# Performance Localisation

Brendan Cody-Kenny TCD & UCD
Michael O'Neill UCD
Stephen Barrett TCD


Natural Computing
Research & Applications
Group

# Improving Performance

Applying mutation - operator design

- Exploitative
  - Alternative implementations - high likelihood code is “correct”
- Explorative
  - Generabperators - large amount of incorrect programs

Where should mutation be applied?

# Focusing Mutation

Profiler
- Hotspots

Fault Localisation
- positive and negative execution traces

Input Sensitive Profiling
- Vary input size

# Performance localisation

Modify the program

- Observe performance and functionality change

Research Question

- Do code changes at the location of performance improvements produce different run-time characteristics than changes in other locations in a program?

Natural Computing Research & Applications Group

# Mutation Operations

Deletion

Exhaustive Mutation

# Evaluation

Profiler

- Line Count

Deletion

- Execution Cost Saved

Exhaustive Mutation

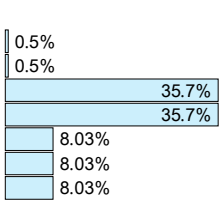
- count of performance reducing mutations / compilable mutants

Deletion + Exhaustive

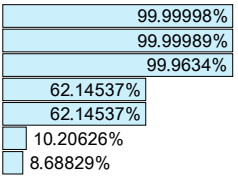
# Example

```
void sort(Integer[]a, int length){
 for (int h=0; h<2; h++) {
  for (int i=0; i<length; i++){
   for (int j=0; j<length-1; j++){
    if (a[j] >a[j +1]){
     int k=a[j ];
     a[j]=a[j +1];
     a[j +1]=k;
}}}}}
```

(a) Redundant Bubblesort




(b) Profiler




(c) Deletion

# Benchmark Set

| Problem Name | LOC | AST Nodes | Imp Nodes | Imp | Improvement Type |
|---|---|---|---|---|---|
| Insertion Sort | 13 | 60 | 5 | 9% | Loop unrolling |
| Bubblesort | 13 | 62 | 3 | 45% | Redundant Traversal (exclude sorted portion) |
| BubbleLoops | 14 | 72 | 5 | 71% | Redundant Traversal (exclude sorted portion) |
| Selection Sort 2 | 16 | 72 | 4 | 11% | Removed redundant increments during tests |
| Selection Sort | 18 | 73 | 4 | 2% | Removed redundant array access |
| Shell Sort | 23 | 85 | 3 | 5% | Various changes in increment size |
| Radix Sort | 23 | 100 | 6 | 3% | Reduced iteration, comparison with 0 |
| Quick Sort | 31 | 116 | 4 | 54% | Reduced iterations, remove tests |
| Cocktail Sort | 30 | 126 | 4 | 15% | Cloned and perforated loops (loop unrolling) |
| | | | | | Redundant Traversal (exclude sorted portion) |
| Merge Sort | 51 | 216 | 1 | 5% | Remove redundant array clone |
| Heap Sort | 62 | 246 | 6 | 41% | Remove redundant array access and assignment |
| Huffman Code-book | 115 | 411 | 3 | 43% | Redundant Traversal (exclude sorted portion) |

Natural Computing Research & Applications Group

# Exhaustive Mutation

| Node | Text | Value |
|---|---|---|
| 1 | for (int h... | 1.0 |
| 2 | int h=0 | - |
| 3 | h=0 | - |
| 4 | h | .6 |
| 5 | 0 | .7 |
| 6 | h < 2 | - |
| 7 | h | .16 |
| 8 | 2 | .85 |
| 9 | h++ | - |
| 10 | h | 0 |

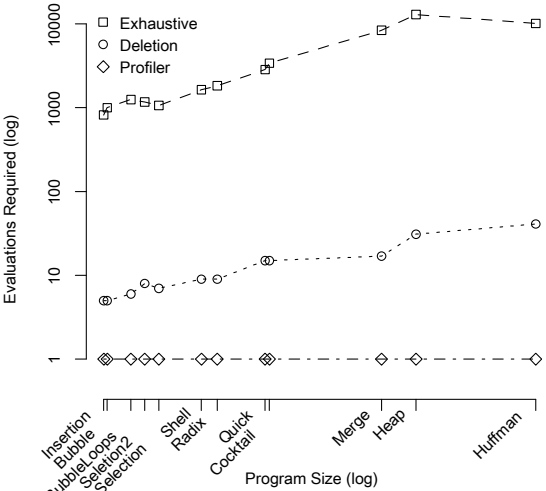
Natural Computing Research & Applications Group

# Results

| Accuracy | Profiler | Deletion | Exhaustive | Ex & De |
|---|---|---|---|---|
| 99-100% | 1 | 0 | 0 | 0 |
| 90-99% | 7 | 8 | 9 | 11 |
| 80-90% | 7 | 2 | 9 | 6 |
| 70-80% | 3 | 10 | 7 | 5 |
| 60-70% | 3 | 6 | 5 | 9 |
| 50-60% | 2 | 4 | 2 | 5 |
| 40-50% | 2 | 3 | 4 | 1 |
| 30-40% | 6 | 2 | 5 | 3 |
| 20-30% | 10 | 5 | 1 | 6 |
| 10-20% | 2 | 2 | 0 | 2 |
| 0-10% | 5 | 6 | 6 | 0 |

Natural Computing Research & Applications Group

# Computational Cost



# Open Future Work

Larger set of larger programs

Modelling

Exploit hierarchical nature of code

- top-down delete/mutate, bifurcate through the code

Combination of techniques

- localise performance based on effect of profiled hotspots
- taint analysis to trace back state involved in hotspots

Perform localisation *during* GP

- an efficient way of exploring program space

Natural Computing Research & Applications Group

# Code

https://github.com/codykenb/locoGP

# Thanks!

codykenny@gmail.com

This research is based upon works supported by Science Foundation Ireland under grant 13/IA/1850 and 13/RC/2094 which is co-funded under the European Regional Development Fund through the Southern & Eastern Regional Operational Programme to Lero - the Irish Software Research Centre (www.lero.ie).

Natural Computing Research & Applications Group