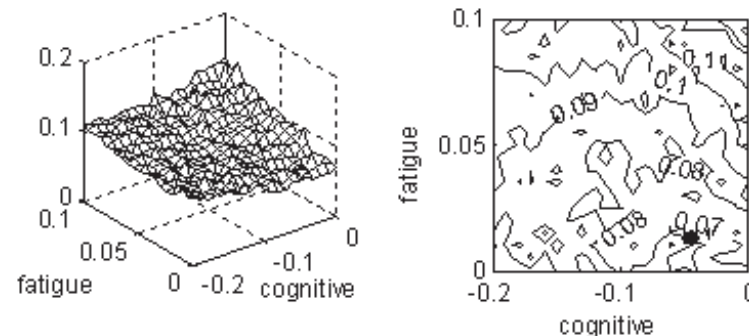
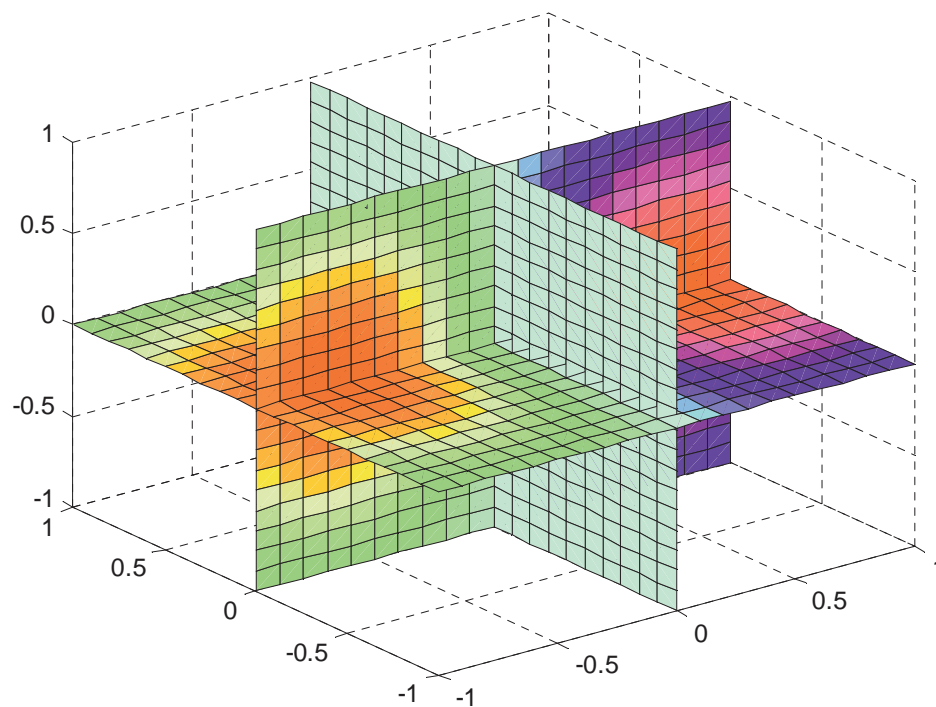


Illustration in a 3-D parameter space

One single slice in each 2-D subspace: most of the parameter space has not been explored.

'Volume' increases extremely fast with number of free parameters.

In engineering applications, there are routinely 1000's of free parameters.



Look at EVERY way of selecting out 2 of the 5 parameters

This search still reaches only a minute part of the full 5-D space

Each slice 21x21 data points

10 slices:

$$10 \times 21^2 = 4,410 \text{ points}$$

Full space at same resolution:

$$21^5 = 4,084,101 \text{ points}$$

If we have 10 parameters:

Full space at same resolution:

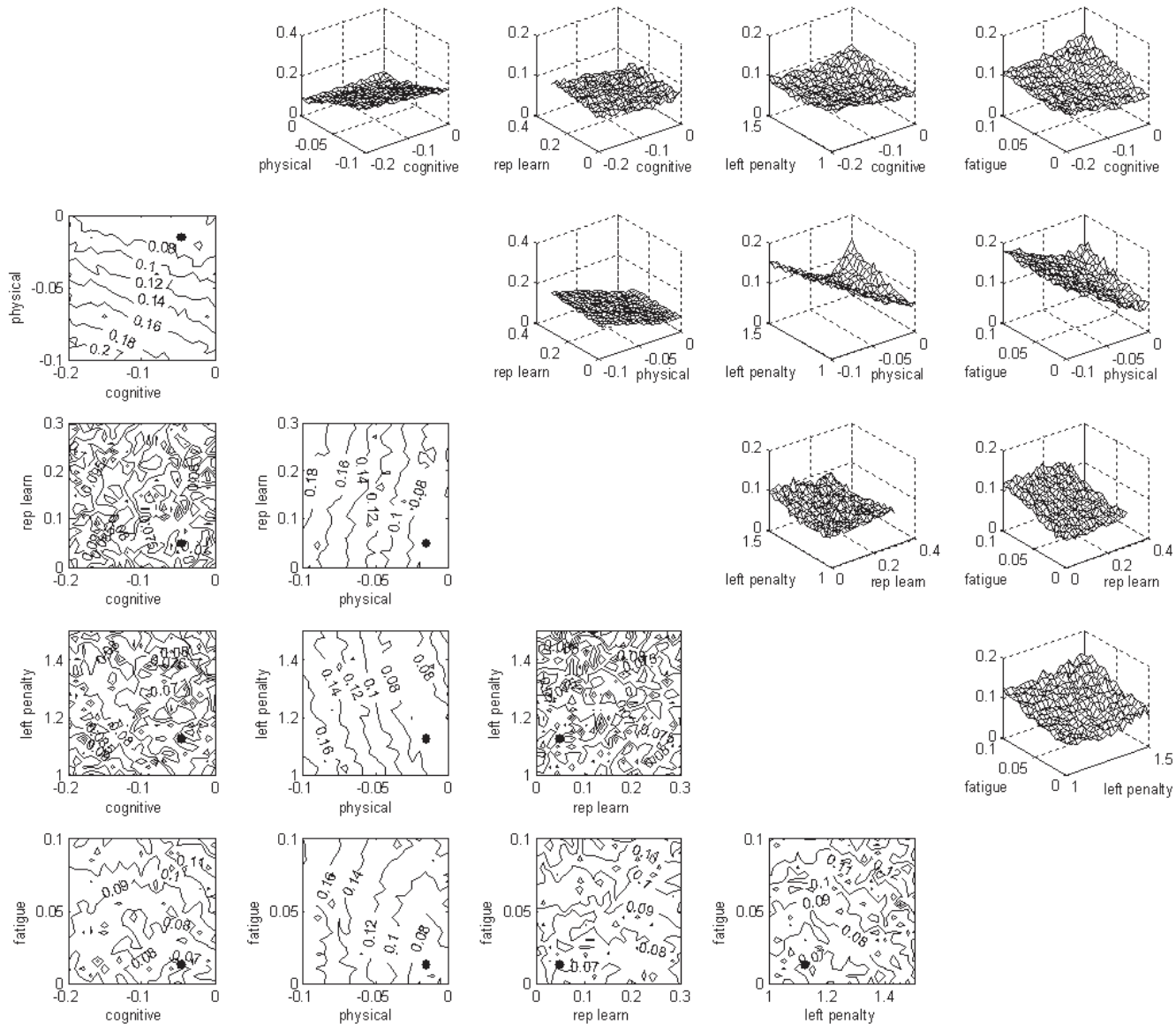
$$21^{10} = 1.7 \cdot 10^{13} \text{ points}$$

IMPRINT: 50 times age of universe

Matlab: 17,000 times faster
(better, but still way too slow)

We need BOTH

- Faster computational environment
- Much better algorithms



Second approach - Heuristic Search

Two major strategies for a much more efficient 'exploration' of the high-dimensional space:

- Simulated annealing (SA)
- Genetic algorithms (GA)

The main challenges are:

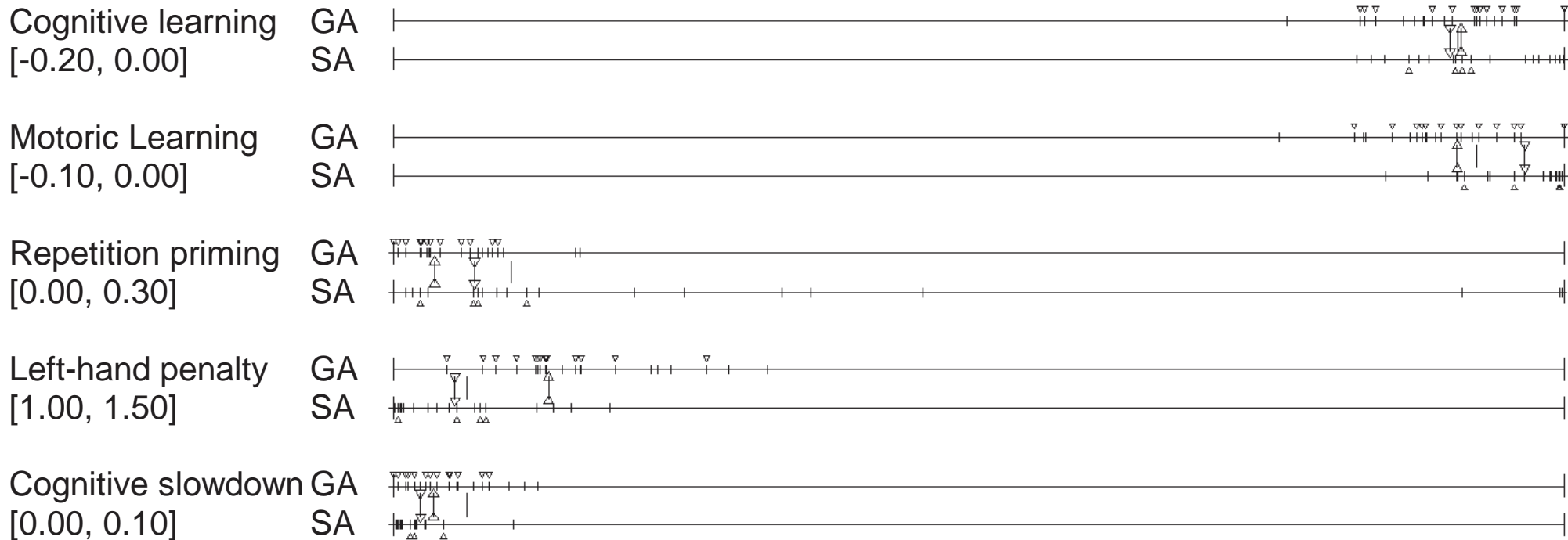
- To explore vast spaces effectively
- To find global minima without getting trapped in local minima.

Both strategies:

- Borrow their concept from processes in nature (formation of a crystal, and evolution of species, resp.),
- Are quite insensitive to 'noise' in the data,
- Are available in a Matlab 'tool box' by just 3 - 4 lines of extra programming.

Real-time demonstration of Genetic Algorithm optimization will be shown after the conclusion slides

GA and SA optimization results (summary of 20 runs)



Each GA: 60 'generations' with a population of 30 'individuals'

Each SA: Single search path taking 1800 steps

Time for each GA or SA optimization: 5 minutes on PC used for earlier timing comparisons;
30 seconds on new dual quad core Dell T5400.

Multiple evaluations of GA and SA offer new opportunities for

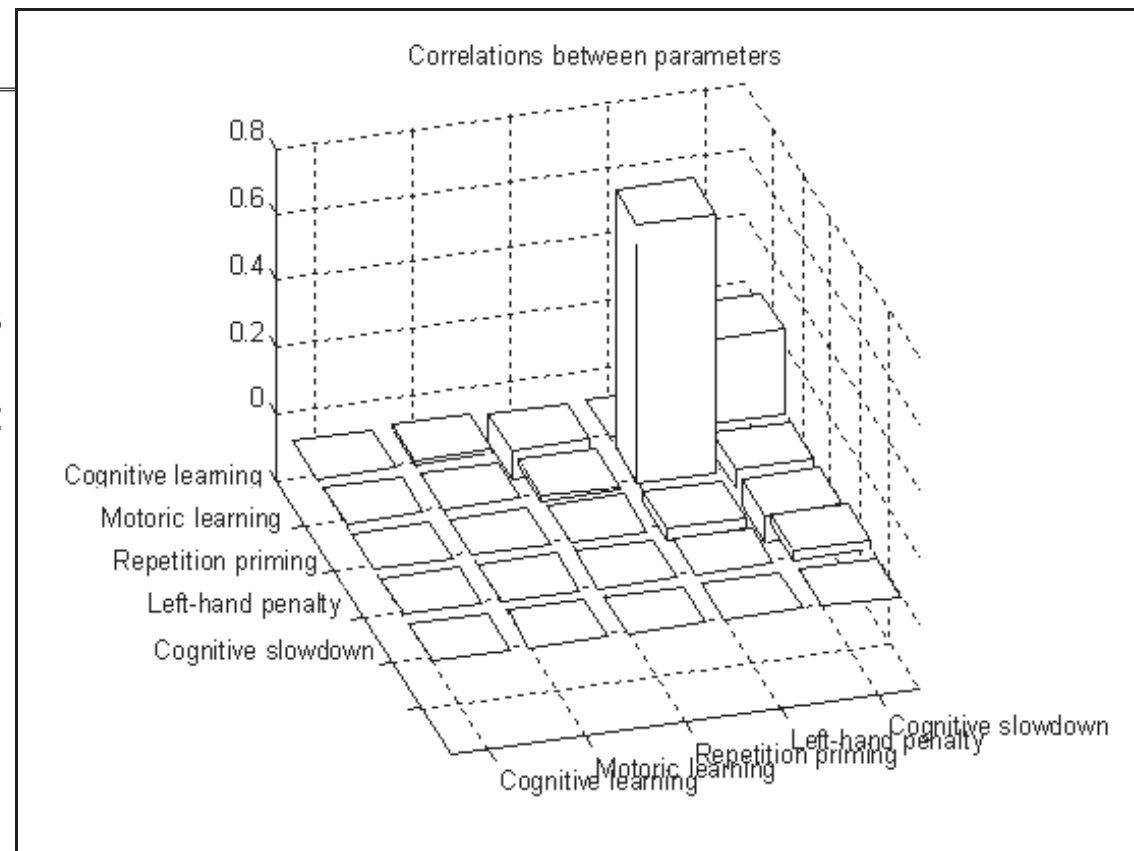
- Estimating model uncertainties
- Finding correlations between parameters

Pairwise optimal parameter approximation correlations

		Cognitive learning	Motoric learning	Repetition priming	Left-hand penalty	Cognitive slowdown
Cognitive learning	GA	-	-0.276	-0.247	-0.013	-0.536
	SA	-	-0.038	0.348	-0.043	-0.466
Motoric learning	GA		-	-0.058	-0.920	0.124
	SA		-	0.277	-0.840	-0.474
Repetition priming	GA			-	0.150	0.397
	SA			-	-0.204	-0.211
Left-hand penalty	GA				-	0.053
	SA				-	0.543
Cognitive slowdown	GA					
	SA					

Above: Parameter correlations between 20 GA and (separately) 20 SA runs

Right: Product of GA and SA correlations: strongly highlights two unexpected but, in retrospect, reasonable relations



Conclusions

Noted earlier:

- ACT-R and IMPRINT are very different systems, with very different built-in capabilities.
- When creating simple models in 'equation form', Matlab can offer vast speed advantages (by factors of 10,000 or more) over ACT-R and IMPRINT codes.

New results:

- GA and SA are fulfilling their promises of providing major opportunities for global parameter optimizations (without any need for using massively parallel computer systems).
- Automated parameter optimization leads to a better understanding of the underlying cognitive processes:

By finding optimal parameter values, meaningful comparisons of parameter values can be made. One such comparison involves cognitive and motoric learning. The parameter for cognitive learning was found to be much larger than that for motoric learning, suggesting that the cognitive component of the task improves more with practice than does the motoric component. Thus, the parameter values provide insight into the underlying psychological processes, and the relative contribution (or weighting) to learning of task taxons.

Plans for the future:

- Conversion of IMPRINT to Matlab for the second 'unified test problem' (RADAR) is in progress (IMPRINT code only very recently available in stable form, so no results yet).
- Develop more graphical tools for visualizing processes that involve several independent variables
- Explore the possible speeding up of GA and SA simulations by artificially lowering the statistical fluctuations going into the simulations (see if parameter results stay the same)
- Explore incorporation of fast equation-based code segments (Matlab, C, Fortran, ...) in IMPRINT and ACT-R environments.
- Compare models in terms of their assumptions about underlying cognitive processes
- Compare models in terms of the accuracy and range of their predictions with respect to the results of future experiments
- Compare models in terms of their ability to simulate the range of individual differences in performance