Concurrent Minimum Spanning Tree Algorithm on JPF

- Paper: by R. Setia, A.Nedunchezhian, S. Balachandaran, in HiPC 2009

Xiwen Chen

DisCoVeri Group, York University, Toronto

Check the properties on the run

- Public static void main(string[] args){

```
Config conf = JPF.createConfig(args);

conf.setProperty("my.property", "whatever");

MyListener myListener = ...

JPF jpf = new JPF(conf);
jpf.addListener(myListener);

jpf.run();
}
```

JPF checking plans

- Threads: 1 – 3.
- Size of the graph: 3 – 5.
- Models: Int[][] v.s. TreeMap
- Search algorithm: DFS, BFSHeuristic, PreferThreads, MostBlocked, RandomHeuristic.
- Lisenter: PreciseDataRaceDector, SimpleDot.

1 threads on 5 nodes

	DFS	BFSH	DFSH	Prefer Thread	Most Block	RandomH
Mem	22MB	49MB	49MB	49MB	49MB	49MB

```
====================================================== statistics
elapsed time:       0:00:04
states:             new=3225, visited=2565, backtracked=5789, end=5
search:             maxDepth=658, constraints hit=0
choice generators:  thread=3225 (signal=0, lock=1, shared ref=2564), data=0
heap:               new=12114, released=7471, max live=829, gc-cycles=4505
instructions:       3838386
max memory:         22MB
loaded code:        classes=123, methods=1846

====================================================== search finished: 11-4-
```

On different size of graph

All using DFSearch	**3 Nodes**	**4 Nodes**	**5 Nodes**
Time	2 s	3 s	4 s
States	1535	2450	3225
Max Depth	320	503	658
Heap: new	5028	8272	12114
Heap: max live	795	811	829
Memory	15MB	20MB	22MB
Error	division by zero	No error	No error

Division by zero error

```
for (int i = 0; i < N; i++) {
        for (int j = i + 1; j < N; j++) {
                E++;
                w = rand.nextInt(N / 4) + 1;
        }
}
```

When N = 3,

w=rand.nextInt(**0**)+1;

StateSpaceDot Listener

- The StateSpaceDot and SimpleDot Listeners give 400 and 200 pages file for 1 thread with 3 nodes.

```
2 -> 5 [arrowhead=onormal, color=gray, style="dotted"]   // backtrack
↵
5 -> 4 [arrowhead=onormal, color=gray, style="dotted"]   // backtrack
↵
4 -> 6 [label="T1"arrowhead=normal, headlabel="get underperform"]
↵
6 -> 7 [label="T0"arrowhead=normal, headlabel="join"]
↵
7 -> 2 [label="T1"arrowhead=vee]
```

Int[][] v.s. TreeMap

All using DFSearch	3 Nodes TreeMap	3 Nodes Int[][]	5 Nodes TreeMap	5 Nodes Int[][]
Time	2 s	2 s	4 s	3 s
States	1535	1845	3225	3880
Max Depth	320	384	658	789
Heap: new	5028	1590	12114	5459
Heap: max live	795	489	829	500
Memory	15MB	15MB	22MB	21MB

State space explosion

- With 2 threads on graph size of 4.

```
====================================================== results
error #1: gov.nasa.jpf.jvm.NoOutOfMemoryErrorProperty

====================================================== statistics
elapsed time:       8:04:20
states:             new=38636247, visited=36291815, backtracked=74927731, end=3813
search:             maxDepth=982, constraints hit=1
choice generators:  thread=38635294 (signal=0, lock=24470, shared ref=36246670), data=0
heap:               new=6638921, released=21500349, max live=845, gc-cycles=69201610
instructions:       -411755468
max memory:         1346MB
loaded code:        classes=124, methods=1857

====================================================== search finished
```

On my laptop

State space explosion

- With 2 threads on graph size of 4.

```
====================================================== results
error #1: gov.nasa.jpf.jvm.NoOutOfMemoryErrorProperty

====================================================== statistics
elapsed time:       6:55:50
states:             new=65681631, visited=73089266, backtracked=138769663, end=4177
search:             maxDepth=1528, constraints hit=1
choice generators:  thread=65680646 (signal=0, lock=56108, shared ref=57032033), data=0
heap:               new=11600968, released=33533186, max live=841, gc-cycles=130317080
instructions:       1862266585
max memory:         2184MB
loaded code:        classes=124, methods=1856

====================================================== search finished
```

Lab machine with 4GB memory

Reduce the state space

- System.out.println();
- Debugging variables.
- @Filterfiled.
- Make methods as native calls
- Add verify.beginAtomic()
  verify.endAtomic().

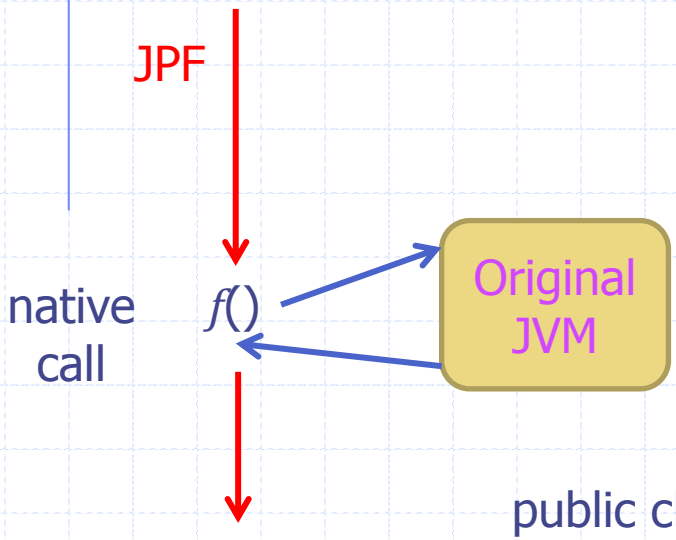
Seems only affect the # of bytecodes

Make variables not relevant for state matching

Control # of interleaving

Native calls

- JPF doesn't check the native calls.




```
public class A{
    ...
    public void f(){
        System.out.println("In JVM");
    }
}
```

put a "copy" of f() in:


```
public class JPF_package_A{
    public static void f(MJIEnv env, int rcls){
        System.out.println("In JVM");
    }
}
```

Results (1 thread)

All using DFSearch	3 Nodes original	3 Nodes reduced	5 Nodes original	5 Nodes reduced
Time	2 s	1 s	4 s	2 s
States	1535	1035	3225	2030
Max Depth	320	214	658	413
Heap: new	5028	3262	12114	6932
Heap: max live	795	798	829	832
Memory	15MB	15MB	22MB	17MB

Result (2 threads)

- Still run more than 10 hours and out of memory…

```
====================================================== results
error #1: gov.nasa.jpf.jvm.NoOutOfMemoryErrorProperty

====================================================== statistics
elapsed time:       9:56:57
states:             new=38636247, visited=53467377, backtracked=92103147, end=1803
search:             maxDepth=813, constraints hit=1
choice generators:  thread=38635883 (signal=0, lock=59258, shared ref=32751065), data=0
heap:               new=5430949, released=13942116, max live=826, gc-cycles=86662523
instructions:       -940434199
max memory:         1351MB
loaded code:        classes=125, methods=1916

====================================================== search finished
```

Conclusion

- State space explosion makes program hard to check using JPF.
- There are many ways to reduce the # of states.
- The concurrent program seems fine at least for those visited 60 millions states.