Comparative Studies for Cell Tower Allocation Problem Using Matlab/HPC

Mina Mamdouh Fahim
Department of Mathematics
Computer Science (1727001)
Faculty of Science
Cairo University

Ahmed Abdallah Mohammed
Department of Mathematics
Computer Science(1727138)
Faculty of Science
Cairo University

Under supervision
Prof. Dr. Nasser Sweilam

Abstract— **A lot of people don't know of parallel computing, and here we are using it day to day without noticing.**

This paper discusses how to use parallel computing to solve an optimization problem called cell tower allocation, and also comparing with serial computing. Finally, comparing results between serial and parallel computations.

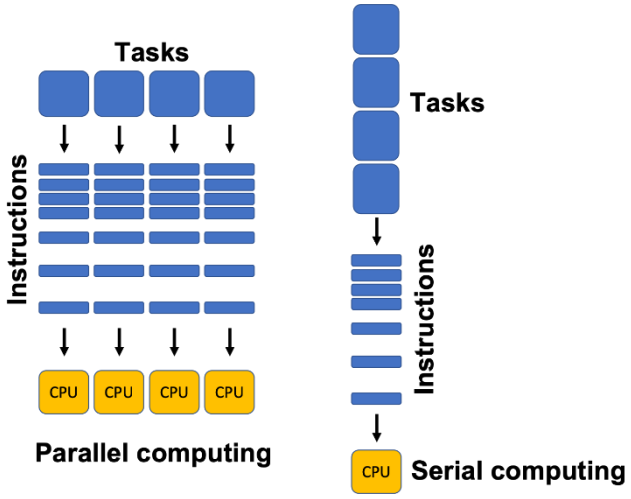
I. INTRODUCTION

By contrast, parallel processing is like cloning yourself 3 or 5 times, then all of you walking side by side, covering many steps along the road at once.
But let's take a closer look at what we are seeing of it in the our daily life. The answer is, mostly every device that becomes in handy now makes use of parallel computing concepts, which now moves us into a deeper look at the details of what is mentioned above.

Parallel computing is about using multi-computing systems or multicore processors to do processes(calculations) in a matter that saves a lot of time

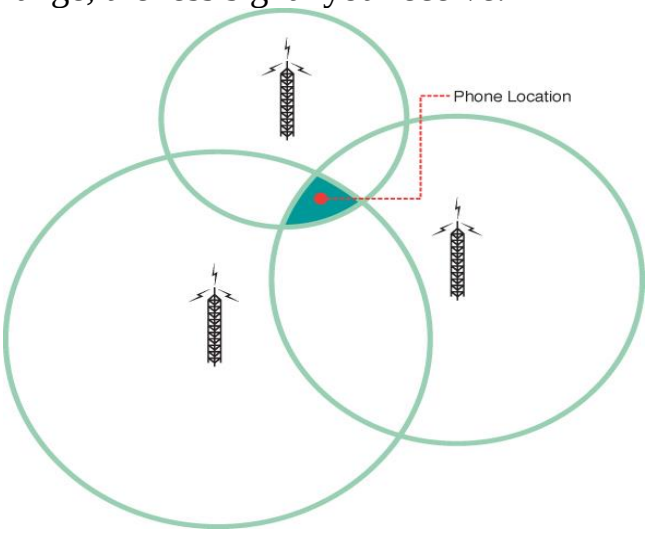
Why was this idea created?
As we saw in the past (back in the latter 90's-earlier 2000's) serial processors had an amazing way in wasting the time you need to get to read an email (almost 30 seconds; a life time); thinking of an email with an attachment, such a nightmare
this was a simple example of what we would suffer if the world continued using serial computing/ as a simple definition, single core computing.

Now, how can we use it?
Well, we are using it. for our closest things more than humans themselves; our smartphones; from after 2011 of course.
Why would we use it?
To get all the benefits of time efficiency, money saving and the ability to solve larger and more complex problems/
system of equations as every core works as a single computer by itself.



This figure illustrates the difference between using parallel computing and serial computing.

Since our study is about cell towers, let's take a brief of what they are. Cell towers are structures built on specific parcels of land that are designed to accommodate wireless tenants. Wireless tenants utilize cell towers to deploy various technologies to a subscriber base, such as telephony, mobile data, television and radio. Cell towers are typically built by tower companies or wireless carriers. Each tower in the world has its own coverage in a range of which can be described by circle with the tower in the center. So the far you get from the tower's range, the less signal you receive.



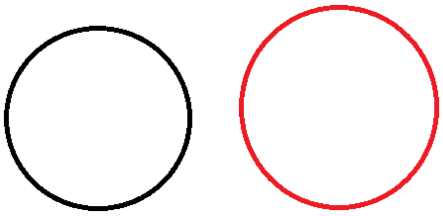
II. Mathematical Model

Minimizing the overlap

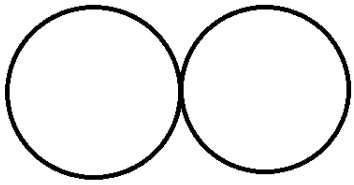
Min F=$\sum_{i=1}^{n}(\sum_{j=i+1}^{n} a_{ij})$ ∋ n is the number of towers, a is the overlap between tower i and tower j.

Cases of overlap; ranges intersected in:

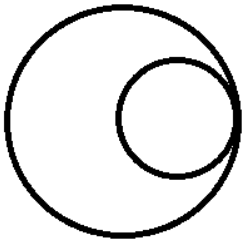
- 0 points (disjoint)➔overlap is 0.



- 1 point (from outside)➔overlap is 0



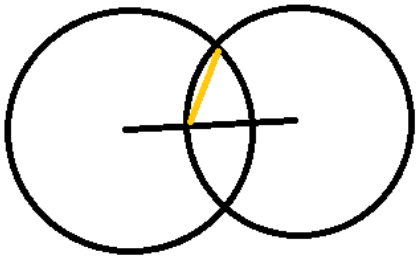
, (from inside)➔overlap is area(inside circle).



area of circle = $\pi \times r^2$

- 2 points, overlap is the area of intersection.

area of intersection = $R^2 \times (\frac{2\pi}{3} - \frac{\sqrt{3}}{2})$; *R is the golden line*.



III. Problem Description and Solving

Our aim is to determine the locations of cell towers, each having various coverage radius, so that there is maximum coverage, or minimum overlap. It is a constrained optimization problem because the cell towers are constrained to lie within a boundary.

Each tower's coverage range is represented by a circle with the location of the tower as its center.

Our objective is Minimizing the overlap:

Minimize the objective function
F= objfcn (X,R)
Where R is the matrix of random radii and X is cell tower coordinates.

Algorithm used:
fmincon-Active set algorithm.

And on the technical explanation side; fmincon is a matlab defined function that finds a minimum of constrained nonlinear multivariable function using a specific algorithm; it takes the objective function as an argument and the set of constraints along with the options it operates with. In our experiment we use the active-set algorithm; this algorithm solves quadratic programming problem, by transforming it into an easier sub-problem.
It's a two-phase iterative method that provide an estimate of the active set at the solution. In the first phase (the feasibility phase or phase 1), the objective is ignored while a feasible point is found for the constraints $Ax = b$ and $Dx \geq f$. In the second phase (the optimality phase or phase 2), the objective is minimized while feasibility is maintained. For efficiency, it is beneficial if the computations of both phases are performed by the same underlying method. The two-phase nature of the algorithm is reflected by changing the function being minimized from a function that reflects the degree of infeasibility to the quadratic objective function.

Doing more iterations can be expressed as minimizing the function.
For parameters

```
towers = 40; number of towers
side = 15; dimension of piece of land (one side)
seed = 5; seed for random initial condition
```

And create a randomly generated celltower problem using:

```
[dimensions,lb,ub,x0] = helper.celltowersetup(towers,side,seed)
```

which generates:

1- random radii matrix
2- bound constraints (xl,yl),(xu,yu) which in our case are equal to (0,0), (15,15)
3- lower and upper bounds to guarantee that no tower range would exceed the bounds by adding the radius to the lower bounds and subtract it from the upper bounds.

This will make our problem structed.

And now, running the code several times with different numbers of workers to compare the time using:

1- to run without using parallel computing

```
tic
[x,fval,exitflag,output] = helper.myOptim(x0,lb,ub,dimensions);
toc
```

`myOptim:` is a function which defines some options for the fmincon function to work with, using the objective function.
To use parallel by fmincon active-set algorithm put this lines of code.

```
options = optimset(options,'UseParallel','always');
options = optimset(options, 'Algorithm', 'active-set');
[x,fval,exitflag,output,lambda,grad,hessian] = ...
```

```
fmincon(@(x)
helper.objFcn(x,dimensions.R),x0,[]
,[],[],[],lb,ub,[],options);
```

2- to run with matlabpool using 1 worker

```
matlabpool open 1
tic
[x,fval,exitflag,output] =
helper.myOptim(x0,lb,ub,dimens
ions);
toc
matlabpool close
```

**Note: in the updated versions of matlab, `matlabpool open #ofCores` is replaced by `parpool(#ofCores)`. And `matlabpool close` by `delete(gcp(nocreate));`.

3- repeat number 2 using different number of workers from 2 to 4 (the number after keyword open).

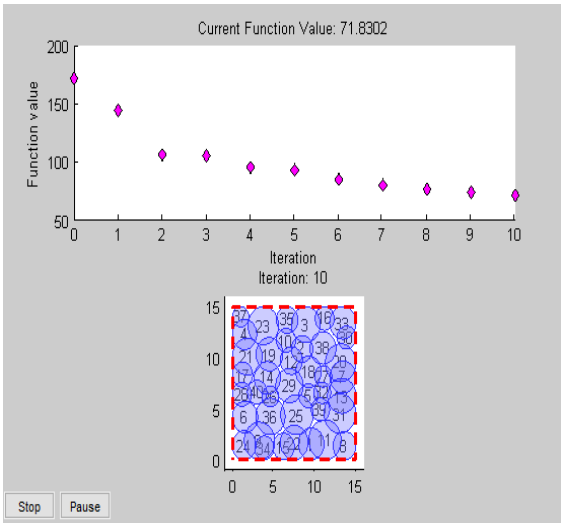
And we record the Elapsed time in each run.

IV. PC Description and Specification

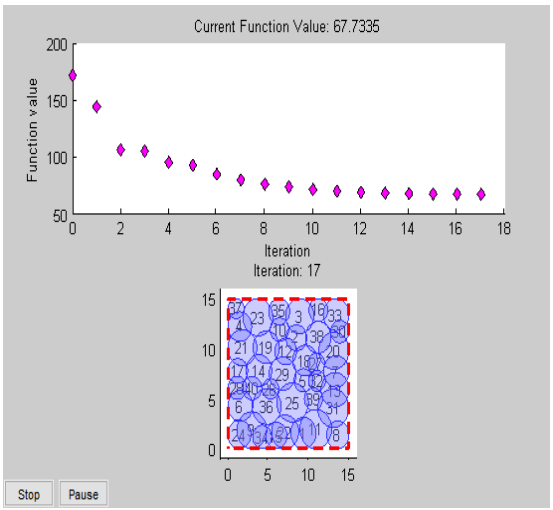
The computer is a laptop from the family of APU powered laptops. An APU is a chip on which has a CPU chip and by its side another chip for graphical use (GPU). It is powered by a quad core AMD A8-6410 @2.00GHz and 8Gb of DDR3 physical RAM.

The operating system is Micorsoft windows 10 Pro for PC's(64-bit), and we are using Matlab version R2013a because it has a parallel toolbox to make use of CPU cores in our hardware; which in turn would help us in our example of the content of this paper. The language used is MATLAB for scripting code and GUI generation and plotting.

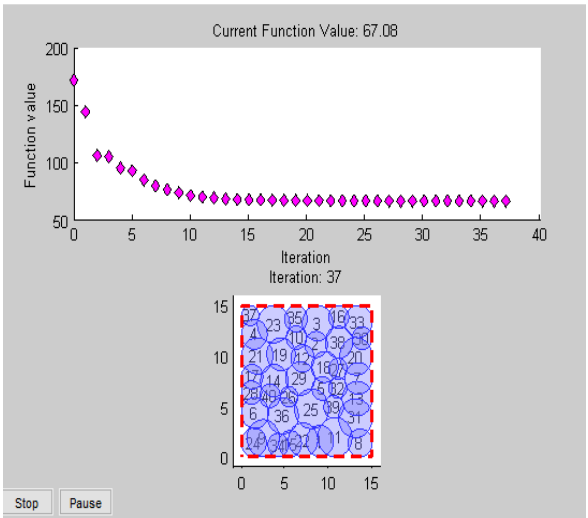
V. Experimental Results



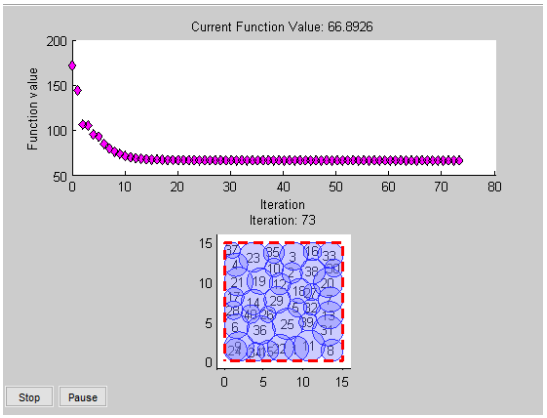
After 10 iterations



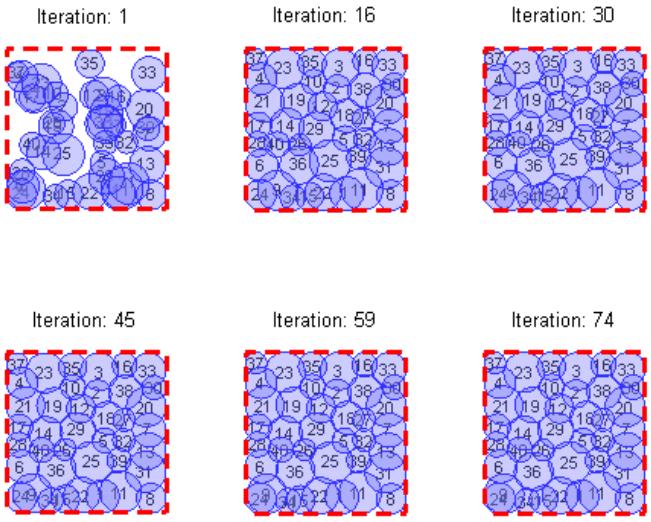
After 17 iterations



After 37 iterations



After 73 iterations, problem was solved



Matlabpool	No of workers used	Elapsed time in seconds	Speed (up/down) With matlabpool clos
Closed	-----	260.307502	-----
Opened	1	264.794998	0.983052
Opened	2	166.522689	1.563195
Opened	3	148.009542	1.758721
Opened	4	144.159657	1.805688

And by doing another experiment

Matlabpool	No of workers used	Elapsed time in seconds	Speed (up/down With matlabpool clo
Closed	-----	291.183481	-----
Opened	1	295.942884	0.983917
Opened	2	193.761181	1.502795
Opened	3	152.724281	1.906595
Opened	4	139.136393	2.092791

After reading these tables, we can notice that:

1- In each time the problem is solved with the same number of iterations so using parallel computing or using more cores doesn't mean that it will be solved in smaller number of iterations.
2- We can see that the values of the function are not close to each other in the first iterations, but later (in the last iteration) the values are very close to each other.
3- We obtain same result in each time and this result is the minimum value for our function which is 66.8926.
4- Elapsed time in solving the problem using 1 worker is a little greater than solving it without opening a parallel pool (approximate equal to 1), so by using 1 worker we don't achieve the meaning of parallel computing. **Or Simply**, when the parallel pool is closed, the CPU cores work together as one powerful core to achieve wanted calculations.
5- Each time we increase the number of workers elapsed time will decrease. Meaning that every worker added saves time and conserves effort of the machine we use.

VI. Conclusion

Solving minimizing problems in matlab is easy using fmincon function algorithms. Parallel computing is a good tool for solving problems, and it is better to use biggest number of cores possible/available.

VII. REFERNCES

[1]T. Gerencer. Parallel computing and its modern uses, HP Tech Takes, October 30, 2019.

[2] MathWorks; Parallel Computing Toolbox

[3]Parallel computing basics; https://pythonnumericalmethods.berkeley.edu/notebooks/chapter13.01-Parallel-Computing-Basics.html

[4]Mathworks: Demo script for cell tower optimization problem; click

[4]Mathworks; Parallel computing, parpool; https://www.mathworks.com/help/parallel-computing/parpool.html

[5] K. Schmidt., cell phone tower types and information, July 14, 2013; Types of Cell Towers

[6]Mathworks; mpcActiveSetSolver

[7]Mathworks; fmincon

[8] R. A. Bartlett and L. T. Biegler. QPSchur: a dual, active-set, Schur-complement method
for large-scale and structured convex quadratic programming. Optim. Eng., 7(1):5{32, 2006.

[9]Area of intersection between two circles