Local Beam Search by Using Parallel Programming

By Isada Sukprapa, Tanadol Mahattanawutakorn, Sudshewin Suebvong

March, 2017

CSS333 Parallel and Distributed Computing
Sirindhorn International Institute of Technology (SIIT)
Thammasat University

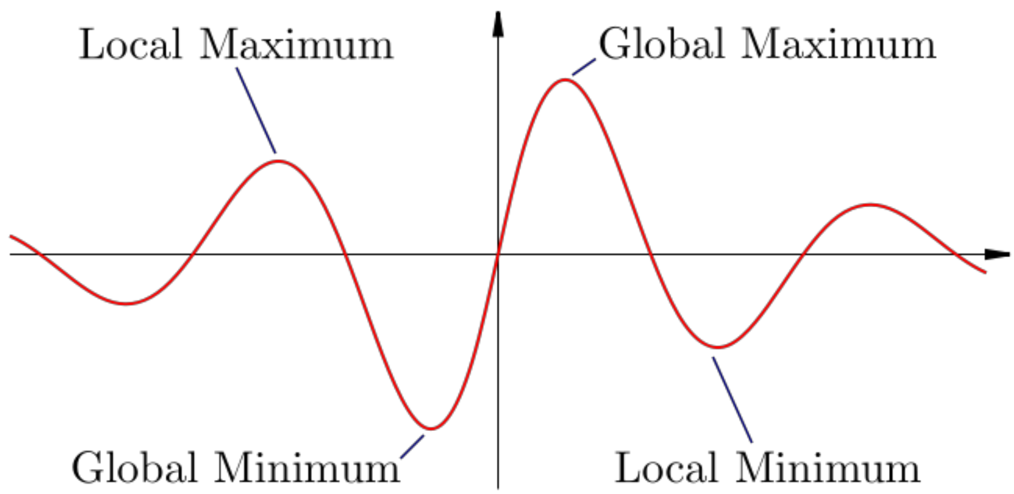
Maxima and Minima

Form mathematical analysis, the Maxima and Minima (the respective plurals of maximum and minimum) of a function, known collectively as extrema (the plural of extremum), are the largest and smallest value of the function, either within a given range (the local or relative extrema) or on the entire domain of a function (the global or absolute extrema). Pierre de Fermat was one of the first mathematicians to propose a general technique, adequality, for finding the maxima and minima of functions.

As defined in set theory, the maximum and minimum of a set are the greatest and least elements in the set, respectively. Unbounded infinite sets, such as the set of real numbers, have no minimum or maximum.

Local Maximum

Define that there are two sets. The one is Universal, the other over a range. From above, Maxima and Minima is extreme values of a function i.e. maximum value and minimum value.



So, when finding the maxima in a 'Ranged' Set → Local Maxima/Minima. And when you find it in Universal Set → Global Maxima/Minima.

In mathematically, there is no Universal, it's rather called a Domain and the other called Range. So that, it just becomes a range of values selected in the Domain. Moreover, for any two real numbers **$a < b$, let $f:[a,b] \to R$** be a real valued function of a real variable. The function f is said to have an absolute maximum (also called the global maximum) at **$a \le z \le b$**, if and only if **$f(z) \ge f(x)$ for all $z,x \in [a,b]$**. Similarly, one can define the same for minima. Remember **z** is a value, **x** is algebraic term. Now when it happens such that **$z,x \in [a,b]$** becomes **$z,x \in [c,d]$** where **$a \le c \le d \le b$** then the above applied in this interval would be called local maxima/minima.

Local Search

In computer science, local search is a heuristic method for solving computationally hard optimization problems. Local search can be used on problems that can be formulated as finding a solution maximizing a criterion among a number of candidate solutions. Local search algorithms move from solution to solution in the space of candidate solutions by applying local changes, until a solution deemed optimal is found or a time bound is elapsed.

Local Beam Search

Local beam search is the applied version of hill climbing search. Instead of keeping track of only 1 node, local beam search algorithm keeps track of k states while conducting the search. It starts from randomly generated k states the algorithm selects the best neighbors. It may select all of their neighbors if there are less than k states, and randomly select in the case of ties. Then, these k states are used in the next iteration. Local beam search, with k states is different from conducting k random, restarts hill climbing search.

Idea of local beam search

function **Beam-Search**(**problem**, **k**) **returns** a solution

state start with **k** randomly generated states

loop

generate all successors of all **k** states

if any of them is a solution **then** return it

else select the **k** best successors

Random pick a point from data
For example : (x,y) = (2,1) the value of that point is 54

	0	1	2	3	4	5
0	81	56	29	70	20	81
1	93	0	54	54	3	93
2	61	17	21	61	97	61
3	73	81	51	41	41	73
4	57	7	19	24	30	57
5	81	56	29	70	20	81

Then we check other points around us.

	0	1	2	3	4	5
0	81	56	29	70	20	81
1	93	0	54	57	3	93
2	61	17	21	61	97	61
3	73	81	51	41	41	73
4	57	7	19	24	30	57
5	81	56	29	70	20	81

If there are points those have value higher or equal current value we call them successors.

	0	1	2	3	4	5
0	81	56	29	70	20	81
1	93	0	54	57	3	93
2	61	17	21	61	97	61
3	73	81	51	41	41	73
4	57	7	19	24	30	57
5	81	56	29	70	20	81

Ex. Only(3,1) has value more than current value (57>54) the (3,1) is successor of (2,1)

We keep do this recursively until there is no more higher value.

- Initial
- First recursive
- Second recursive
- Third recursive

Finally, we get local maximum point(4,2) with value equal to 97 at third recursive

	0	1	2	3	4	5
0	81	56	29	70	20	81
1	93	0	54	57	3	93
2	61	17	21	61	97	61
3	73	81	51	41	41	73
4	57	7	19	24	30	57
5	81	56	29	70	20	81

Methods: Java Eclipse

Initialize

1. Matrix 10000 x 10000 with random data in range 0-100

```
//------------Generate Matrix 10000x10000 data-------------
int number = 10000;
int[][] data = new int[number][number];
for (int coloumn = 0; coloumn < number; coloumn++)
    for (int row = 0; row < number; row++) {
        data[coloumn][row] = (int) (Math.random() * 100);
    }
//------------Write example plot to Excel------------------
WriteExcel(data);
```

2. Random 10000 pairs of (x,y) coordinate as starting point

Sequential

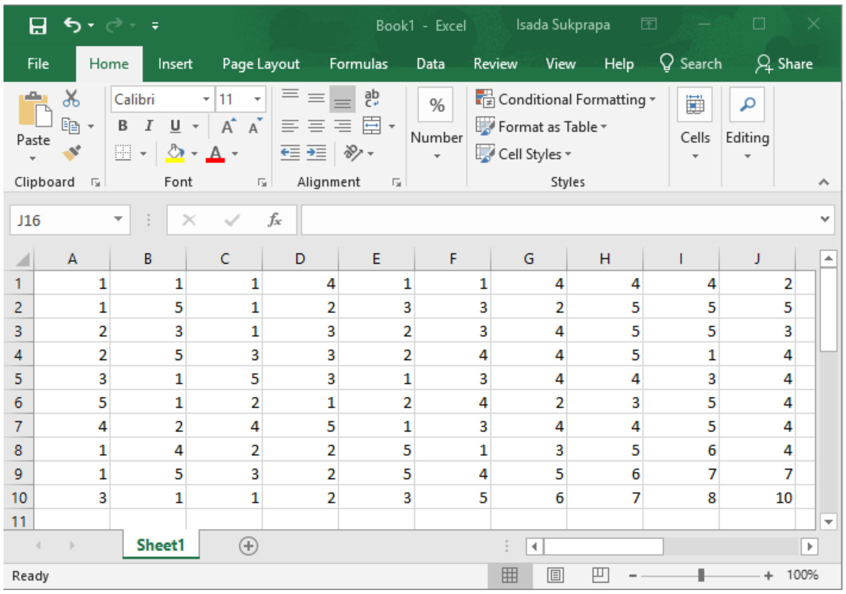
```
//------------Start SEQUENTIAL Calculation----------------
count=0;
long start = System.currentTimeMillis();
int localmax=0;
while (count != 10000) {

    System.out.println("(" + m + "," + n + ")");
    localmax = localSearch(data,randomx[count],randomy[count]);
    if(max<localmax){
        max=localmax;
        System.out.println("max: "+max);
    }


        System.out.println("-----");
    count++;
}
long end = System.currentTimeMillis() - start;
System.out.println("sequential: " + end + " ms");
```

<u>Parallel</u>

```java
//------------Start PARALLEL Calculation---------------
long startp = System.currentTimeMillis();
Thread t1 = new Thread(task1);
Thread t2 = new Thread(task2);


t1.start();
t2.start();

t1.join();
t2.join();

long endp = System.currentTimeMillis() - startp;
System.out.println("parallel: " + endp + " ms");
System.out.println("sequential: " + end + " ms");
```

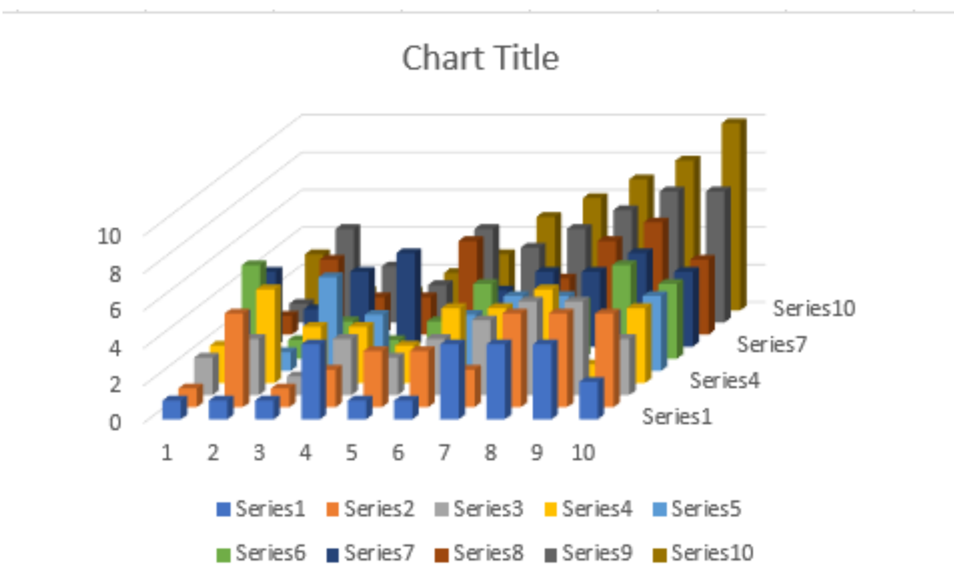
Goal : Find max of each (x,y) pair

Result :

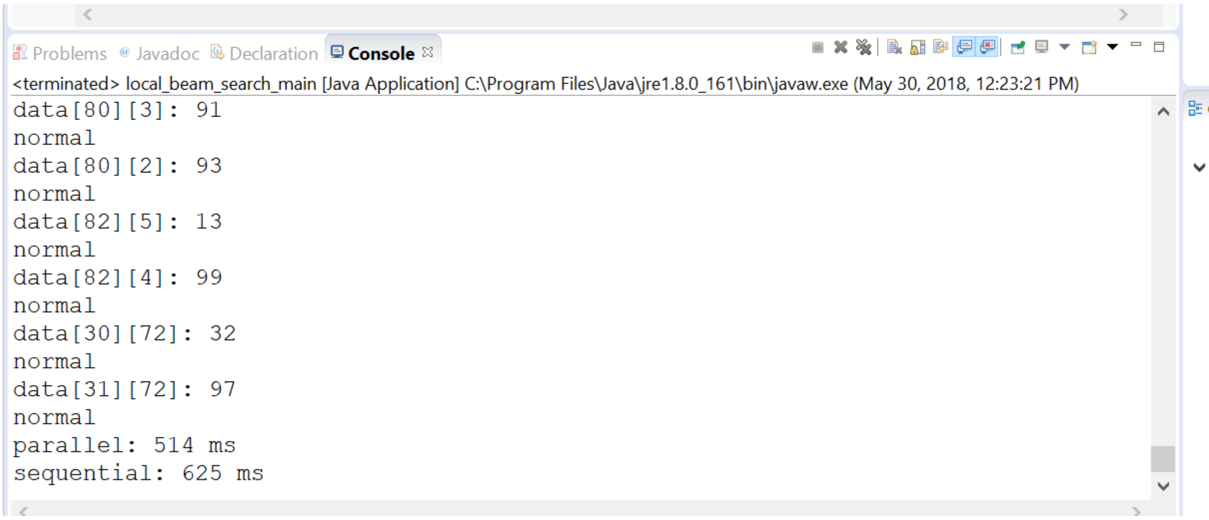


	A	B	C	D	E	F	G	H	I	J
1	1	1	1	4	1	1	4	4	4	2
2	1	5	1	2	3	3	2	5	5	5
3	2	3	1	3	2	3	4	5	5	3
4	2	5	3	3	2	4	4	5	1	4
5	3	1	5	3	1	3	4	4	3	4
6	5	1	2	1	2	4	2	3	5	4
7	4	2	4	5	1	3	4	4	5	4
8	1	4	2	2	5	1	3	5	6	4
9	1	5	3	2	5	4	5	6	7	7
10	3	1	1	2	3	5	6	7	8	10

This table shows the sampling data.



Plot graph from table above.



Result from java compiling.

Conclusion:

The parallel programming works with local beam search quiet well. The calculation time reduces around 17% of sequential programming.