

AI Programming

[Week 13] Practice

2024. 11. 28.



부산대학교
PUSAN NATIONAL UNIVERSITY

- 실습 준비
- 실습 목표
- 과제 안내

HW08 본인 제출 파일 준비

- main.py
- optimizer.py
- problem.py
- setup.py

HW09 파일 준비

- exp.txt
- GA skeleton.py

GA 알고리즘 이해하고 구현하기

- main.py 변경
- Numeric – GA 구현
- Non-Convex 문제 실험
- TSP – GA 설명

main.py 실습 - 파일 I/O 수정하기

1. readPlan 함수 수정
exp.txt에서 GA 수행에 필요한
변수 읽어오기
2. createOptimizer 함수 수정
GA 추가

```
#
# If you are running a metaheuristic algorithm,
#   give the total number of evaluations until temination.
#   Enter the number (limitEval) : 50000
#
# If you are running GA,
#   give the values of the following parameters.
#   Population size (popSize) : 100
#
# If you are using GA for numerical optimization,
#   give the values of the following parameters.
#   Resolution for binary encoding in number of bits (resolution) : 10
#   Swap probability for uniform crossover (uXp) : 0.2
#   Multiplication factor to 1/n for mutation (mrF) : 1
#
# If you are using GA for TSP,
#   give the values of the following parameters.
#   Crossover rate (XR) : 0.1
#   Mutation rate (mR) : 0.9
#
# Enter the total number of experiments
#   Enter the number (numExp) : 2
```

main.py 실습 – 파일 I/O 수정하기 (3분)

```
def readPlan():
    fileName = input("Enter the file name of experimental setting: ")
    infile = open(fileName, 'r')

    ##### GA를 위한 parameter 추가하기; start #####
    parameters = { 'pType':0, 'pFileName':'', 'aType':0, 'delta':0,
                   'limitStuck':0, 'alpha':0, 'dx':0, 'numRestart':0,
                   'limitEval':0, 'numExp':0 }
    ##### GA를 위한 parameter 추가하기; end #####
    ...
```

GA 관련 변수 - popSize, resolution, uXp, mrF, XR, mR

```
def createOptimizer(parameters):
    # GA optimizer 추가하기
```

optimizer.py 실습 – GA interface 추가해주기 (3분)

1. GA skeleton 코드를 optimizer.py에 추가
2. exp.txt에서 Search algorithm GA로 변경
3. main.py – readPlanAndCreate() 실행 후 생성된 alg(GA)에 변수가 제대로 할당되는지 확인(디버깅)

```
# Select the search algorithm:
# Hill Climbing algorithms:
#   1. Steepest-Ascent
#   2. First-Choice
#   3. Stochastic
#   4. Gradient Descent
# Metaheuristic algorithms:
#   5. Simulated Annealing
#   6. GA
Enter the number (aType) : 6
```

```
5 def main():
6     p, alg = readPlanAndCreate() # Setup and create (problem, algorithm)
7     conductExperiment(p, alg)   # Conduct experiment & produce results
```

```
▼ alg = <optimizer.GA object at 0x0000...
> special variables
> function variables
 XR = 0.1
 aType = 6
 alpha = 0.01
 delta = 0.01
 dx = 0.0001
 limitEval = 50000
```

```
_mR = 0.9
_mrF = 1
_numExp = 2
_pC = 0.1
_pM = 0.9
_pType = 2
_popSize = 100
_uXp = 0.2
_whenBestFound = 0
```

optimizer.py 실습 – GA run 구현 (8분)

hint)

Numeric, TSP 문제 유형에 따라 함수 구현을 달리해야 하는 경우, **problem.method** 처럼 구현

공통적으로 쓰일 수 있는 경우, **GA.method**로 구현

전체 흐름을 이해하고 흐름을 따라가며 필요한 함수 구현 실습

*실습은 Numeric으로 수행, Tsp 구현은 과제에서..

```
def run(self, p):
    # Population 생성
    pop = p.initializePop(self._popSize)
    # Population 중 최적해 찾기
    best = self.evalAndFindBest(pop, p)
    numEval = p.getNumEval()
    whenBestFound = numEval
    # limitEval 까지 [다음세대 생성-평가] 반복
    while numEval < self._limitEval:
        newPop = []
        I = 0
        # 다음 세대 생성; start
        while I < self._popSize:
            par1, par2 = self.selectParents(pop)
            ch1, ch2 = p.crossover(par1, par2, self._pC)
            newPop.extend([ch1, ch2])
            I += 2
        newPop = [p.mutation(ind, self._pM) for ind in newPop]
        pop = newPop
        # 다음 세대 생성; end
        # 다음 세대 값 평가 및 best 업데이트
        newBest = self.evalAndFindBest(pop, p)
        numEval = p.getNumEval()
        if newBest[0] < best[0]:
            best = newBest
            whenBestFound = numEval
        self._whenBestFound = whenBestFound
        bestSolution = p.indToSol(best)
        p.storeResult(bestSolution, best[0])
```


optimizer.py 실습 – GA run – Problem.InitializePop 구현 (8분)

```
def initializePop(self, size):
    pop = []
    for i in range(size):
        chromosome = self.randBinStr()
        pop.append([0, chromosome])
    return pop

def randBinStr(self):
    # Numeric 문제의 변수 N (self._domain[0]) 개에 대해서,
    # 각 변수 별 self._resolution 크기의 random binary 생성
    # N=5, self._resolution=10 이라면,
    # 50길이의 [1, 0, 1, 0, 0, 1, 0, 1, ...] 배열 생성하여 반환
    return chromosome
```

resolution이 의미하는 것은 각 변수가 가질 수 있는 값의 해상도,
만약 resolution이 8bit라면 $-30 < x < 30$ 범위 사이를 2^8 개로 잘라서 표현할 수 있음
Resolution이 높을수록 소수점 자리를 더 세밀하게 표현

optimizer.py 실습 – GA run – self.evalAndFindBest 구현 (8분)

Optimizer.py – GA class

```
def evalAndFindBest(self, pop, p):  
    best = pop[0]  
    p.evalInd(best)  
    bestValue = best[0]  
    for i in range(1, len(pop)):  
        p.evalInd(pop[i])  
        newValue = pop[i][0]  
        if newValue < bestValue:  
            best = pop[i]  
            bestValue = newValue  
    return best
```

Problem.py – Numeric Class

```
def binaryToDecimal(self, binCode, l, u):  
    # binCode [0, 1, 0, 1, ...]을  
    # low, upper bound내의 값으로 표현해서 return  
    # l=0, u=10, binCode=[x, x, x, x] 일 때,  
    # [0,0,0,0]=0, [0,0,0,1]=0.625, [0,0,1,0]=1.25  
    # [0,0,1,1]=1.875, ..., [1,1,1,1]=9.375  
    # u가 포함 안되는데, binCode가 충분히 길다는 가정  
    하에 포함 여부는 구현에 큰 문제없음
```

Problem.py – Numeric Class

```
def evalInd(self, ind):  
    ind[0] = self.evaluate(self.decode(ind[1]))
```

[1, 0, 1, 0, 0, 1, ...] 로 표현된 값을 evaluate할 수 있도록 실수로 변환하는 decode 함수 필요

Problem.py – Numeric Class

```
def decode(self, chromosome):  
    r = self._resolution  
    low = self._domain[1]  
    up = self._domain[2]  
    genotype = chromosome[:]  
    phenotype = []  
    start = 0  
    end = r  
    for var in range(len(self._domain[0])):  
        value = self.binaryToDecimal(genotype[start:end],  
                                     low[var], up[var])  
        phenotype.append(value)  
        start += r  
        end += r  
    return phenotype
```

optimizer.py 실습 – GA run – self.selectParents 구현 (5분)

```
def selectParents(self, pop):
    ind1, ind2 = self.selectTwo(pop)
    par1 = self.binaryTournament(ind1, ind2)
    ind1, ind2 = self.selectTwo(pop)
    par2 = self.binaryTournament(ind1, ind2)
    return par1, par2

def selectTwo(self, pop):
    # pop에서 random하게 2개의 individuals 선택해서 반환

def binaryTournament(self, ind1, ind2):
    # 2개의 individuals 중 더 좋은 ind 선택해서 반환
```

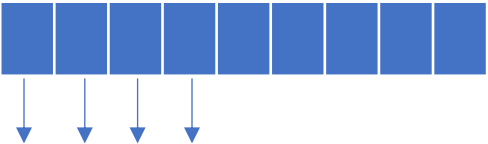
optimizer.py 실습 – GA run – p.crossover 구현 (5분)

```
def crossover(self, ind1, ind2, uXp):  
    chr1, chr2 = self.uXover(ind1[1], ind2[1], uXp)  
    return [0, chr1], [0, chr2]  
  
def uXover(self, chrInd1, chrInd2, uXp): # uniform crossover  
    # chrInd1, chrInd2의 각 원소를 확률적(uXp)으로 crossover  
    chr1 = chrInd1[:] # Make copies  
    chr2 = chrInd2[:] # implement  
    return chr1, chr2
```



optimizer.py 실습 – GA run – p.mutation 구현 (5분)

```
def mutation(self, ind, mrF): # bit-flip mutation
    # mrF * (1/ length of individual) 확률로 ind의 개별 원소 bit-flip
    child = ind[:] # Make copy
    # implement
    return child
```

child  개별 원소마다 $mrF * (1/\text{length of ind})$ 확률로 flip

optimizer.py 실습 – GA run – problem.indToSol 구현

```
def indToSol(self, ind):  
    return self.decode(ind[1])
```

optimizer.py 실습 – GA – Numeric 수행해보기 (5분)

실험 값 변경해가며 GA – Numeric – Ackley.txt 문제 수행해보기

Best Value가 0에 가깝게 나와야 함

Average Iteration of Finding the best가 작게 나오게 효율적인 값 찾기

```
# If you are running a metaheuristic algorithm,  
# give the total number of evaluations until temination.
```

```
Enter the number (limitEval) : 1000
```

```
# If you are running GA,  
# give the values of the following parameters.
```

```
Population size (popSize) : 100
```

```
# If you are using GA for numerical optimization,  
# give the values of the following parameters.
```

```
Resolution for binary encoding in number of bits (resolution) : 10  
Swap probability for uniform crossover (uXp) : 0.2  
Multiplication factor to 1/n for mutation (mrF) : 1
```

```
# If you are using GA for TSP,  
# give the values of the following parameters.
```

```
Crossover rate (XR) : 0.1
```

```
Mutation rate (mR) : 0.9
```

```
# Select the problem type:
```

```
# 1. Numerical Optimization
```

```
# 2. TSP
```

```
Enter the number (pType) : 1
```

```
# Enter the name of the file : problem/Convex.txt
```

```
# Enter the name of the file : problem/Griewank.tx
```

```
Enter the name of the file : problem/Ackley.txt
```

```
# Enter the name of the file : problem/tsp30.txt
```

```
# Enter the name of the file : problem/tsp50.txt
```

```
# Enter the name of the file : problem/tsp100.txt
```

```
# Select the search algorithm:
```

```
# Hill Climbing algorithms:
```

```
# 1. Steepest-Ascent
```

```
# 2. First-Choice
```

```
# 3. Stochastic
```

```
# 4. Gradient Descent
```

```
# Metaheuristic algorithms:
```

```
# 5. Simulated Annealing
```

```
# 6. GA
```

```
Enter the number (aType) : 6
```

GA TSP 실습 – initializePop (3분)

```
def initializePop(self, size):  
    n = self._numCities  
    pop = []  
    for i in range(size):  
        # tsp.randomInit 메서드 이용하여 chromosome 생성하고 pop에 추가  
        # chromosome = [eval_value, [tour order]]  
        # = [0, [5, 12, 17, 11, 7, 22, ...]]  
  
    return pop
```


GA TSP 실습 – evalInd, crossover

```
def evalInd(self, ind):  
    ind[0] = self.evaluate(ind[1])  
  
def crossover(self, ind1, ind2, XR):  
    if random.uniform(0, 1) <= XR:  
        chr1, chr2 = self.oXover(ind1[1], ind2[1])  
    else:  
        chr1, chr2 = ind1[1][:], ind2[1][:]  
    return [0, chr1], [0, chr2]
```

TSP에서도 GA 알고리즘이 정상적으로 수행될 수 있도록 구현하기

- tsp.oXover 구현
- tsp.mutation 구현

oXover 설명

1. ind 길이 사이의 두 값 a, b 생성

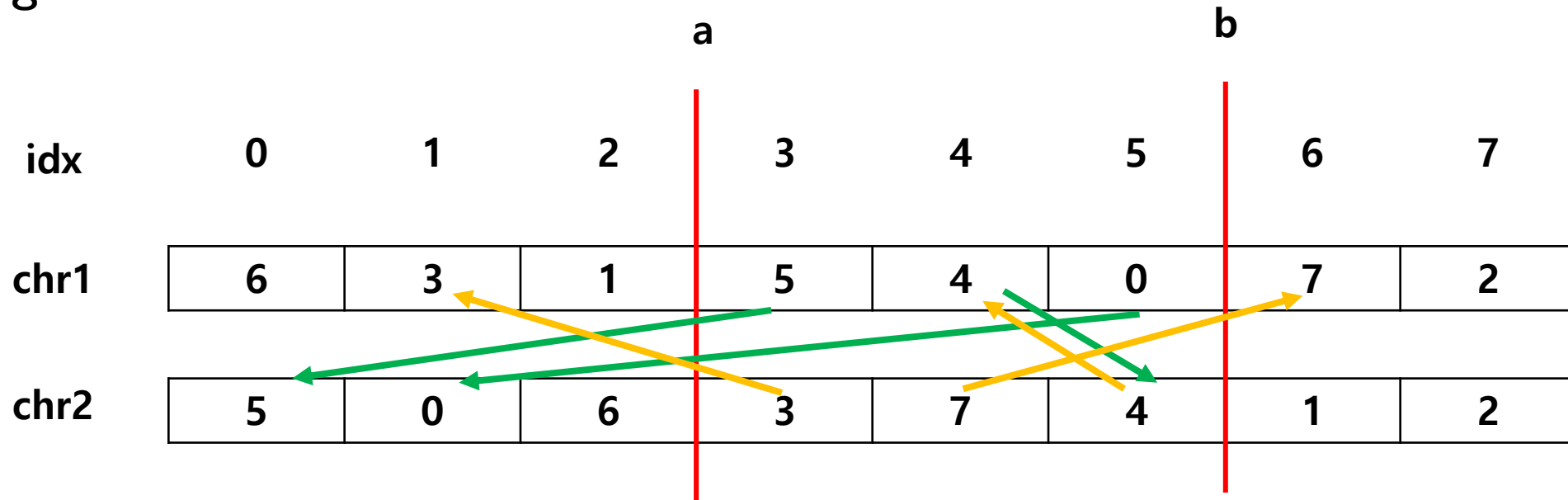
$\text{Min_individual_length} < a < b < \text{Max_individual_length}$
예제에서는 ind 길이가 8이고, a, b를 각각 2, 5라고 가정

2. 각 chr에서 해당 부분에 대응하는 원소 찾기

- chr1에서는 index a~b 사이 값이 [5, 4, 0]
- Chr2에서는 index a~b 사이 값이 [3, 7, 4]

			a			b		
idx	0	1	2	3	4	5	6	7
chr1	6	3	1	5	4	0	7	2
chr2	5	0	6	3	7	4	1	2

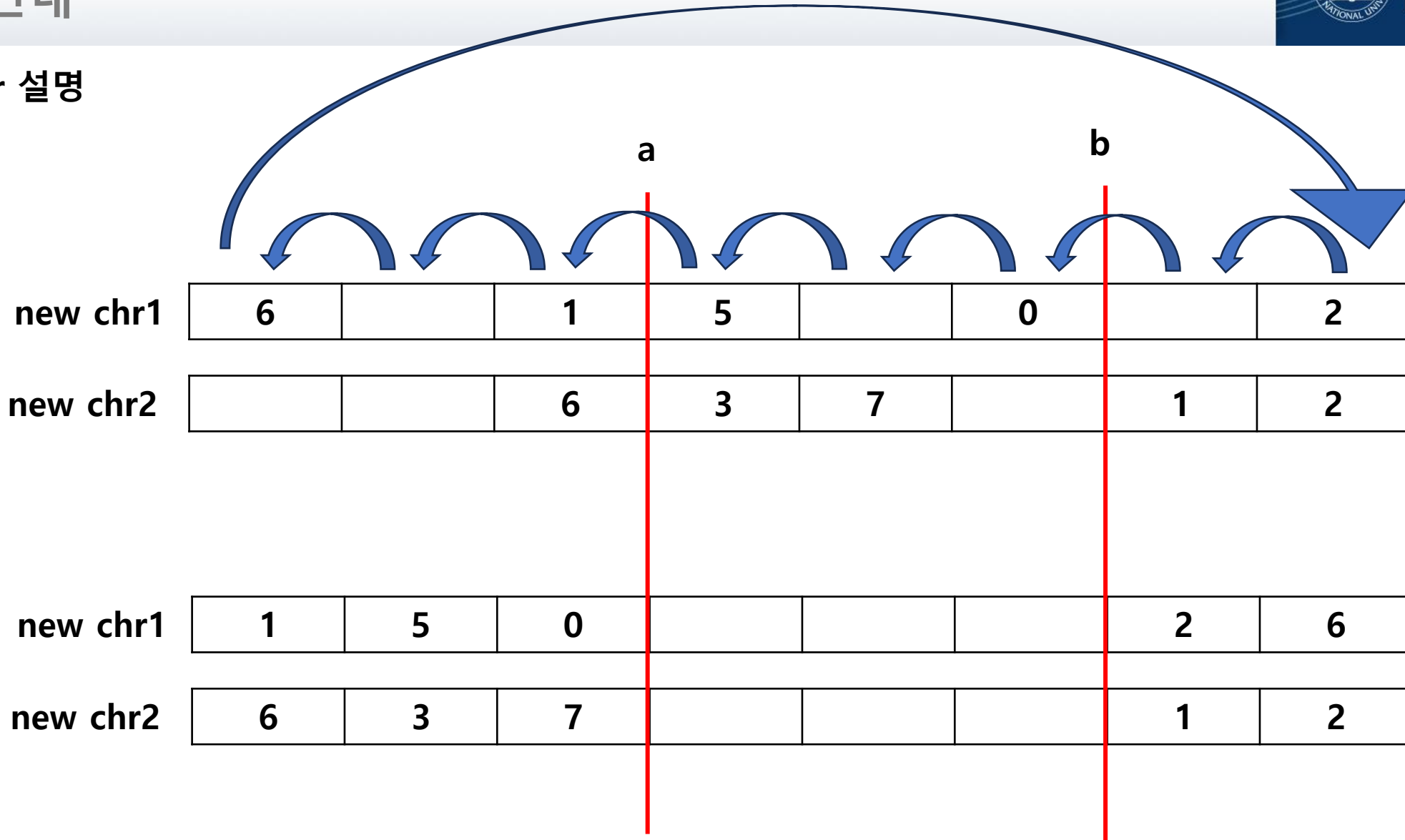
oXover 설명



3. 사이 값과 같은 값을 상대 individual에서 삭제

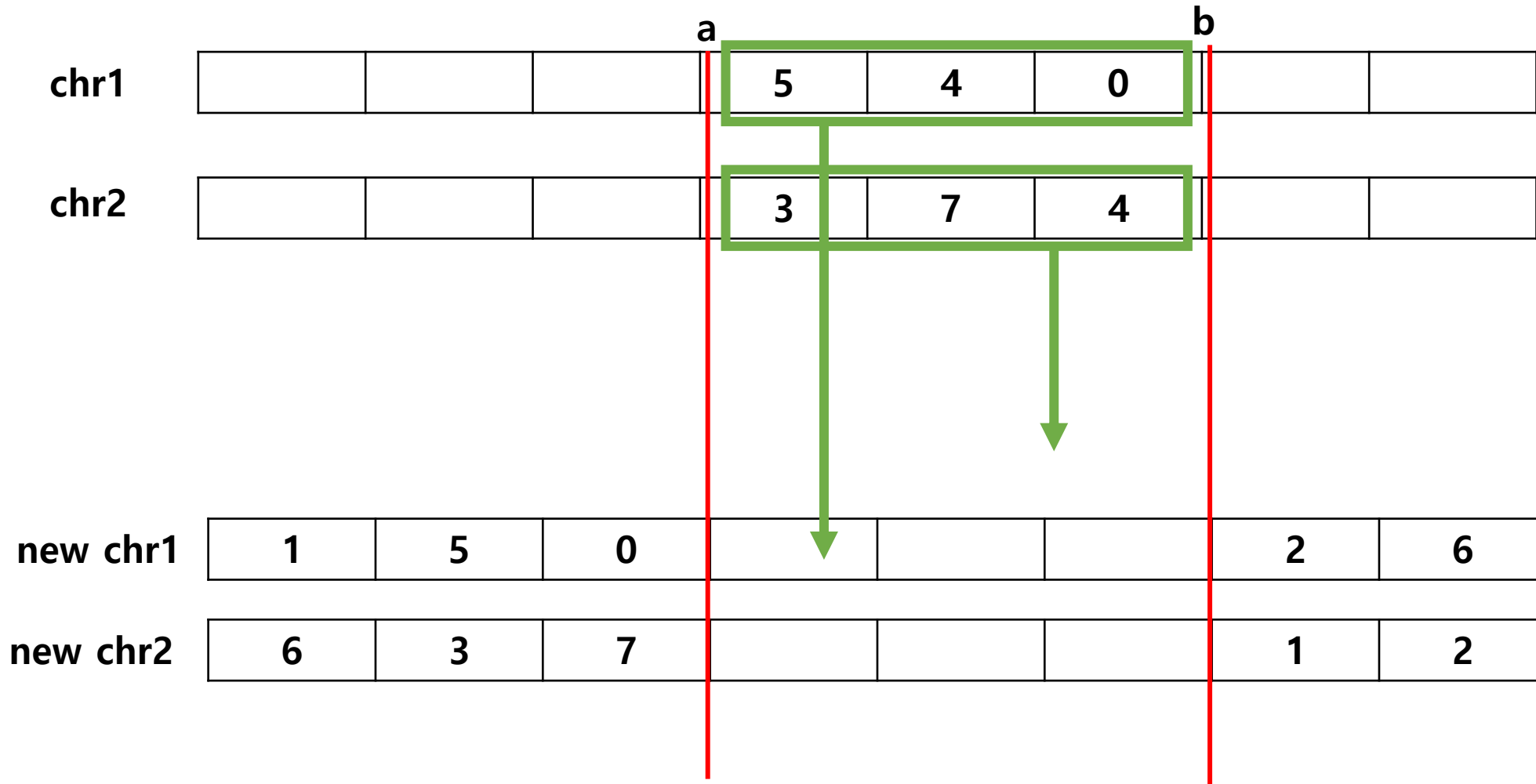
new chr1	6		1	5		0		2
new chr2			6	3	7		1	2

oXover 설명



4. 각 원소를 계속해서 left shift하며 a:b 사이가 완전히 비워지게 구성

oXover 설명



5. 처음 찾았던 chr1의 사이값 [5,4,0]을 new chr2에 대입, chr2도 동일하게

mutation 설명

```
def mutation(self, ind, mR):  
    # mR 확률로 inversion 수행  
    # ind 길이 사이의 수 두 i,j (where i < j) 선정  
    # self.inversion 메소드 활용하여 inversion  
  
    child = ind[:] # Make copy  
    # implement  
    return child
```

각 알고리즘으로 **Numeric, TSP 문제 모두** 수행, **최적의 parameter**를 찾고 해당 파라미터로 실행시켰을 때 **결과값**을 표로 작성해서 리포트에 첨부

		Convex		Griewank		Ackley		Tsp30	...
Steepest Ascent	Delta = 0.01	0.0	774,692	0.260	67,144	17.832	12,182		
		0.0		0.108		14.029			
First Choice	Delta = 0.01 Limitstuck =1000	Avg objective Value	Avg Num of evaluations			
		Best Objective Value	Avg iteration of find best solution(GA/SA)						
...	...								
GA	Resolution = 10 uXp = 0.2 ...	3.920	500,000	0.036	500,000	0.214	500,000		
		0.766	227,140	0.015	220,390	0.141	173,900		

제출물:

파이썬 파일 총 4개를 **HW09_NAME** 폴더로 묶어서 **압축**하여 제출 (.zip)

- main.py
- optimizer.py
- problem.py
- setup.py

리포트 제출 (.pdf)

- 모든 알고리즘을 이용하여 문제 모두(총 6문제)를 최적화
- 수행한 알고리즘의 파라미터 및 결과를 반드시 리포트에 작성**
- 각 알고리즘 비교 결과 및 문제별 성능에 대한 상세한 해석과 결론 작성**