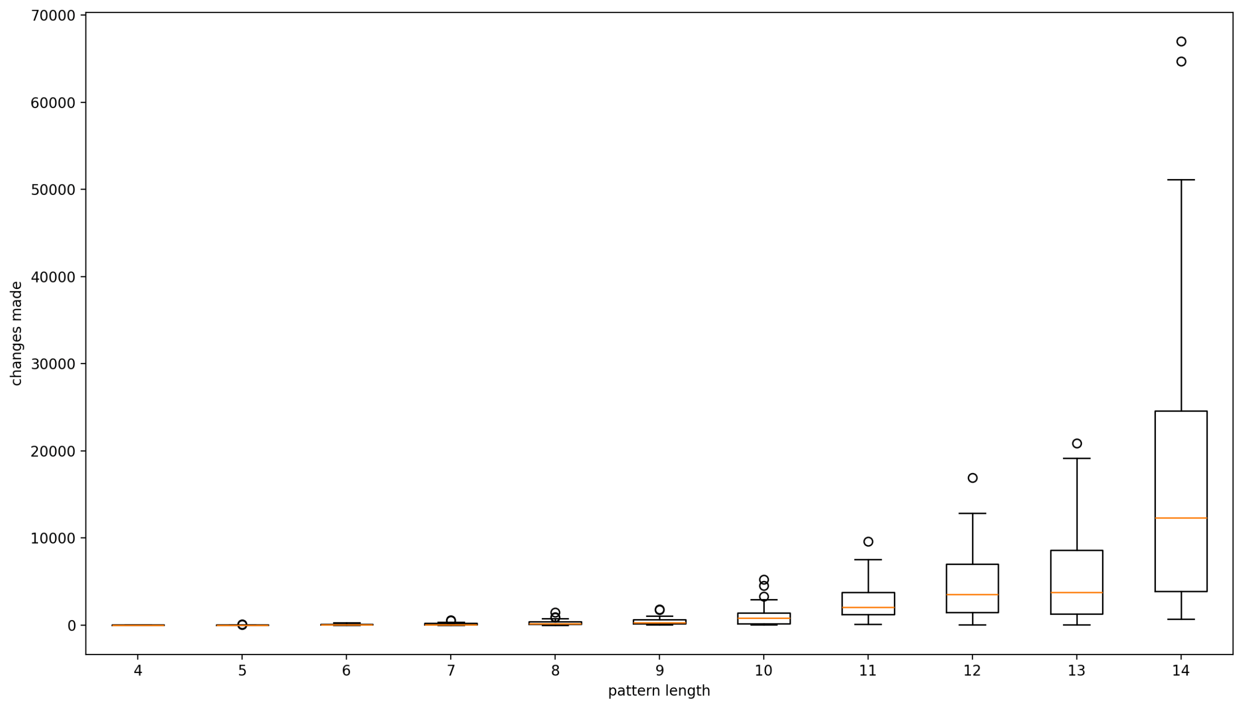
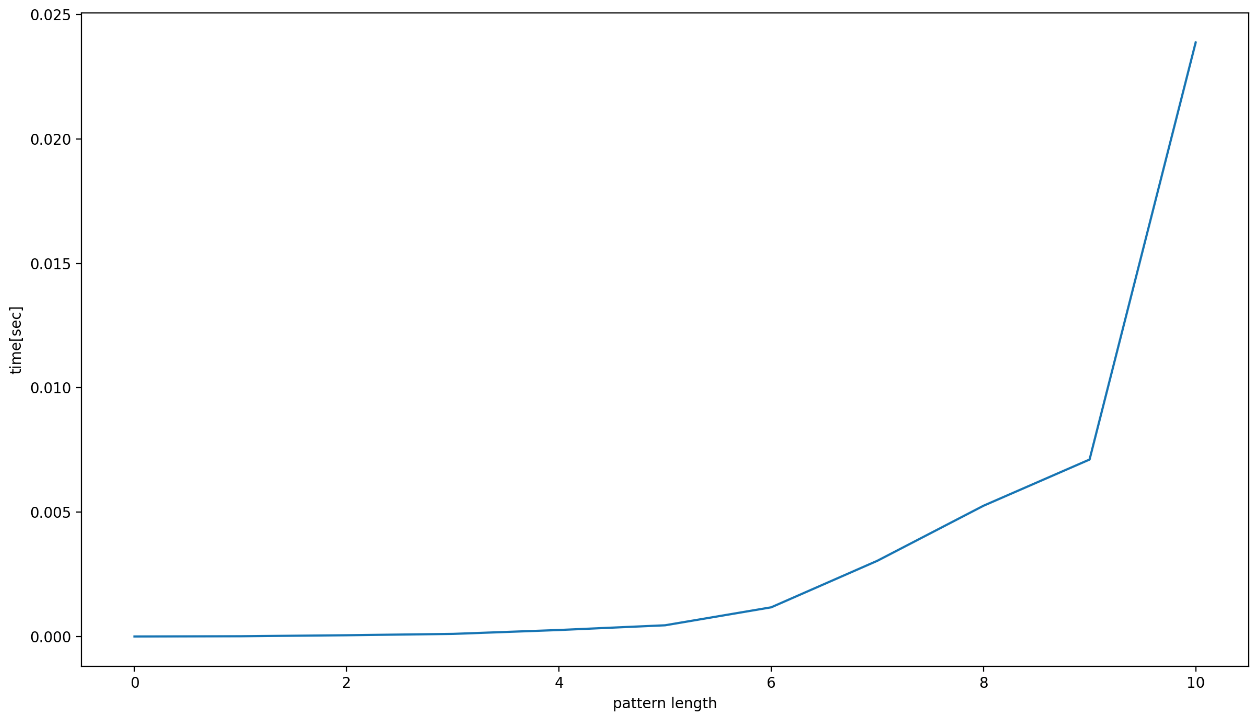
Jan Tacakiewicz
Jędrzej Friska

Evolutionary Algorithms

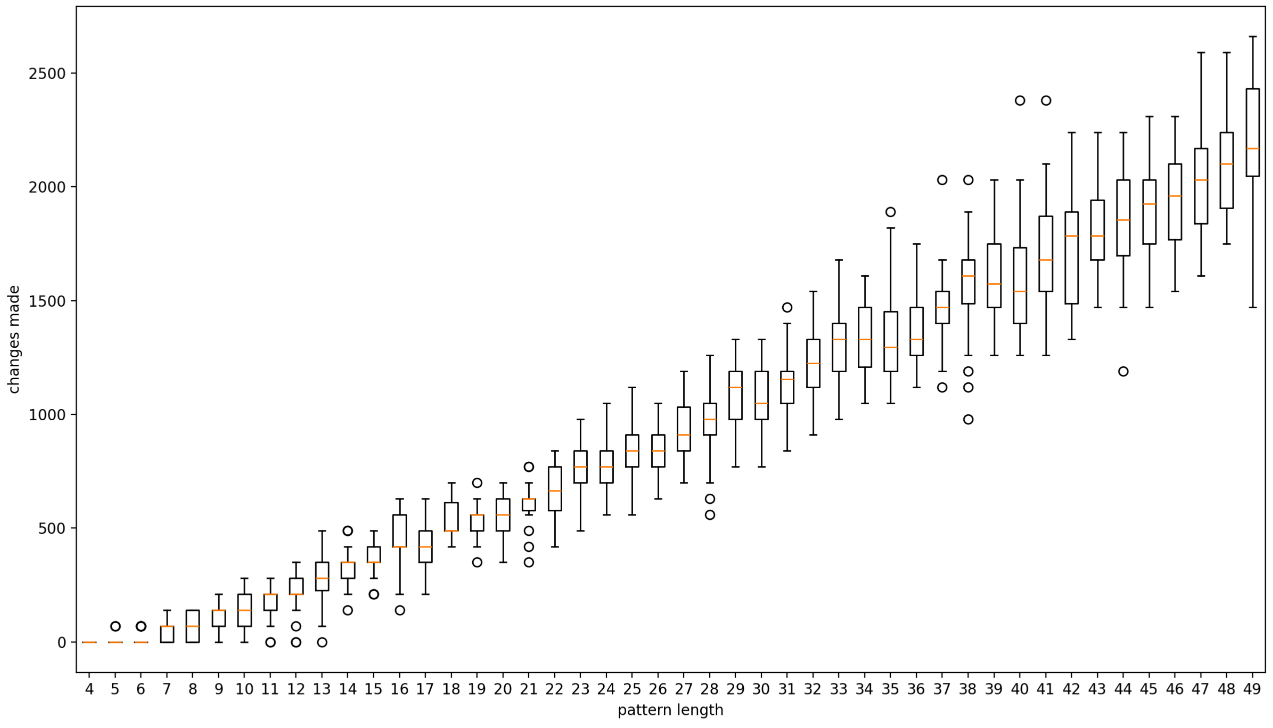
Ex.1

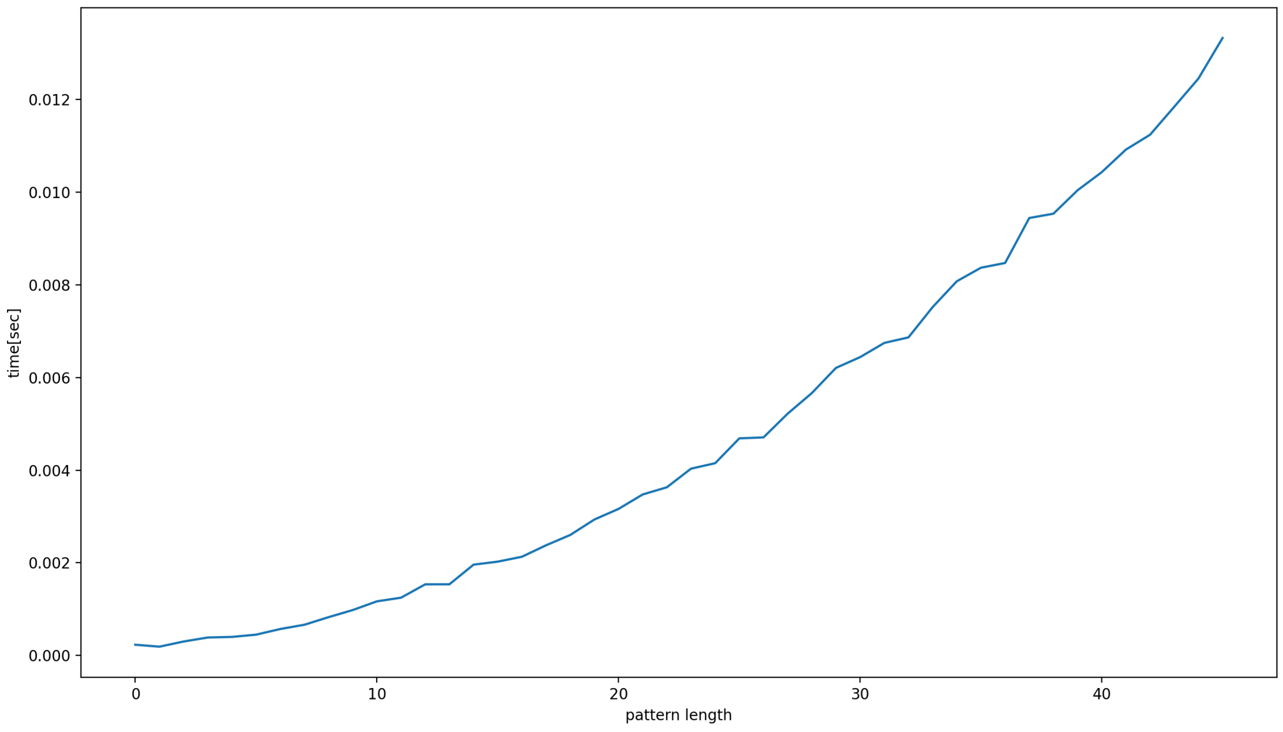
1.1 Random guesses:



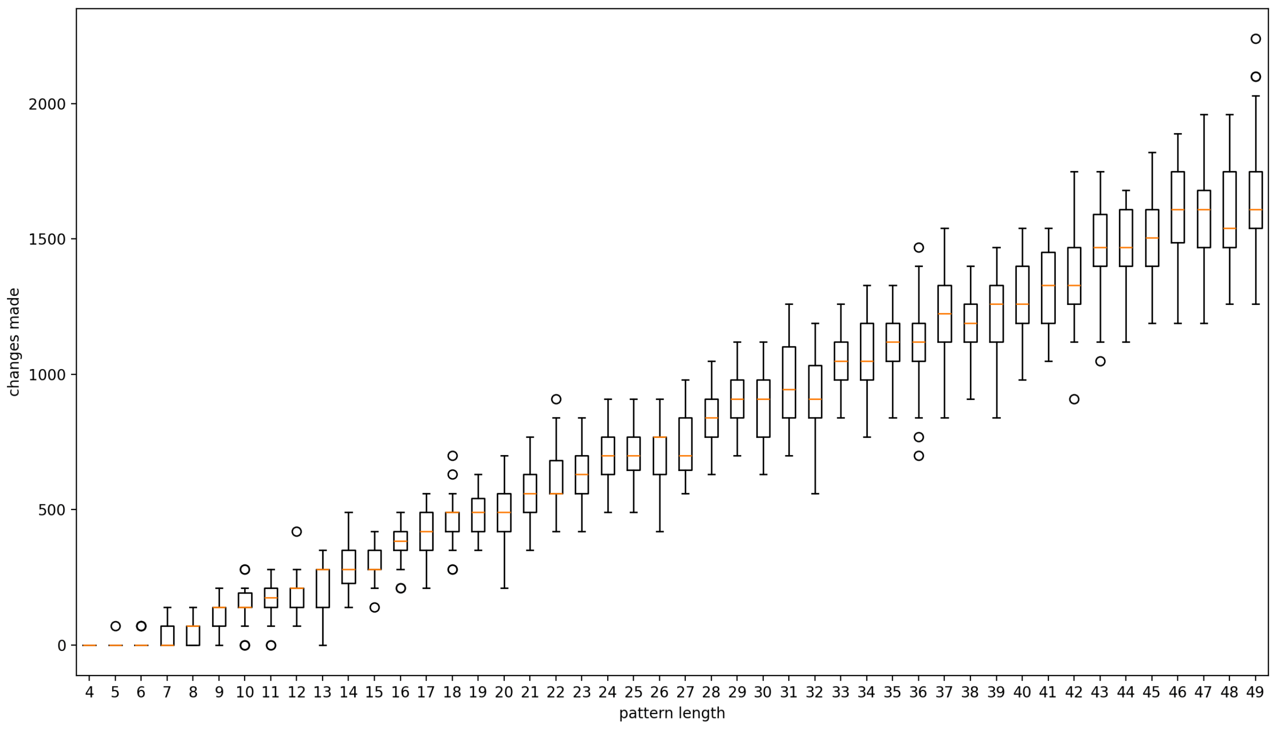


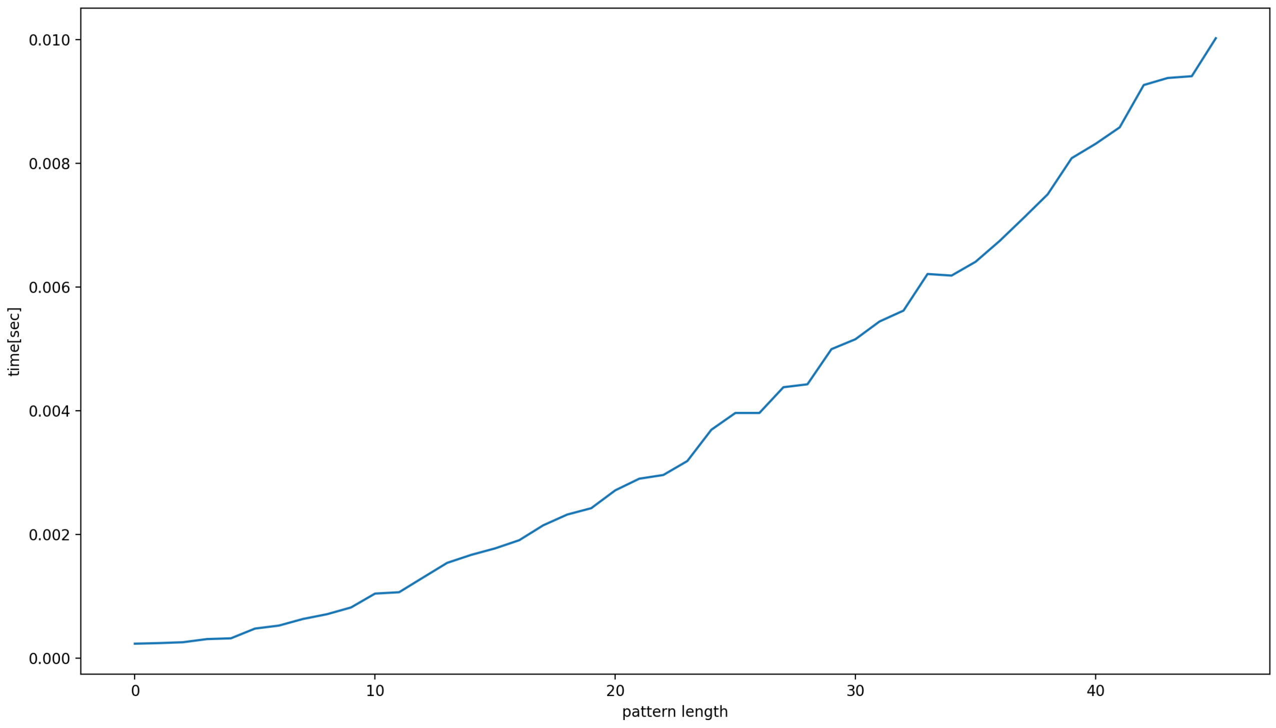
1.2 Mutations:





1.3 Mutations with 10% of the population being generated as crossover:





Ex.2 Explain the changes that would be necessary in the evaluation function, mutation, and crossover to deal with a similar problem where the size of the target bit pattern would be unknown.

The **evaluation** function would have to heavily penalise mismatch of pattern lengths, rising exponentially.

```
def fitness(target, guess):
    min_size = min(len(target), len(guess))
    Fit = sum([1 - abs(target[i] - guess[i]) for i in range(min_size)])
    if len(target) != len(guess):
      fit -= abs(len(target) - len(guess))**2
    return fit
```

Mutation function should also have a small chance to change the patterns length:

```
def mutateOne(pattern):
    i = rnd.randrange(len(pattern))
    pattern[i] = 1 - pattern[I]
    if rnd.random() < size_change_chance:
        pattern += [round(rnd.random())]
```

Crossover would average out both patterns lengths and proceed as normal (making sure that cutoff is within the shorter patterns length):

```
def crossover(guess1, guess2):
    avg_len = int((len(guess1) + len(guess2)) / 2)
    longer = guess1 if len(guess1) > len(guess2) else guess2
    shorter = guess2 if len(guess1) > len(guess2) else guess1
    cutoff = rnd.randrange(len(shorter))
    return shorter[:cutoff] + longer[cutoff:avg_len]
```

Ex.3 Explain the changes that would be necessary in the evaluation function, mutation, and crossover to deal with a similar problem where the pattern would be of decimal digits and not binary.

Jędrzej Friska
We could convert the decimal digits pattern so that each digit is represented by 4 bits and proceed with our ready implementation that takes 0s and 1s as input, but the patterns length would increase by a factor of 4 (4 bits per number). If we had to keep pattern size unchanged:

Evaluation would take in the amount each digit differs from the original pattern, so if the digits are the same, the score for that digit would be

```
def fitness(target, guess):

    return sum([-abs(target[I] - guess[i]) for i in range(len(target))])
```

Mutation would have a chance to either increase random digit by one or decrease random digit by one, not allowing decreasing of ‘0’ or increasing of ‘9’

```
def mutateOne(pattern):

    i = rnd.randrange(len(pattern))

    if pattern[i] == 9:

        pattern[i] = 8

    else if pattern[i] == 0:

        pattern[i] = 1

    Else:

        if random.random() > 0.5:

             pattern[i] += 1

        else:

             pattern[i] -= 1
```

Crossover would not be impacted by this change.