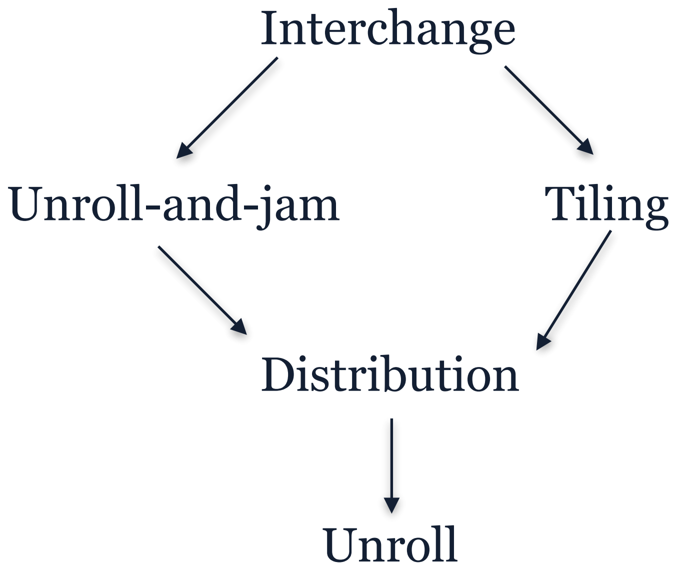
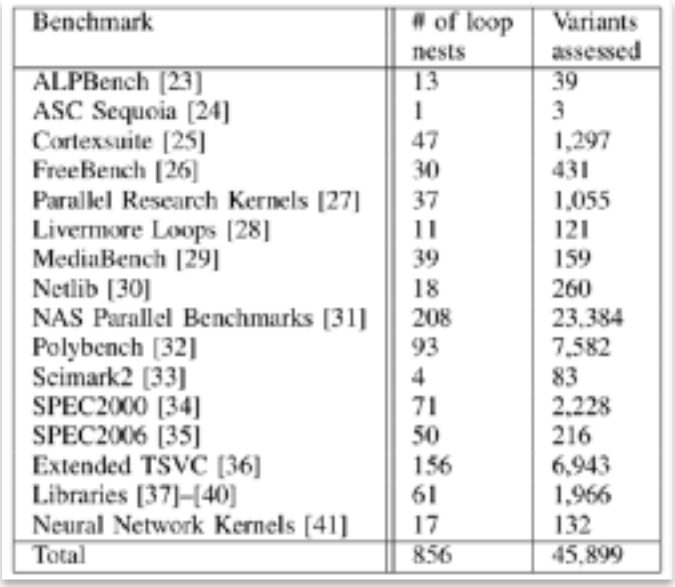
Optimization of Arbitrary Loop Nests

- Generic Locus program to optimize source codes unknown beforehand
- Goal: reproduce Gong Zhangxiaowen et al.[1] work using Locus
- Selected 856 loops from 16 benchmarks
- Transformed loops with all subsets of two sequences:

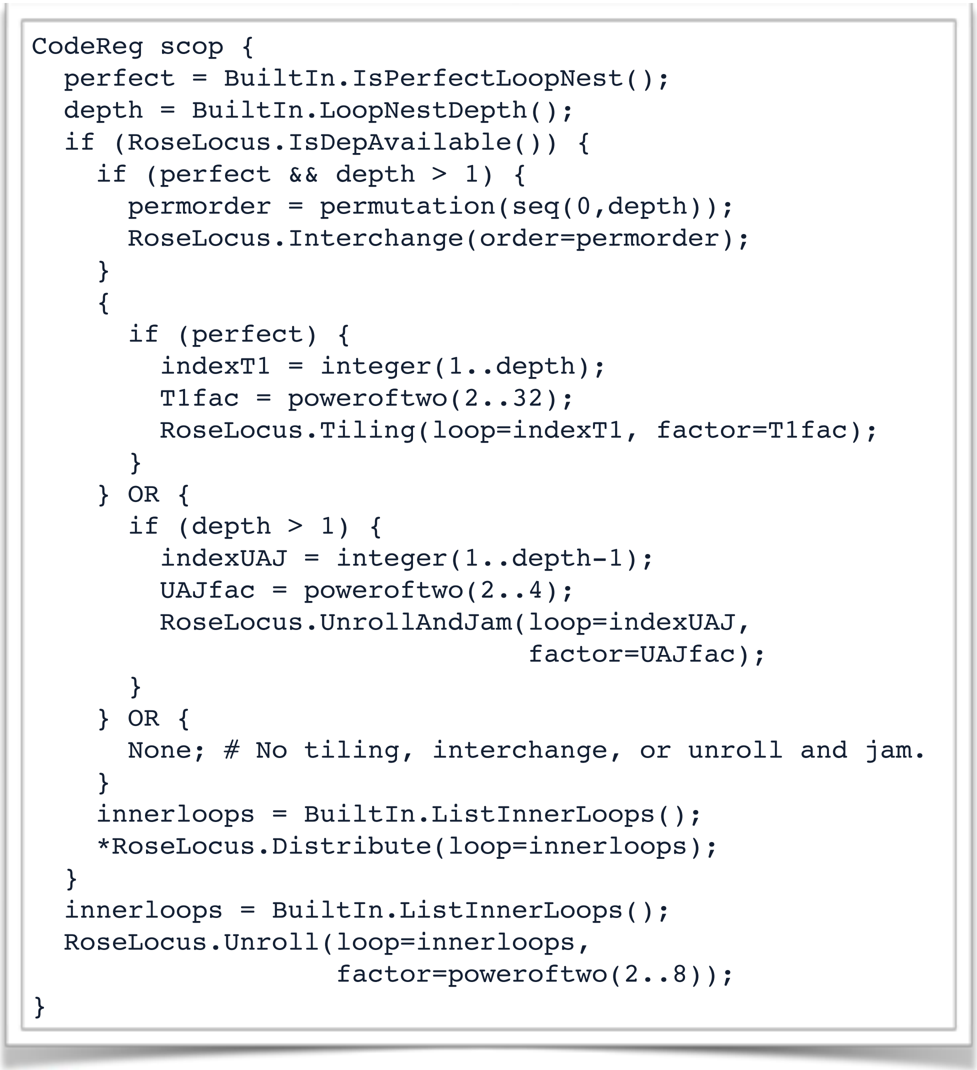




Benchmark	# of loop nests	Variants assessed
ALPBench [23]	13	39
ASC Sequoia [24]	1	3
Cortexsuite [25]	47	1,297
FreeBench [26]	30	431
Parallel Research Kernels [27]	37	1,055
Livermore Loops [28]	11	121
MediaBench [29]	39	159
Netlib [30]	18	260
NAS Parallel Benchmarks [31]	208	23,384
Polybench [32]	93	7,582
Scimark2 [33]	4	83
SPEC2000 [34]	71	2,228
SPEC2006 [35]	50	216
Extended TSVC [36]	156	6,943
Libraries [37]–[40]	61	1,966
Neural Network Kernels [41]	17	132
Total	856	45,899

[1] Gong Zhangxiaowen et al. “An empirical study of the effect of source-level loop transformations on compiler stability”.

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();
  RoseLocus.Unroll(loop=innerloops,
                   factor=poweroftwo(2..8));
}
```