BALANCED STAFF ROUTING FOR MAINTENANCE

Đinh Nguyễn Công Quý - 20214927
Vũ Tuấn Minh - 20210597
Hoàng Tú Quyên - 20214929
Lê Tuấn Anh - 20214874

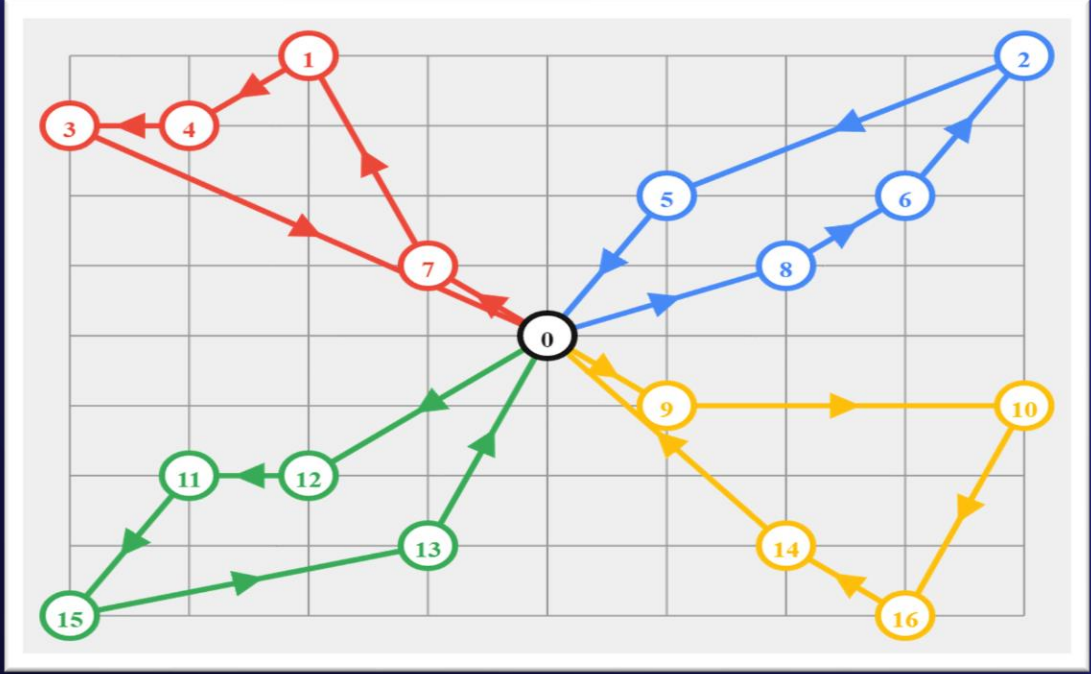
Table of contents

01.

PROBLEM DESCRIPTION

Details about idea of the problem

PROBLEM DESCRIPTION



Customer

+ N customers
+Each customer served by only one employee
+Maintenance time (d)

Employee

+M employees
+Depot=>Customers=>Depot
+Travelling time (t)
+Working time (c) = Travelling time + Maintenance time

GOAL

Minimize the longest working time of an employee

02.

Modelling

Decision variables, Constraint, Objective function

INTEGER PROGRAMMING

Variables

- $x[i, k] = \begin{cases} 1, & if\ customer\ \boldsymbol{i}\ is\ served\ by\ employee\ \boldsymbol{k} \\ 0, & otherwise \end{cases}$

  with: 1 <= i <= N; 1 <= k <= M.

- $y[i, j, k] = \begin{cases} 1, & if\ employee\ \boldsymbol{k}\ travels\ directly\ from\ \boldsymbol{i}\ to\ \boldsymbol{j} \\ 0, & otherwise \end{cases}$

  with: 0 <= i, j <= N; 1 <= k <= M

- **c[k]**: positive integer variable indicate the working time of employee k
  with: 1<=k<=M

- **z**: positive integer variable which represents the maximum working time among the employees

INTEGER PROGRAMMING

Constraints

(1) $\sum_{j=1}^{N} y[0,j,k] = \sum_{i=1}^{N} y[i,0,k] = 1$ $\quad (\mathbf{1 \le k \le M})$

(2) $\sum_{j=1}^{N} y[i,j,k] = \sum_{j=1}^{N} y[j,i,k] = x[i,k]$ $\quad (\mathbf{1 \le i \le N; i \ne j;\ 1 \le k \le M})$

(3) $\sum_{k=1}^{K} x[i,k] = 1$ $\quad (1 \le i \le N)$

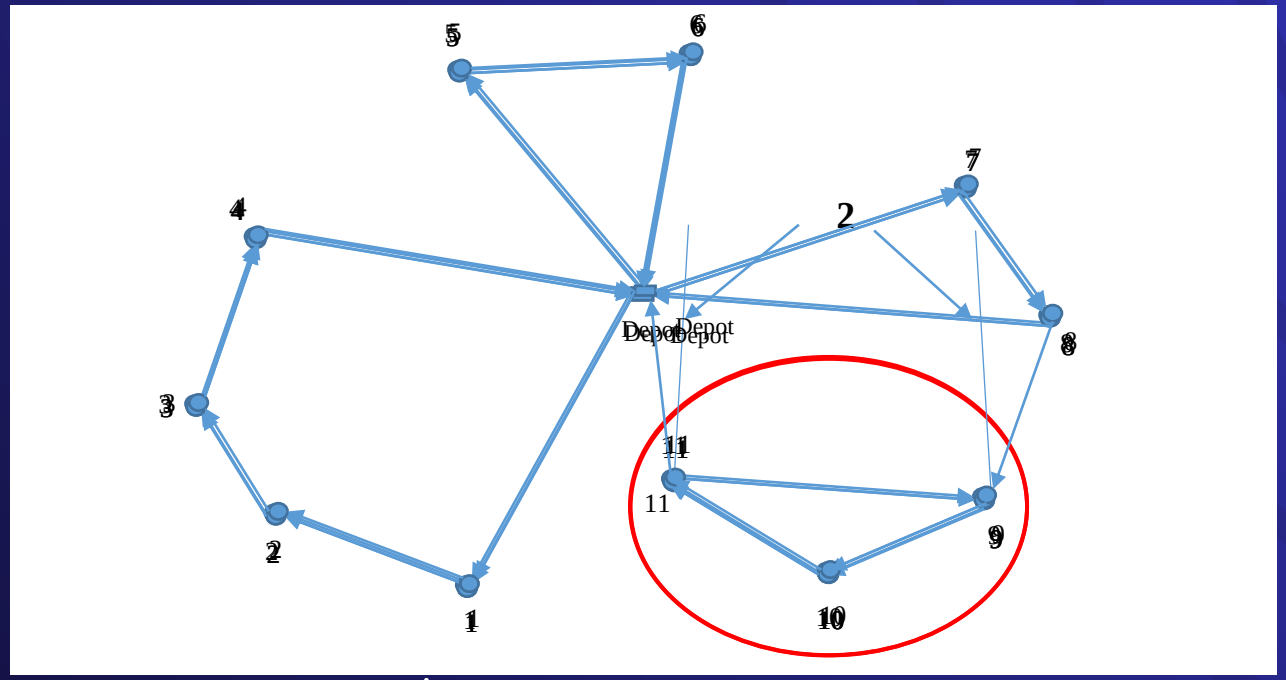
(4) $c[k] = \sum_{i=1}^{N} x[i,k] * d[i] + \sum_{i=0}^{N}\sum_{j=0}^{N} y[i,j,k] * t[i][j]$ $\quad (1 \le k \le M)$

(5) $c[k] \le z$ $\quad (1 \le k \le M)$

INTEGER PROGRAMMING

Sub-tour elimination Constraint [1]

$$(6)\ \sum_{(i,j)\in S} y[i,j,k] \geq 2\ (1 \leq k \leq K;\ \forall S \subseteq \{1,2,\dots,N\}; 2 \leq |S| \leq N - 2$$



INTEGER PROGRAMMING

The objective function:

Minimizing the maximum working time among the employees:

$$z \rightarrow min$$

03.

PROPOSED ALGORITHM

Offer some algorithms to solve the problem

EXACT ALGORITHMS

Applying 2 solvers

SCIP solver

Employ a combination of branching, cutting plane algorithms, and heuristics [2]

CP-SAT solver

Utilizes **SAT (Satisfiability)** method [3]

=> CP-SAT gives faster performance than SCIP solver

HEURISTICS

Method 1

Greedy algorithm

Method 2

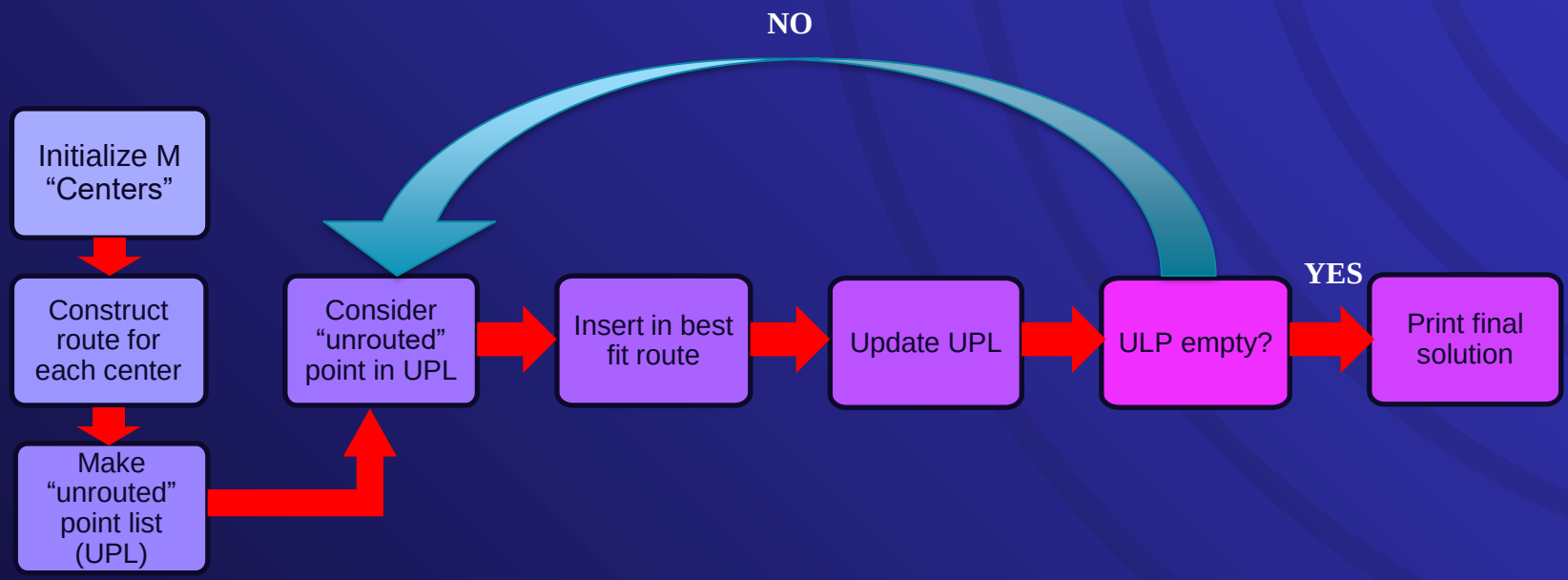
Local search algorithm

Method 3

Hybrid of Greedy and Local search algorithm

Method 1: Greedy

The algorithm involves the following steps:



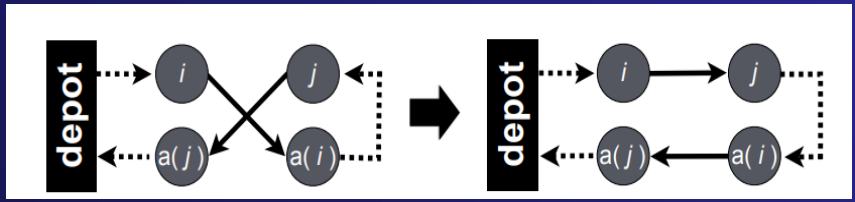
Method 1: Greedy

Result for sorted and unsorted maintenance time

	f_min	f_max	f_avg	std_dev
Unsorted	3410	4080	3750.6	147.8
Sorted in ascending order	3450	4180	3788.4	163.6
Sorted in descending order	3380	4090	3757.2	133.1

Method 2: Local Search

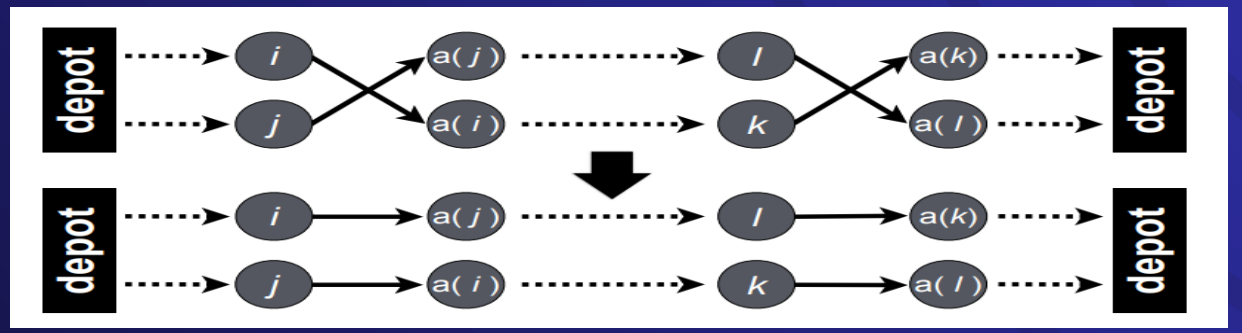
Idea: **Utilizes the 2-opt algorithm and its expansion (CROSS-exchange)**



2-opt algorithm [4]

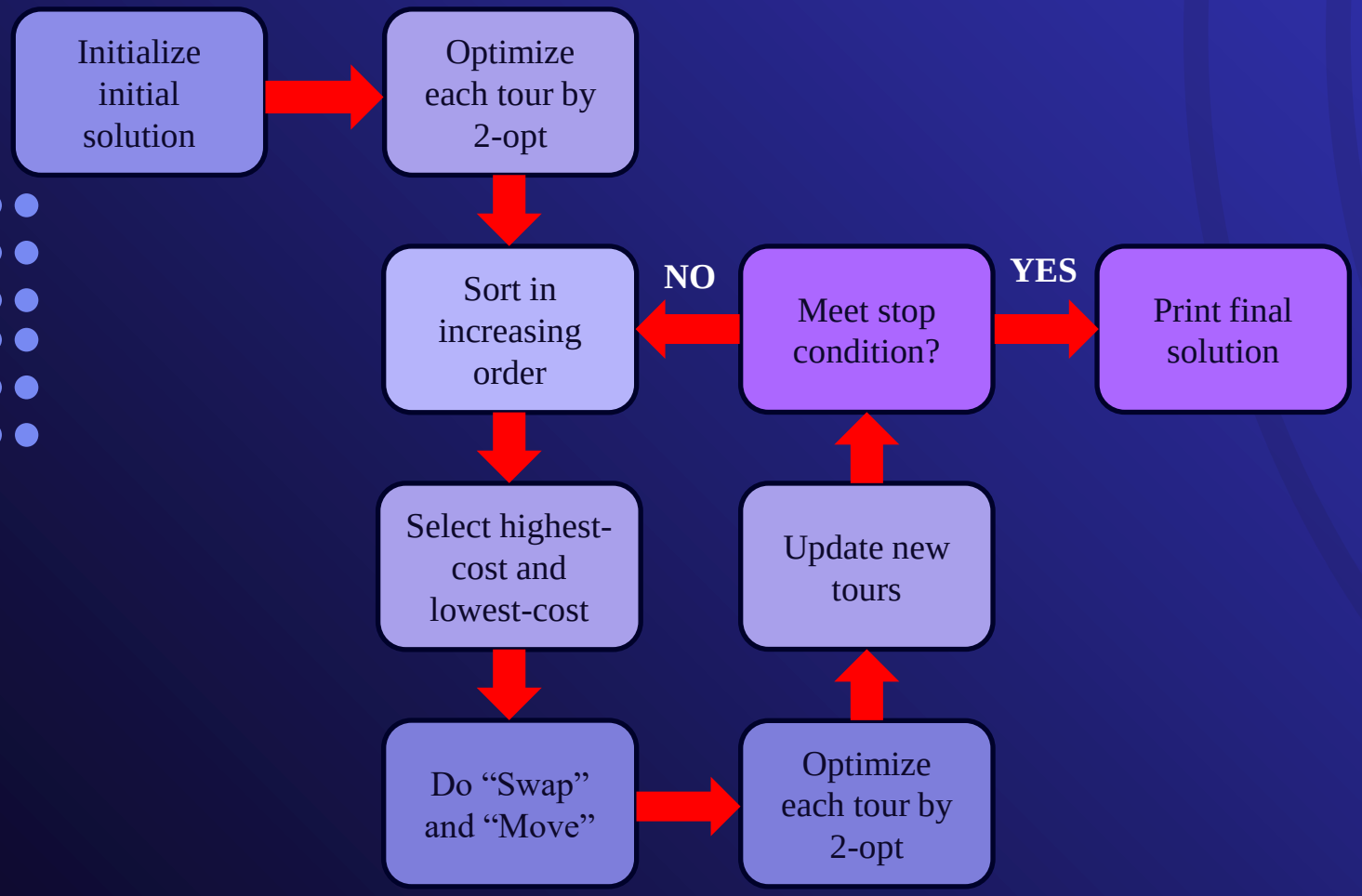
```
procedure 2optSwap(route, v1, v2) {
    1. take route[0] to route[v1] and add them in order to new_route
    2. take route[v1+1] to route[v2] and add them in reverse order to new_route
    3. take route[v2+1] to route[start] and add them in order to new_route
    return new_route;
}
```

Swaping in 2-opt [6]



CROSS-exchange algorithm [5]

Method 2: Local Search



Two operators created for the initial solution:

+ Move: Move a city from the tour has maximum cost to the tour has minimum cost.
+ Swap: Swap randomly two locations from maximum tours and minimum tours to find the best neighborhood.

=> In general, local search methods cannot find optimal solutions due to the local minimum problem.

Method 3: Greedy + Local Search

Purpose: reduce the number of times the swapping and moving operators are called

Idea:

1. Using the greedy algorithm to create initial feasible solution.

2. Once the initial solution is obtained from the greedy algorithm, the local search algorithm starts by exploring its neighbors.

3. The local search algorithm looks for a neighbor that satisfies all the constraints, and also has a higher objective function value than the current solution

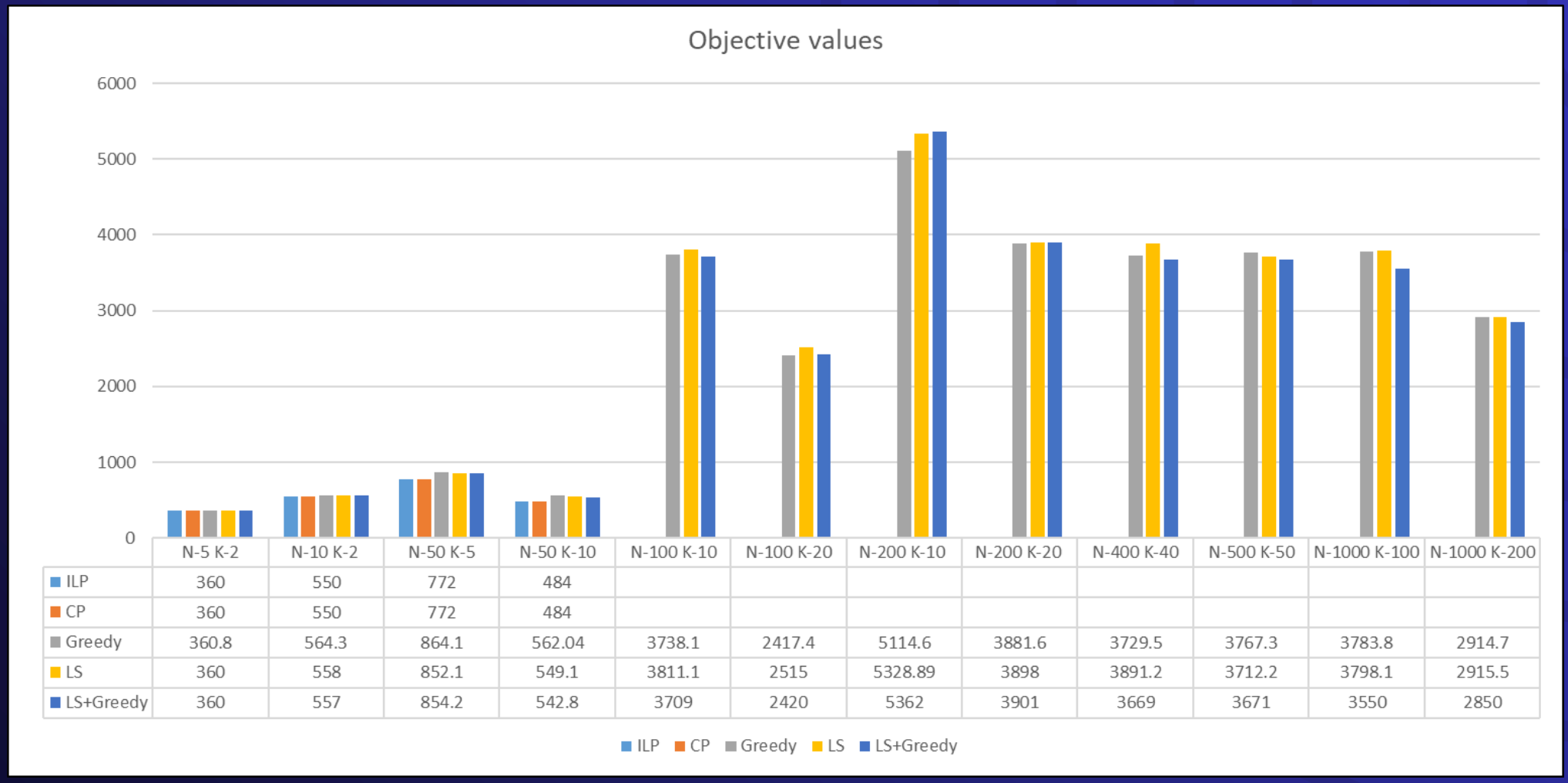
4. This process is repeated until a local optimum is reached

04.

EXPERIMENTAL RESULT

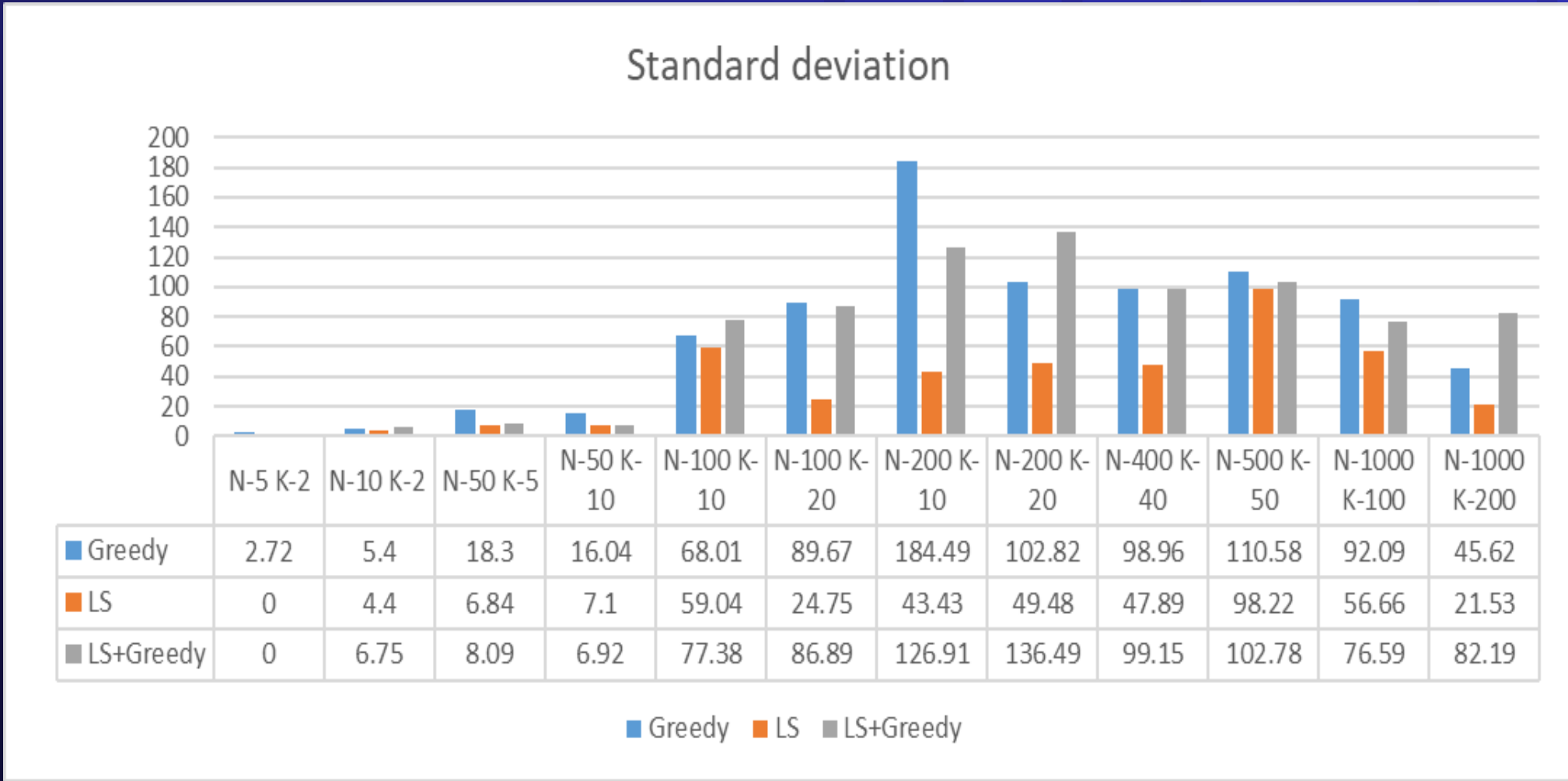
Show experimental result and analysis

EXPERIMENTAL RESULT



	N-5 K-2	N-10 K-2	N-50 K-5	N-50 K-10	N-100 K-10	N-100 K-20	N-200 K-10	N-200 K-20	N-400 K-40	N-500 K-50	N-1000 K-100	N-1000 K-200
ILP	360	550	772	484								
CP	360	550	772	484								
Greedy	360.8	564.3	864.1	562.04	3738.1	2417.4	5114.6	3881.6	3729.5	3767.3	3783.8	2914.7
LS	360	558	852.1	549.1	3811.1	2515	5328.89	3898	3891.2	3712.2	3798.1	2915.5
LS+Greedy	360	557	854.2	542.8	3709	2420	5362	3901	3669	3671	3550	2850

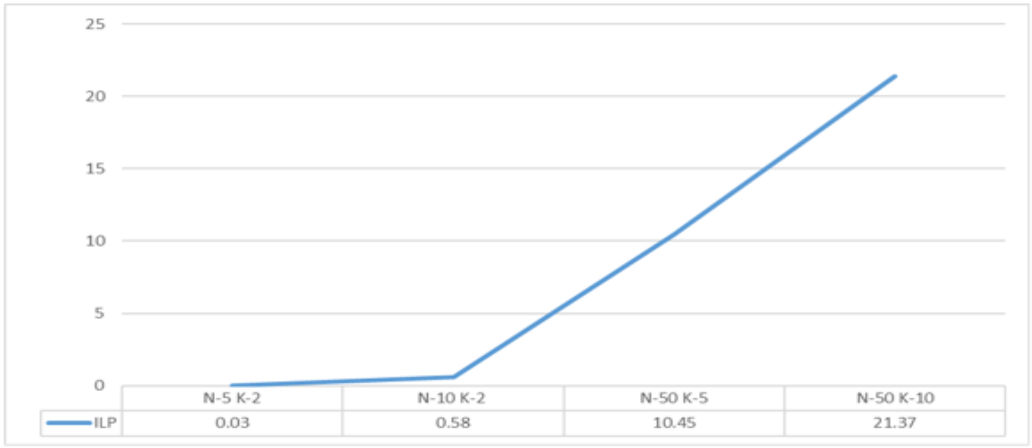
EXPERIMENTAL RESULT



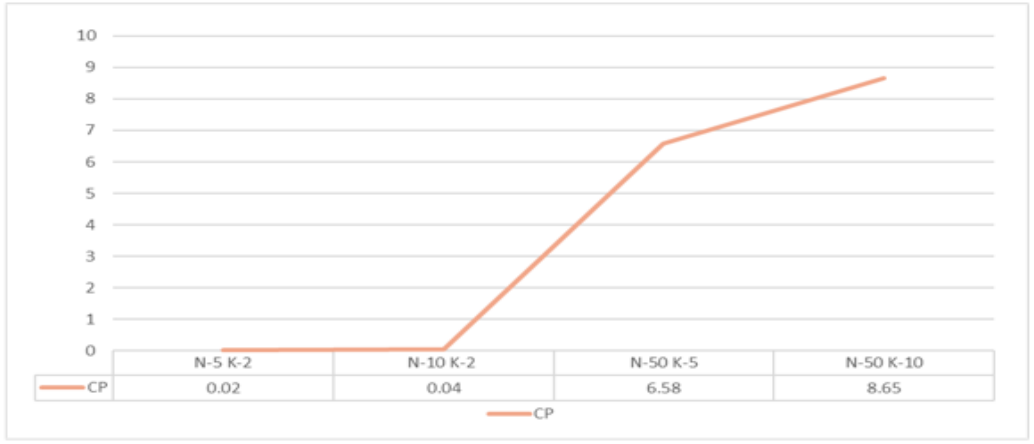
	N-5 K-2	N-10 K-2	N-50 K-5	N-50 K-10	N-100 K-10	N-100 K-20	N-200 K-10	N-200 K-20	N-400 K-40	N-500 K-50	N-1000 K-100	N-1000 K-200
Greedy	2.72	5.4	18.3	16.04	68.01	89.67	184.49	102.82	98.96	110.58	92.09	45.62
LS	0	4.4	6.84	7.1	59.04	24.75	43.43	49.48	47.89	98.22	56.66	21.53
LS+Greedy	0	6.75	8.09	6.92	77.38	86.89	126.91	136.49	99.15	102.78	76.59	82.19

EXPERIMENTAL RESULT

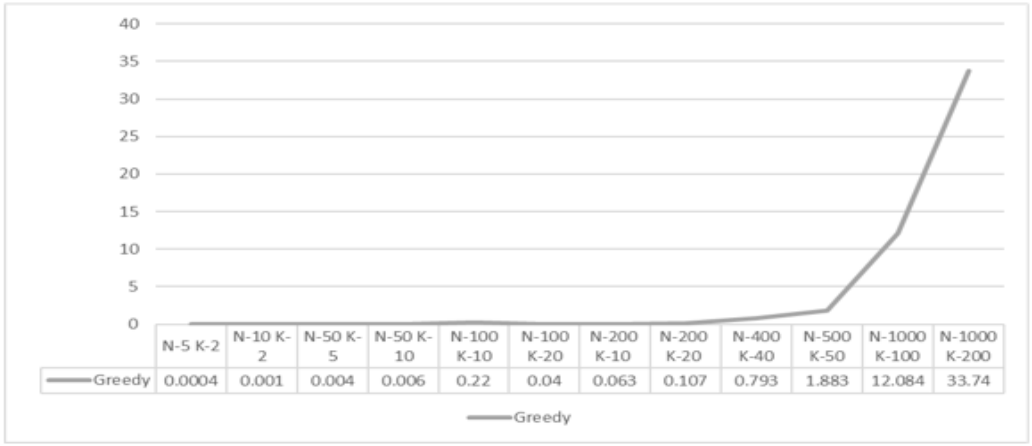
RUNNING TIME



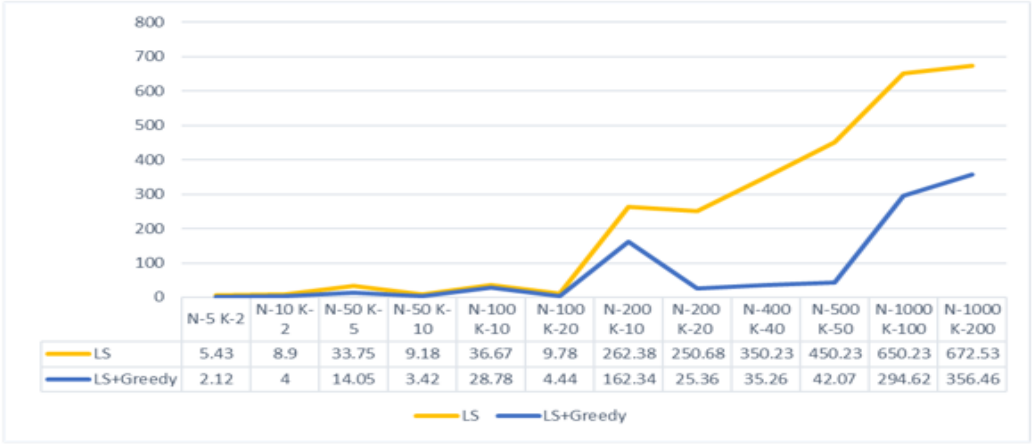
	N-5 K-2	N-10 K-2	N-50 K-5	N-50 K-10
ILP	0.03	0.58	10.45	21.37



	N-5 K-2	N-10 K-2	N-50 K-5	N-50 K-10
CP	0.02	0.04	6.58	8.65



	N-5 K-2	N-10 K-2	N-50 K-5	N-50 K-10	N-100 K-10	N-100 K-20	N-200 K-10	N-200 K-20	N-400 K-40	N-500 K-50	N-1000 K-100	N-1000 K-200
Greedy	0.0004	0.001	0.004	0.006	0.22	0.04	0.063	0.107	0.793	1.883	12.084	33.74



	N-5 K-2	N-10 K-2	N-50 K-5	N-50 K-10	N-100 K-10	N-100 K-20	N-200 K-10	N-200 K-20	N-400 K-40	N-500 K-50	N-1000 K-100	N-1000 K-200
LS	5.43	8.9	33.75	9.18	36.67	9.78	262.38	250.68	350.23	450.23	650.23	672.53
LS+Greedy	2.12	4	14.05	3.42	28.78	4.44	162.34	25.36	35.26	42.07	294.62	356.46

05.

CONCLUSION

Final conclusion for the problem

CONCLUSION

SHORTEST TIME; Greedy Algorithm
-> Recommended in reality, suitable for enormous datasets
-> its failure rate is high when N and M are close
to each other

Exact algorithms:
-> With small datasets: Both solvers returns the solutions in a nearly equal amount of time. The CP-SAT gives a more slow and steady increase in running time than the SCIP.
-> When N and M get larger (N > 50, M > 10), the solver cannot return the solution due to the overflow problem

CONCLUSION

Local Search:

-> Gives more stable performance than Greedy Algorithm. The performance gap gets narrower when M increases

-> Depends much on the initial solution. If using the Greedy initial solution, the running time could be much faster, and the standard deviation doesn’t rise much.

-> The slow running time is probably due to the programming language.

REFERENCES

[1] *"Explicit Dantzig-Fulkerson-Johnson formulation"*
https://how-to.aimms.com/Articles/332/332-Explicit-Dantzig-Fulkerson-Johnson-formulation.html

[2], [3] *"OR-Tools Guides"*
https://developers.google.com/optimization/introduction

[3] Qing Guo, Jian Peng, Wenzheng Xu, *"Minimizing the Longest Tour Time Among a Fleet of UAVs for Disaster Area Surveillance"*
http://users.cecs.anu.edu.au/~Weifa.Liang/papers/GPXLJXYW20.pdf

[4], [5] Takafumi Matsuura, Kazumiti Numata, *"Solving Min-Max Multiple Traveling Salesman Problems by Chaotic Neural Network"*
https://www.ieice.org/nolta/symposium/archive/2014/articles/B1L-C1-6137.pdf

[6] https://en.wikipedia.org/wiki/2-opt


THANK YOU :)