Two methods for optimising cognitive model parameters

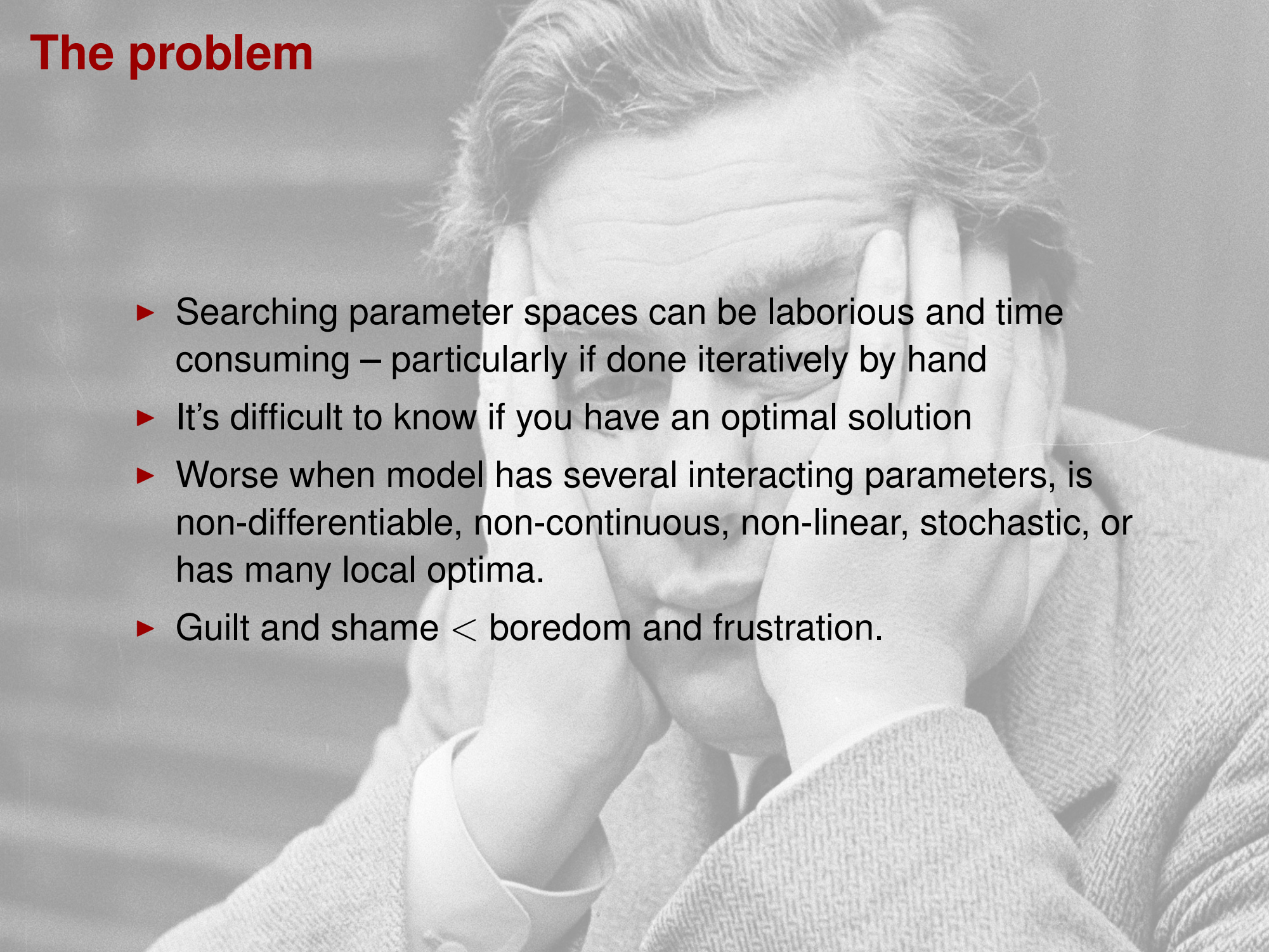
David Peebles

—

9th August 2016

University of HUDDERSFIELD

The problem



- Searching parameter spaces can be laborious and time consuming – particularly if done iteratively by hand
- It's difficult to know if you have an optimal solution
- Worse when model has several interacting parameters, is non-differentiable, non-continuous, non-linear, stochastic, or has many local optima.
- Guilt and shame $<$ boredom and frustration.

One solution, two methods

- Use a population of models to explore the parameter space.
- **Differential evolution (DE)**
  - Genetic algorithm for multidimensional real-valued functions.
  - Ideal for optimising models with relatively few parameters or short run times on a single computer.
- **High-throughput computing with HTCondor**
  - Open source software for managing job scheduling on computer clusters or *cycle scavenging* idle computers.
  - Ideal for larger, more complex models or when simulating the behaviour of many participants.
  - Used by many universities and easy to set up on local networks (or so I'm told).

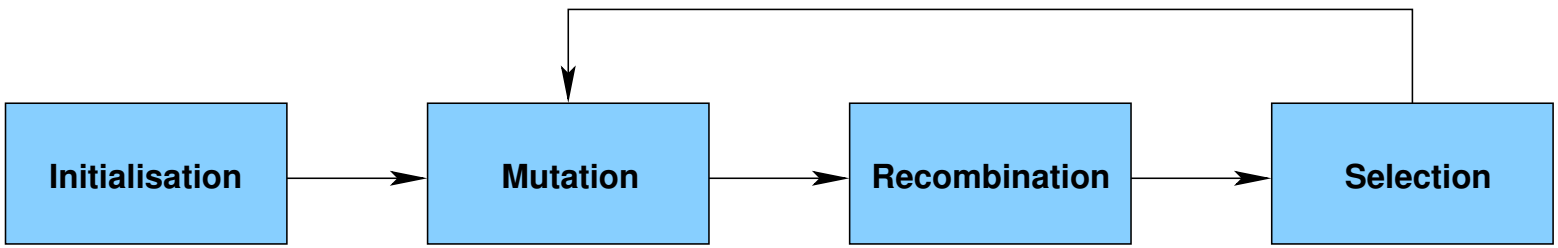
Differential evolution

- An evolutionary strategy for real numbers used for multidimensional numerical optimisation [2, 3].
- **Attractions:**
  - Simplicity of the algorithm
  - Only three control parameters
  - Fast, accurate, and robust performance
  - Wide applicability
- **Control parameters:**

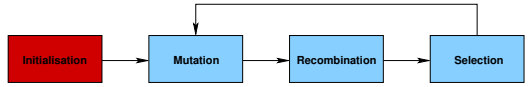
  NP the population size
  F a scale factor applied to the mutation process
  Cr a constant that regulates the crossover process

The DE algorithm in a nutshell

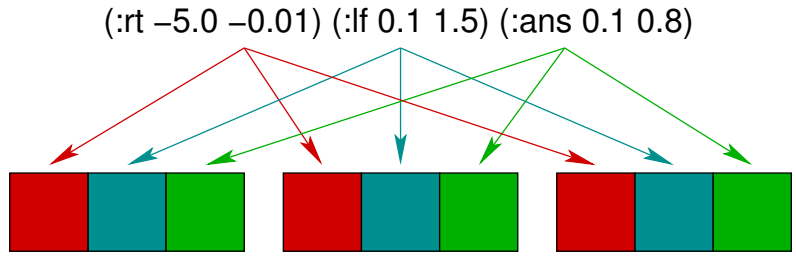
DE applies repeated cycles of *mutation*, *recombination*, and *selection* on an initial, randomly generated population of vectors to create a single vector that produces the best solution to a problem.



Initialisation

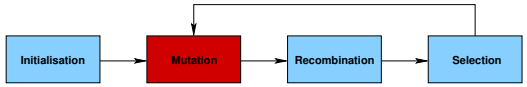


- Create a population of real-valued vectors – one dimension for each of the model parameters to be optimised.
- Initialise vectors with uniformly distributed random numbers within maximum and minimum bounds set for each dimension.

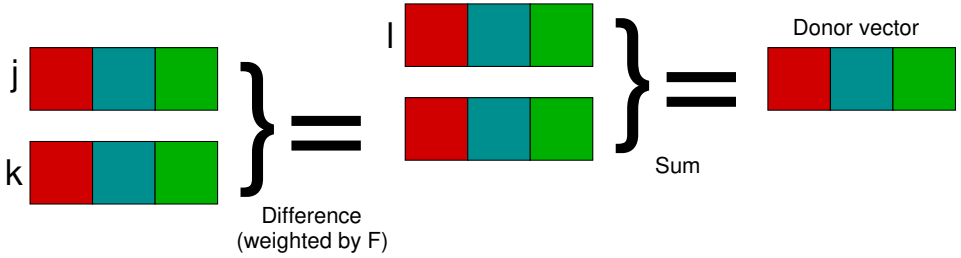


- To create the next population of vectors, each vector $\mathbf{i}$ in the current population is selected in sequence, designated as the **target** vector, and subjected to a competitive process. . .

Mutation

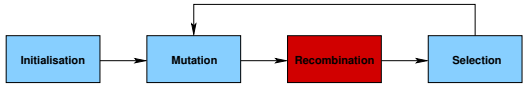


- Mutation is the weighted difference between two vectors in the current population.
- A mutated *donor* vector is created by randomly selecting three unique vectors, $\mathbf{j}$, $\mathbf{k}$ and $\mathbf{l}$ from the population and adding the difference between $\mathbf{j}$ and $\mathbf{k}$ (scaled by the F parameter) to $\mathbf{l}$.

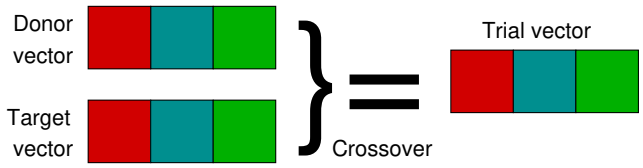


- This ensures that differences in the scale and sensitivity of different vector parameters are taken into account and that the search space is explored equally on all dimensions.

Recombination

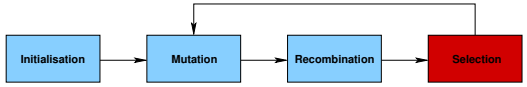


- Cross the donor vector with the target vector to create the *trial* vector to allow successful solutions from the previous generation to be incorporated into the trial vector.



- Achieved by series of Bernoulli trials which determine for each dimension which parent will donate its value.
- Moderated by the crossover rate parameter Cr $(0 \leq Cr \leq 1.)$.

Selection



- Run the model with the parameter values from the trial vector.
- If $\text{fitness}_{trial} \geq \text{fitness}_{target}$ replace target vector with trial vector in next generation, else retain target vector.
- Repeat for each vector in the current population until the next population is created and the evolutionary process continues for a user-defined number of cycles.
- Retain vector with highest fitness in each population.
- Winning vector in the final population is considered the best solution to the problem.

Paired Associate model example

1. Load (a) ACT-R, (b) DE code file, and (c) the fan effect model.
2. Modify the output function to just return one correlation.
3. Define a function to run the task suppressing the model output and any warnings resulting from poor parameter choices.
4. Specify the three parameters to adjust with their ranges and use the default DE configuration.

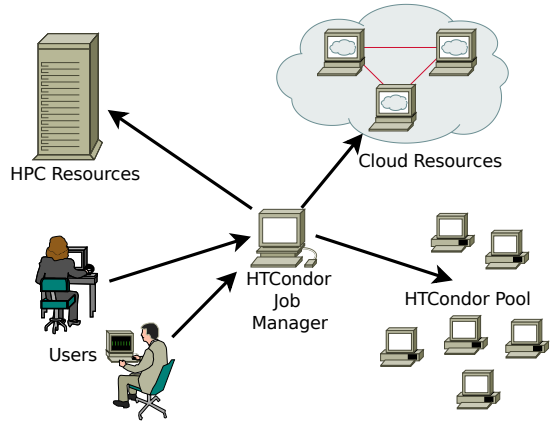
```
(defun output-data (data n)
    (let ((probability (mapcar (lambda (x) (/ (first x) n)) data))
          (latency (mapcar (lambda (x) (/ (or (second x) 0) n)) data)))
;;(correlation latency *paired-latencies*)
  (correlation probability *paired-probability*)))

(defun run-paired-with-no-output ()
    (let ((*standard-output* (make-string-output-stream)))
          (suppress-warnings (paired-experiment 100))))

(optim 'run-paired-with-no-output '((:rt -5.0 -0.01) (:lf 0.1 1.5) (:ans 0.1 0.8)))
```

HTCondor

- DE useful for models with relatively few parameters or short run times on a single computer.
- DE too slow for large, complex models or when simulating the behaviour of many participants.



- Run a population of models over a network using *HTCondor*.
- HTCondor – open source, cross-platform software system for managing and scheduling tasks over computer networks [1].

Running ACT-R models on HTCondor

1. Modify your model to allocate random values to parameters.
2. Create a *submit description file* specifying job details (executable, platform, the model files, commands etc.).

```
requirements =(OpSys == “WINNT61” && Arch == “INTEL”) ||||
                    (OpSys == “WINDOWS” && Arch == “INTEL”) ||||
                    (OpSys == “WINDOWS” && Arch == “X86_64”))

executable = actr-s-64.exe
arguments = “-l ’paired.lisp’ -e ’(collect-data 20)’ -e ’(quit)’ ”
transfer_executable = ALWAYS
transfer_input_files = paired.lisp

output = out.stdout.$(Cluster).$(Process)
error = out.err.$(Cluster).$(Process)
log = out.clog.$(Cluster).$(Process)

queue 100
```

Random number generation

- A potential problem arises with how the pseudorandom number generator (PRNG) seed is set.
- Each model instance must be initialised with a unique seed.
- Many Lisps (and the ACT-R random module) use the computer's clock to generate seed. Problem if different instances are given the same clock time.
- To avoid this, use an ACT-R PRNG that doesn't using the system clock, uses a random number generated from CCL (the Lisp used to create the MS Windows standalone ACT-R).
- This method is employed in a new ACT-R *create-random-module* which is loaded prior to loading the model file.

Why not use MindModeling@Home?

- NOT a toothbrush thing
- MAYBE a lack of knowledge thing
- MAYBE a *World Community Grid* thing (Zika, cancer cures).
- Some models are small/trivial
- Speed, convenience and flexibility

Try them out

Code and instructions for using both methods using the *paired associate* model from tutorial unit 4.

- **DE:** https://github.com/peebz/actr-paired-de
- **HTCondor:** https://github.com/peebz/actr-paired-htc

Thanks as always to **Dan Bothell** for his patience and advice.

References I


M. J. Litzkow, M. Livny, and M. W. Mutka. 'Condor—A Hunter of Idle Workstations'. In: *Proceedings of the 8th International Conference on Distributed Computing Systems*. IEEE. June 1988, pp. 104–111.


R. Storn and K. Price. *Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces*. Tech. rep. TR-95-012. Berkeley, CA: ICSI Berkeley, 1995.


R. Storn and K. Price. 'Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces'. In: *Journal of global optimization* 11.4 (1997), pp. 341–359.