Master in Innovation and Research in Informatics

Algorithmic Methods For Mathematical Models

Course Project Presentation

Roger Gili i Coscojuela
roger.gili.i@estudiantat.upc.edu

Edgar Rodríguez de la calle
edgar.rodriguez.de.la.calle@estudiantat.upc.edu

“Forget about Mr. Bond and forget about the door. This is a ***TSP*** *problem”.*

1.

Problem statement

Initial Information

- A set *S* that contains a fixed *n* number of codes.
- Each code is a sequence of *m* binary digits.

```
m = 10;
n = 6;
S =
[
        [ 0 0 0 0 0 0 0 0 0 0 ]
        [ 1 0 1 0 1 0 0 0 0 1 ]
        [ 0 0 1 0 1 1 0 1 0 1 ]
        [ 0 1 0 1 0 1 1 0 1 1 ]
        [ 0 1 1 0 1 1 0 0 0 0 ]
        [ 0 1 1 1 0 0 1 0 0 1 ]
];
```

2.

Mixed Integer Linear Programming Model

Inputs and Variables of the Model

Inputs

n : The number of codes.

m : The number of binary digits that each code has.

S : The set of n codes with m digits

Variables

f: Discrete variable -> Flips

$\mathbf{d_{i,j}}$: Discrete variable ->Cost. ($0 \leq i,j \leq n$)

$\mathbf{y_{i,j}}$: Binary variable -> Path ($0 \leq i,j < n$)

$\mathbf{x_{k,j}}$: Binary variable -> Subtour ($1 \leq k,j < n$)

Objective Function & Constraints

Objective Function:

Minimize: *f*

Subject To:

1: Intended Meaning of *f* → $\sum_{i,j=0}^{n} d_{i,j} y_{i,j}$

Objective Function & Constraints

Subject To:

2: Out Constraint → $$\sum_{j=1}^{n} x_{k,j} = 1$$

3: In Constraint → $$\sum_{k=1}^{n} x_{k,j} = 1$$

4: First Code is 0..0 → $$y_{0,j} = x_{1,j} : \forall j \ (1 \le j < n)$$

5: Last Code is 0..0 → $$y_{j,0} = x_{(n-1),j} \ \forall j: \ (1 \le j < n)$$

5: No subtour → $$x_{k,i} + x_{(k+1),j} - y_{i,j} \le 1$$
$$\forall k: \ (1 \le k < n - 2) \ \forall i,j: \ (1 \le i,j < n)$$

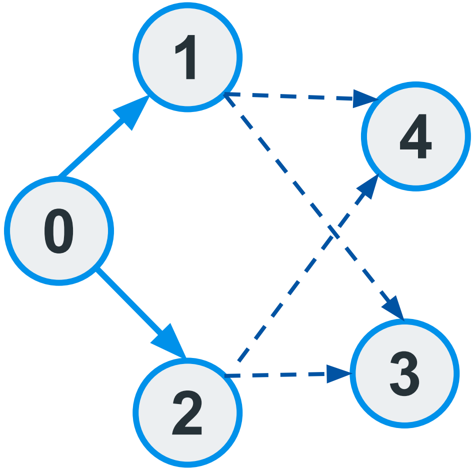
3.

Heuristic Algorithms

Greedy

Algorithm 1 GREEDY

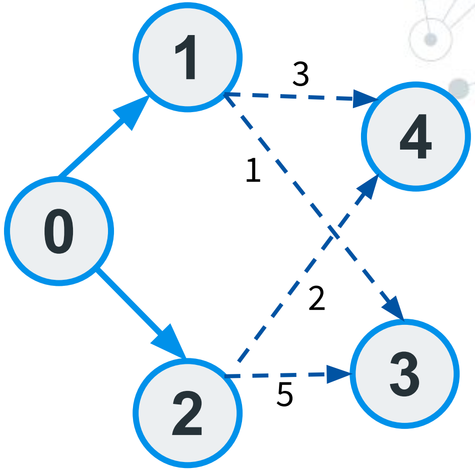
```
function GetFeasibleLinks(ω)
    Candidates = {}
    for i in ω do
        if node[i] connections < 2 then
            for each j in N do
                if Node[j] connections == 0 then
                    c = Cost of moving from Node[i] to Node[j]
                    candidate = (c, Node[j])
                    Add candidate to Candidates
                end if
            end for
        end if
    end for
    Sort Candidates by its cost
    return Candidates
end function

function GreedySolver
    Initialize N, the initial set of codes
    ω ← {}
    Add to ω the first node in N
    while ω is not a solution do
        Candidates = getFeasibleLinks(ω)
        Update ω with first node of Candidates
    end while
    return ω
end function
```



GRASP

Algorithm 2 GRASP

function GRASPSOLVER
  Initialize N, the initial set of codes
  $\omega \leftarrow \{\}$
  Add to ω the first node in N
  while ω is not a solution **do**
    Candidates = getFeasibleLinks(ω)
    $q_{min} \leftarrow min\{q(c) \mid c \in Candidates\}$
    $q_{max} \leftarrow max\{q(c) \mid c \in Candidates\}$
    $RCL_{min} \leftarrow \{c \in Candidates \mid q(c) \leq q_{min} + \alpha(q_{max} - q_{min})\}$
    Select $c \in$ *RCL at random*
    Update ω with random selected node form RCL
  end while
  return ω
end function



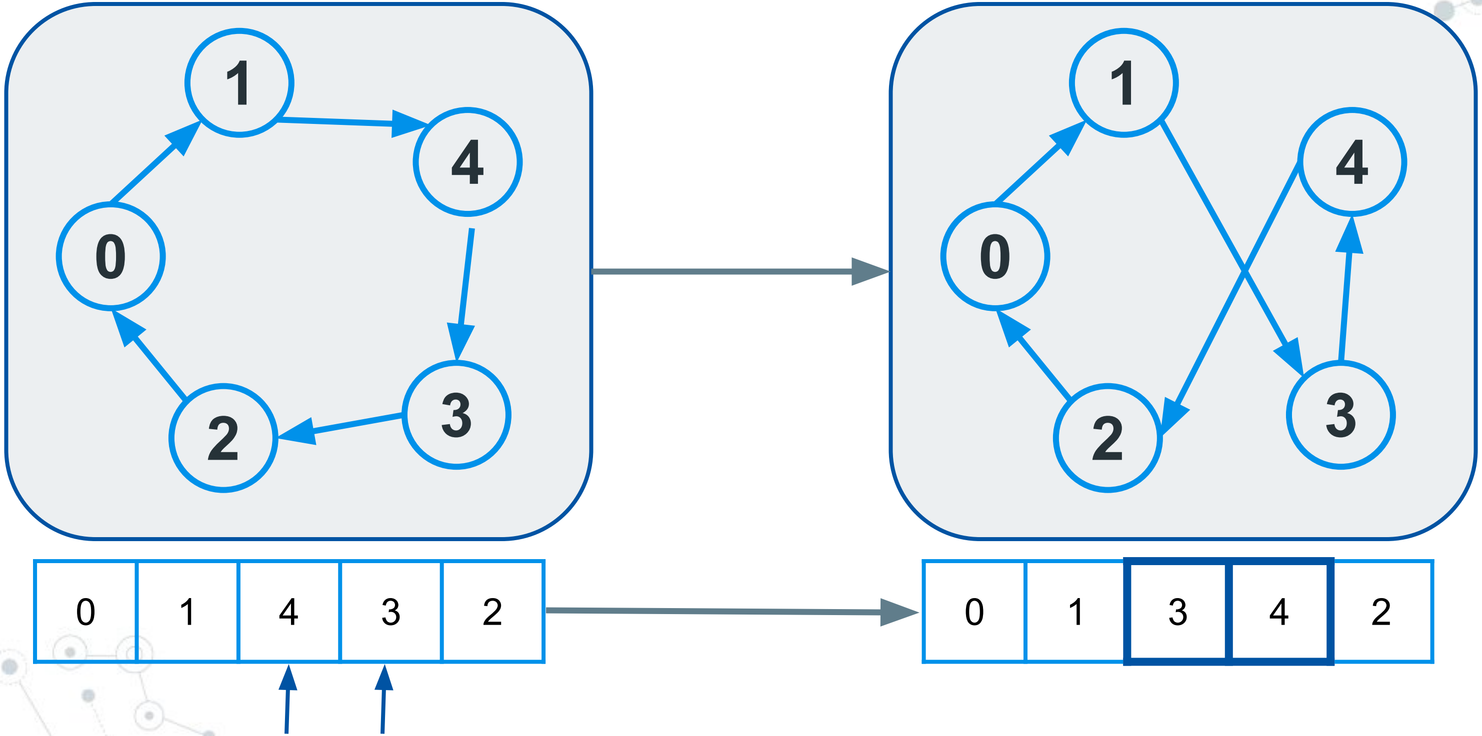
1	2	3	5

Given **α** of 0.7:
RCL_{min} = 1+0.7(5-1) = 3.8

1	2	3	5

Local Search

2 Edge Exchange



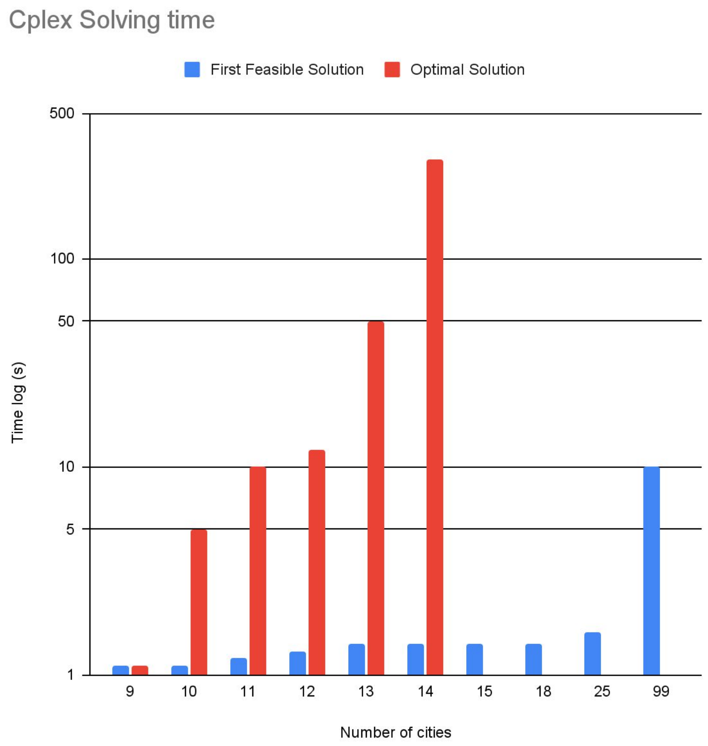
4.

Results

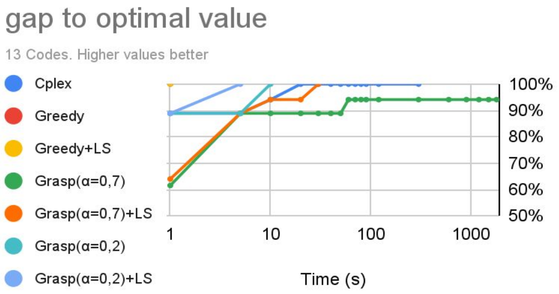
Cplex solving time

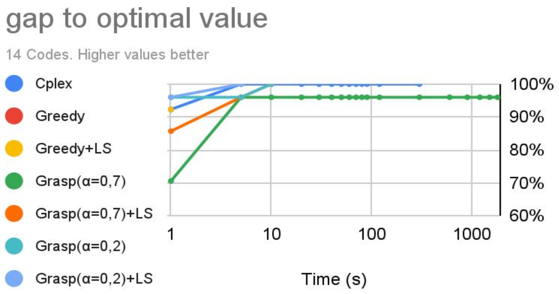
- Our Cplex model time grows exponentially

- With more than 15 codes our model is time infeasible



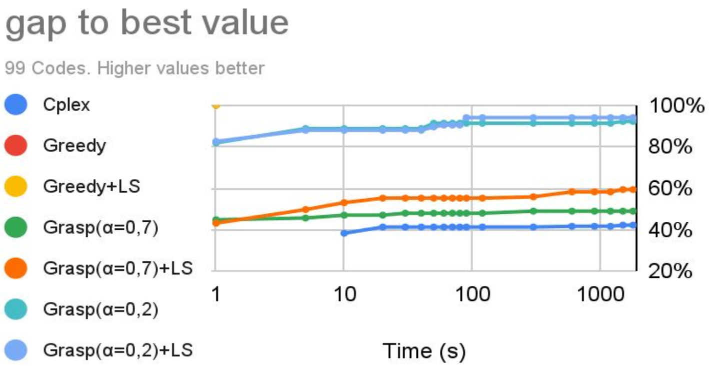
Algorithm comparison up to 15 codes





- Greedy algorithm gives good and fast solutions.
- After some tuning, lower **α** values give better results in GRASP.
- Combining Local Search with the Greedy and GRASP provides better results than his counterparts.

Algorithm comparison 25 and 99 codes



5.

Conclusions & Future Work

Conclusions

Future work

- Grasp with increasing **α** value
- Try other local search algorithms (3-opt, Lin–Kernighan heuristic)
- Try other Cplex models

Thanks!

Any questions?