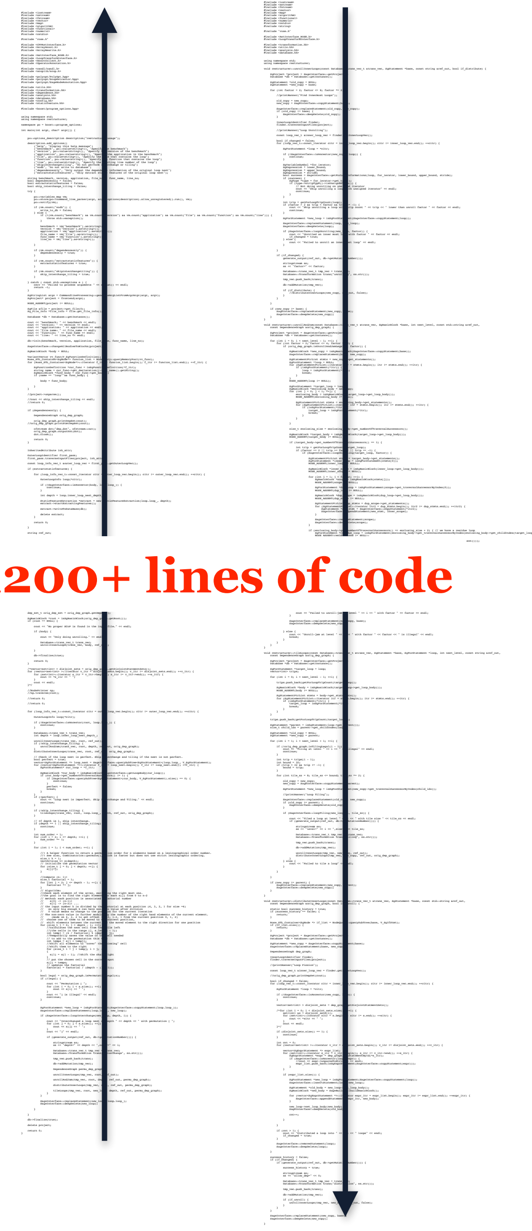
Optimization of Arbitrary Loop Nests

```
CodeReg scop {
  perfect = BuiltIn.IsPerfectLoopNest();
  depth = BuiltIn.LoopNestDepth();
  if (RoseLocus.IsDepAvailable()) {
    if (perfect && depth > 1) {
      permorder = permutation(seq(0,depth));
      RoseLocus.Interchange(order=permorder);
    }
    {
      if (perfect) {
        indexT1 = integer(1..depth);
        T1fac = poweroftwo(2..32);
        RoseLocus.Tiling(loop=indexT1, factor=T1fac);
      }
    } OR {
      if (depth > 1) {
        indexUAJ = integer(1..depth-1);
        UAJfac = poweroftwo(2..4);
        RoseLocus.UnrollAndJam(loop=indexUAJ,
                               factor=UAJfac);
      }
    } OR {
      None; # No tiling, interchange, or unroll and jam.
    }
    innerloops = BuiltIn.ListInnerLoops();
    *RoseLocus.Distribute(loop=innerloops);
  }
  innerloops = BuiltIn.ListInnerLoops();
```

37 lines of code



- Reproduced Gong Zhangxiaowen et al. results
- Much more concise and flexible

Conclusions

- Locus is able to represent *complex* optimization spaces for different code regions
- Easy to use fine-grain *optimizations* in fine-grain *regions of code* to improve performance
- *Share* resulting optimization programs to amortize the search time
- Keep the baseline version *cleaner* and *simpler* for the long term
- Future work:
  - Use multiple search modules concurrently to speed up the search process
  - Help users at designing optimization sequences