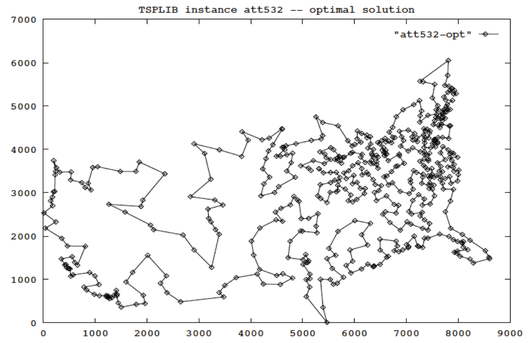
High Performance GPU Accelerated TSP Solver

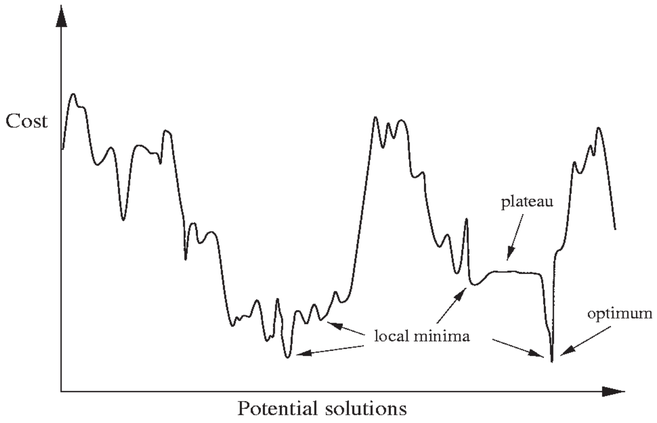
Kamil Rocki and Reiji Suda, Department of Computer Science, Graduate School of Information Science and Technology, The University of Tokyo

GPU TECHNOLOGY CONFERENCE
MARCH 18-21, 2013 | SAN JOSE, CALIFORNIA

TSP instance: shortest round trip through 532 US cities



- Computer wiring
- Vehicle routing
- Crystallography
- Robot control



LOGO TSP Solver - Open available
LOGO - **LO**cal **G**
CUDA and OpenCL

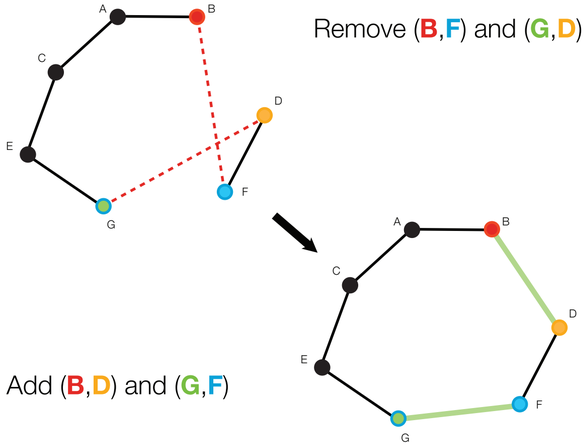
http://olab.is.s.u-tokyo.ac.jp/~kamil.rocki/projects.html

Global Optimization

```
1: procedure ITERATED LOCAL SEARCH
2:     s0 := GenerateInitialSolution()
3:     s* := 2optLocalSearch (s0)
4:     while (termination condition not met)
5:         s' := Perturbation (s*)
6:         s*' := 2optLocalSearch (s')
7:         s* := AcceptanceCriterion (s*, s*')
8:     end while
9: end procedure
```

GPU acceleration

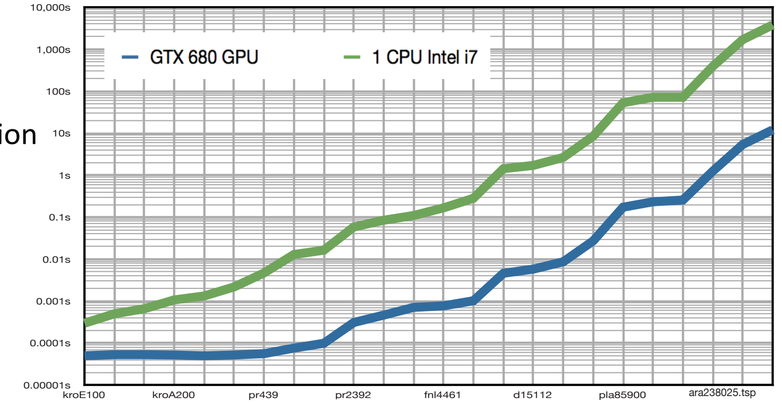
Local Optimization



Each thread has to check:

if (distance(i, i-1) + distance(j+1, j) > distance(i, j+1) + distance(i-1, j)) update best i and j;

2-opt step processing time



For 200,000 cities approximately
25,000,000,000 swaps/second checked on
GTX 680 ~ 700 GFLOPS

The 2-opt algorithm basically removes two edges from the tour, and reconnects the two new sub-tours created. This is often referred to as a 2-opt move. There is only one way to reconnect the two sub-tours so that the tour remains valid. The steps is repeated only as long as the new tour is shorter.

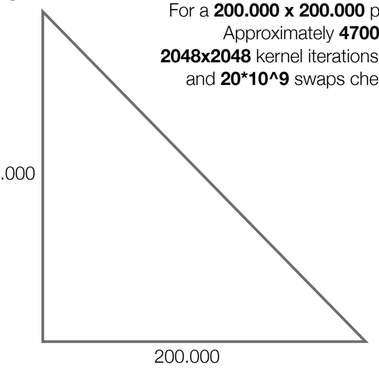
GPU uses this simple function for EUC_2D problems

```
__device__ int calculateDistance2D
(unsigned int i, unsigned int j, city_coords* coords) {

        register float dx, dy;

        dx = coords[i].x - coords[j].x;
        dy = coords[i].y - coords[j].y;

        return (int)(sqrtf(dx * dx + dy * dy) + 0.5f);
}
```



For a **200.000 x 200.000** problem
Approximately **4700**
2048x2048 kernel iterations needed
and **20*10^9** swaps checked