

21 July 2022

Mathematics Department, University of Padova

Number Mind Game

Knowledge and Data Mining Project

ADVISOR

Luciano Serafin

STUDENT

Mojtaba Amini
2041518

Table of
Contents

| | | Page |
|-----|---------------------------------|------|
| I | Problem and Method | 3 |
| II | Solving Procedure and Functions | 8 |
| III | Results | 13 |

Chapter I

Problem and Method

Number Mind Game



• Initial Theory

- first of all, we considered a number for every number in every position

```
def digit(n,r):
    return n*numdigit + r + 1
```

- The first constraint in the problem is :

#each position should have at least a number

$$\bigwedge_{r=1}^{\text{number of digit}} \bigvee_{n=0}^9 \text{digit}(n,r)$$

```
1 for n in range(10):
2     for r in range(numdigit):
3         print("digit({},{})->{}".format(n,r,digit(n,r)))
```

```
digit(0,0)->1
digit(0,1)->2
digit(0,2)->3
digit(0,3)->4
digit(0,4)->5
digit(0,5)->6
digit(0,6)->7
digit(1,0)->8
digit(1,1)->9
digit(1,2)->10
digit(1,3)->11
digit(1,4)->12
digit(1,5)->13
digit(1,6)->14
digit(2,0)->15
digit(2,1)->16
digit(2,2)->17
digit(2,3)->18
digit(2,4)->19
digit(2,5)->20
digit(2,6)->21
digit(3,0)->22
digit(3,1)->23
digit(3,2)->24
digit(3,3)->25
digit(3,4)->26
digit(3,5)->27
digit(3,6)->28
digit(4,0)->29
digit(4,1)->30
digit(4,2)->31
digit(4,3)->32
digit(4,4)->33
digit(4,5)->34
digit(4,6)->35
digit(5,0)->36
```

- Initial Theory

- The second constraint in the problem is :

#A position can not have two value

$$\bigwedge_{r=1}^{\text{number of digit}} \bigwedge_{n=0}^9 \bigwedge_{n \neq n', n'=0}^9 \text{digit}(n, r) \rightarrow \neg \text{digit}(n', r)$$



• Updating Theory

- If the number of the correct answer is less than the length of the initial guess :
- #Use exactly K propositional variables is true:

```
if num_correct >= 0:
    cnf_least = list(itertools.combinations(guess_c, len(guess_c) - int(num_correct) + 1))
    guess_neg = [ -x for x in guess_c]
    cnf_most = list(itertools.combinations(guess_neg, 1 + int(num_correct)))
    for x,y in zip(cnf_least, cnf_most):
        num.add_clause(list(x))
        num.add_clause(list(y))
```

$$\bigwedge_{\substack{I \subseteq [n] \\ |I|=n-k+1}} \bigvee x_i \quad \bigwedge_{\substack{I \subseteq [n] \\ |I|=k+1}} \bigvee_{i \in I} \neg x_i$$

- If the number of the correct answer is equal to the length of the initial guess:

#Guess is found



Chapter II

Solving Procedure and Functions

Number Mind Game



Solving Procedure

- Generating the secret code:

```
1 numdigit = 12
2 #initial guess
3
4 secret_code = random_with_N_digits(numdigit)
5 print ("The secret code is :",secret_code)
6 code_c = num2CNFcode(secret_code,numdigit)
7 print ("The secret code is :",code_c)
~
```

```
The secret code is : 306146929874
The secret code is : [37, 2, 75, 16, 53, 78, 115, 32, 117, 106, 95, 60]
```

```
1 def random_with_N_digits(n):
2     range_start = 10**(n-1)
3     range_end = (10**n)-1
4     return randint(range_start, range_end)
~
```

```
def num2CNFcode(num,numdigit):

    CNFcode = [0] * numdigit
    for x in range(numdigit):
        CNFcode[numdigit - x -1]= digit(int(num%10), numdigit - x -1)
        num=int(num/10)
    return CNFcode
```




Solving Procedure

- Generating an initial guess:

```
9
10 initial_guess = int (''.join(str(int(i)) for i in [8] * numdigit))
11 print ("initial guess is :", initial_guess)
12 guess_c = num2CNFcode(initial_guess, numdigit)
13
```

- Initializing SAT solver and adding general constraints.

```
14 clauses=[]
15 num = Minisat22()
16 #each position should have at least a number
17 for r in range(numdigit):
18     num.add_clause([digit(n,r) for n in range(10)])
19 #A position can not have two value
20 for r in range(numdigit):
21     for n in range(10):
22         for n1 in range(10):
23             if n!=n1:
24                 num.add_clause([-digit(n,r), -digit(n1,r)])
25
```



Solving Procedure

- Using a WHILE loop to find the solutions.
- First of all, I will check the number of coincidences of guess and secret code:

```
STOP = False
```

```
while STOP==False :  
    # find the number of correct digit  
    num_correct= 0  
    for i in guess_c:  
        if i in code_c:  
            num_correct = num_correct+1  
    if num_correct==numdigit:  
        break  
    print("Method's guess is : {}, {} correct".format(decodify(guess_c),num_correct))  
    --
```

Solving Procedure

- Updating the SAT solver model and finding new guess:

```
num_correct= 0
for i in guess_c:
    if i in code_c:
        num_correct = num_correct+1
if num_correct==numdigit:
    break
print("Method's guess is : {}, {} correct".format(decodify(guess_c),num_correct))
new_guess = []
if num_correct >= 0:
    cnf_least = list(itertools.combinations(guess_c,len(guess_c) - int(num_correct) + 1))
    guess_neg = [ -x for x in guess_c]
    cnf_most = list(itertools.combinations(guess_neg, 1 + int(num_correct)))
    for x,y in zip(cnf_least, cnf_most):
        num.add_clause(list(x))
        num.add_clause(list(y))

elif num_correct == len(guess_c):
    STOP=True

# solving SAT solver
num.solve()
s=num
for m in s.enum_models():
    new_guess = [x for x in m if x>0 ]
    break
#Updating for new loop
guess_c = new_guess
```



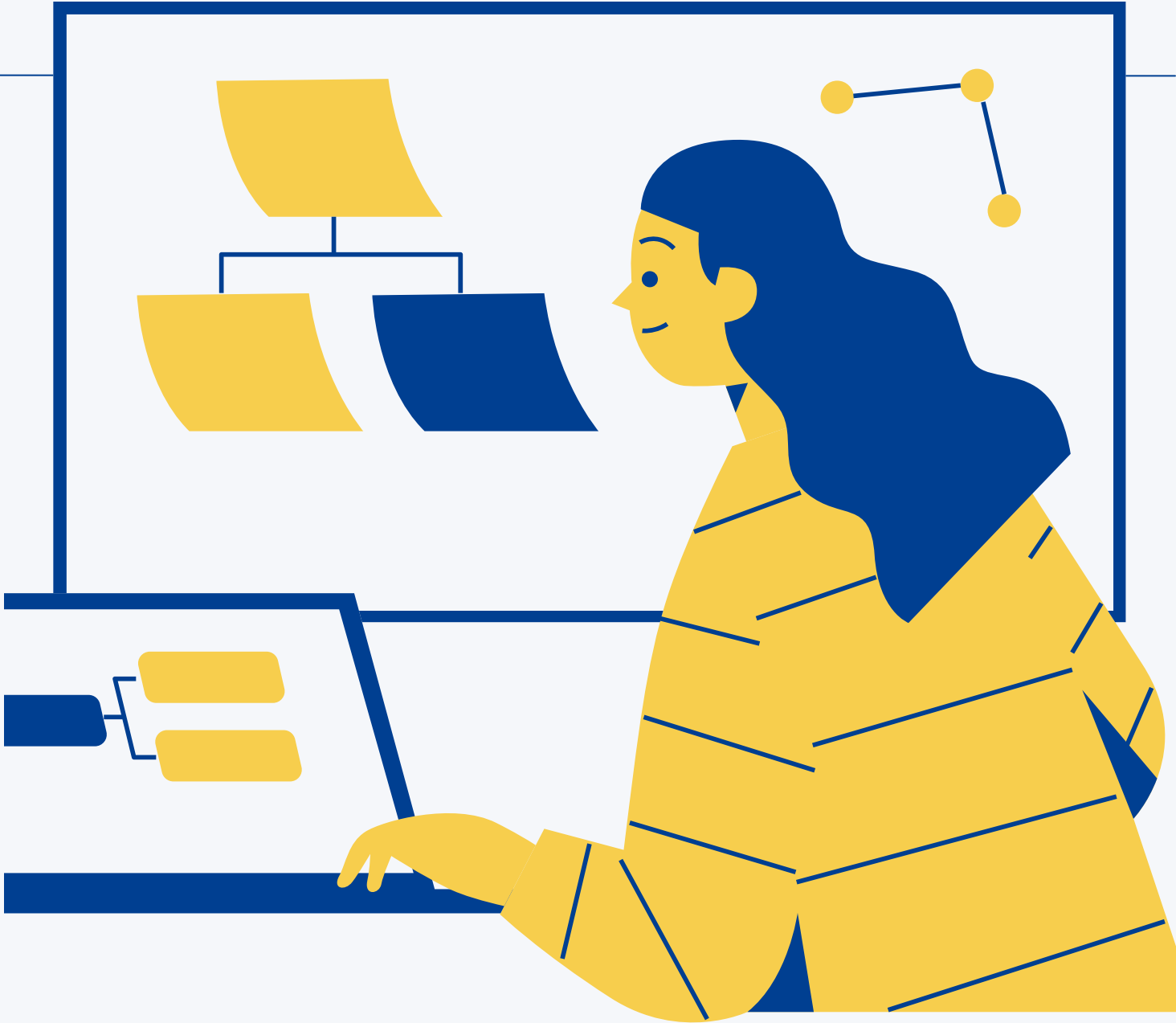
Chapter III

Results

Number Mind Game

- Average running time for 5 times

| Length | Time (ms) | Length | Time(mS) |
|--------|-----------|--------|-----------|
| 4 | 8.6 | 13 | 124.6 |
| 5 | 11.4 | 14 | 115.8 |
| 6 | 17.6 | 15 | 147.4 |
| 7 | 14 | 16 | 941.4 |
| 8 | 19.6 | 17 | 1857.8 |
| 9 | 19.2 | | |
| 10 | 31 | | |
| 11 | 41.6 | | |
| 12 | 60.6 | | |



21 July 2022

Mathematics Department, University of Padova

Thank you for listening

ADVISOR

Luciano Serafini

STUDENT

Mojtaba Amini
2041518