# Optimization of vehicle routing problem using artificial bee colony algorithm

**Advisor**
Zack Butler

**Author**
Nikhil Keswaney

# Problem Statement

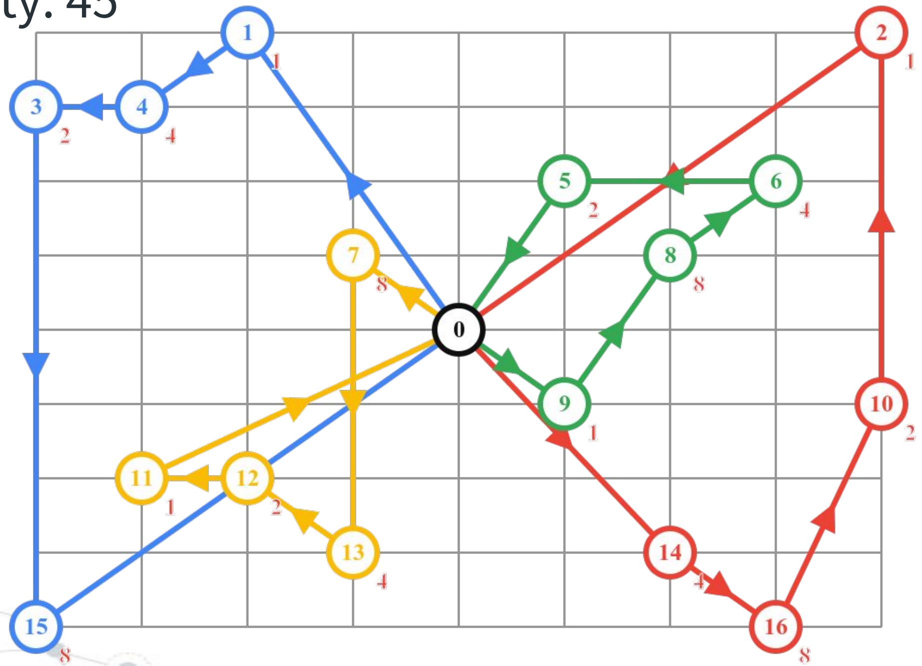
Capacitated vehicle routing problem(CVRP) is a combinatorial optimization problem which states as follows:

*“ Find the optimal delivery routes for a set of vehicles to supply the set of customers with given demands minimizing the total cost of all the routes. ”*

# Example

Number of vehicles: 4

Vehicle Capacity: 45



# Computation complexity

| Problem Size (Number of Nodes) | Approximate Solution Time |
|---|---|
| 10 | 3 milli-seconds |
| 20 | 77 years |
| 25 | 490 million years |
| 30 | $8.4*10^{15}$ years |
| 50 | $9.6*10^{47}$ years |

# Milestone 1

1. Finding dataset
2. Implementation of an exact algorithm

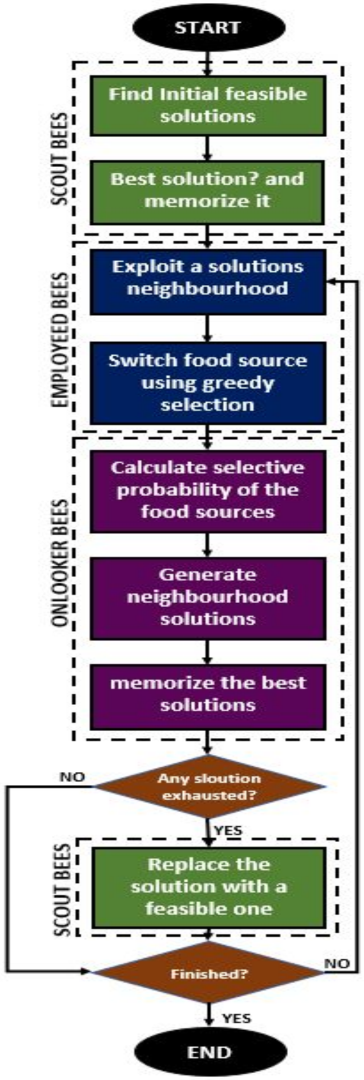
# Milestone 1 - Result

| Nodes | | Exact algorithm (Time Taken) | |
|---|---|---|---|
| **Number of Nodes** | **Optimal Answer** | **Brute Force** | **Optimized** |
| 10 | 100 | 3 ms | 5s |
| **20** | 216 | 77 Years | 4 Days |
| **25** | 569 | 490 Million Years | 122 Days |
| **30** | 534 | 8.4xE15 Years | 2 Years |
| **50** | 741 | 9.7xE47 Years | 136 Years |

# Milestone 2

1. Implement an approximate algorithm (Artificial bee colony algorithm)
2. Improve the performance of artificial bee colony algorithm

# Algorithm



START
SCOUT BEES
Find Initial feasible solutions
Best solution? and memorize it
EMPLOYEED BEES
Exploit a solutions neighbourhood
Switch food source using greedy selection
ONLOOKER BEES
Calculate selective probability of the food sources
Generate neighbourhood solutions
memorize the best solutions
Any sloution exhausted?
NO
YES
SCOUT BEES
Replace the solution with a feasible one
Finished?
NO
YES
END

# Time taken

| Nodes | | Exact algorithm (Time Taken) | | Approximate Algorithm (Swarm Size = 70) Iterations = 1500 | | |
|---|---|---|---|---|---|---|
| **Number of Nodes** | **Optimal Answer** | **Brute Force** | **Optimized** | **Approximate Answer** | **Avg. ans** | **Time Taken** |
| 10 | 100 | 3 ms | 5s | 111 | 123 | 1000ms |
| **20** | 216 | 77 Years | 4 Days | 224 | 231 | 1062 ms |
| **25** | 569 | 490 Million Years | 122 Days | 654 | 671 | 1048 ms |
| **30** | 534 | 8.4xE15 Years | 2 Years | 575 | 603 | 1125 ms |
| **50** | 741 | 9.7xE47 Years | 136 Years | 884 | 925 | 1551 ms |

# Milestones

1. Milestone 1 ✔
   a. Finding dataset
   b. Implementation of an exact algorithm
2. Milestone 2 ✔
   a. Implement an approximate algorithm(Artificial bee colony algorithm)
   b. Improve the performance of artificial bee colony algorithm
3. Milestone 3 ✔
   a. Parallelize the artificial bee colony algorithm using CPU.
   b. Compare and contrast the run-time of all the implementations

# Milestones

1. Milestone 1 ✔
    a. Finding dataset
    b. Implementation of an exact algorithm
2. Milestone 2 ✔
    a. Implement an approximate algorithm(Artificial bee colony algorithm)
    b. Improve the performance of artificial bee colony algorithm
3. Milestone 3 ✔
    a. Parallelize the artificial bee colony algorithm using ~~CPU~~ GPU and CUDA
    b. Compare and contrast the run-time of all the implementations

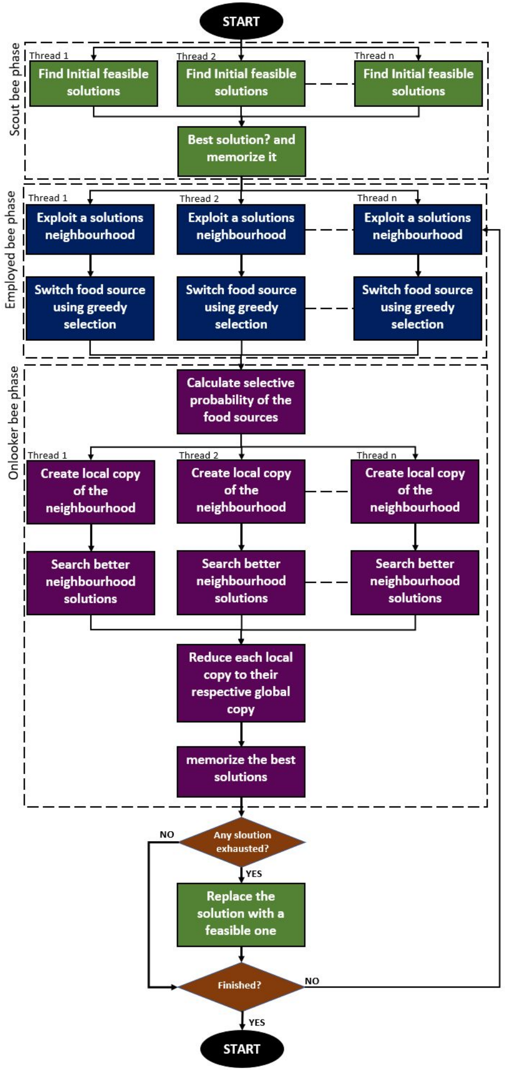
# Milestones

1. Milestone 1 ✔
    a. Finding dataset
    b. Implementation of an exact algorithm
2. Milestone 2 ✔
    a. Implement an approximate algorithm(Artificial bee colony algorithm)
    b. Improve the performance of artificial bee colony algorithm
3. Milestone 3 ✔
    a. Parallelize the artificial bee colony algorithm using ~~CPU~~ ~~GPU~~ CPU
    b. Compare and contrast the run-time of all the implementations

# Software & Hardware

- **Software:**
  - Java, PJ2(Parallel Java 2) library.
- **Hardware:**
  - tardis.cs.rit.edu cluster (12 Nodes)
  - Each Node has.
    - Two Intel Xeon E5-2603 v4 (x2) processors
    - Six CPU cores per processor
    - 12 threads
    - 1.7 GHz clock
    - 64 GB main memory

# Algorithm



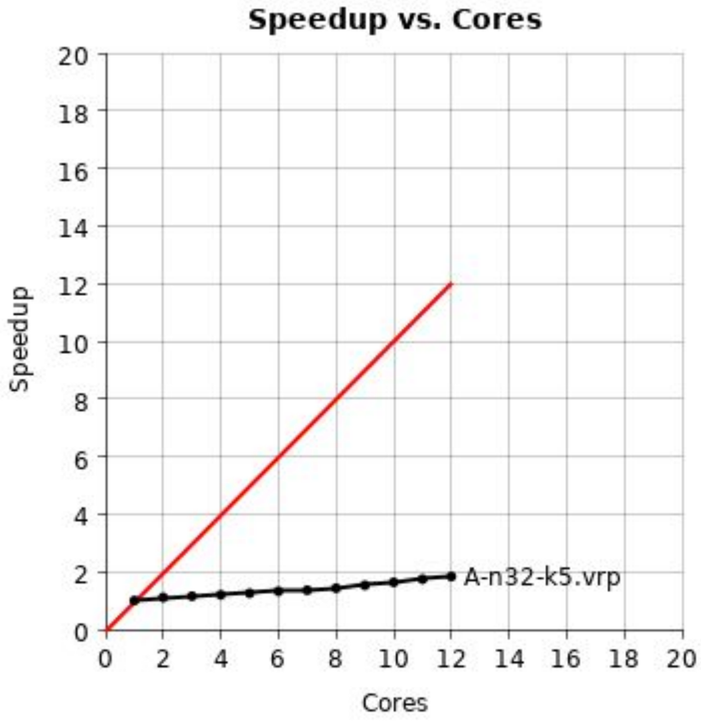
START
Scout bee phase
Thread 1
Thread 2
Thread n
Find Initial feasible solutions
Find Initial feasible solutions
Find Initial feasible solutions
Best solution? and memorize it
Employed bee phase
Thread 1
Thread 2
Thread n
Exploit a solutions neighbourhood
Exploit a solutions neighbourhood
Exploit a solutions neighbourhood
Switch food source using greedy selection
Switch food source using greedy selection
Switch food source using greedy selection
Onlooker bee phase
Calculate selective probability of the food sources
Thread 1
Thread 2
Thread n
Create local copy of the neighbourhood
Create local copy of the neighbourhood
Create local copy of the neighbourhood
Search better neighbourhood solutions
Search better neighbourhood solutions
Search better neighbourhood solutions
Reduce each local copy to their respective global copy
memorize the best solutions
Any sloution exhausted?
NO
YES
Replace the solution with a feasible one
Finished?
NO
YES
START

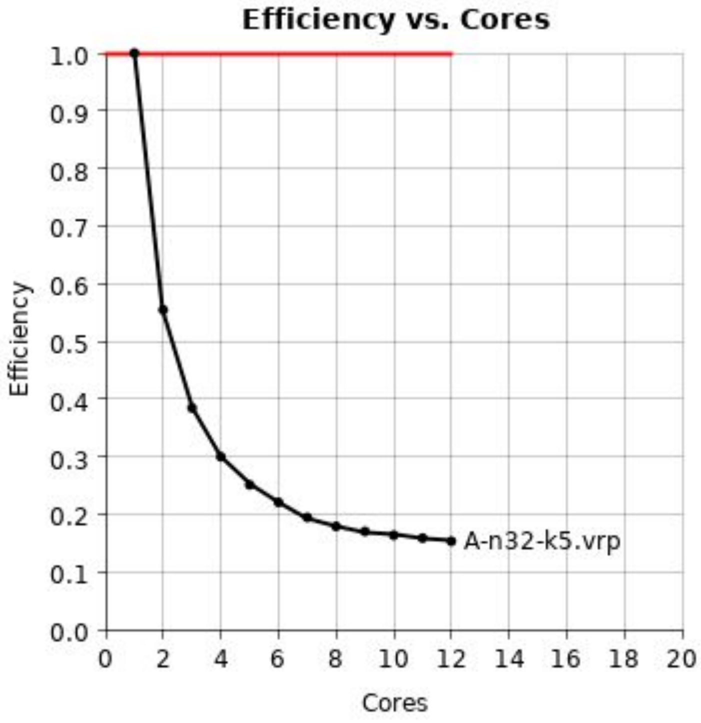
# Scaling

Scaling refers to running the program on increasing numbers of cores.

1. **Strong scaling** : As the number of cores increases, the problem size stays the same. This means that the program should ideally take 1/K the amount of time to compute the answer for the same problem

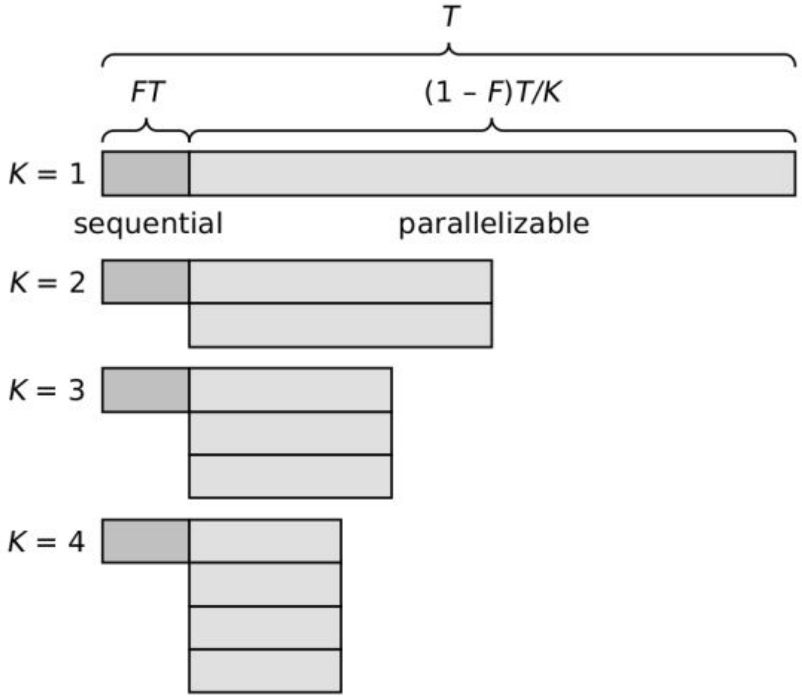
# Strong Scaling Result



# Problems

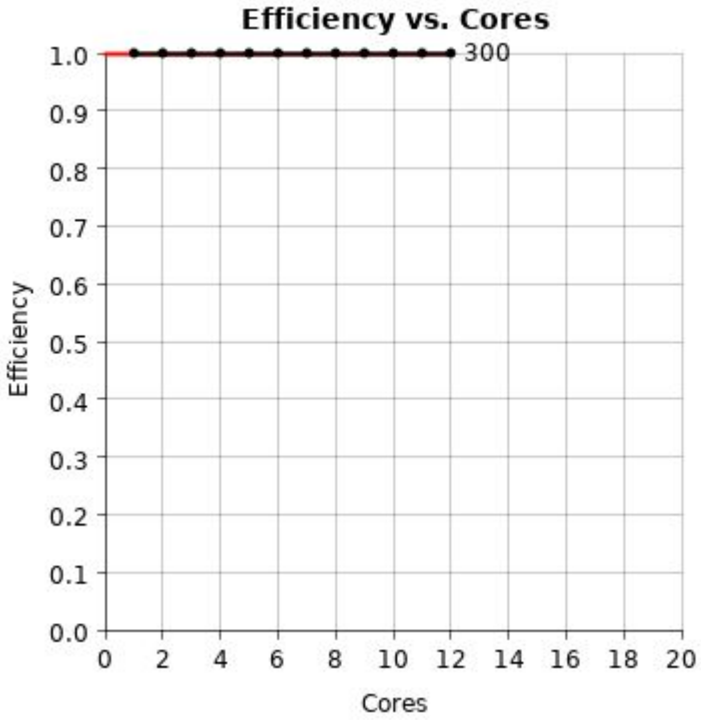
Sequential dependencies:

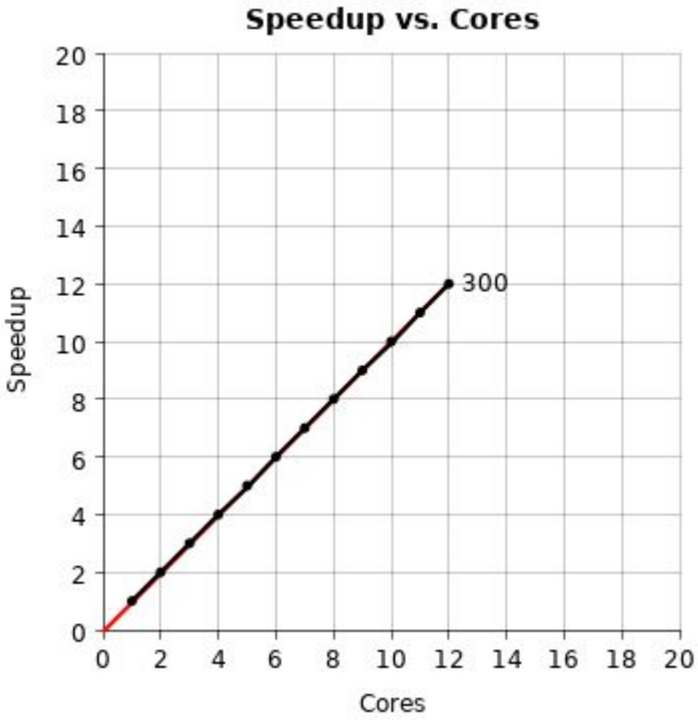


2. **Week scaling** : run on more cores, increase the size of the problem being solved as you increase the number of cores.

For week scaling I use the same file for the data and just increase the number of bees in the search space.

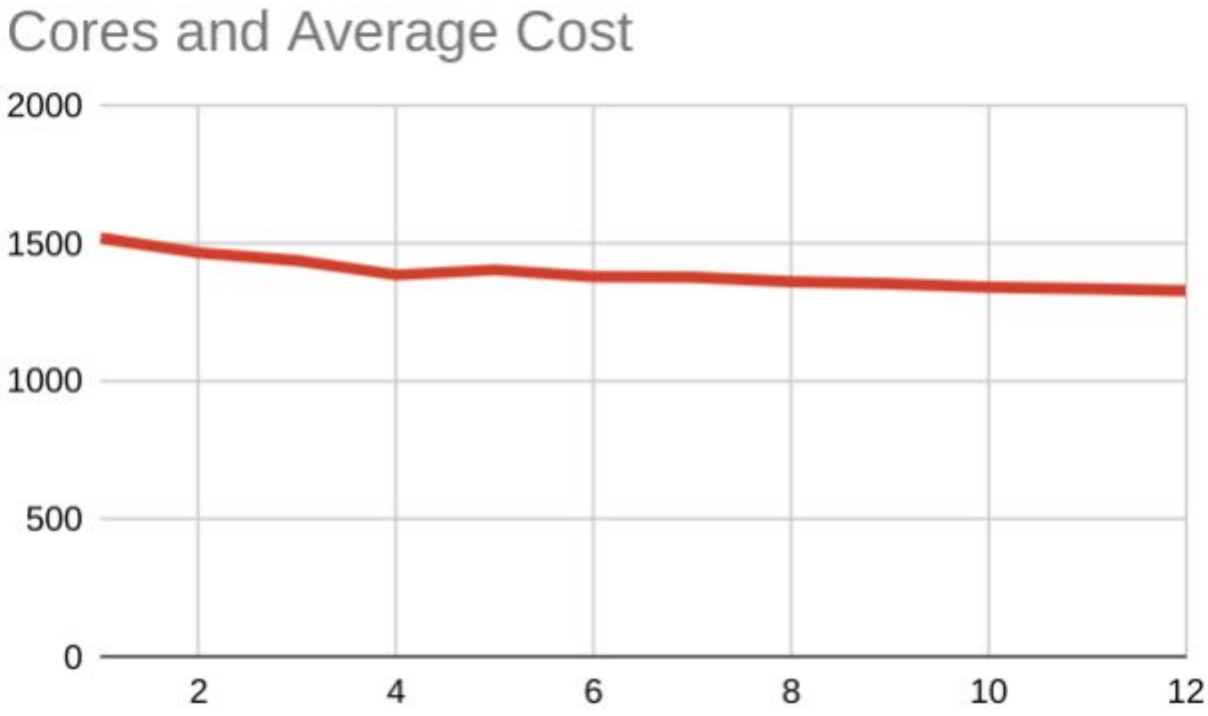
# Week Scaling Result



# This was about time taken!
# What about the results?

# Final Result



# Questions?

# Thank you.