Sparse Adapton

Low-Overhead Incremental Computation

By Kyle Headley

kyle.headley@colorado.edu
Incremental Computation

A computation is incremental if repeating it with a changed input is faster than from-scratch recomputation.
Adapton – General Purpose Incremental Computation

• Memoization
• Demand driven computation
• Dynamic dependency tracking
• Demand driven incremental updates
Problem:

General purpose incremental computation relies heavily on memoized results, which requires a lot of memory
Problem:

General purpose incremental computation relies heavily on memoized results, which requires a lot of memory

Solution:

Memoize fewer results!
It works!

- Required memory is reduced
- Memory management is eased
- Update times are decreased

We gain control over the balance between speed and memory
But of course, there are complications...
We can no longer use the strategy ‘memorize every function call’, we have to make a choice.
Working with data structures
Working with data structures
D – Data
M – Memo Point
Insert Memo Points Evenly
D – Data
M – Memo Point

Insert Memo Points Evenly
Modify Data

D – Data
M – Memo Point
Incremental Insert

D  DMD  D  DMD  DMD  DMD  DMD  DMD  DMD  DMD  DMD  D
Incremental Insert

D – Data
M – Memo Point
Probabilistic distribution

Hash and Insert Memo Points
D – Data
M – Memo Point

Hash and Insert Memo Points
D – Data
M – Memo Point

Incremental Insert
Now we’re committed to referencing a previous memo point to create a new one
Divide and Conquer Example
Input list: M5 M6 2M4 M1 M3
Build Balanced Tree

```
  M
 /   \
M     M
|     |
5     4   1
|     |
M     M   3
|     |
2     1
|     |
6     3
```

MR
Build Balanced Tree

Poor Results:

Emit a memo point when you encounter a memo point.
Pass References to memo points as parameters:

First class names
Build Balanced Tree

Good Results:
Maintain association through merge steps
Success!

Onward

• Increasingly complex algorithms
• Automation through libraries
• Interactivity of common tools
Conclusion

• Increase incremental efficiency by managing overhead
• Memo point placement matters
• Associate memo points with data