Using Genetic Programming to predict GeneChip performance on an nVidia 8800

W. B. Langdon

Mathematical and Biological Sciences
and Computing and Electronic Systems

University of Essex

Evolving GeneChip Correlation Predictors on Parallel Graphics Hardware

CIGPU 2008

Predicting GeneChip Probe Performance by Interpreting Genetic Programming on a GPU

- What are GeneChips
- Why are GeneChip correlations important
- Preparation of training data
- Interpreting multiple GP programs simultaneously on GPU
- Simultaneously interpreting 256000 programs
  - 16 384 (used in GeneChip analysis)
- Actual speed 0.3 - 1.0 billion GP ops /second
- Evolved predictor

Affymetrix HG-U133A

- Simultaneously measure activity of (almost) all human genes.
- mRNA concentration low, so data noisy.
- 21 765 probesets with exactly 11 pairs of probes per gene.
- GeneChips cost approx £500 each.
- 6685 human tissue samples.

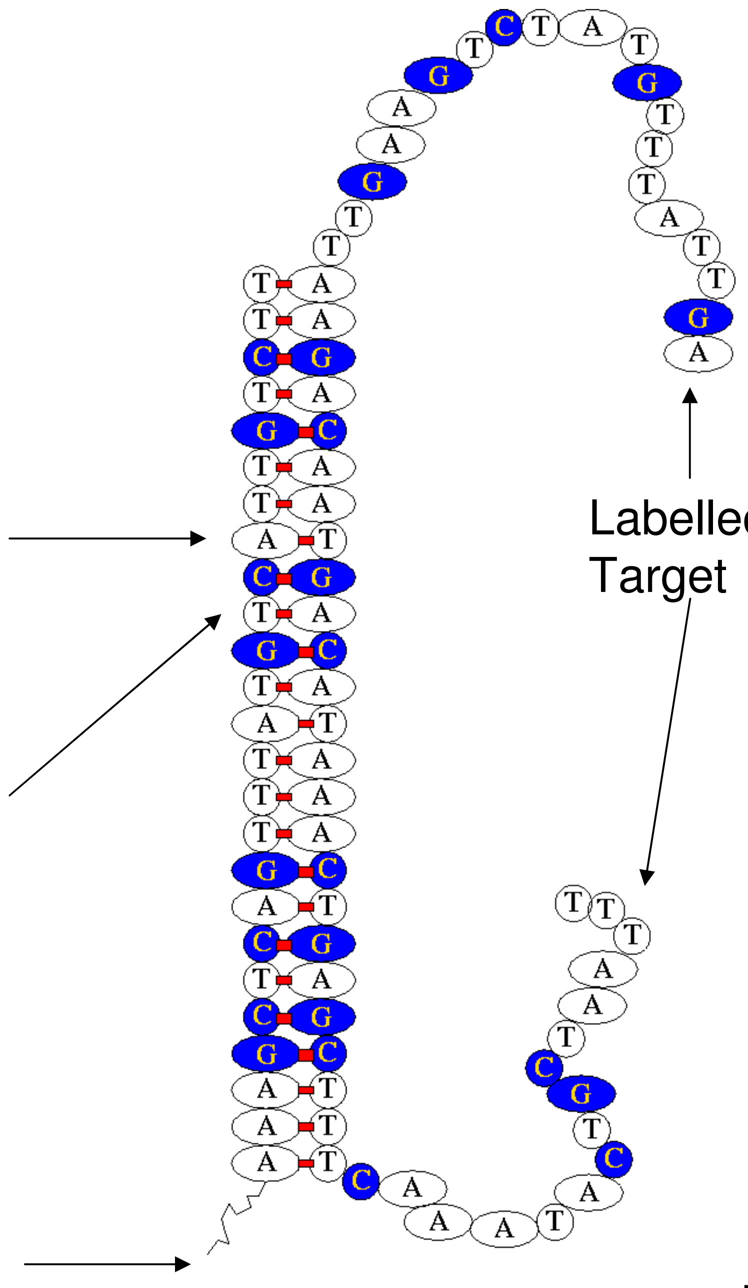
How GeneChips work

- Gene produces messenger RNA
- mRNA treated with fluorescent maker
- Labelled marker prefentially binds to complementary base sequence on chip.
- Laser scans chip to measure concentration and location of fluorescent markers.

Target bound to DNA on chip

Probe and target linked by complementary bases to form double helix

DNA probe 25 bases long



Labelled Target

DNA tied to chip

A ↔ T Adenine binds to Thymine.

C ↔ G Cytosine binds to Guanine

Probeset Correlations

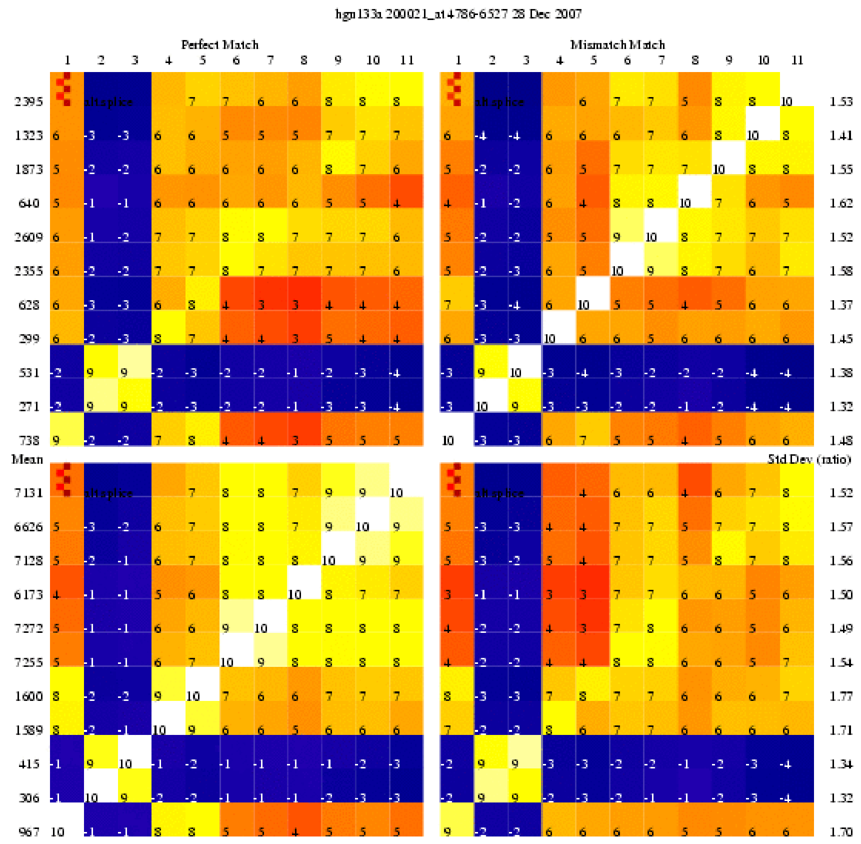
- 11 pairs (PM and MM) of measurements
- All measurements are designed to measure activity of same gene. They should be correlated.
- Calculate correlation. This shows some probes are NOT correlated with others.
- Use genetic programming to find systematic patterns which suggest a probe will be poor.
- Pattern can give insight into biochemistry and physics of GeneChips.

Example Correlation Matrix

Calculated correlations between all probe pairings for every probeset on HG-U133A.

Yellow high correlation.
Blue low/no correlation.

Interpretation of Affymetrix data controversial.
Some signals not behaving as wanted.



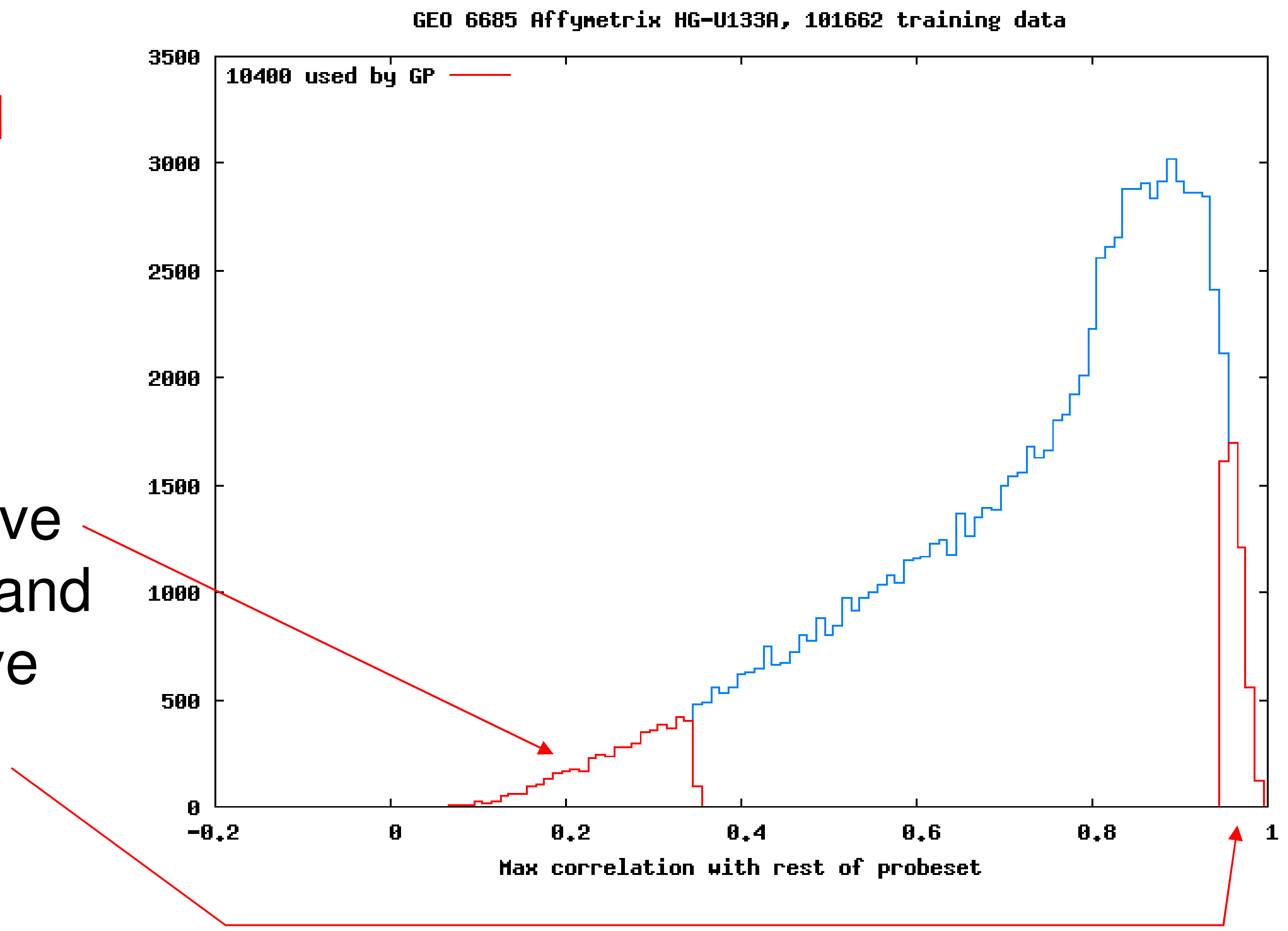
Training data

- 5.3 million correlation calculated.
- Exclude probesets with little or no signal
  - 13 863 probesets with ≥3 pairs of highly correlated (>0.8) probes.
  - 13863×22 probes (3.2 million pairs)
- Max correlation with rest of probeset
- Randomly split: training, validation, holdout:
  - 101662 training examples.
  - 5200 highest and 5200 lowest used.

Distribution of Max Correlation with rest of Probeset

GP uses lowest and highest

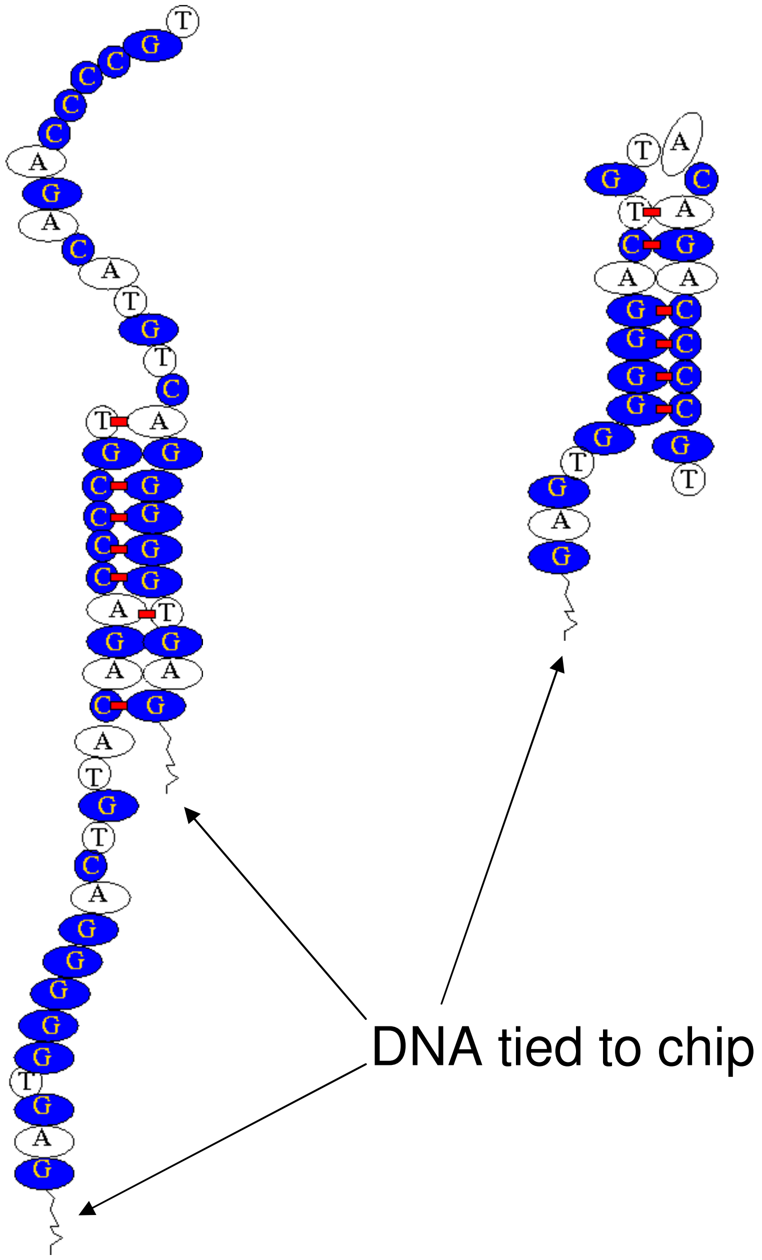
Each generation 100 negative examples and 100 positive examples



Poor Correlation due to Probe Binding?

Looked at two possible probe interactions: Watson-Crick base pairing between adjacent probes (left) and Watson-Crick binding of a probe to itself.

Binding strength based on counting number of bonds.



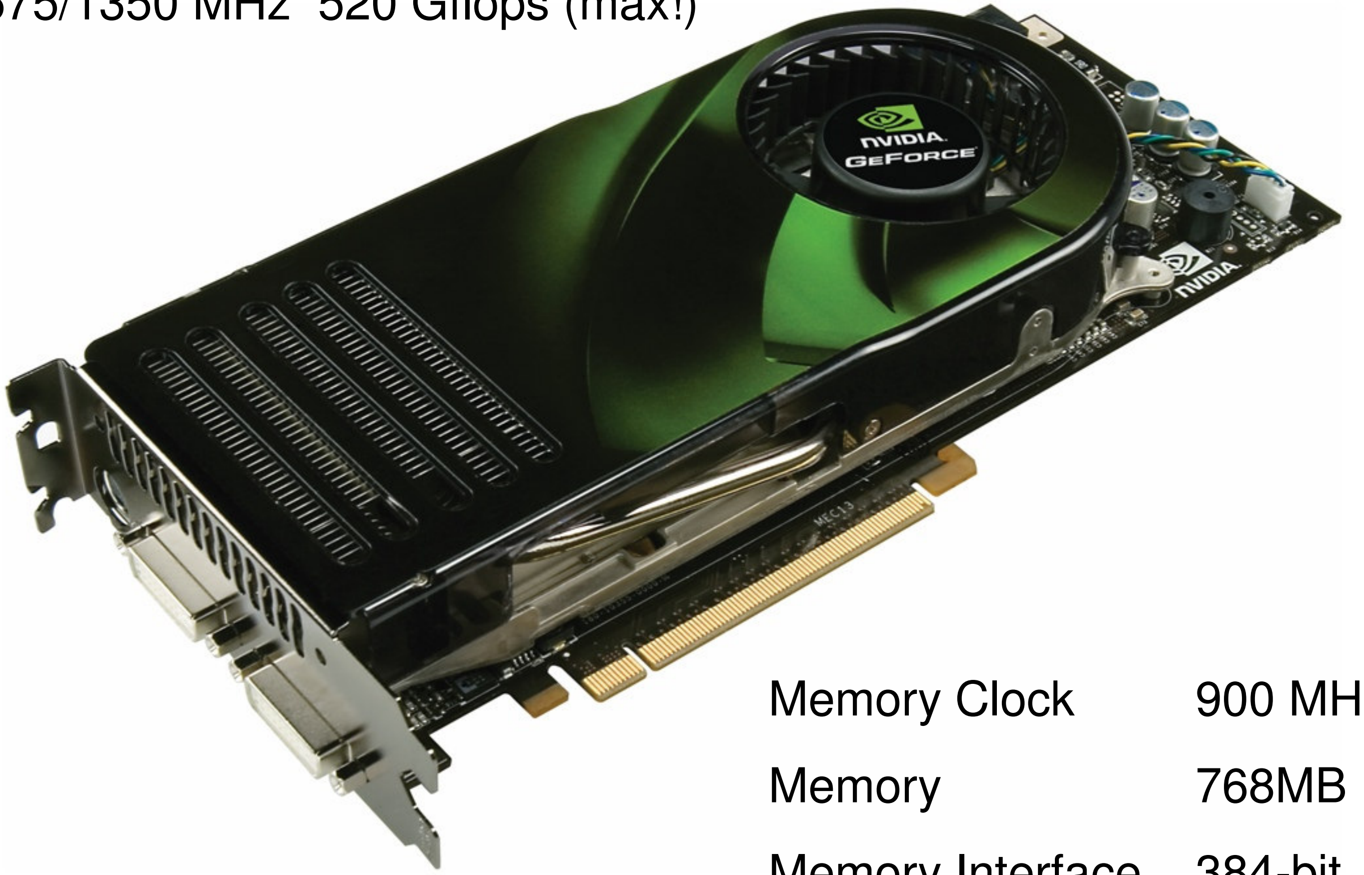
Training Data-Summary

- 47 inputs. (Goal: predict maximal correlation between probe pairs)
- Index of both probes in their probeset.
- Flag to indicate PM or MM (both probes).
- Distances along transcript: between probes and distance from end of probeset (as integers and as fraction of distance spanned by probeset).
- Number of As, Ts, Gs and Cs (as integers and as fractions).
- 25 ATGC values (irrationally coded: $-1/\pi$, $1/\pi$, $-e^{-3/4}$ and $e^{-3/4}$).
- Fraction of probe exposed assuming Watson-Crick probe-probe binding or probe hairpin.

GPU nVidia GTX 8800

128 Stream Processors

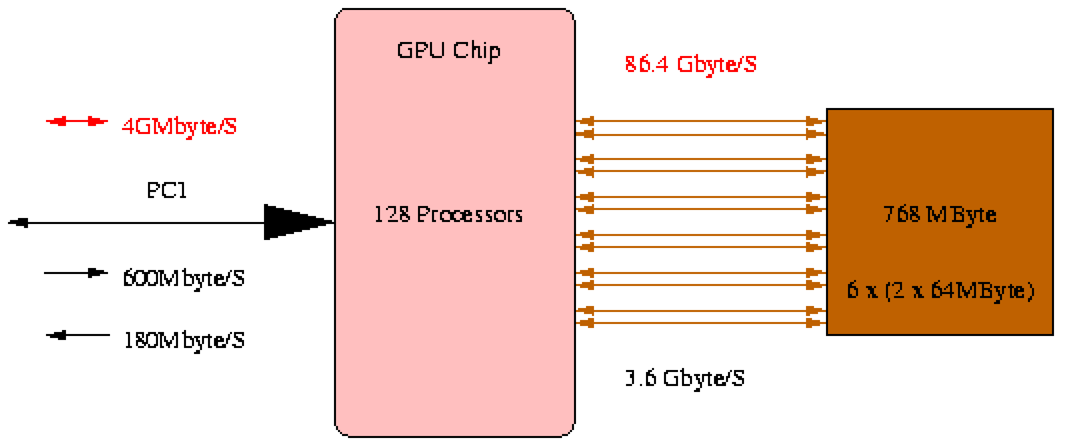
Clock 575/1350 MHz 520 Gflops (max!)



Memory Clock	900 MHz
Memory	768MB (6 × 128)
Memory Interface	384-bit (6 × 64)
Memory Bandwidth	86.4 GB/sec (max!)

GPU chip connections

Linux PC

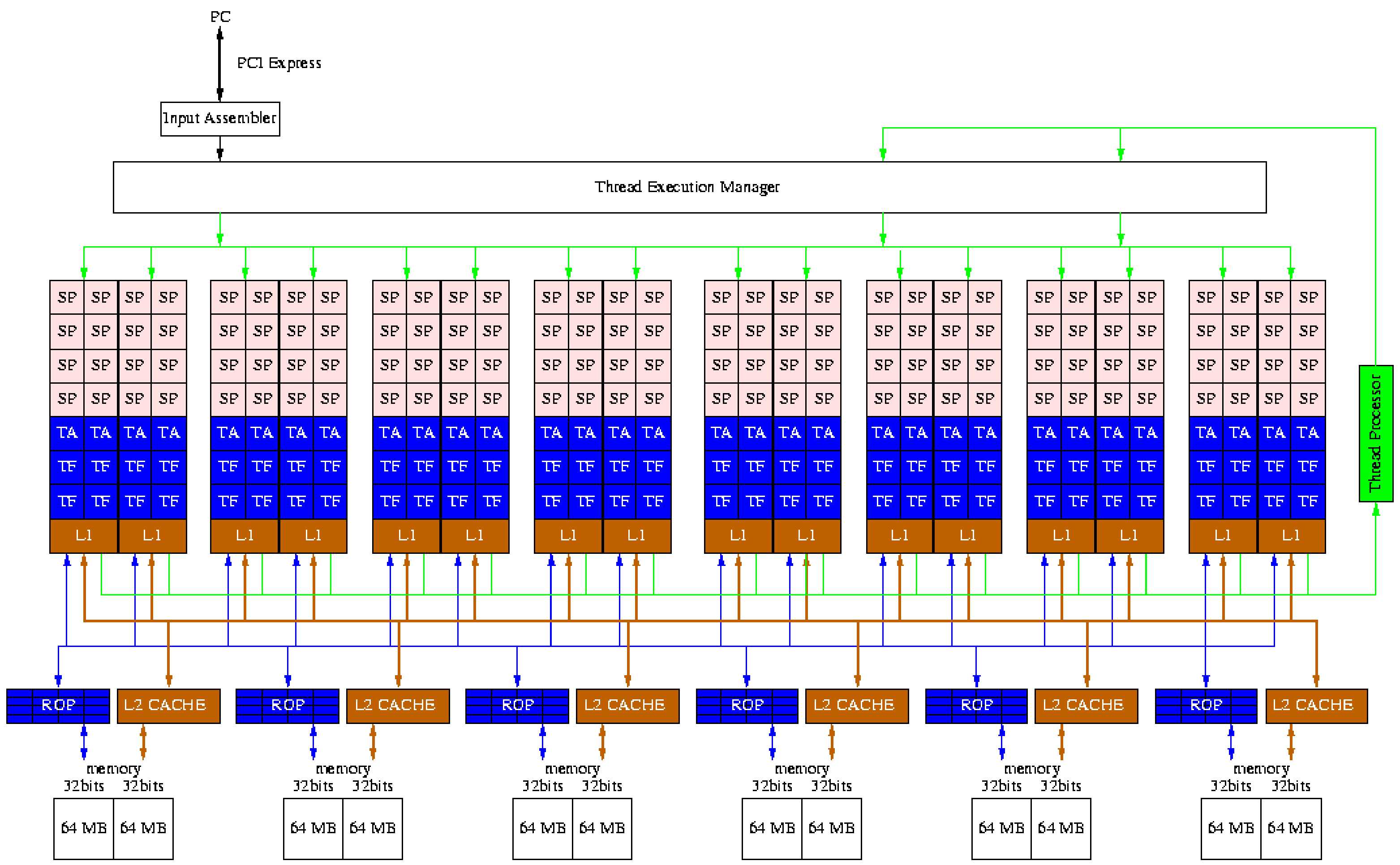


Hype

Actual?

Memory, GPU chip, video hardware etc on one card

128 SP processors = 16 independent blocks of 8



Blue hardware dedicated to graphics

General Purpose GPU Software Options

Most software aimed at graphics. Interest in using them (and CELL processors, XBox, PS3, game consoles) for general purpose computing: GPGPU.

- Microsoft Research windows/DirectX
- BrookGPU stanford.edu
- GPU specific assemblers
- nVidia CUDA
- nVidia Cg
- PeakStream
- Sh no longer active. Replaced by
- RapidMind [Langdon, EuroGP 2008]

RapidMind

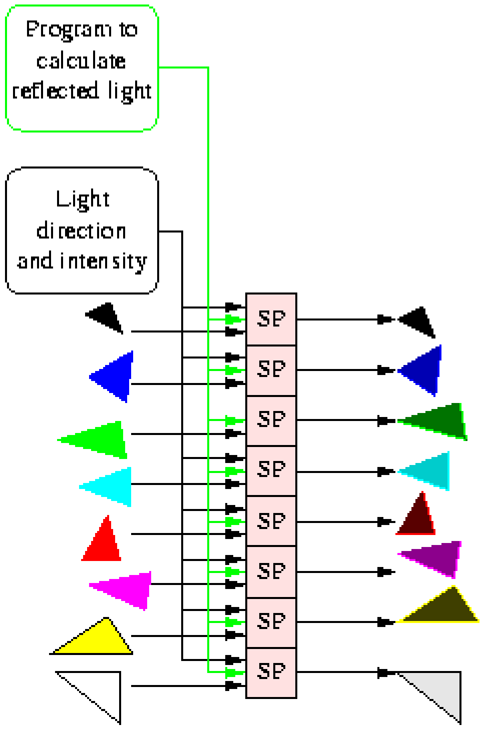
- High level, C++,
- OpenGL/DirectX, Microsoft, Linux, notMac
- CELL and multi-core CPU as well as GPU.
- **Supported**
  - Not free but academics can get a developers license on request.
- Portable between GPUs (many) CELL but code locked-in to RapidMind

RapidMind Software

- Grew out of Sh meta-programming (Waterloo)
- Not source compatible with Sh but very similar concepts.
- High level, C++ very heavy use of templates
- Compatible with free GNU C++
- Templates/GDB on occasion produce huge incomprehensible error messages leading to a difficult learning path.
- Very active, new releases, targeting new hardware. Suggests RapidMind will be a viable option in the future as well as now.
- Still feels like beta release. 18 bugs/gotchas reported.
- Active developer support
- Integrated compiler for GPU works almost without problem.

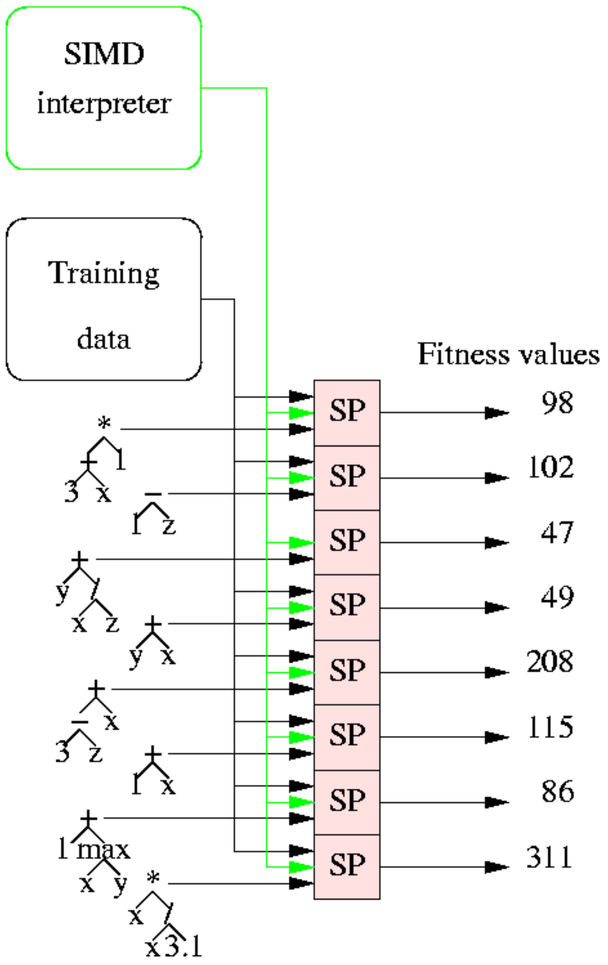
Single Instruction Multiple Data

- GPU designed for graphics
  - 32 bit floating point (2^{-23}) precision
  - Arrays max 4 million elements
- Same operation done on many objects
  - Eg appearance of many triangles, different shapes, orientations, distances, surfaces
  - One program, many data → Simple (fast) parallel data streams
  - GPU does not allow random write access to large arrays. (stack depth)
- How to run many programs on SIMD computer?


Program to calculate reflected light
Light direction and intensity
SP
SP
SP
SP
SP
SP
SP
SP

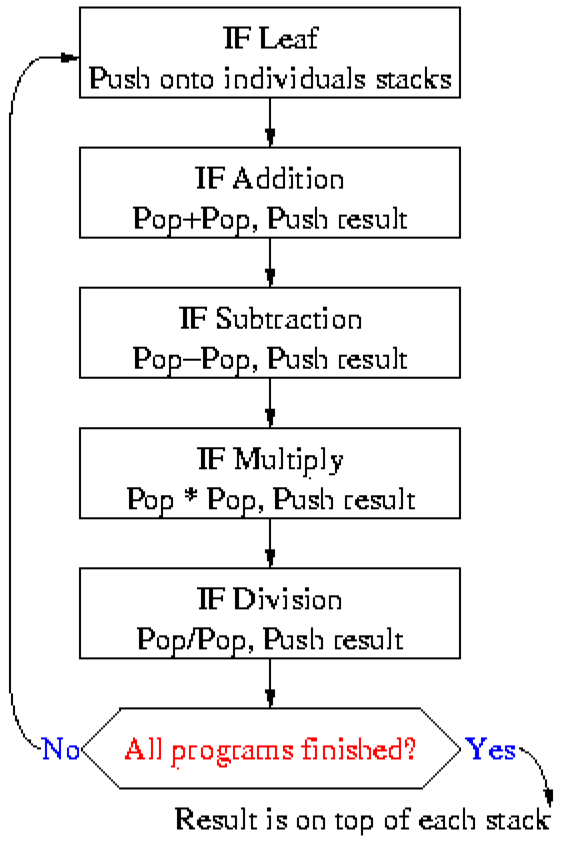
Interpreting many programs simultaneously

- Previous gpu gp used PC to compile individuals to gpu code. Then run one program in multiple data (training cases).
- Avoid compilation by interpreting tree
- Run single SIMD interpreter on GPU on many trees.



GPU Genetic Programming Interpreter

- Programs wait for the interpreter to offer an instruction they need evaluating.
- For example an addition.
  - When the interpreter wants to do an addition, everyone in the whole population who is waiting for addition is evaluated.
  - The operation is ignored by everyone else.
  - They then individually wait for their next instruction.
- The interpreter moves on to its next operation.
- The interpreter runs round its loop until the whole population has been interpreted. (Or a timeout?)



Representing the Population

- Data is pushed onto stack before operations pop them (i.e. reverse polish. x+y $\rightarrow$ $\boxed{x \mid y \mid +}$)
- The tree is stored as linear expression in reverse polish.
- Same structure on host as GPU.
  - Avoid explicit format conversion when population is loaded onto GPU.
- Genetic operations act on reverse polish:
  - random tree generation (eg ramped-half-and-half)
  - subtree crossover
  - 4 types of mutation
- Requires only one byte per leaf or function.
  - So large populations (millions of individuals) are possible.

Cost

- Interpreters avoid compilation but exec is slow
- SIMD two main sources of additional waste
  - Synchronisation means short programs take as long to execute as long programs.
  - Most operations (80%) are not wanted and their results are thrown away.
- Leafs access data and so are much more expensive than functions?
  - A multiplication takes only 4 clock cycles = 3nS
  - Main memory read takes up to 300 clock cycles
  - 50% of trees are leafs.
  - so cost is dominated by leafs not functions?
- We accept other interpreter overheads (eg Lisp, Perl, Python, PHP), so why not SIMD overhead

Examples

- Approximating Pi
- Chaotic Time Series Prediction
- Mega population. Bioinformatics protein classification
    - Is protein nuclear based on num of 20 amino acids
- Predicting Breast Cancer fatalities
    - HG-U133A/B probes →10year outcome
- Predicting problems with DNA GeneChips
    - HG-U133A correlation between probes in probesets →MM, A/G ratio and A×C

Speed of GPU interpreter GeForce 8800 GTX.

Experiment	**Number of Terminals**	**|F|**	**Population**	**Program size**	**Stack depth**	**Test cases**	**Speed (million OPs/sec)**
Mackey-Glass	8+128	4	204 800	11.0	4	1200	895
Mackey-Glass	8+128	4	204 800	13.0	4	1200	1056
Protein	20+128	4	1 048 576	56.9	8	200	504
Laser_a	3+128	4	18 225	55.4	8	151 360	656
Laser_b	9+128	4	5 000	49.6	8	376 640	190
Cancer	1 013 888+1001	4	5 242 880	≤15.0	4	128	535
GeneChip	47+1001	6	16 384	≤ 63.0	8	⅓M, sample 200	314

Lessons

- Suggest interpreting GP trees on the GPU is dominated by leafs:
  - since there are lots of them and typically they require data transfers across the GPU.
  - adding more functions will slow interpreter less than might have been expected.
- To get the best of the GPU it needs to be given large chunks of work to do:
  - Aim for 1-10 seconds.
  - More than about 10 seconds and Linux dies
    - Solved by not using GPU as main video interface??
  - Less than 1millisec Linux task switching dominates
- Poor debug, performance tools
- Code via FTP

GeneChip Results

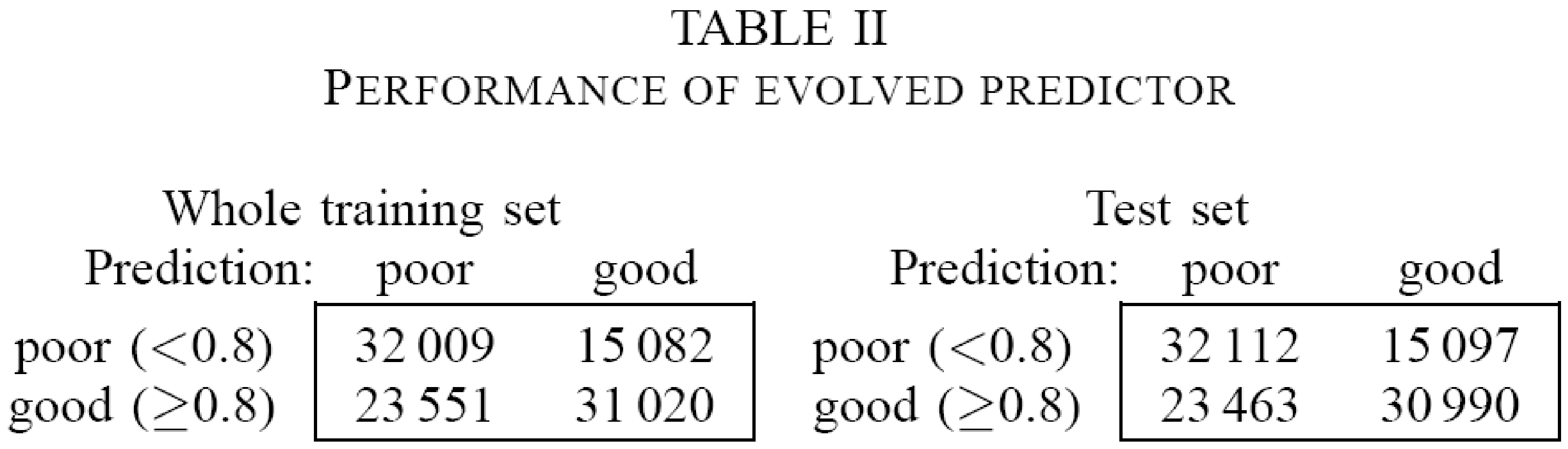


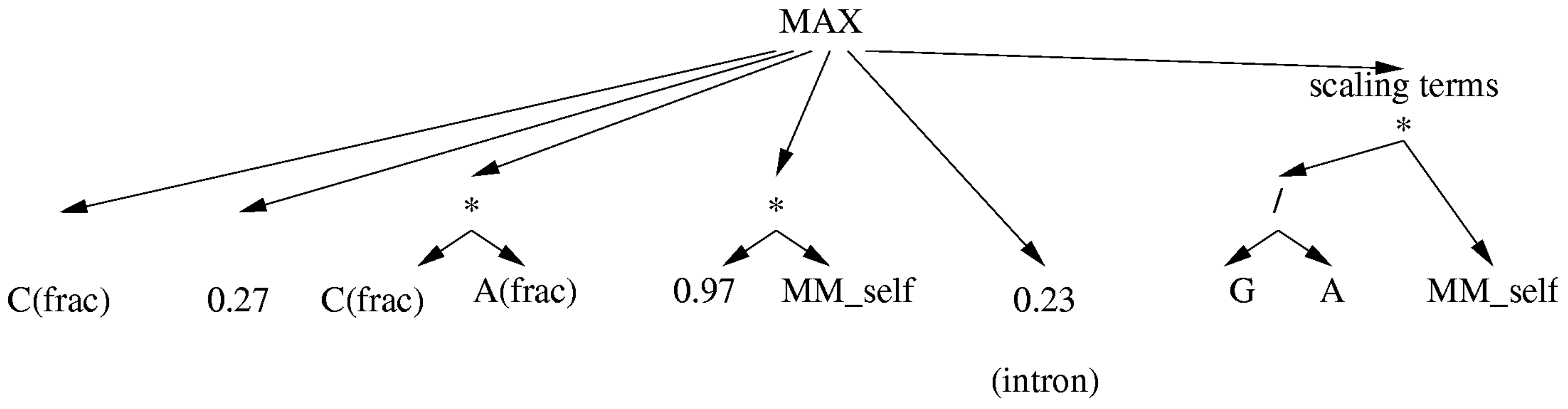
TABLE II
PERFORMANCE OF EVOLVED PREDICTOR

	Whole training set			Test set	
Prediction:	poor	good	Prediction:	poor	good
poor (<0.8)	32 009	15 082	poor (<0.8)	32 112	15 097
good (≥0.8)	23 551	31 020	good (≥0.8)	23 463	30 990

No over fitting.

Evolved predictor on average within 0.16 of actual correlation

Evolved GeneChip Predictor

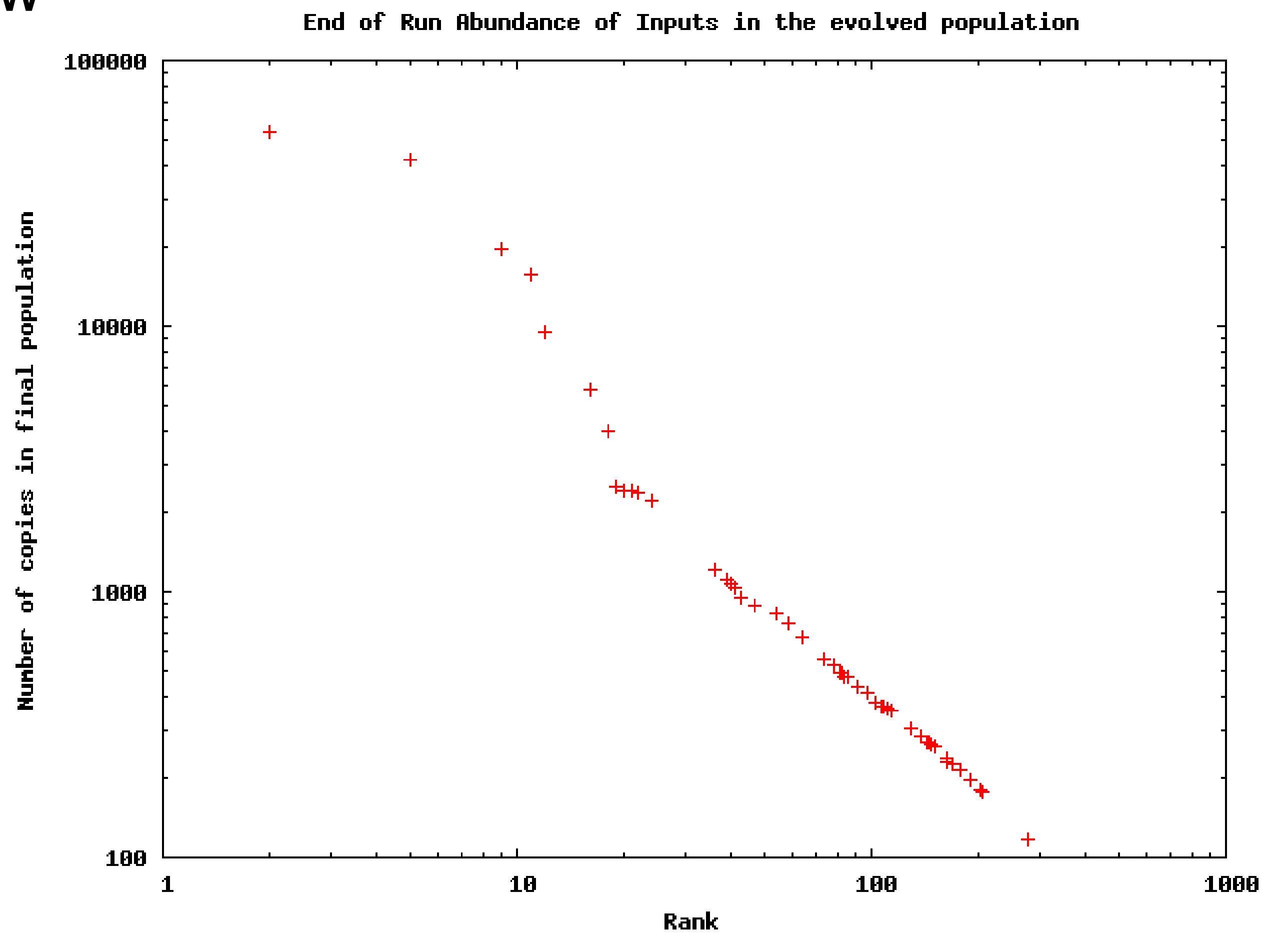


Simplification of evolved HG-U133A probe correlation predictor.

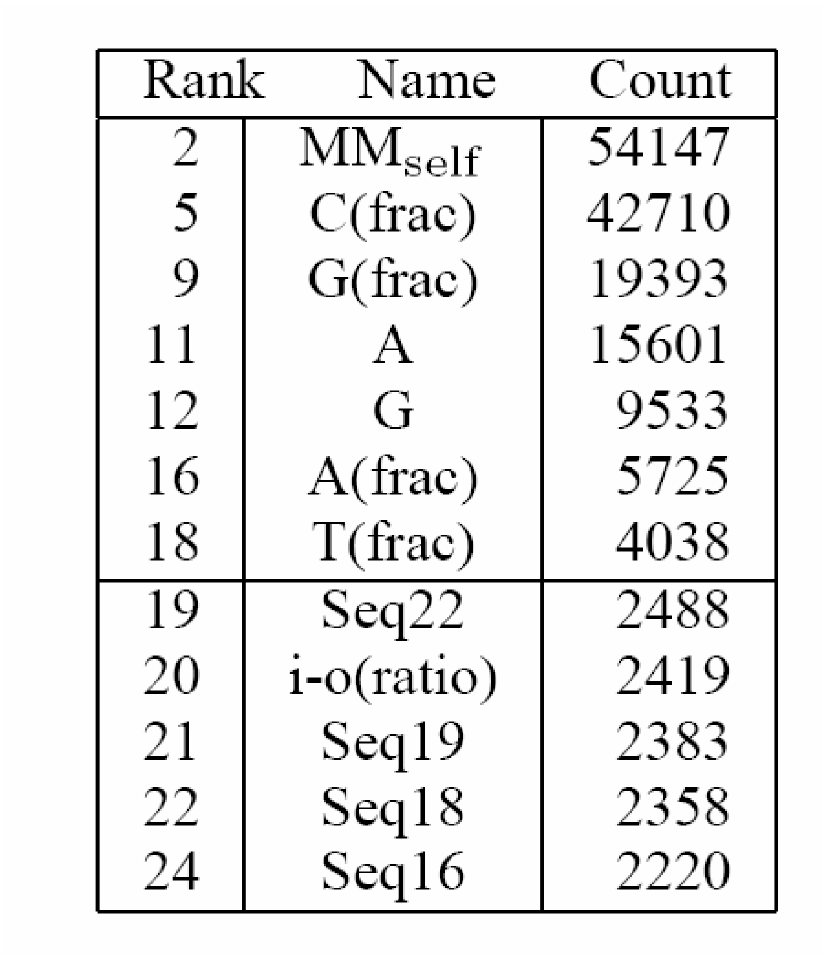
The most important factors are if the probe is MM or PM and the G/A ratio.

Importance of the 47 Inputs

Zipf law



Relative Importance of Inputs



Rank	Name	Count
2	MM_{self}	54147
5	C(frac)	42710
9	G(frac)	19393
11	A	15601
12	G	9533
16	A(frac)	5725
18	T(frac)	4038
19	Seq22	2488
20	i-o(ratio)	2419
21	Seq19	2383
22	Seq18	2358
24	Seq16	2220

Table gives values for data in previous graph.

MM important (cf. evolved predictor) followed by total number of each base. (cf A/G ratio).

Hairpin and Watson-Crick pairing not much used.

Discussion

- PM/MM dominates, i.e. if probe is PM or MM is the most important.
- Followed by number of each base in probe
- Difficult to recognise patterns “motifs” in probe sequence.
- Two predetermined probe-probe bindings do not appear important.
- Supplied simplified Watson-Crick type probe-probe interactions. Difficult for GP to consider other types of binding (**G-quadruplex**, i-motif, etc).

Conclusions

- Use GPUs cheap, convenience, fast, getting faster (now 256×1.5GHz $500)
- GPU difficult to program, but GPGPU tools
- Running multiple trees on "single instruction multiple data" (SIMD) parallel computer
- Simultaneously interpreting 256000 programs
- Actual speed 0.2 - 1.0 billion GP ops /second
  - 0.1 peta GP opcodes per day $400
- GP automatically finding information on Affymetrix. This has feed into potential bio-physical explanation and so to improved data analysis.

END

Questions

- Code via ftp
  - ftp://cs.ucl.ac.uk/genetic/gp-code/gpu_gp_1.tar.gz
- Correlations

  http://bioinformatics.essex.ac.uk/users/wlangdon/