Optimization of vehicle routing problem using artificial bee colony algorithm

Advisor
Zack Butler

Author
Nikhil Keswaney

Problem Statement

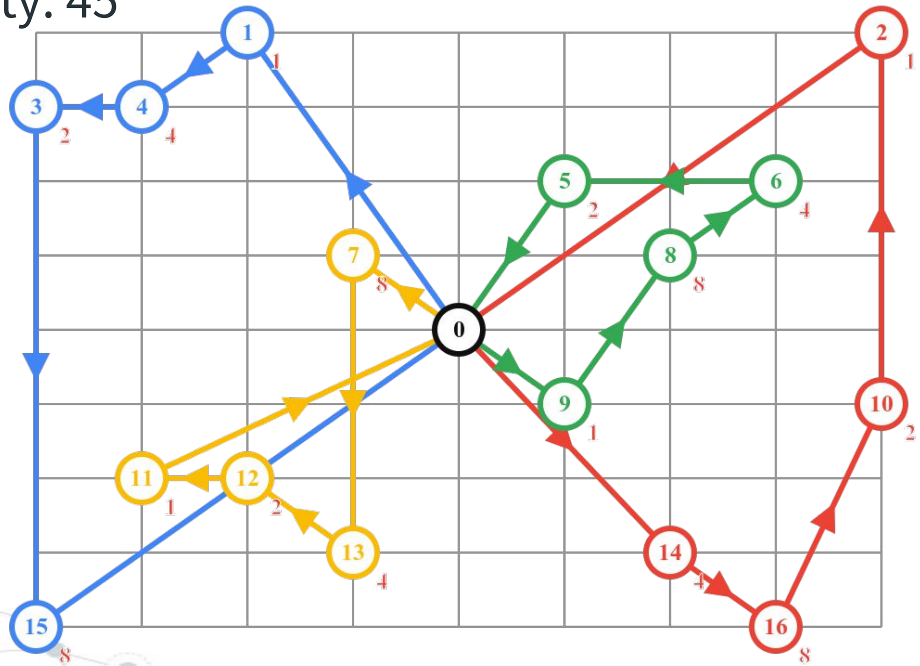
Capacitated vehicle routing problem(CVRP) is a combinatorial optimization problem which states as follows:

> “ *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

Dataset

- The dataset I selected is from branchandcut.org
- It has datasets for Traveling salesman CVRP etc.
- It has ~ 107 files for CVRP problem

```
NAME : A-n32-k5
COMMENT : (Augerat et al, No of trucks: 5, Optimal value: 784)
TYPE : CVRP
DIMENSION : 32
EDGE_WEIGHT_TYPE : EUC_2D
CAPACITY : 100
NODE_COORD_SECTION
 1 82 76
 2 96 44
 3 50 5
 4 49 8
DEMAND_SECTION
1 0
2 19
3 21
4 6
DEPOT_SECTION
 1
 -1
EOF
```

Algorithm

1. Given a set of customers find all the ways in which the customers can be served given a set of vehicles. (Subset finder)
2. Out of all the the ways found what is the best way (Traveling salesman)

Recurrence relation

```
S(N, K) = k * S(N - 1, K) + S(N - 1, K - 1)
```

Problem statement

There are 4 customers numbered from 1 to 4 and let depot be 0.

Distance Matrix

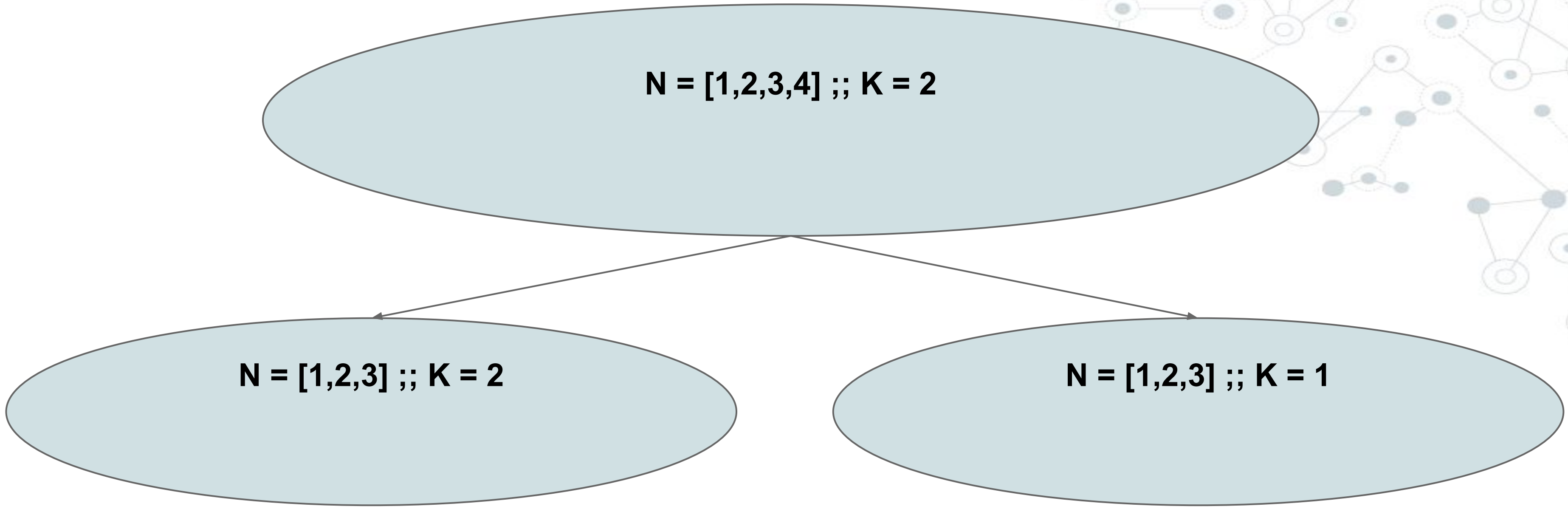
	0	1	2	3	4
0	0	2	31	4	12
1	3	0	12	4	5
2	5	9	0	10	11
3	3	2	12	0	4
4	13	7	10	8	0

Demand

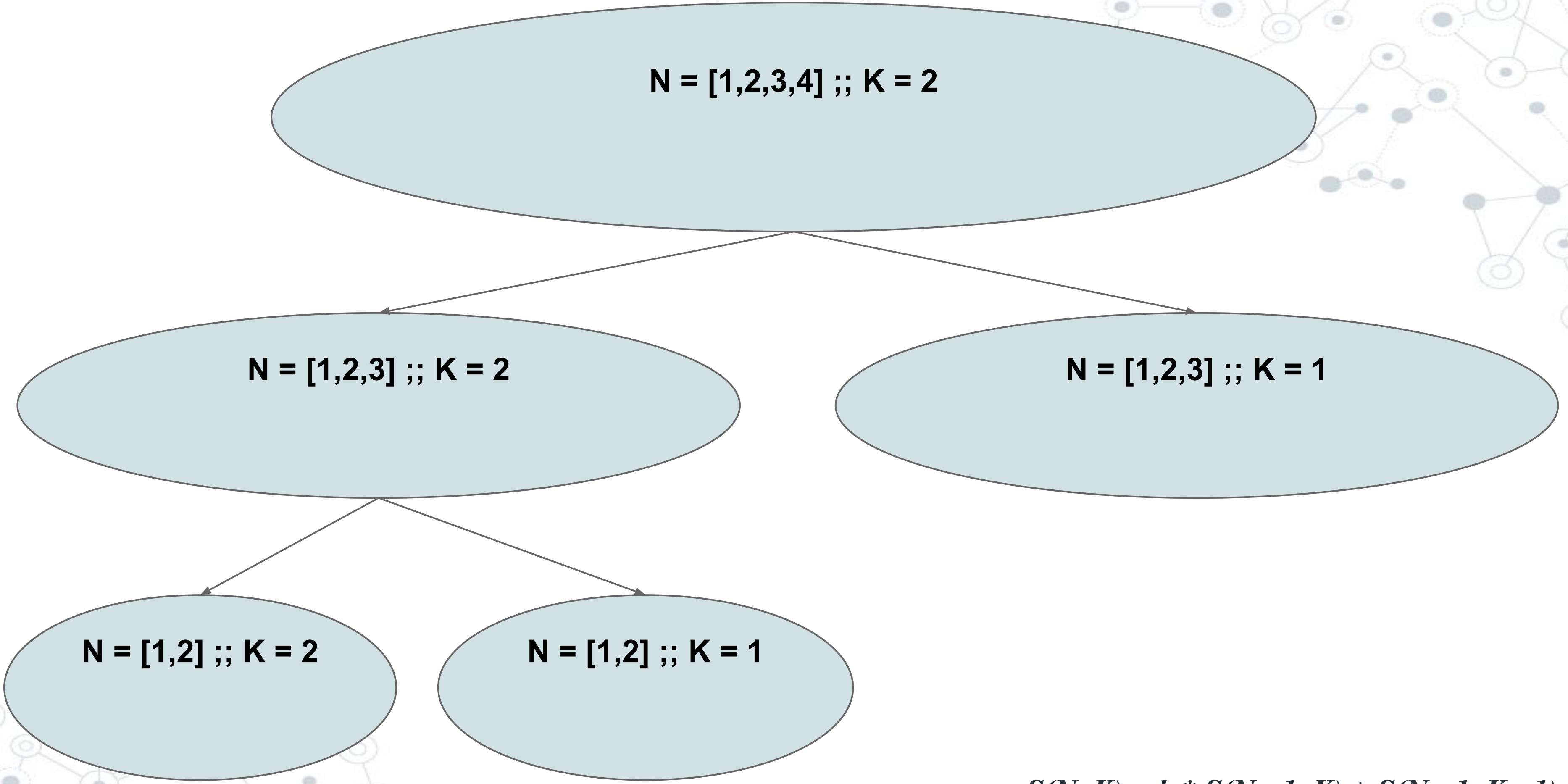
0	-1
1	12
2	8
3	20
4	9

N = [1,2,3,4] ;; K = 2

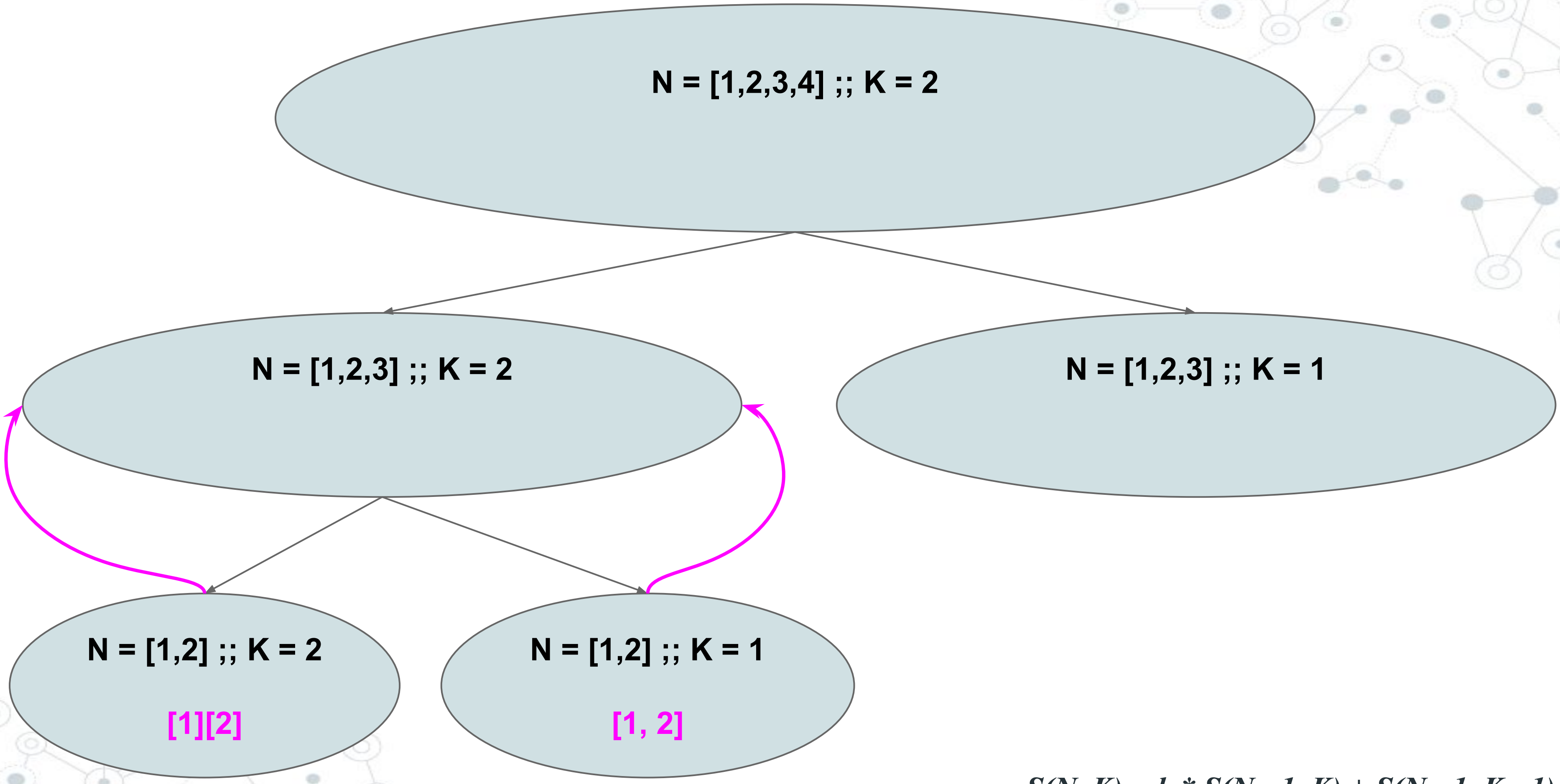
$$S(N, K) = k * S(N - 1, K) + S(N - 1, K - 1)$$



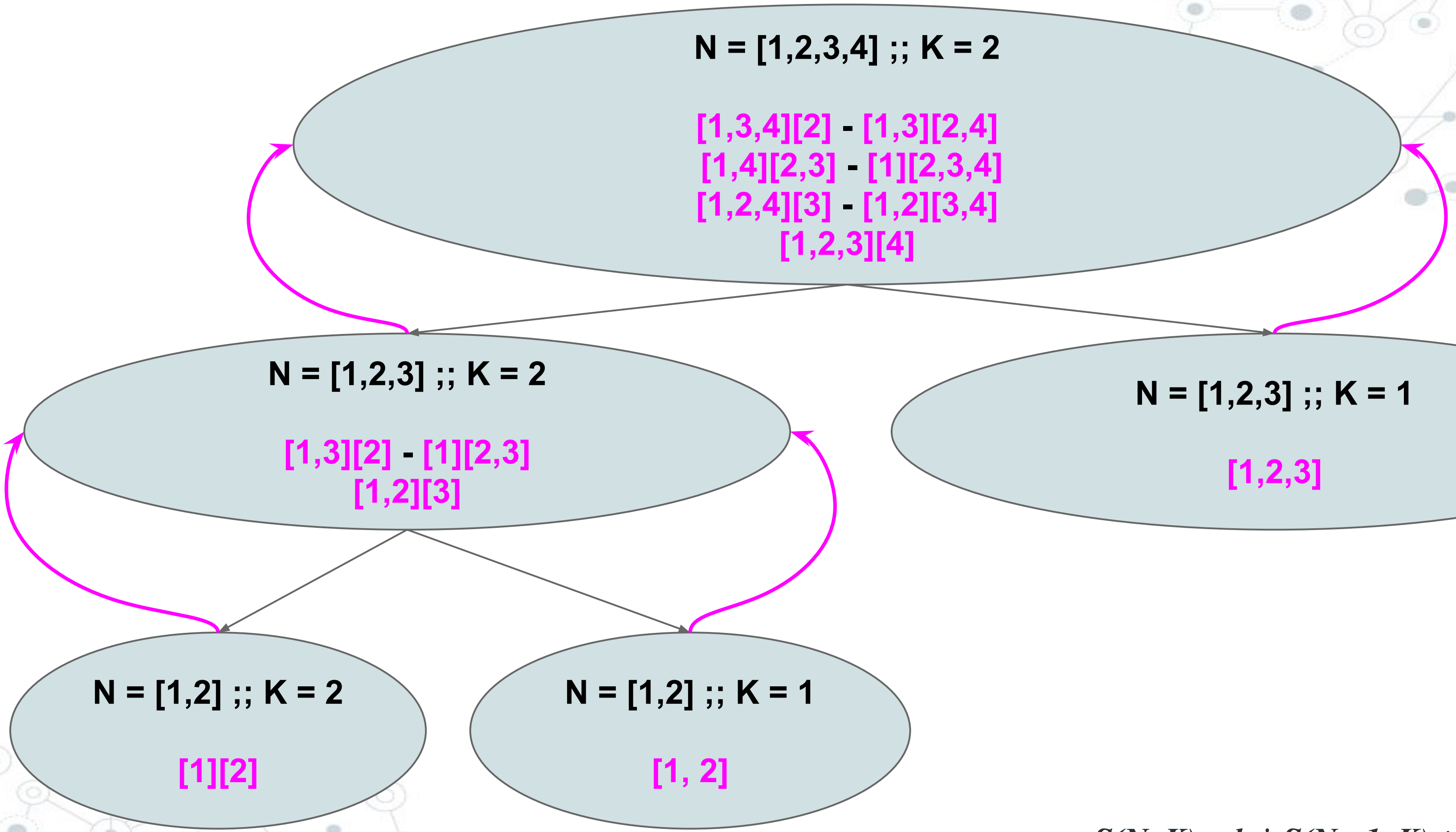
$$S(N, K) = k * S(N - 1, K) + S(N - 1, K - 1)$$



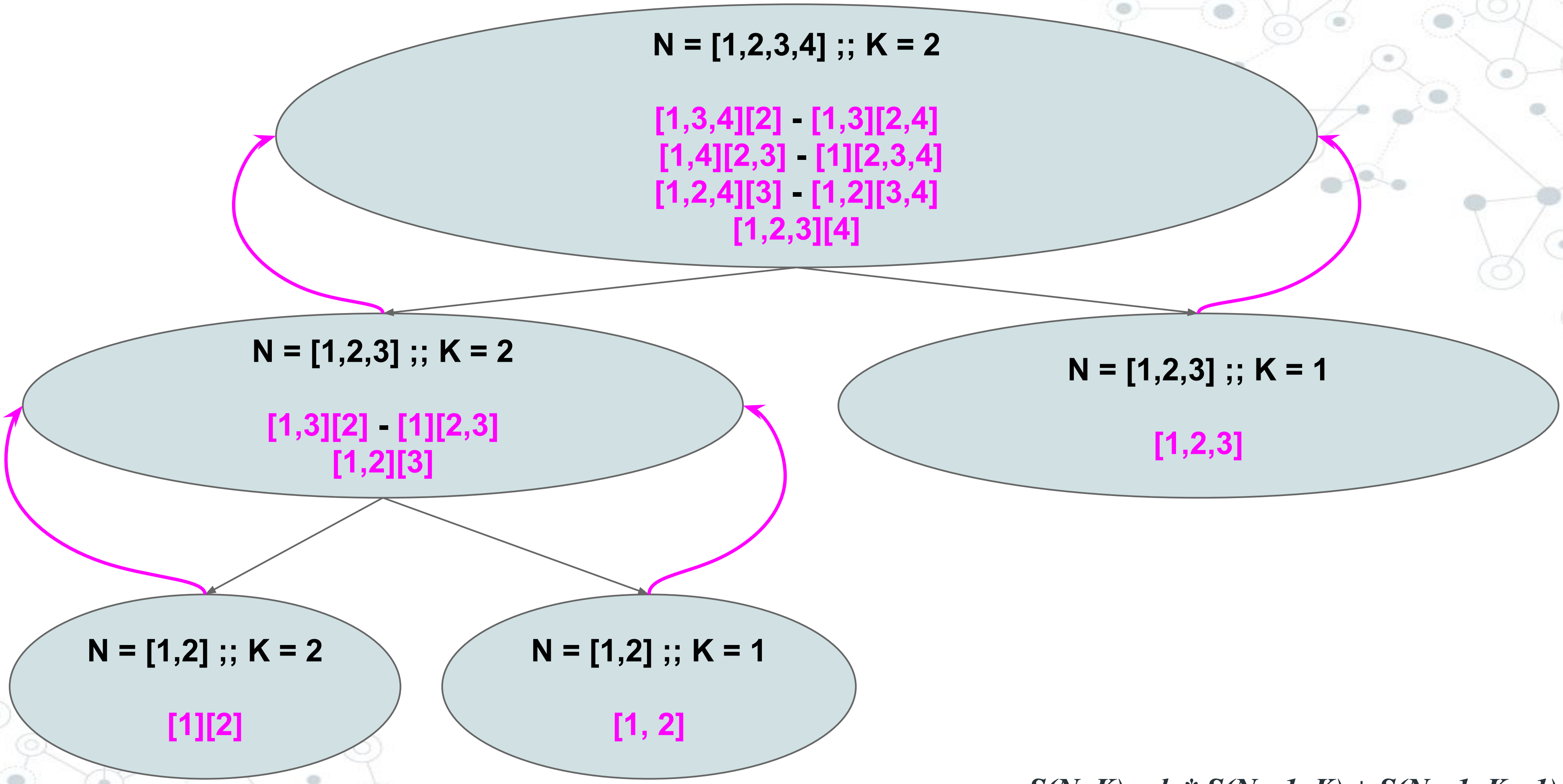
$$S(N, K) = k * S(N - 1, K) + S(N - 1, K - 1)$$



$$S(N, K) = k * S(N - 1, K) + S(N - 1, K - 1)$$



$$S(N, K) = k * S(N - 1, K) + S(N - 1, K - 1)$$



$$S(N, K) = k * S(N - 1, K) + S(N - 1, K - 1)$$

Subset finder (Dynamic Programing)

N ⬇️ K ➡️	0	1	2
0			
1			
2			
3			
4			

$$S(N, K) = k * S(N - 1, K) + S(N - 1, K - 1)$$

BaseCases

N ⬇️ K ➡️	0	1	2
0	[]		
1	[]	[1]	
2	[]	[1,2]	[1][2]
3	[]	[1,2,3]	
4	[]	[1,2,3,4]	

$$S(N, K) = k * S(N - 1, K) + S(N - 1, K - 1)$$

N ⬇️ K ➡️	0	1	2
0	[]		
1	[]	[1]	
2	[]	[1,2]	[1][2]
3	[]	[1,2,3]	[1,3][2] [1][2,3] [1,2][3]
4	[]	[1,2,3,4]	

$$S(N, K) = k * S(N - 1, K) + S(N - 1, K - 1)$$

N ⬇️ K ➡️	0	1	2
0	[]		
1	[]	[1]	
2	[]	[1,2]	[1][2]
3	[]	[1,2,3]	[1,3][2] [1][2,3] [1,2][3]
4	[]	[1,2,3,4]	[1,3,4][2] [1,3][2,4] [1,4][2,3] [1][2,3,4] [1,2,4][3] [1,2][3,4] [1,2,3][4]

$$S(N, K) = k * S(N - 1, K) + S(N - 1, K - 1)$$

Pruned Brute Force

Truck capacity = 30
Demand = {1 : 12, 2 : 8, 3 : 20, 4 :9}

N K ➡️ ⬇️	0	1	2
0	[]		
1	[]	[1]	
2	[]	[1,2]	[1][2]
3	[]	[1,2,3]	[1,3][2] [1][2,3] [1,2][3]
4	[]	[1,2,3,4]	[1,3,4][2] [1,3][2,4] [1,4][2,3] [1][2,3,4] [1,2,4][3] [1,2][3,4][1,2,3][4]

Pruned Brute Force

Truck capacity = 30
Demand = {1 : 12, 2 : 8, 3 : 20, 4 :11}

N K ➡️ ⬇️	0	1	2
0	[]		
1	[]	[1]	
2	[]	[1,2]	[1][2]
3	[]	~~[1,2,3]~~	~~[1,3][2]~~ [1][2,3] [1,2][3]
4	[]	~~[1,2,3,4]~~	~~[1,3,4][2]~~ ~~[1,3][2,4]~~ [1,4][2,3] ~~[1][2,3,4]~~ ~~[1,2,4][3]~~ ~~[1,2][3,4]~~ ~~[1,2,3][4]~~

Performance Comparison

For,
K = 4

N	Dynamic Approach	Prued Dynamic Approach	Percentage Drop
4	1	1	0 %
5	10	9	10 %
6	65	40	38.4 %
7	350	152	56.5 %
8	1701	359	78.8 %
9	7770	1357	82.5 %
10	34105	1958	94.2 %
11	145750	3674	97.4 %
12	611501	5173	99.1 %

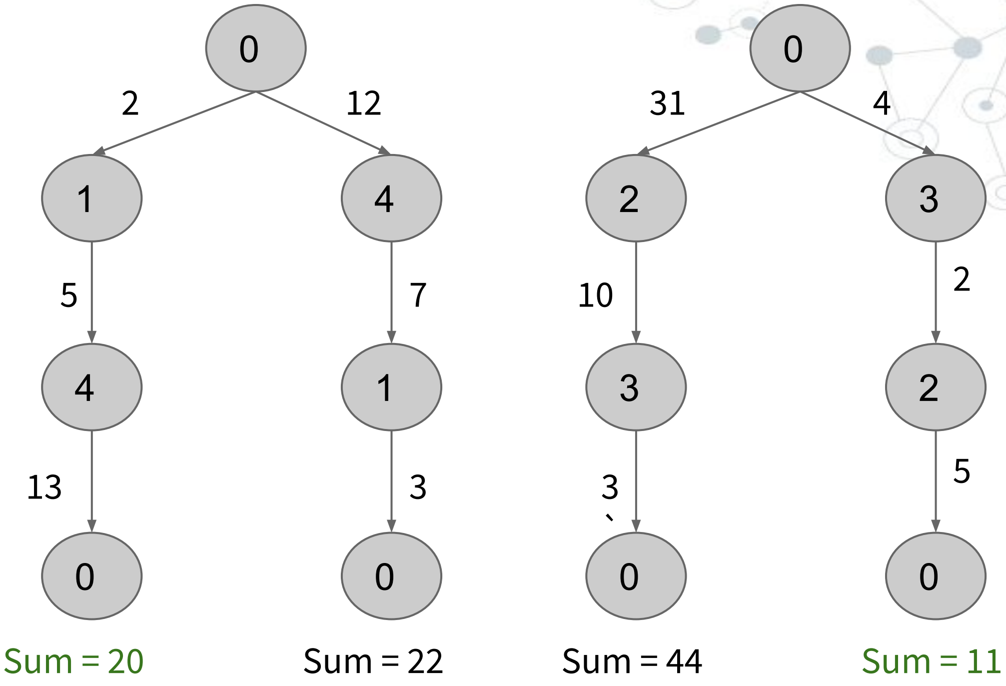
Traveling salesman

- Since I have all the ways that the customers can be served given a set of vehicle
- We need to find which solution is the best
- And also what order the customers should be visited

Traveling salesman

Distance Matrix

	0	1	2	3	4
0	0	2	31	4	12
1	3	0	12	4	5
2	5	9	0	10	11
3	3	2	12	0	4
4	13	7	10	8	0



Result

Truck1 : 0 → 1 → 4 → 0

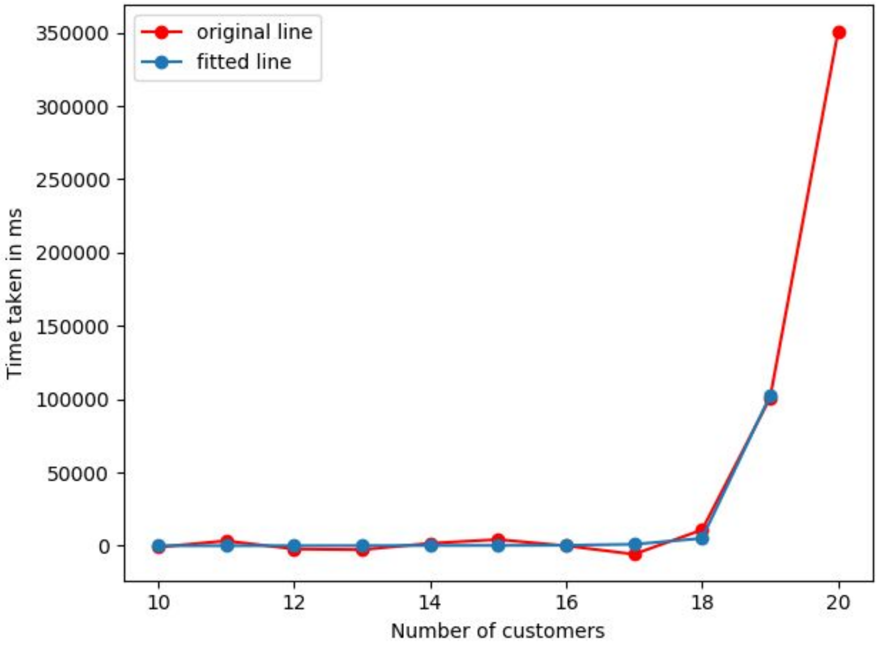
Truck2 : 0 → 3 → 2 → 0

Total Cost = 31

Result comparison

Customers	Time taken(ms)
10	5.0
11	12.0
12	29.0
13	51.0
14	146.0
15	209.0
16	238.0
17	273.0
18	5026.0
19	8447.0

Result

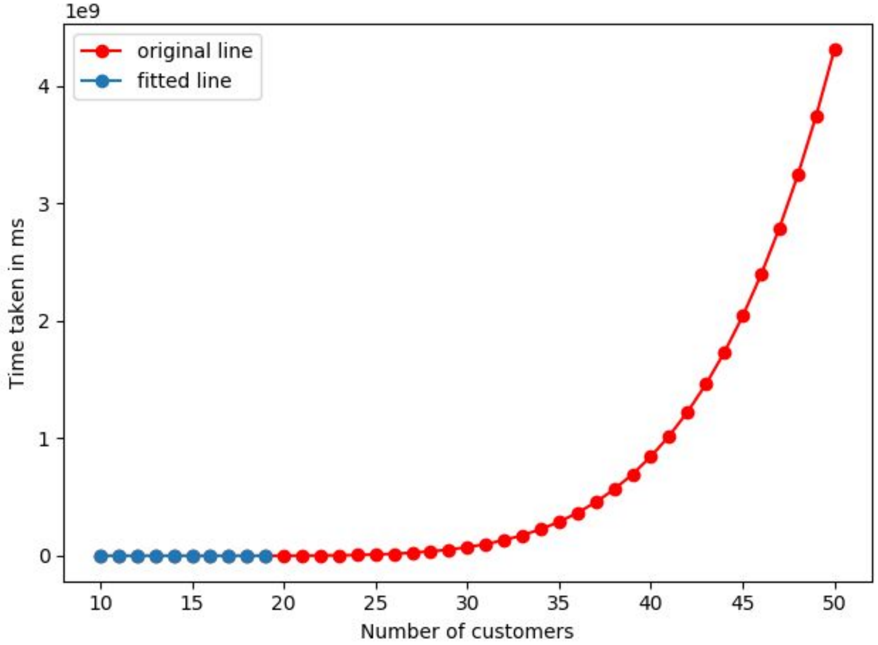


original line
fitted line
350000
300000
250000
200000
150000
100000
50000
0
Time taken in ms
10
12
14
16
18
20
Number of customers

Result

50 Nodes

4.30915618e+09 ms = 49874.49282407 days = **136.64244609334247 years**



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
   b. Compare and contrast the run-time of all the implementations

Questions?

Thank you.