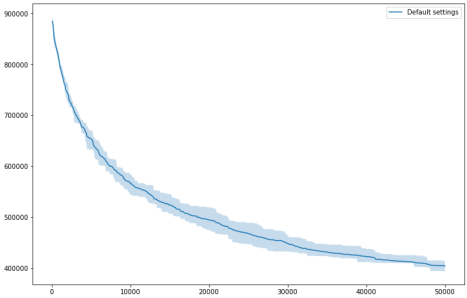
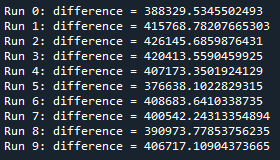
Assignment 8

1) Experiment with the traveling salesman problem and let me know, what you tried and what was the results. You have a few possibilities.

* Try to create another crossover and compare it to the default version.
* Try to make another mutation.
* Try to change the fitness function, 1/path length may not be the best one (it does not matter if you use the tournament selection)

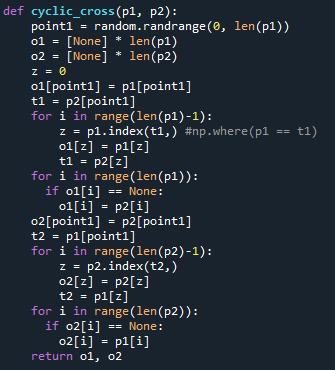
**Results without making any changes**:



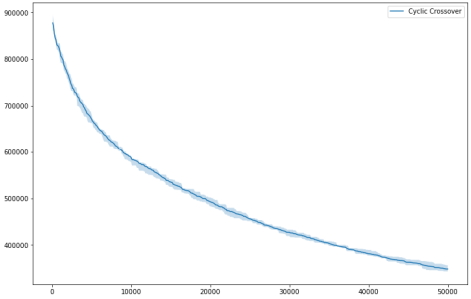


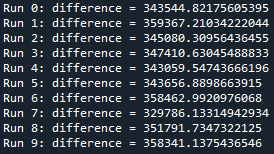
**With Cyclic Crossover**:

Cyclic\_cross code:



Output obtained:

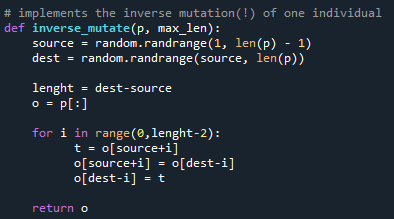




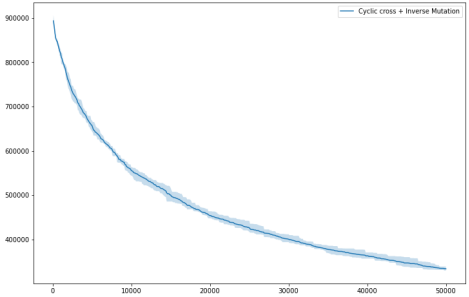
Least difference obtained was -> **329786.13314942934**

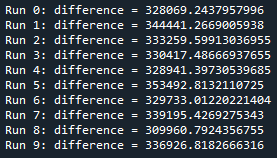
**With Cyclic Crossover & Inverse Mutation**:

Inverse Mutation code:



Output obtained:





Least difference obtained was -> **309960.7924356755**

**Tried to implement PMX, it worked with lecture example, but failed when implementing on the code:**

Code:

def partially\_mapped\_cross(p1, p2):

point1 = random.randrange(1, len(p1))

point2 = random.randrange(1, len(p1))

start = min(point1, point2)

end = max(point1, point2)

o1 = [None] \* len(p1)

o2 = [None] \* len(p2)

for i in range(start,end+1):

o1[i] = p2[i]

for i in range(len(p1)):

if o1[i] == None:

count = 0

for j in range(start,end+1):

if p1[i] == o1[j]:

break

else:

count+=1

s = end+1-start

if count == s and o1[i] == None:

o1[i] = p1[i]

z = o1.count(None)+1

o = [None]\*z

s1 = len(p1)-1

v = 0

#print(z, o1, o1.index(o1[8]))

u=0

if o1[s1] != None:

u = o1[s1]

#print('U->',u)

for j in range(1,len(p1)+1):

c = 0

for k in range(s1):

#print('S1->', k, o1[k])

if o1[-1] == o1[k] and o1[-1] != None:

c-= 7

print('o1[k] != None')

if o1[k] == j:

break

else: c+=1

#print(s1, c, j)

if s1 == c and u!=j: #and o1[k] != None:

#print('V->', v)

o[v] = j

v+=1

w = 0

for i in range(len(p1)):

if o1[i] == None:

o1[i] = o[w]

w+=1

for i in range(start,end+1):

o2[i] = p1[i]

for i in range(len(p2)):

if o2[i] == None:

count = 0

for j in range(start,end+1):

if p2[i] == o2[j]:

break

else:

count+=1

s = end+1-start

if count == s and o2[i] == None:

o2[i] = p2[i]

z = o2.count(None)+1

ot = [None]\*z

s2 = len(p2)-1

v = 0

#print(z, o2, o2.index(o2[8]))

u=0

if o2[s2] != None:

u = o2[s2]

#print('U2->',u)

for j in range(1,len(p2)+1):

c = 0

print('J->',j)

for k in range(s2):

#print('S1->', k, o1[k])

#if o2[-1] == o2[k] and o2[-1] != None:

# c-= 7

# print('o1[k] != None')

if o2[k] == j:

break

else: c+=1

#print(s2, c, j)

#print('J->',j)

if s2 == c and u!=j: #and o1[k] != None:

print('V->', v)

ot[v] = j

v+=1

w = 0

for i in range(len(p2)):

if o2[i] == None:

o2[i] = ot[w]

w+=1

return o1, o2