AMMM Project

Bertille Temple
David Masek

Formalisation: optimisation problem with integers

For a given number n we want to find a set of different natural numbers such that **no two different pairs of numbers in the set are the same distance apart** and the difference between the maximum and minimum numbers in the set is minimized.

VARIABLES:

- x[i],
- d[i,j],

OBJECTIVE FUNCTION: min z

CONSTRAINTS: z is the maximum of the set, the set of x is ordered without repetitions, and the variables have the intending meaning

… how to implement the constraint “**no two different pairs of numbers in the set are the same distance apart”** in a linear way ?

Contents

OPL

Linear model

Not linear implementation

Meta-Heuristics

Greedy+Local Search

GRASP

Comparison

Linear model

VARIABLES: x[i], d[i,j]

- av[i,j], and binary variable e[i,j]. e[i,j] = 1 if d[i,j]-av[i,j] ⩾ 0; e[i,j] = 0 if d[i,j]+av[i,j] ⩽ 0
- b[i,j,k,l] is 1 when the distance between x[i] and x[j] is the same as the distance between x[k] and x[l], else 0

OBJECTIVE FUNCTION: min z

CONSTRAINTS: z is the maximum of the set, the set of x is ordered without repetitions, av[i,j] has the intended meaning:

-av[i,j] ⩽ d[i,j] ⩽ av[i,j]

d[i,j] - av[i,j] ⩾ -2*11 * (1-e[i,j]); -2*11 is a lower bound of d[i,j] - av[i,j]

d[i,j] + av[i,j] ⩽ 2*11 * e[i,j]; 2*11 is an upper bound of d[i,j] - av[i,j]

no two different pairs of numbers in the set are the same distance apart:

$$\sum_{k=1}^{N}\sum_{j=1}^{N}\sum_{l=1}^{N} b[i,j,k,l] == 2, 1 \leq i \leq n, i \neq k$$

Not linear implementation

$$\forall 1 \leq i,j,k,l \leq n : ((i,j) = (k,l)) \vee (i = j \wedge k = l) \iff |d_{i,j}| = |d_{k,l}|$$

$$((i,j) \neq (k,l)) \wedge (i \neq j \vee k \neq l) \iff |d_{i,j}| \neq |d_{k,l}|$$

We can see that:

$$i \neq j \wedge i \neq k \wedge k \neq l \wedge j \neq l \implies ((i,j) \neq (k,l)) \wedge (i \neq j \vee k \neq l)$$

Thus:

$$i \neq j \wedge i \neq k \wedge k \neq l \wedge j \neq l \implies |d_{i,j}| \neq |d_{k,l}|$$

translated into

```
forall (i,j in N: i < j)
    forall (k,l in N: k < l)
        (i != k && j != l) => d[i,j] != d[k,l];
```

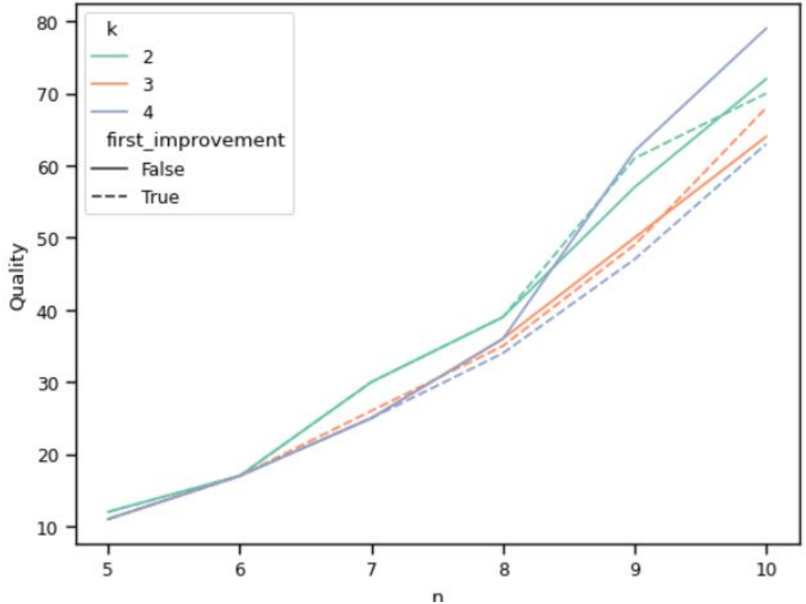
opl constraint

Greedy algorithm

```
n = ... # input, target set size
M = 1 # hyperparameter, size of the candidate set
C = {} # candidate set
X = {} # solution
while |X| < n:
    # update C
    C = {}
    c = max(X) # 0 for empty set
    while |C| < M:
        if is_feasible( X ∪ {c} ):
            C.insert({c})
            c += 1
    # evaluation function
    q(c, X) = max(X ∪ {c}) = c
    # selection function
    c_best = argmin ( q(c, X) for c in C )
    # update solution
    X = X ∪ {c_best}
return q(X), X
```

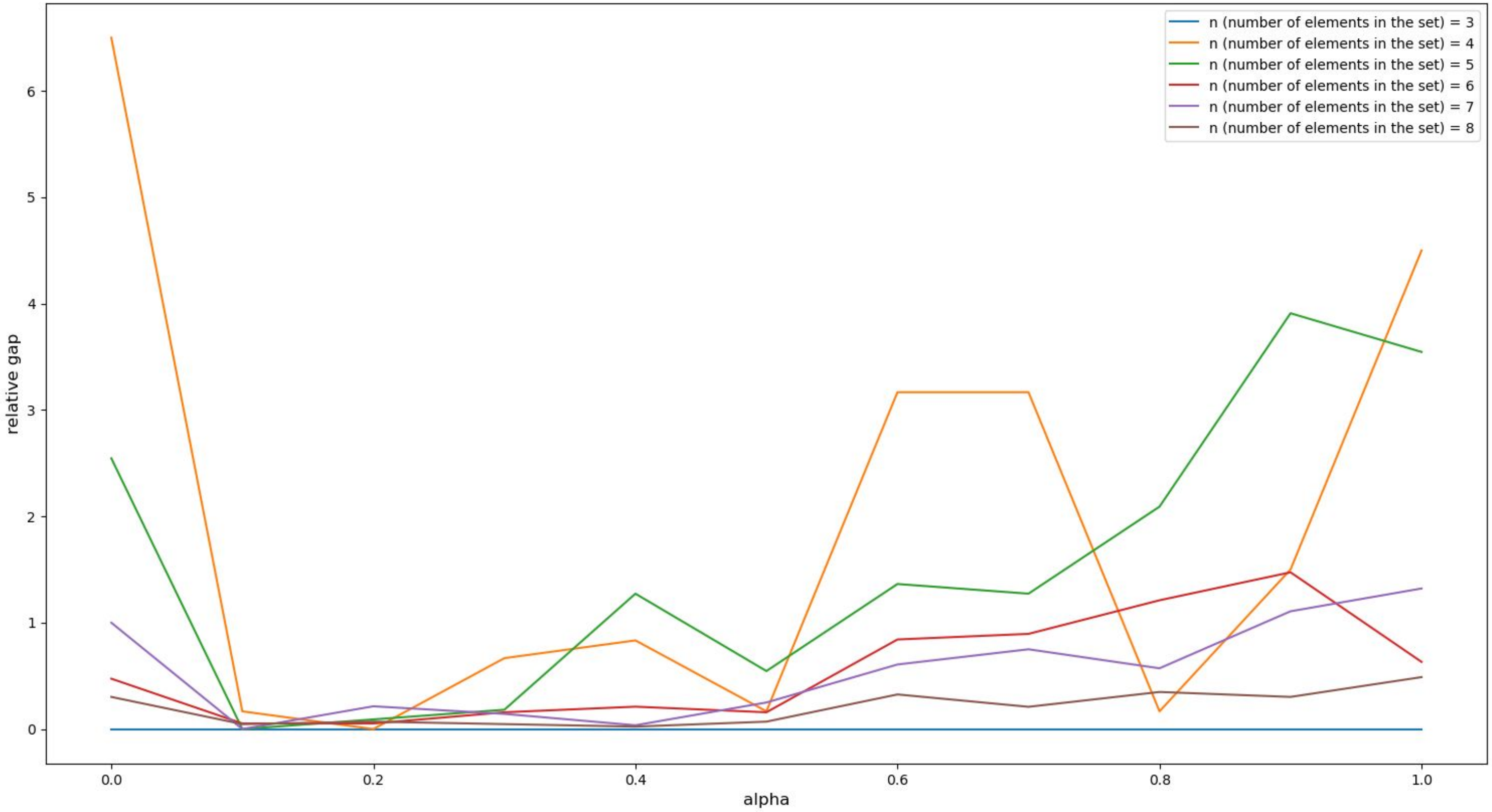
Local search

- exchange K numbers
- first improving strategy



GRASP

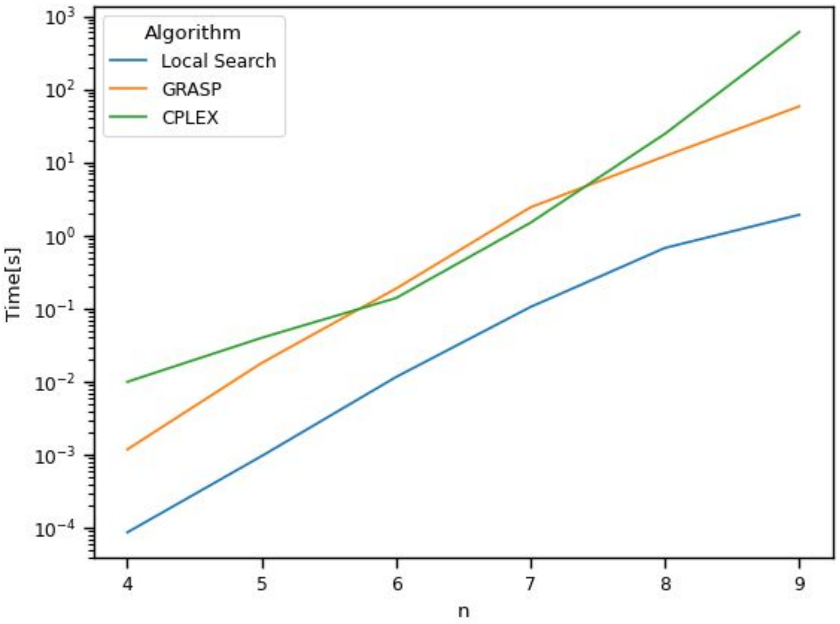
Alpha tuning



Comparison of algorithms performance, computing time

Algorithm n	CPLEX	GRASP	Local Search
2	0.00	0.00	0.00
3	0.02	0.00	0.00
4	0.01	0.00	0.00
5	0.04	0.02	0.00
6	0.14	0.19	0.01
7	1.50	2.44	0.11
8	24.66	12.19	0.68
9	610.09	58.46	1.92

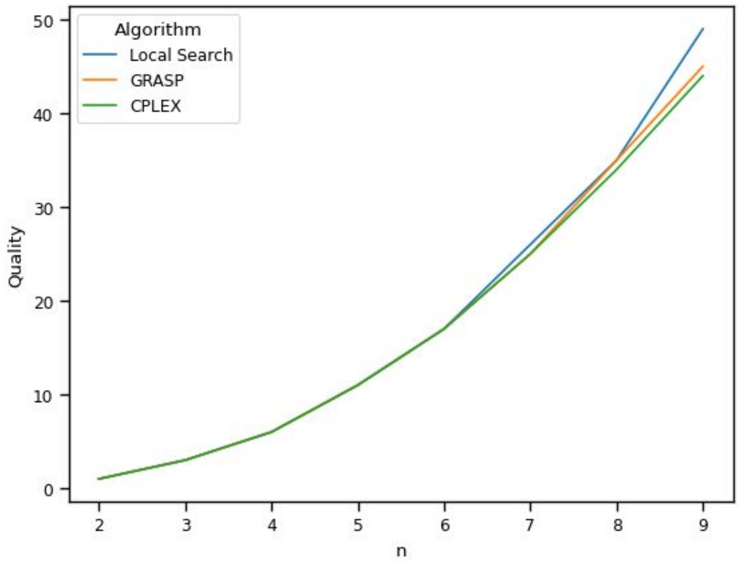
Table 3: Comparison of algorithms: time in seconds



Comparison of algorithms performance, solution quality

Algorithm n	CPLEX	GRASP	Local Search
2	1	1	1
3	3	3	3
4	6	6	6
5	11	11	11
6	17	17	17
7	25	25	26
8	34	35	35
9	44	45	49

Table 2: Comparison of algorithms: quality of the found solution



Recap/conlusion