# Assignment 7: Novelty Search

In our last assignment, we explored the idea of measuring diversity. This week we'll turn it up to eleven, and directly incentivize diversity by playing around with novelty search.

While not quite a deceptive landscape, we'll see how novelty search interacts with a rugged fitness landscape by revisiting our prior work on NK-landscapes from Assignment 3.

In [1]:
```
# imports
import numpy as np
import copy
import matplotlib.pyplot as plt
plt.style.use('seaborn')

import scikits.bootstrap as bootstrap
import warnings
warnings.filterwarnings('ignore') # Danger, Will Robinson! (not a scalable hack, and may s

import scipy.stats # for finding statistical significance

import time
```

Our NK fitness landscape function from Assignment 3.

In [2]:
```
class Landscape:
    """ N-K Fitness Landscape
    """


    def __init__(self, n=10, k=2):
        self.n = n # genome length
        self.k = k # number of other loci interacting with each gene
        self.gene_contribution_weight_matrix = np.random.rand(n,2**(k+1)) # for each gene,

    # find values of interacting loci
    def get_contributing_gene_values(self, genome, gene_num):
        contributing_gene_values = ""
        for i in range(self.k+1): # for each interacing loci (including the location of th
            contributing_gene_values += str(genome[(gene_num+i)%self.n]) # for simplicity
        return contributing_gene_values # return the string containing the values of all l

    # find the value of a partiuclar genome
    def get_fitness(self, genome):
        gene_values = np.zeros(self.n) # the value of each gene in the genome
        for gene_num in range(len(genome)): # for each gene
            contributing_gene_values = self.get_contributing_gene_values(genome, gene_num)
            gene_values[gene_num] = self.gene_contribution_weight_matrix[gene_num,int(cont
        return np.mean(gene_values) # define the fitness of the full genome as the average
```

## Q1: Baseline implementation

Let's copy our usual `Individual` and `evolutionary_algorithm` setup from before. For simplicity in future questions, let's simplify our algorithm as much as possible, working with bit-string (as per the NK fitness function), mutation only (just flipping one bit) and no crossover, and simple truncation selection rather than tournament selection. Like last week, let's also record genotypic diverity over time.

In [3]:
```
class Individual:

    def __init__(self, fitness_function, bit_string_length):
        self.fitness_function = fitness_function
        self.genome = np.random.randint(2, size = bit_string_length)
        self.fitness = 0

    def eval_fitness(self):
        self.fitness = self.fitness_function(self.genome)
```

In [4]:
```
def evolutionary_algorithm(fitness_function=None, total_generations=100, num_parents=10, r
    """ Evolutinary Algorithm (copied from the basic hillclimber in our last assignment)

        parameters:
        fitness_funciton: (callable function) that return the fitness of a genome
                          given the genome as an input parameter (e.g. as defined in Lanc
        total_generations: (int) number of total iterations for stopping condition
        num_parents: (int) the number of parents we downselect to at each generation (mu)
        num_childre: (int) the number of children (note: parents not included in this cour
        bit_string_length: (int) length of bit string genome to be evoloved
        num_elements_to_mutate: (int) number of alleles to modify during mutation (0 = no
        crossover (bool): whether to perform crossover when generating children

        returns:
        fitness_over_time: (numpy array) track record of the top fitness value at each ger
    """

    # initialize record keeping
    solution = None # best genome so far
    solution_fitness = -99999 # fitness of best genome so far
    fitness_over_time = np.zeros(total_generations)
    solutions_over_time = np.zeros((total_generations,bit_string_length))
    diversity_over_time = np.zeros(total_generations)

    # the initialization proceedure
    population = [] # keep population of individuals in a list
    for i in range(num_parents): # only create parents for initialization (the mu in mu+la
        population.append(Individual(fitness_function,bit_string_length)) # generate new i

    # get population fitness
    for i in range(len(population)):
        population[i].eval_fitness() # evaluate the fitness of each parent

    for generation_num in range(total_generations): # repeat

        # the modification procedure
        new_children = [] # keep children separate for now (lambda in mu+lambda)
        while len(new_children) < num_children:

            # inheretance
            [parent1, parent2] = np.random.choice(population, size=2) # pick 2 random pare
            child1 = copy.deepcopy(parent1) # initialize children as perfect copies of the
            child2 = copy.deepcopy(parent2)

            # crossover
            if crossover:
                [crossover_point1, crossover_point2] = sorted(np.random.randint(0,bit_stri
                child1.genome[crossover_point1:crossover_point2+1] = parent2.genome[crossc
                child2.genome[crossover_point1:crossover_point2+1] = parent1.genome[crossc

            # mutation
            for this_child in [child1,child2]:
                elements_to_mutate = set()
                while len(elements_to_mutate)<num_elements_to_mutate:
```

```
                    elements_to_mutate.add(np.random.randint(bit_string_length)) # random
                for this_element_to_mutate in elements_to_mutate:
                    this_child.genome[this_element_to_mutate] = (this_child.genome[this_el

            new_children.extend((child1,child2)) # add children to the new_children list

        # the assessement procedure
        for i in range(len(new_children)):
            new_children[i].eval_fitness() # assign fitness to each child

        # selection procedure
        population += new_children # combine parents with new children (the + in mu+lambda
        population = sorted(population, key=lambda individual: individual.fitness, reverse
        population = population[:num_parents] # perform truncation selection (keep just to

        # record keeping
        if population[0].fitness > solution_fitness: # if the new parent is the best found
            solution = population[0].genome                 # update best solution records
            solution_fitness = population[0].fitness
            solution_generation = generation_num
        fitness_over_time[generation_num] = solution_fitness # record the fitness of the

        solutions_over_time[generation_num,:] = solution

        genome_list = np.array([individual.genome for individual in population])
        diversity = np.mean(genome_list.std(axis=0))
        diversity_over_time[generation_num] = diversity

    return fitness_over_time, solutions_over_time, diversity_over_time
```

Initialize recordkeeping

In [5]:

```
experiment_results = {}
solutions_results = {}
diversity_results = {}
```

## Q1b: Baseline Results

Let's pull all the pieces together and run `20` repitions of `100` generations of a population with `20` parents and `20` children. Let's use a NK-landscape with a bitstring length ( `N` ) of `15` and a highly rugged landscape of `K = 14` . (My repitions take about 1.5 second each)

In [6]:

```
num_runs = 20
total_generations = 100
num_elements_to_mutate = 1
bit_string_length = 15
num_parents = 20
num_children = 20
crossover = False

n = bit_string_length
k = bit_string_length - 1

for run_name in ["fitness_only"]:
    experiment_results[run_name] = np.zeros((num_runs, total_generations))
    solutions_results[run_name] = np.zeros((num_runs, total_generations, bit_string_length
    diversity_results[run_name] = np.zeros((num_runs, total_generations))
    for run_num in range(num_runs):
        landscape = Landscape(n=n, k=k)
        start_time = time.time()
        fitness_over_time, solutions_over_time, diversity_over_time = evolutionary_algorit
```

```
        experiment_results[run_name][run_num] = fitness_over_time
        solutions_results[run_name][run_num] = solutions_over_time
        diversity_results[run_name][run_num] = diversity_over_time
        print(run_name, run_num, time.time()-start_time, fitness_over_time[-1])
```

```
fitness_only 0 1.5843682289123535 0.6700631545436494
fitness_only 1 1.7294938564300537 0.7180342683385563
fitness_only 2 1.747509479522705 0.725422960038818
fitness_only 3 1.6028845310211182 0.7121564712013957
fitness_only 4 1.6969656944274902 0.675897759307328
fitness_only 5 1.724489450454712 0.6873133727738885
fitness_only 6 1.616396188735962 0.7082651283421195
fitness_only 7 1.7054729461669922 0.6830213662021148
fitness_only 8 1.7039716243743896 0.7166178809076704
fitness_only 9 1.5808649063110352 0.7361934383838192
fitness_only 10 1.6879584789276123 0.745936617115449
fitness_only 11 1.727492332458496 0.7190478187699124
fitness_only 12 1.6058869361877441 0.6824006959373967
fitness_only 13 1.782038927078247 0.7293510357706994
fitness_only 14 1.7430055141448975 0.7344847564939153
fitness_only 15 1.753014326095581 0.7469785199745973
fitness_only 16 1.621901273727417 0.6763835665462674
fitness_only 17 1.7525136470794678 0.6842248971630713
fitness_only 18 1.775533676147461 0.6698910792947962
fitness_only 19 1.7174835205078125 0.7047581068169072
```

## Q1c: Plotting

Please plot both the fitness over time and diversity over time of this run.

In [7]:

```
def plot_mean_and_bootstrapped_ci_over_time(input_data = None, name = "change me", x_label
    """

    parameters:
    input_data: (numpy array of shape (max_k, num_repitions)) solution metric to plot
    name: (string) name for legend
    x_label: (string) x axis label
    y_label: (string) y axis label

    returns:
    None
    """

    fig, ax = plt.subplots() # generate figure and axes

    if isinstance(name, str): name = [name]; input_data = [input_data]

    # for this_input_data, this_name in zip(input_data, name):
    for this_name in name:
        print("plotting",this_name)
        this_input_data = input_data[this_name]
        total_generations = this_input_data.shape[1]

        if plot_bootstrap:
            boostrap_ci_generation_found = np.zeros((2,total_generations))
            for this_gen in range(total_generations):
                if this_gen%10==0: print(this_gen)
                boostrap_ci_generation_found[:,this_gen] = bootstrap.ci(this_input_data[:,


        ax.plot(np.arange(total_generations), np.mean(this_input_data,axis=0), label = thi
        if plot_bootstrap:
            ax.fill_between(np.arange(total_generations), boostrap_ci_generation_found[0,:
        ax.set_xlabel(x_label) # add axes labels
```

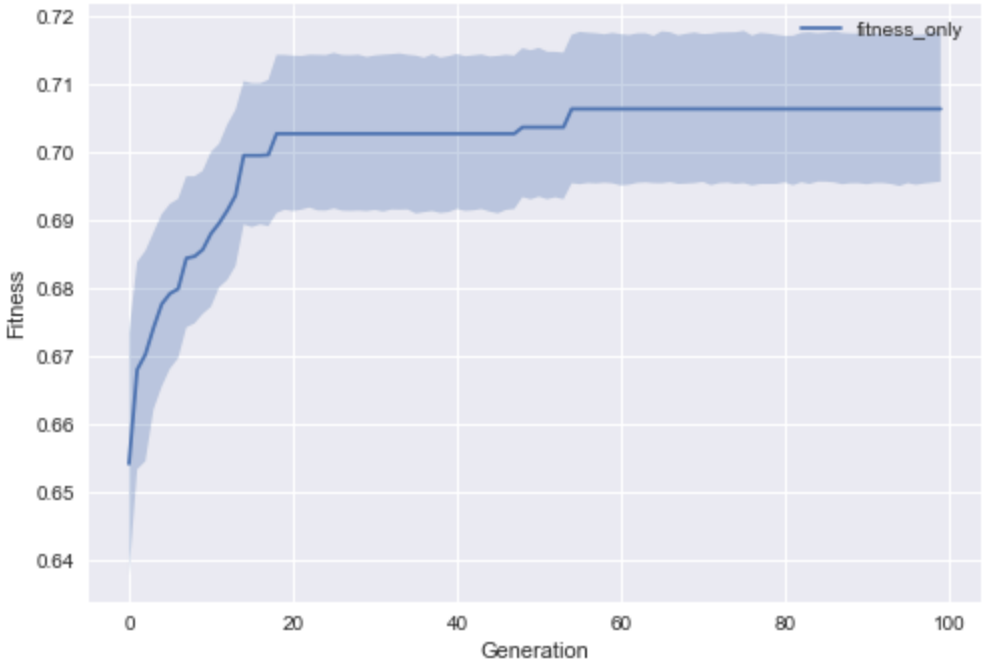
```
        ax.set_ylabel(y_label)
        if y_limit: ax.set_ylim(y_limit[0],y_limit[1])
        plt.legend(loc='best'); # add legend

def plot_mean_and_bootstrapped_ci_multiple(input_data = None, title = 'overall', name = "c
    """

    parameters:
    input_data: (numpy array of numpy arrays of shape (max_k, num_repitions)) solution met
    name: numpy array of string names for legend
    x_label: (string) x axis label
    y_label: (string) y axis label

    returns:
    None
    """

    generations = len(input_data[0])

    fig, ax = plt.subplots()
    ax.set_xlabel(x_label)
    ax.set_ylabel(y_label)
    ax.set_title(title)
    for i in range(len(input_data)):
        CIs = []
        mean_values = []
        for j in range(generations):
            mean_values.append(np.mean(input_data[i][j]))
            CIs.append(bootstrap.ci(input_data[i][j], statfunction=np.mean))
        mean_values=np.array(mean_values)

        print(CIs)
        high = []
        low = []
        for j in range(len(CIs)):
            low.append(CIs[j][0])
            high.append(CIs[j][1])

        low = np.array(low)
        high = np.array(high)

        y = range(0, generations)
        ax.plot(y, mean_values, label=name[i])
        ax.fill_between(y, high, low, alpha=.2)
        ax.legend()
```

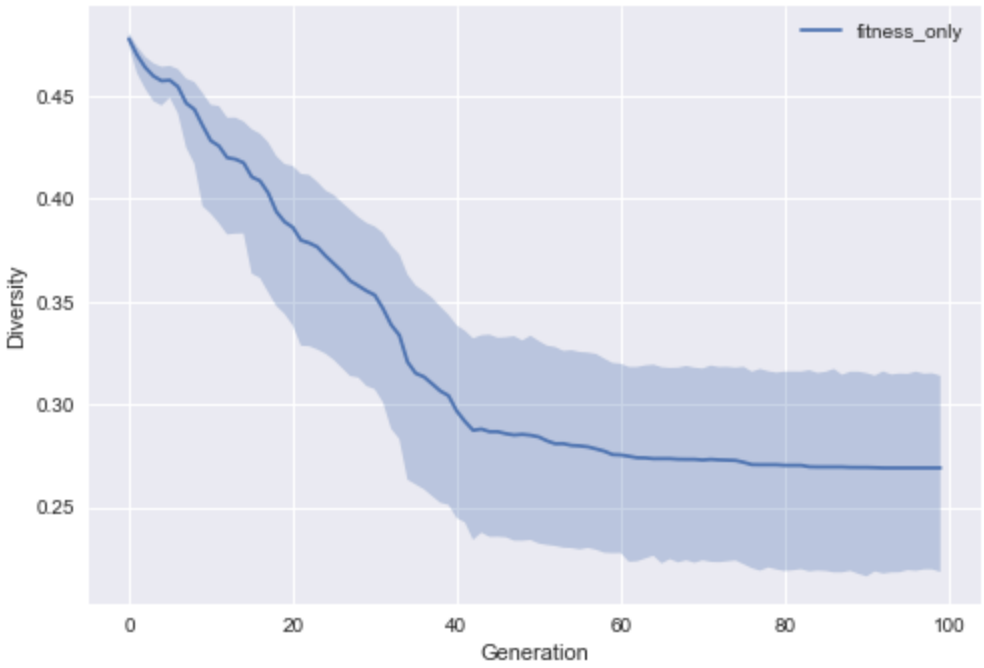
In [8]:

```
# plotting
plot_mean_and_bootstrapped_ci_over_time(input_data = experiment_results, name = ["fitness_
plot_mean_and_bootstrapped_ci_over_time(input_data = diversity_results, name = ["fitness_c
```

```
plotting fitness_only
0
10
20
30
40
50
60
70
80
90
plotting fitness_only
0
10
```

20
30
40
50
60
70
80
90




## Q2: Analysis

What do your results look like? In what ways, if any, do you expect that they'll change if we search for novelty instead of fitness? Why?

**The results look like fitness goes up over time and diversity goes down over time. I expect that this will change with novelty in that the diversity will bounce around but stay within a higher range than the final diversity here. I also expect for the fitness to climb more slowly but steadily over time as the discovery of better solutions are found.**

## Q3: Implementing Novelty

Let's implement novelty search! First, modify your `Individual` class to record `novelty` as an attribute (in additon to `fitness`).

*Hint:* As ususal, you may want to skip ahead to the modification of your `evolutionary_algorithm` function in `Q4` and come back to fill in the helper functions in `Q3` once you have a better idea of when they'll be used (they just appear first to establish defniitions in case you `Restart and Run All`).

In [9]:
```
class Individual:

    def __init__(self, fitness_function, bit_string_length):
        self.fitness_function = fitness_function
        self.novelty_function = get_novelty
        self.genome = np.random.randint(2, size = bit_string_length)
        self.fitness = 0
        self.novelty = 0

    def eval_fitness(self):
        self.fitness = self.fitness_function(self.genome)

    def eval_novelty(self, archive, k):
        if self.novelty == 0:
            self.novelty = self.novelty_function(archive, self, k)
```

## Q3b: Caclulate Novelty

Let's define the novelty of a solution to be the average hamming/euclidean distance (i.e. number of differing bits) between the closest `k` genomes to it within an archive of prior solutions. It may be helpful to define a helper function to calculate this quanity.

In [10]:
```
# Function to find the cross over point (the point before which elements are
# smaller than or equal to x and after which greater than x)
def findCrossOver(arr, low, high, x) :

    # Base cases
    if (arr[high] <= x) : # x is greater than all
        return high

    if (arr[low] > x) : # x is smaller than all
        return low

    # Find the middle point
    mid = (low + high) // 2 # low + (high - low)// 2

    # If x is same as middle element, then return mid
    if (arr[mid] <= x and arr[mid + 1] > x) :
        return mid

    # If x is greater than arr[mid], then either arr[mid + 1] is ceiling of x
    # or ceiling lies in arr[mid+1...high]
    if(arr[mid] < x) :
        return findCrossOver(arr, mid + 1, high, x)

    return findCrossOver(arr, low, mid - 1, x)

# This function prints k closest elements to x in arr[]. n is the number of elements in ar
def getKclosest(solutions, x, k, n) :

    arr = [s.fitness for s in solutions]

#     arr = solutions
```

```
    neighbors = []

    # Find the crossover point
    l = findCrossOver(arr, 0, n - 1, x)
    r = l + 1 # Right index to search
    count = 0 # To keep track of count of elements already printed

    # If x is present in arr[], then reduce left index. Assumption: all elements
    # in arr[] are distinct
    if (arr[l] == x) :
        l -= 1

    # Compare elements on left and right of crossover point to find the k closest elements
    while (l >= 0 and r < n and count < k) :

        if (x - arr[l] < arr[r] - x) :
            neighbors.append(solutions[l])
#             print(arr[l], end = " ")
            l -= 1
        else :
            neighbors.append(solutions[r])
#             print(arr[r], end = " ")
            r += 1
        count += 1

    # If there are no more elements on right side, then print left elements
    while (count < k and l >= 0) :
        neighbors.append(solutions[l])
#         print(arr[l], end = " ")
        l -= 1
        count += 1

    # If there are no more elements on left side, then print right elements
    while (count < k and r < n) :
        neighbors.append(solutions[r])
#         print(arr[r], end = " ")
        r += 1
        count += 1

    return neighbors

def hamming_distance(s1, s2):
    """Calculate the Hamming distance between two bit strings"""
    return sum(c1 != c2 for c1, c2 in zip(s1, s2))


def get_novelty(solution_archive, individual, k):

    # get the k closest neighbors from solution_archive according to fitness
    closest = getKclosest(solution_archive, individual.fitness, k, n=len(solution_archive)
#     hamming_distances = [hamming_distance(individual.genome, s.genome) for s in solutio
#     closest = getKclosest(hamming_distances, 0, k+1, n=len(solution_archive))
    novelty = 0
    novelty = sum([hamming_distance(individual.genome, c.genome) for c in closest])
#     novelty = sum(closest)

    return novelty
```

## Q3c: Selecting for Novelty

Please modify your evolutionary algorithm code to select (again, using the truncation selection as above) for the most novel solutions in our population (according to the novelty metric defined above), regardless of their

fitness.

In order to keep down the cost of computing the distance between a new genome and all those that have previously existed, let's also set a finite size to our novelty archive (as a parameter we can pass to the algorithm). When we trying to add new genomes to the novelty archive, only add the the new individual if it has a higher novelty score than an individual already in the novelty archive (and remove that prior individual from the archive to keeep the archive size the same). Let's also say that the novelty of an individual will be its novelty score when first being considered for additon to the archive (i.e. we do not have to re-calculate it in the future as the makeup of the archive changes).

It may also be helpful to build a helper function for (potentially) updating the archive with a new individual.

Feel free to use the indidivuals in the current generation for your archive calculation or not, whichever is more convenient for your implementation.

*Hint:* If you've sorted the population by novelty for selection, don't forget that the population will no longer be in order of fitness when you go to record the fitness of most fit indidivual for record keeping!

In [11]:
```
def update_archive(solution_archive, individual, max_archive_length):
    if len(solution_archive) == max_archive_length:
        if individual.novelty < sorted(solution_archive, key=lambda i: i.novelty)[0].novel
            return solution_archive
        else:
            solution_archive = sorted(solution_archive, key=lambda i: i.novelty)
            solution_archive[0] = copy.deepcopy(individual)
    else:
        solution_archive.append(copy.deepcopy(individual))

    return solution_archive
```

In [12]:
```
def evolutionary_algorithm(fitness_function=None, total_generations=100, num_parents=10, r
    """ Evolutinary Algorithm (copied from the basic hillclimber in our last assignment)

        parameters:
        fitness_funciton: (callable function) that return the fitness of a genome
                          given the genome as an input parameter (e.g. as defined in Lanc
        total_generations: (int) number of total iterations for stopping condition
        num_parents: (int) the number of parents we downselect to at each generation (mu)
        num_childre: (int) the number of children (note: parents not included in this cour
        bit_string_length: (int) length of bit string genome to be evoloved
        num_elements_to_mutate: (int) number of alleles to modify during mutation (0 = no
        crossover (bool): whether to perform crossover when generating children

        returns:
        fitness_over_time: (numpy array) track record of the top fitness value at each ger
    """

    # initialize record keeping
    solution = None # best genome so far
    solution_fitness = -99999 # fitness of best genome so far
    fitness_over_time = np.zeros(total_generations)
    solutions_over_time = np.zeros((total_generations,bit_string_length))
    diversity_over_time = np.zeros(total_generations)
    solution_archive = []
    max_archive_length = 100

    # the initialization proceedure
    population = [] # keep population of individuals in a list
    for i in range(num_parents): # only create parents for initialization (the mu in mu+la
```

```
    population.append(Individual(fitness_function,bit_string_length)) # generate new i

# get population fitness
for i in range(len(population)):
    population[i].eval_fitness() # evaluate the fitness of each parent
    if len(solution_archive) < max_archive_length:
        solution_archive.append(population[i])
    else:
        population[i].eval_novelty(solution_archive, novelty_k)
        solution_archive = update_archive(solution_archive, population[i], max_archive



for generation_num in range(total_generations): # repeat

    # the modification procedure
    new_children = [] # keep children separate for now (lambda in mu+lambda)
    while len(new_children) < num_children:

        # inheretance
        [parent1, parent2] = np.random.choice(population, size=2) # pick 2 random pare
        child1 = copy.deepcopy(parent1) # initialize children as perfect copies of the
        child2 = copy.deepcopy(parent2)

        # crossover
        if crossover:
            [crossover_point1, crossover_point2] = sorted(np.random.randint(0,bit_stri
            child1.genome[crossover_point1:crossover_point2+1] = parent2.genome[crosso
            child2.genome[crossover_point1:crossover_point2+1] = parent1.genome[crosso

        # mutation
        for this_child in [child1,child2]:
            elements_to_mutate = set()
            while len(elements_to_mutate)<num_elements_to_mutate:
                elements_to_mutate.add(np.random.randint(bit_string_length)) # random
            for this_element_to_mutate in elements_to_mutate:
                this_child.genome[this_element_to_mutate] = (this_child.genome[this_el

        new_children.extend((child1,child2)) # add children to the new_children list

    # the assessement procedure
    for i in range(len(new_children)):
        new_children[i].eval_fitness() # assign fitness to each child
        if len(solution_archive) < max_archive_length:
            solution_archive.append(new_children[i])
        else:
            new_children[i].eval_novelty(solution_archive, novelty_k)
            solution_archive = update_archive(solution_archive, new_children[i], max_a

    # selection procedure
    population += new_children # combine parents with new children (the + in mu+lambda
    for i in range(len(population)):
        population[i].eval_novelty(solution_archive, novelty_k)
    population = sorted(population, key=lambda individual: individual.novelty, reverse
    population = population[:num_parents] # perform truncation selection (keep just to
    for i in range(len(population)):
        if len(solution_archive) < max_archive_length:
            solution_archive.append(population[i])
        else:
            solution_archive = update_archive(solution_archive, population[i], max_arc

    # record keeping
    population = sorted(population, key=lambda individual: individual.fitness, reverse
    if population[0].fitness > solution_fitness: # if the new parent is the best found
        solution = population[0].genome                 # update best solution records
        solution_fitness = population[0].fitness
```

```
            solution_generation = generation_num
        fitness_over_time[generation_num] = solution_fitness # record the fitness of the c

        solutions_over_time[generation_num,:] = solution

        genome_list = np.array([individual.genome for individual in population])
        diversity = np.mean(genome_list.std(axis=0))
        diversity_over_time[generation_num] = diversity

    return fitness_over_time, solutions_over_time, diversity_over_time
```

## Q4: Run

Run novelty search with the same hyperparameter settings as above, and with a `novelty_k` value of `5` nearest neighbors for calculating the novelty metrics, from an archive of the `100` most novel individuals found thus far. (My runs take about 3 seconds for each repitition)

In [13]:

```
num_runs = 20
total_generations = 100
num_elements_to_mutate = 1
bit_string_length = 15
num_parents = 20
num_children = 20
novelty_k = 5
crossover = False

n = bit_string_length
k = bit_string_length - 1

for run_name in ["novelty_k=5"]:
    experiment_results[run_name] = np.zeros((num_runs, total_generations))
    solutions_results[run_name] = np.zeros((num_runs, total_generations, bit_string_length
    diversity_results[run_name] = np.zeros((num_runs, total_generations))
    for run_num in range(num_runs):
        landscape = Landscape(n=n, k=k)
        start_time = time.time()
        fitness_over_time, solutions_over_time, diversity_over_time = evolutionary_algorit
        experiment_results[run_name][run_num] = fitness_over_time
        solutions_results[run_name][run_num] = solutions_over_time
        diversity_results[run_name][run_num] = diversity_over_time
        print(run_name, run_num, time.time()-start_time, fitness_over_time[-1])
```

```
novelty_k=5 0 4.3457536697387695 0.7002817637336534
novelty_k=5 1 4.493380784988403 0.6469157352752016
novelty_k=5 2 4.421819448471069 0.6212326504451838
novelty_k=5 3 4.54392409324646 0.6212042257202249
novelty_k=5 4 4.404804706573486 0.6758338387270231
novelty_k=5 5 4.425823450088501 0.6618340881418373
novelty_k=5 6 4.5249083042144775 0.6302628315446329
novelty_k=5 7 4.374778747558594 0.6789913795811735
novelty_k=5 8 4.502889394760132 0.6884825967949598
novelty_k=5 9 4.361767530441284 0.6204395363191967
novelty_k=5 10 4.60597825050354 0.6613660737133944
novelty_k=5 11 4.386789083480835 0.6647120850166514
novelty_k=5 12 4.604477167129517 0.7274479781661463
novelty_k=5 13 4.4253222942352295 0.7070093989377695
novelty_k=5 14 4.6129841804504395 0.6505770546725429
novelty_k=5 15 4.370274782180786 0.6068271240976065
novelty_k=5 16 4.616487503051758 0.6437421547901813
novelty_k=5 17 4.482872247695923 0.6913272237660385
novelty_k=5 18 4.5589377880096436 0.7104980882149302
novelty_k=5 19 4.467358827590942 0.6405140855236062
```

## Q4b: Plot

Please visualize the fitness and diversity over time for novelty search vs. fitness-based search

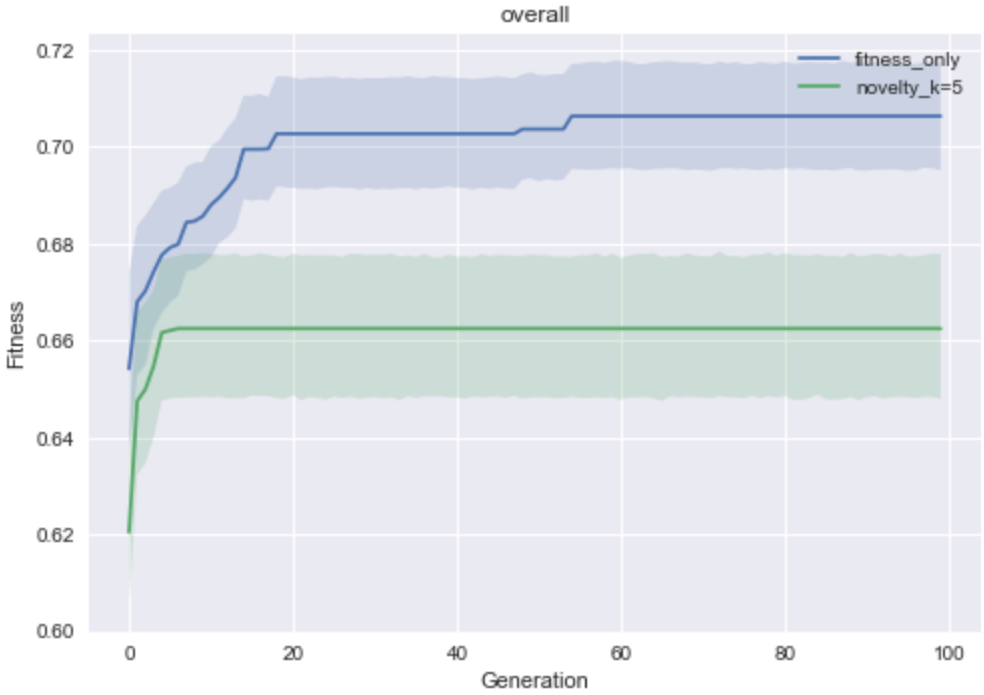
In [14]:

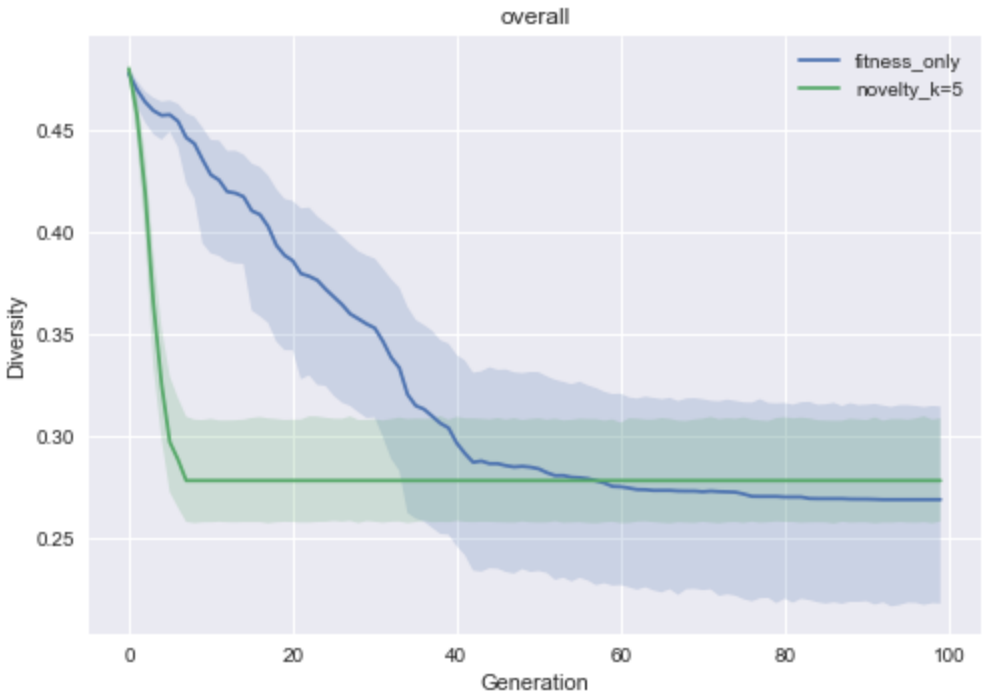
```
# plotting
plot_mean_and_bootstrapped_ci_multiple(input_data=[np.transpose(x) for k, x in experiment_
plot_mean_and_bootstrapped_ci_multiple(input_data=[np.transpose(x) for k, x in diversity_r
```

```
[array([0.6377392 , 0.67391834]), array([0.65282924, 0.68364408]), array([0.65495696, 0.68
579633]), array([0.66240331, 0.68830029]), array([0.66572508, 0.69095345]), array([0.66788
177, 0.69152931]), array([0.66926958, 0.69254894]), array([0.67434339, 0.69589481]), array
([0.67464196, 0.6966715 ]), array([0.67576252, 0.69681353]), array([0.67716313, 0.7001207
8]), array([0.68013038, 0.70123612]), array([0.68134082, 0.70382541]), array([0.68319553,
0.70585514]), array([0.68920416, 0.71056439]), array([0.68888448, 0.71052548]), array([0.6
8913345, 0.71089757]), array([0.68886781, 0.71026647]), array([0.69187015, 0.71449377]), a
rray([0.69160031, 0.71450085]), array([0.69151438, 0.71434814]), array([0.69132488, 0.7138
6977]), array([0.69142789, 0.71430081]), array([0.6912199 , 0.71421722]), array([0.6916513
4, 0.71435975]), array([0.69173059, 0.71449289]), array([0.691532  , 0.71407332]), array
([0.69157616, 0.7142831 ]), array([0.69116706, 0.71396007]), array([0.69151524, 0.7141825
8]), array([0.69121452, 0.71415326]), array([0.69136306, 0.71415791]), array([0.6912587 ,
0.71433361]), array([0.69142172, 0.71429446]), array([0.6914885 , 0.71452576]), array([0.6
9121216, 0.71424743]), array([0.69158683, 0.71415075]), array([0.69133573, 0.71456099]), a
rray([0.69160104, 0.71437673]), array([0.69181026, 0.71434302]), array([0.6915544 , 0.7141
9028]), array([0.69154428, 0.71411592]), array([0.69117279, 0.71396338]), array([0.6915616
8, 0.71411673]), array([0.69129578, 0.71431705]), array([0.69161544, 0.71447647]), array
([0.6914009 , 0.71430299]), array([0.69128601, 0.71408407]), array([0.6932291 , 0.7149151
2]), array([0.6935161 , 0.71531811]), array([0.69287072, 0.71477389]), array([0.6932982 ,
0.71472964]), array([0.69303841, 0.7149847 ]), array([0.69324223, 0.71495209]), array([0.6
9543356, 0.71704496]), array([0.69534315, 0.7171951 ]), array([0.69524423, 0.71742618]), a
rray([0.6953007 , 0.71755625]), array([0.69540708, 0.7172697 ]), array([0.6955342 , 0.7176
7431]), array([0.6953893 , 0.71773301]), array([0.69560036, 0.71749958]), array([0.6952805
5, 0.71700675]), array([0.6951103 , 0.71709805]), array([0.69540173, 0.71741188]), array
([0.69542126, 0.71754385]), array([0.69554824, 0.71768224]), array([0.69545755, 0.7174068
3]), array([0.69516856, 0.7173046 ]), array([0.69529847, 0.71742367]), array([0.69518959,
0.71710935]), array([0.69502359, 0.71697317]), array([0.69550637, 0.71753797]), array([0.6
9558266, 0.71719555]), array([0.69538771, 0.71741774]), array([0.69549441, 0.71708196]), a
rray([0.69527778, 0.71705503]), array([0.69517592, 0.71744388]), array([0.69542285, 0.7172
6934]), array([0.69526127, 0.71711792]), array([0.69554428, 0.71757263]), array([0.6954003
8, 0.71716466]), array([0.69554505, 0.71731084]), array([0.69531163, 0.71744336]), array
([0.69517247, 0.71730532]), array([0.69529055, 0.71740232]), array([0.69510841, 0.7171532
6]), array([0.69537862, 0.71766854]), array([0.69556237, 0.71737826]), array([0.69507969,
0.71735877]), array([0.6950768 , 0.71721891]), array([0.6953001 , 0.71754578]), array([0.6
9529796, 0.71714716]), array([0.69533507, 0.71729773]), array([0.69526205, 0.71708941]), a
rray([0.69503757, 0.71714658]), array([0.69508711, 0.71717282]), array([0.69570478, 0.7174
2339]), array([0.6955003 , 0.71732254]), array([0.6951547 , 0.71732434])]
[array([0.60553519, 0.64064987]), array([0.6323327 , 0.66578192]), array([0.63470797, 0.66
780983]), array([0.64005777, 0.67040933]), array([0.64759898, 0.67669005]), array([0.64809
216, 0.67713573]), array([0.64816866, 0.67744709]), array([0.6482522 , 0.67786442]), array
([0.64832448, 0.67782185]), array([0.64838134, 0.67799915]), array([0.64826134, 0.6779125
6]), array([0.64842374, 0.67759338]), array([0.64812639, 0.67765396]), array([0.64809262,
0.67794909]), array([0.64811289, 0.67740815]), array([0.64867115, 0.67758808]), array([0.6
4854107, 0.67797463]), array([0.64856699, 0.67764975]), array([0.64811784, 0.67731042]), a
rray([0.64799939, 0.67717113]), array([0.64867187, 0.67787693]), array([0.64774794, 0.6774
8532]), array([0.64834816, 0.67765357]), array([0.64817045, 0.67749726]), array([0.6478236
5, 0.67759378]), array([0.6484307 , 0.67749595]), array([0.64814578, 0.67722005]), array
([0.64844509, 0.6778424 ]), array([0.64799233, 0.67783884]), array([0.64819351, 0.6776381
7]), array([0.64834931, 0.67773407]), array([0.64813465, 0.67763318]), array([0.64830887,
0.67793057]), array([0.64774978, 0.6773245 ]), array([0.64852082, 0.67752232]), array([0.6
4818844, 0.67720255]), array([0.64848502, 0.67789994]), array([0.64842295, 0.67717383]), a
rray([0.64850123, 0.67710809]), array([0.64826367, 0.67774022]), array([0.64822601, 0.6772
7238]), array([0.64800672, 0.67737517]), array([0.64847827, 0.67719063]), array([0.6483287
2, 0.67779639]), array([0.6483601 , 0.67742923]), array([0.64845025, 0.67767899]), array
([0.6481957 , 0.67757629]), array([0.64862021, 0.67752644]), array([0.64798882, 0.6774549
7]), array([0.64825416, 0.67769121]), array([0.64831562, 0.67760808]), array([0.64830598,
```

0.67758142]), array([0.64810658, 0.67726535]), array([0.6481586 , 0.67730881]), array([0.6
4842231, 0.67720907]), array([0.64813113, 0.67763906]), array([0.64813743, 0.67755206]), a
rray([0.64830109, 0.67801435]), array([0.64819397, 0.67729301]), array([0.64832293, 0.6778
9216]), array([0.64777216, 0.67708317]), array([0.64826805, 0.67753291]), array([0.6482517
8, 0.67780124]), array([0.64839062, 0.67785426]), array([0.64803244, 0.67802287]), array
([0.64758662, 0.67721188]), array([0.64849726, 0.67738812]), array([0.64827095, 0.6772430
9]), array([0.64841811, 0.67796722]), array([0.64835535, 0.67757048]), array([0.64831443,
0.67732579]), array([0.64816104, 0.677366 ]), array([0.64854355, 0.67833071]), array([0.6
4830438, 0.67746684]), array([0.64828479, 0.67738521]), array([0.64810772, 0.67716778]), a
rray([0.64867207, 0.67765727]), array([0.64806761, 0.67766885]), array([0.64854037, 0.6781
3989]), array([0.64824485, 0.67750317]), array([0.64850851, 0.67802041]), array([0.6478012
2, 0.67765703]), array([0.6478123 , 0.67761491]), array([0.64814352, 0.67750334]), array
([0.64884592, 0.67783444]), array([0.64792952, 0.67739013]), array([0.64783399, 0.6775167
1]), array([0.64793745, 0.67719831]), array([0.64839899, 0.67799786]), array([0.64774032,
0.67739644]), array([0.64836603, 0.6775713 ]), array([0.64849926, 0.67785694]), array([0.6
4801572, 0.67771912]), array([0.64832487, 0.67737149]), array([0.64837257, 0.67790261]), a
rray([0.64831304, 0.67731578]), array([0.64813906, 0.67734639]), array([0.6484249 , 0.6776
0801]), array([0.64835951, 0.67790625]), array([0.64802354, 0.67780248])]
[array([0.47483795, 0.48085652]), array([0.46187632, 0.47426204]), array([0.45399111, 0.46
918228]), array([0.44827233, 0.46605822]), array([0.44548978, 0.46418336]), array([0.44957
341, 0.46475366]), array([0.44197734, 0.46305374]), array([0.42434572, 0.45851535]), array
([0.41695037, 0.45678498]), array([0.39473648, 0.45140441]), array([0.38982632, 0.4453421
6]), array([0.38857743, 0.44514625]), array([0.38580922, 0.44000539]), array([0.38488157,
0.44011874]), array([0.38466719, 0.43826412]), array([0.36173458, 0.43377996]), array([0.3
58996 , 0.43211586]), array([0.3556873 , 0.42788333]), array([0.34657634, 0.42127268]), a
rray([0.34238531, 0.41648745]), array([0.34230043, 0.41549204]), array([0.32809577, 0.4118
3659]), array([0.3300755 , 0.41224974]), array([0.32532276, 0.40787112]), array([0.3248186
6, 0.40472215]), array([0.32001354, 0.40170537]), array([0.31660228, 0.39799622]), array
([0.31502079, 0.39428445]), array([0.31192326, 0.39081902]), array([0.30913951, 0.3887661
4]), array([0.30946167, 0.38725531]), array([0.29974803, 0.38232532]), array([0.28911063,
0.37719537]), array([0.28326539, 0.37292806]), array([0.26249145, 0.3634927 ]), array([0.2
5978952, 0.35716258]), array([0.25831955, 0.35467127]), array([0.25522064, 0.35186082]), a
rray([0.25228359, 0.34719359]), array([0.25190011, 0.3455323 ]), array([0.24627566, 0.3399
8739]), array([0.24186766, 0.33719871]), array([0.23427608, 0.33126837]), array([0.2337625
1, 0.33193372]), array([0.23542726, 0.33391381]), array([0.23520883, 0.33268326]), array
([0.23358746, 0.33284375]), array([0.23276607, 0.33144761]), array([0.23504158, 0.3310292
3]), array([0.23356648, 0.33166151]), array([0.23404166, 0.3314998 ]), array([0.23311214,
0.32946748]), array([0.22971968, 0.32781476]), array([0.23087843, 0.32676326]), array([0.2
2891698, 0.32562848]), array([0.23073037, 0.32686009]), array([0.22943717, 0.32504791]), a
rray([0.22714233, 0.32376336]), array([0.22809973, 0.32190813]), array([0.22689047, 0.3207
9501]), array([0.22715109, 0.32098302]), array([0.22611666, 0.31915842]), array([0.2248767
8, 0.31859943]), array([0.22611747, 0.31956905]), array([0.22454847, 0.31892472]), array
([0.22360044, 0.31831827]), array([0.22516985, 0.31893656]), array([0.22238095, 0.3173763
8]), array([0.22514822, 0.31860536]), array([0.22535593, 0.31880519]), array([0.22469402,
0.31828812]), array([0.22506889, 0.31750984]), array([0.22185813, 0.31725784]), array([0.2
2264731, 0.31807965]), array([0.22197069, 0.31744387]), array([0.22024099, 0.31671137]), a
rray([0.21919851, 0.31660667]), array([0.22071649, 0.31828358]), array([0.2208961, 0.31640
33]), array([0.21829049, 0.31620049]), array([0.22052412, 0.31643344]), array([0.21851427,
0.31557088]), array([0.22047427, 0.3167736 ]), array([0.21978145, 0.31698718]), array([0.2
1875758, 0.3155664 ]), array([0.21806885, 0.31575091]), array([0.21874921, 0.31652346]), a
rray([0.21795585, 0.31490914]), array([0.21896993, 0.31634861]), array([0.21892413, 0.3153
8052]), array([0.21925396, 0.31557323]), array([0.21756938, 0.31599855]), array([0.2187434
5, 0.31517899]), array([0.21670392, 0.31447761]), array([0.21736469, 0.315352 ]), array
([0.21854433, 0.3153966 ]), array([0.21842503, 0.31499157]), array([0.21741058, 0.3144117
4]), array([0.21829839, 0.31498996]), array([0.21811085, 0.31467261])]
[array([0.47547316, 0.48322899]), array([0.44492044, 0.46682176]), array([0.39692502, 0.43
828949]), array([0.33429456, 0.3899535 ]), array([0.29901998, 0.3529944 ]), array([0.27276
837, 0.32943504]), array([0.26527019, 0.31882492]), array([0.25816248, 0.30971656]), array
([0.25753673, 0.30825553]), array([0.25796547, 0.30803348]), array([0.2579752 , 0.3086184
3]), array([0.25823018, 0.30788015]), array([0.25799687, 0.30787456]), array([0.25791249,
0.30797422]), array([0.25806538, 0.30824454]), array([0.25806526, 0.30908147]), array([0.2
5804736, 0.3095281 ]), array([0.25746145, 0.30885624]), array([0.25795406, 0.30854419]), a
rray([0.25819373, 0.30807167]), array([0.25806466, 0.3082078 ]), array([0.25781036, 0.3082
7461]), array([0.25784482, 0.30987592]), array([0.2581164, 0.3097988]), array([0.25818072,
0.30919224]), array([0.25861083, 0.30876182]), array([0.25823516, 0.30865319]), array([0.2
5866597, 0.30987151]), array([0.25756778, 0.30806623]), array([0.2585524 , 0.30820929]), a

rray([0.2582327, 0.3084545]), array([0.25772361, 0.3078544 ]), array([0.2582382 , 0.308515
27]), array([0.25834066, 0.30925424]), array([0.25752921, 0.30858944]), array([0.25805929,
0.30882793]), array([0.25866305, 0.30864842]), array([0.25797424, 0.30874853]), array([0.2
5810412, 0.3100184 ]), array([0.25800505, 0.30823939]), array([0.25861301, 0.30962724]), a
rray([0.25816145, 0.30892766]), array([0.25802653, 0.30889049]), array([0.25820719, 0.3086
6466]), array([0.2581631 , 0.30838338]), array([0.25796902, 0.30908028]), array([0.2578632
5, 0.30822656]), array([0.2577414 , 0.30898911]), array([0.25837045, 0.30807295]), array
([0.25791727, 0.30857674]), array([0.25779219, 0.3089303 ]), array([0.25787819, 0.3081603
1]), array([0.25847973, 0.30871239]), array([0.25770175, 0.3078796 ]), array([0.25788334,
0.30814086]), array([0.25799762, 0.30867875]), array([0.25759636, 0.30849547]), array([0.2
5742221, 0.30848279]), array([0.25793538, 0.30774777]), array([0.25825181, 0.30834331]), a
rray([0.25754013, 0.30673821]), array([0.25764377, 0.30887764]), array([0.25828255, 0.3087
6684]), array([0.25841186, 0.30829833]), array([0.25804526, 0.30830287]), array([0.2577122
4, 0.30905182]), array([0.25743353, 0.30859436]), array([0.25800054, 0.30776113]), array
([0.2577956 , 0.30762109]), array([0.25764868, 0.30829903]), array([0.25820307, 0.3087073
1]), array([0.25780321, 0.30958831]), array([0.25744618, 0.30845638]), array([0.25796521,
0.30894023]), array([0.2581219 , 0.30863874]), array([0.25763071, 0.30936567]), array([0.2
5831163, 0.3083299 ]), array([0.25774707, 0.30790405]), array([0.25799463, 0.30760506]), a
rray([0.2583697, 0.3090391]), array([0.25812092, 0.30944291]), array([0.25810035, 0.308874
06]), array([0.25771784, 0.30927032]), array([0.25791465, 0.30886441]), array([0.25794869,
0.30798498]), array([0.25790235, 0.30781921]), array([0.25729336, 0.30773805]), array([0.2
5824321, 0.30774969]), array([0.25778366, 0.30868092]), array([0.25791176, 0.30834259]), a
rray([0.25791325, 0.30798371]), array([0.25741334, 0.30934688]), array([0.25781739, 0.3077
4029]), array([0.25775577, 0.3085264 ]), array([0.25808721, 0.30857306]), array([0.2578307
4, 0.30821955]), array([0.25772804, 0.30883819]), array([0.258034  , 0.30991372]), array
([0.25747491, 0.30822296]), array([0.25832951, 0.30893287])]






## Q5: Analysis

How does novelty search perform in this domain? Is it what you expected? If no, why might that be the case?

**Novelty seems to perform very poorly. This is not what I was expecting, but upon thinking more deeply about this particular situation, it might start to make sense. We are calculating novelty as the hamming distance between the genomes in the archive with the closest fitness. This is a fitness-based (and, in this case, phenotypic) measurement of novelty, whereas our diversity calculation is completed genotypic. This still doesn't explain everything since we can see even our fitness values begin to converge fairly quickly. I believe this is because we only calculate novelty the first time an individual is assessed. We do actually maintain a higher level of diversity in the end than the fitness based evolution, but our fitness scores do not do as well as I imagined. I also wonder if this has something to do with the ruggedness of the landscape and the calculation of novelty only being the 5 nearest fitness values.**

## Q6: Larger Neighborhoods

How might you expect the result to change if you were to use a larger neighborhood ( `novelty_k` value) for calculating a soluton's novelty.

Let's try it! Please run and plot the same settings as above, but with a `novelty_k` of `100` (i.e. using the full archive for novelty calcuation).

In [15]:

```
num_runs = 20
total_generations = 100
num_elements_to_mutate = 1
bit_string_length = 15
num_parents = 20
num_children = 20
novelty_k = 100
crossover=False

n = bit_string_length
k = bit_string_length-1

for run_name in ["novelty_k=100"]:
    experiment_results[run_name] = np.zeros((num_runs, total_generations))
```

```
    solutions_results[run_name] = np.zeros((num_runs, total_generations, bit_string_length
    diversity_results[run_name] = np.zeros((num_runs, total_generations))
    for run_num in range(num_runs):
        landscape = Landscape(n=n, k=k)
        start_time = time.time()
        fitness_over_time, solutions_over_time, diversity_over_time = evolutionary_algorit
        experiment_results[run_name][run_num] = fitness_over_time
        solutions_results[run_name][run_num] = solutions_over_time
        diversity_results[run_name][run_num] = diversity_over_time
        print(run_name, run_num, time.time()-start_time, fitness_over_time[-1])
```

```
novelty_k=100 0 4.4798688888549805 0.6619972382123518
novelty_k=100 1 4.373277902603149 0.636580596515919
novelty_k=100 2 4.388790845870972 0.6867917179054371
novelty_k=100 3 4.560939311981201 0.686845927043569
novelty_k=100 4 4.359265089035034 0.6533997035658649
novelty_k=100 5 4.544925689697266 0.6662763292584775
novelty_k=100 6 4.454347372055054 0.668930726685939
novelty_k=100 7 4.546927213668823 0.6409351751460461
novelty_k=100 8 4.418816566467285 0.6426378413465449
novelty_k=100 9 4.557936906814575 0.6746594534195992
novelty_k=100 10 4.4338295459747314 0.6499868381943565
novelty_k=100 11 4.490378379821777 0.6617703671206865
novelty_k=100 12 4.460352897644043 0.6795110561402821
novelty_k=100 13 4.587462425231934 0.6414139039781336
novelty_k=100 14 4.398799419403076 0.6247527447029849
novelty_k=100 15 4.633001804351807 0.6029885433896425
novelty_k=100 16 4.42432165145874 0.757907598039554
novelty_k=100 17 4.618989706039429 0.6065104868740989
novelty_k=100 18 4.4003005027771 0.6157258921315397
novelty_k=100 19 4.588963985443115 0.6309416329106585
```

In [16]:

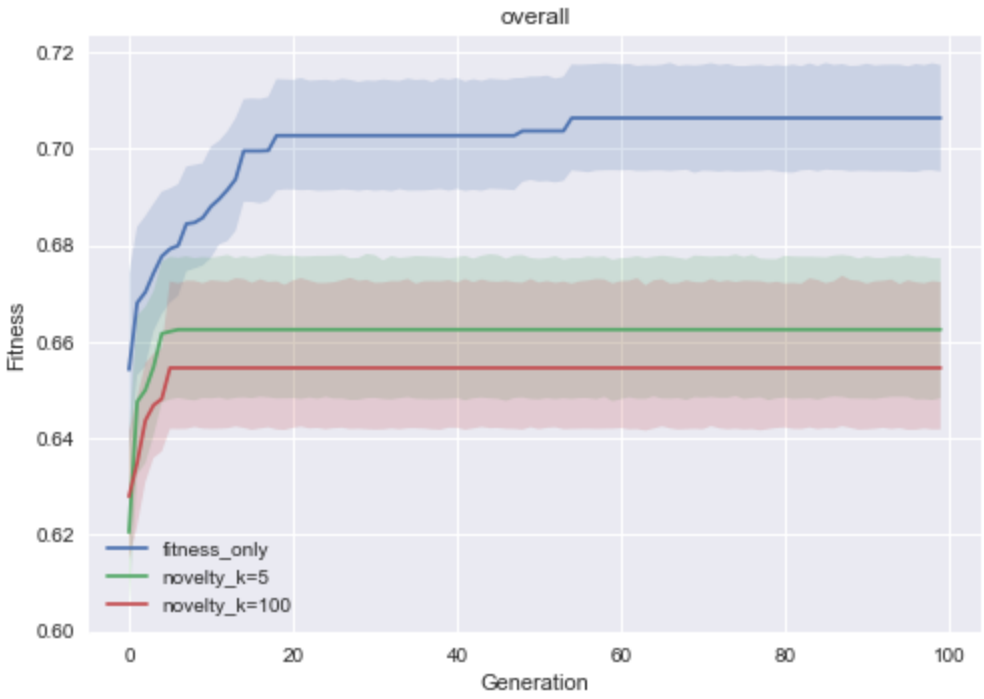
```
# plotting
plot_mean_and_bootstrapped_ci_multiple(input_data=[np.transpose(x) for k, x in experiment_
plot_mean_and_bootstrapped_ci_multiple(input_data=[np.transpose(x) for k, x in diversity_r
```

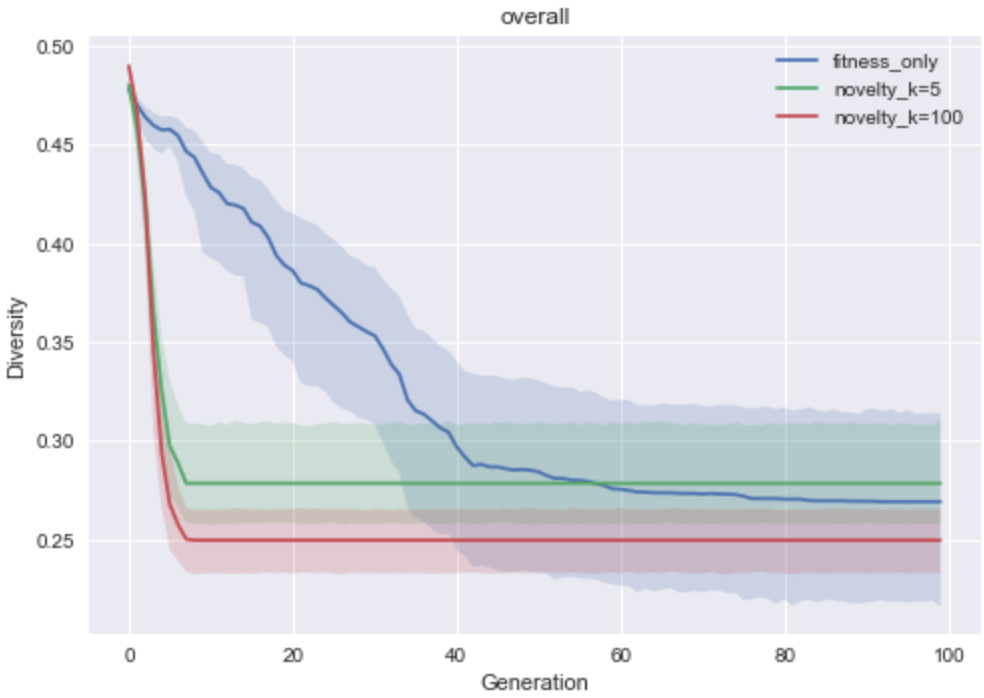
```
[array([0.63742968, 0.6740432 ]), array([0.65299521, 0.68376136]), array([0.65492503, 0.68
600854]), array([0.66217143, 0.68856043]), array([0.66569527, 0.69105584]), array([0.66809
702, 0.69171371]), array([0.6694831 , 0.69260739]), array([0.67457356, 0.69627246]), array
([0.67513691, 0.69660181]), array([0.6757525 , 0.69695926]), array([0.67732741, 0.7003849
6]), array([0.68013567, 0.70157308]), array([0.68081404, 0.70350971]), array([0.68289203,
0.70625186]), array([0.68899009, 0.71021516]), array([0.68893301, 0.71037203]), array([0.6
8862595, 0.71026713]), array([0.68918249, 0.7106933 ]), array([0.69132947, 0.71439202]), a
rray([0.69151559, 0.71429776]), array([0.6914957, 0.7142263]), array([0.69137644, 0.714594
08]), array([0.69144073, 0.7140422 ]), array([0.69123258, 0.71414342]), array([0.69127533,
0.71423778]), array([0.69133357, 0.71401024]), array([0.69150895, 0.71443494]), array([0.6
9143116, 0.71420649]), array([0.69121627, 0.71380393]), array([0.69145689, 0.71409557]), a
rray([0.69123294, 0.71400927]), array([0.6913894 , 0.71429101]), array([0.69114762, 0.7141
1602]), array([0.69158993, 0.71400957]), array([0.69125723, 0.71450073]), array([0.6912929
3, 0.71419526]), array([0.69130426, 0.71431135]), array([0.69145284, 0.7138169 ]), array
([0.69160655, 0.71451152]), array([0.69126736, 0.71422413]), array([0.69163875, 0.7146159
7]), array([0.69120487, 0.71389171]), array([0.69144521, 0.71435257]), array([0.69121738,
0.71400414]), array([0.69165916, 0.71444672]), array([0.69159576, 0.7143316 ]), array([0.6
9150983, 0.71421016]), array([0.69137318, 0.71455809]), array([0.69311115, 0.71462393]), a
rray([0.69348558, 0.71489022]), array([0.69314745, 0.71489355]), array([0.69323496, 0.7151
1393]), array([0.69298008, 0.71447902]), array([0.69328462, 0.71485317]), array([0.6954616
, 0.71740977]), array([0.69528952, 0.71732301]), array([0.69508819, 0.71742254]), array
([0.69543012, 0.71750259]), array([0.69548516, 0.71764171]), array([0.6955253 , 0.7175434
5]), array([0.69547275, 0.71739138]), array([0.69555491, 0.71753956]), array([0.69535028,
0.71710841]), array([0.69535999, 0.71727105]), array([0.69545762, 0.71719815]), array([0.6
9545903, 0.7173469 ]), array([0.69532063, 0.71776192]), array([0.69547893, 0.71711634]), a
rray([0.69561387, 0.71729444]), array([0.6949484 , 0.71718481]), array([0.69541156, 0.7173
4228]), array([0.6953499, 0.7176977]), array([0.69519632, 0.71728881]), array([0.6953596 ,
0.71751941]), array([0.69521525, 0.71711935]), array([0.69526837, 0.71705219]), array([0.6
9530219, 0.71721453]), array([0.69534519, 0.71732829]), array([0.69552651, 0.71705519]), a
```

rray([0.69529059, 0.71729503]), array([0.69505052, 0.71710986]), array([0.69500481, 0.7173
8667]), array([0.69546029, 0.7170565 ]), array([0.6956101, 0.717638 ]), array([0.69527088,
0.71711306]), array([0.69564303, 0.71773275]), array([0.69552263, 0.7173023 ]), array([0.6
95508  , 0.71749621]), array([0.69526739, 0.71708901]), array([0.69536659, 0.71718387]), a
rray([0.69536751, 0.71757916]), array([0.69527924, 0.71720283]), array([0.69532599, 0.7173
3545]), array([0.69529428, 0.7173744 ]), array([0.69528083, 0.71736359]), array([0.6951823
7, 0.71701997]), array([0.69513144, 0.71750117]), array([0.69547653, 0.71727013]), array
([0.69549254, 0.7176625 ]), array([0.69522833, 0.71727626])]
[array([0.60510057, 0.64021418]), array([0.63242978, 0.66532691]), array([0.63479613, 0.66
719835]), array([0.64053347, 0.6704023 ]), array([0.64739053, 0.67656581]), array([0.64800
507, 0.67752991]), array([0.64833136, 0.67734598]), array([0.64802042, 0.677644  ]), array
([0.64781752, 0.67727436]), array([0.64830216, 0.67749139]), array([0.64823688, 0.6773782
3]), array([0.64841158, 0.67759173]), array([0.64802634, 0.6779672 ]), array([0.64834616,
0.6772778 ]), array([0.64844829, 0.67767444]), array([0.64827085, 0.67734766]), array([0.6
4859083, 0.67763388]), array([0.64844267, 0.67805087]), array([0.64822132, 0.67790661]), a
rray([0.6482472 , 0.67771796]), array([0.64826913, 0.67782727]), array([0.64811804, 0.6775
8015]), array([0.64842583, 0.67763905]), array([0.64816045, 0.67774278]), array([0.6485528
4, 0.67773885]), array([0.6482062 , 0.67781212]), array([0.64819767, 0.67792012]), array
([0.64788061, 0.67722099]), array([0.64792374, 0.67731935]), array([0.64827976, 0.6781560
9]), array([0.64834565, 0.6778952 ]), array([0.64810334, 0.67682477]), array([0.64799435,
0.67770878]), array([0.64827084, 0.67731391]), array([0.64848337, 0.67746971]), array([0.6
4815093, 0.67771915]), array([0.64810261, 0.67764981]), array([0.64804224, 0.67754927]), a
rray([0.64810295, 0.6774917 ]), array([0.64814958, 0.67720919]), array([0.6483008 , 0.6774
4068]), array([0.64783586, 0.67775357]), array([0.6482528 , 0.67716276]), array([0.6484019
9, 0.67775987]), array([0.64825423, 0.6780883 ]), array([0.64812854, 0.67793103]), array
([0.64770586, 0.67715074]), array([0.64832542, 0.67780197]), array([0.64842898, 0.6779181
7]), array([0.64809008, 0.67729836]), array([0.6480036 , 0.67713251]), array([0.64828638,
0.67728243]), array([0.64806124, 0.67769766]), array([0.64827417, 0.67699126]), array([0.6
4810857, 0.67729473]), array([0.64829372, 0.67726013]), array([0.6483024 , 0.67762989]), a
rray([0.64804413, 0.67742104]), array([0.64855997, 0.67764575]), array([0.6476435 , 0.6769
2996]), array([0.64805489, 0.67726519]), array([0.64845837, 0.67737715]), array([0.6481370
2, 0.67768679]), array([0.64813288, 0.6772915 ]), array([0.64864292, 0.67755701]), array
([0.64821636, 0.67777801]), array([0.64818309, 0.67761228]), array([0.64827923, 0.6773437
8]), array([0.64789617, 0.67705137]), array([0.64849583, 0.67781775]), array([0.64835896,
0.67769235]), array([0.64852418, 0.67757404]), array([0.64836313, 0.67770372]), array([0.6
4834049, 0.67761621]), array([0.64869797, 0.67804201]), array([0.64810177, 0.67750375]), a
rray([0.64832399, 0.67797381]), array([0.64846086, 0.67719181]), array([0.64785389, 0.6770
8435]), array([0.64824523, 0.67708126]), array([0.64839858, 0.67777069]), array([0.6481045
1, 0.67762491]), array([0.6481744 , 0.67757423]), array([0.64825168, 0.67758171]), array
([0.64796856, 0.67751768]), array([0.6483758 , 0.67743238]), array([0.64862771, 0.6778087
7]), array([0.64852955, 0.67794129]), array([0.64821367, 0.67772975]), array([0.64818929,
0.67719549]), array([0.64816417, 0.67762803]), array([0.64808924, 0.67731748]), array([0.6
4835902, 0.67731909]), array([0.64855447, 0.67760763]), array([0.64855654, 0.67756728]), a
rray([0.6485088, 0.6779678]), array([0.64800317, 0.67751633]), array([0.64803797, 0.677591
59]), array([0.64788963, 0.67726886]), array([0.64825009, 0.67726997])]
[array([0.61493288, 0.64145772]), array([0.62179884, 0.64841837]), array([0.63085439, 0.65
553752]), array([0.63590318, 0.65754889]), array([0.63719469, 0.65858408]), array([0.64190
145, 0.6724272 ]), array([0.64185715, 0.67206548]), array([0.64190378, 0.67260843]), array
([0.64230799, 0.67257135]), array([0.64197075, 0.67206287]), array([0.64189621, 0.6727012
7]), array([0.64192499, 0.67255416]), array([0.64202766, 0.67240813]), array([0.64239571,
0.6731077 ]), array([0.64209762, 0.67293404]), array([0.64169175, 0.67216   ]), array([0.6
4198047, 0.67288591]), array([0.64222652, 0.67234054]), array([0.64176828, 0.67248   ]), a
rray([0.64167637, 0.67172344]), array([0.64210215, 0.67265136]), array([0.64238609, 0.6731
2573]), array([0.64199193, 0.6726132 ]), array([0.64209959, 0.67212719]), array([0.6418190
5, 0.67212664]), array([0.64204888, 0.67227903]), array([0.64214593, 0.67272532]), array
([0.64185026, 0.67318513]), array([0.64192875, 0.67257144]), array([0.64181655, 0.6727725
]), array([0.64174206, 0.67214906]), array([0.64177102, 0.67221595]), array([0.64199173,
0.67251626]), array([0.64180161, 0.67250209]), array([0.64202454, 0.67297873]), array([0.6
4203513, 0.67197923]), array([0.64196313, 0.67250182]), array([0.64179438, 0.67252929]), a
rray([0.64172177, 0.67249436]), array([0.64215825, 0.67273123]), array([0.64212544, 0.6722
2657]), array([0.64170796, 0.67294327]), array([0.64198832, 0.67227507]), array([0.6423094
4, 0.67286614]), array([0.64205774, 0.6724596 ]), array([0.64210164, 0.67235745]), array
([0.64192595, 0.67257333]), array([0.64188712, 0.67227466]), array([0.64174077, 0.6721755
7]), array([0.64225287, 0.67294065]), array([0.64183715, 0.67218985]), array([0.64211998,
0.67254095]), array([0.64224415, 0.6721526 ]), array([0.64182274, 0.67243243]), array([0.6
4187371, 0.67199857]), array([0.64192073, 0.67247484]), array([0.6419365 , 0.67262658]), a

rray([0.64188103, 0.67294581]), array([0.64191493, 0.67283757]), array([0.64194537, 0.6720
3136]), array([0.64206864, 0.67252717]), array([0.64180055, 0.67258479]), array([0.6418475
, 0.67312331]), array([0.64153853, 0.67162043]), array([0.64195476, 0.67268637]), array
([0.64241945, 0.67288674]), array([0.64200485, 0.67201431]), array([0.64174777, 0.6721228
1]), array([0.64184185, 0.67267114]), array([0.64191255, 0.67228691]), array([0.64182219,
0.67221922]), array([0.64247018, 0.67292859]), array([0.64201937, 0.67234096]), array([0.6
4180708, 0.67264646]), array([0.64192703, 0.67229505]), array([0.64177155, 0.67261655]), a
rray([0.64184284, 0.67255055]), array([0.641589  , 0.67250374]), array([0.64184966, 0.6721
5039]), array([0.64208925, 0.67237506]), array([0.64219591, 0.67203302]), array([0.6419043
3, 0.67213626]), array([0.64185981, 0.67196688]), array([0.64221387, 0.67308326]), array
([0.64185641, 0.67275148]), array([0.64205495, 0.67207952]), array([0.64197673, 0.6725024
2]), array([0.64245589, 0.67362019]), array([0.64211159, 0.67274609]), array([0.64174164,
0.67259847]), array([0.6417798, 0.6719875]), array([0.6419167 , 0.67246894]), array([0.642
12993, 0.67258715]), array([0.64187361, 0.67285703]), array([0.64205854, 0.67189652]), arr
ay([0.64170207, 0.67267685]), array([0.64171336, 0.67227028]), array([0.64197264, 0.672056
51]), array([0.64176899, 0.67250575]), array([0.64179671, 0.67232654])]
[array([0.47487337, 0.48088531]), array([0.4617969 , 0.47431714]), array([0.45318141, 0.46
9145  ]), array([0.44807834, 0.46616277]), array([0.44548658, 0.46405107]), array([0.44940
873, 0.46456218]), array([0.44118047, 0.46319937]), array([0.42378929, 0.45852942]), array
([0.41652837, 0.45710363]), array([0.39562325, 0.45139127]), array([0.39243384, 0.4457614
8]), array([0.39049436, 0.44505245]), array([0.38597163, 0.44005888]), array([0.38365022,
0.43982712]), array([0.38334809, 0.43788258]), array([0.36157429, 0.43347071]), array([0.3
5976632, 0.43218128]), array([0.35743737, 0.4274167 ]), array([0.3476462 , 0.42141718]), a
rray([0.34349843, 0.41647067]), array([0.33993568, 0.41534452]), array([0.32920902, 0.4124
9136]), array([0.32740053, 0.41071696]), array([0.32768136, 0.40910494]), array([0.3246413
9, 0.40458851]), array([0.3192408 , 0.40194871]), array([0.3164438 , 0.39809246]), array
([0.31476497, 0.39355771]), array([0.31147276, 0.39198025]), array([0.31052107, 0.3886831
4]), array([0.30836796, 0.38784773]), array([0.29915739, 0.38174575]), array([0.28965134,
0.37716133]), array([0.28428881, 0.37323715]), array([0.2654053 , 0.36375038]), array([0.2
6101255, 0.35661086]), array([0.25868391, 0.35570335]), array([0.255146  , 0.35182735]), a
rray([0.25260453, 0.34797525]), array([0.25280916, 0.34550671]), array([0.24463865, 0.3403
0785]), array([0.24175493, 0.33528931]), array([0.23582543, 0.33405781]), array([0.2369556
5, 0.33272131]), array([0.23584753, 0.33300343]), array([0.23422272, 0.33277704]), array
([0.23341821, 0.33128268]), array([0.23348719, 0.33095042]), array([0.23417018, 0.3318752
3]), array([0.23358986, 0.33164104]), array([0.23305899, 0.33046159]), array([0.23258211,
0.32886138]), array([0.22939211, 0.32740416]), array([0.23051401, 0.32682939]), array([0.2
3083614, 0.32683899]), array([0.23035367, 0.32465217]), array([0.22990419, 0.32531786]), a
rray([0.22960722, 0.32425395]), array([0.22786146, 0.32159381]), array([0.22729696, 0.3201
8485]), array([0.22652909, 0.32073197]), array([0.22641746, 0.32041735]), array([0.2232655
5, 0.31804759]), array([0.22498763, 0.31789171]), array([0.22371941, 0.31761387]), array
([0.22431904, 0.31895561]), array([0.22550892, 0.31818609]), array([0.22329425, 0.3180899
1]), array([0.22310948, 0.31835368]), array([0.22389992, 0.3184901 ]), array([0.22336232,
0.31763271]), array([0.22392186, 0.31712252]), array([0.22409877, 0.31810803]), array([0.2
2270348, 0.31800634]), array([0.22264151, 0.31815153]), array([0.21927735, 0.31687206]), a
rray([0.21958879, 0.31611458]), array([0.22089468, 0.3168916 ]), array([0.21973327, 0.3171
1504]), array([0.21737126, 0.31597071]), array([0.21972734, 0.31687302]), array([0.2161997
1, 0.31499466]), array([0.21878987, 0.31590742]), array([0.21950658, 0.31685421]), array
([0.21801515, 0.31537854]), array([0.21959995, 0.31465035]), array([0.21905474, 0.3159274
7]), array([0.21854068, 0.31428132]), array([0.21771292, 0.31433779]), array([0.21750785,
0.31362919]), array([0.21806177, 0.31600082]), array([0.21860427, 0.31601253]), array([0.2
1922678, 0.31503443]), array([0.21812288, 0.31583545]), array([0.218519  , 0.31429106]), a
rray([0.21806987, 0.31545514]), array([0.21852598, 0.31363122]), array([0.21825475, 0.3137
355 ]), array([0.21896431, 0.31400674]), array([0.21599992, 0.31398222])]
[array([0.47548974, 0.48321941]), array([0.44413061, 0.46656194]), array([0.39621342, 0.43
839207]), array([0.33419453, 0.38855083]), array([0.29795988, 0.35212547]), array([0.27297
079, 0.33018092]), array([0.26575698, 0.3180956 ]), array([0.25804373, 0.30845298]), array
([0.25802699, 0.30877431]), array([0.25778949, 0.30832483]), array([0.2571954 , 0.3073852
8]), array([0.25805691, 0.30923863]), array([0.25772929, 0.30799525]), array([0.25814013,
0.30989119]), array([0.25811058, 0.30881835]), array([0.25821892, 0.30828913]), array([0.2
581582, 0.3093448]), array([0.25858539, 0.30969396]), array([0.25823301, 0.30858377]), arr
ay([0.25836032, 0.30890226]), array([0.25820526, 0.30887967]), array([0.25809096, 0.307659
47]), array([0.25803168, 0.30740362]), array([0.25773399, 0.30838368]), array([0.25830746,
0.30858831]), array([0.25848055, 0.30945595]), array([0.25742323, 0.30842819]), array([0.2
581704 , 0.30862954]), array([0.25801773, 0.30891072]), array([0.25765256, 0.30859129]), a
rray([0.25786524, 0.3080372 ]), array([0.25802306, 0.30860632]), array([0.2583795 , 0.3089
8694]), array([0.25804443, 0.30913597]), array([0.2574317 , 0.30817448]), array([0.2579569

7, 0.30823797]), array([0.25828469, 0.30858065]), array([0.25768985, 0.30866103]), array
([0.25786888, 0.30821735]), array([0.25799745, 0.30952743]), array([0.25833865, 0.3079681
3]), array([0.25829931, 0.30850807]), array([0.25770367, 0.30877926]), array([0.25805025,
0.30885432]), array([0.2582341 , 0.30819822]), array([0.25857007, 0.30943216]), array([0.2
5854366, 0.30895969]), array([0.25838272, 0.30888009]), array([0.25805487, 0.30824927]), a
rray([0.25750761, 0.30862533]), array([0.25746159, 0.3093232 ]), array([0.25804425, 0.3087
0962]), array([0.25785851, 0.30845388]), array([0.25816388, 0.30828944]), array([0.257509
, 0.30937536]), array([0.25779691, 0.30807496]), array([0.25813914, 0.30904826]), array
([0.25832553, 0.30897435]), array([0.2580504, 0.3079248]), array([0.25787005, 0.3093089
6]), array([0.25820846, 0.30840387]), array([0.25819209, 0.30917592]), array([0.25788113,
0.30860425]), array([0.25796384, 0.3095648 ]), array([0.25762055, 0.30940652]), array([0.2
583503 , 0.30961328]), array([0.25878744, 0.30889623]), array([0.25811055, 0.30876701]), a
rray([0.25776742, 0.30796675]), array([0.25816322, 0.30874948]), array([0.25792796, 0.3080
9119]), array([0.2579135 , 0.30808924]), array([0.25772873, 0.30760355]), array([0.2582002
8, 0.30830233]), array([0.25811145, 0.30921655]), array([0.25830857, 0.30773463]), array
([0.25806918, 0.30783869]), array([0.25826742, 0.3077649 ]), array([0.25832738, 0.3089164
8]), array([0.25809489, 0.30819614]), array([0.25824459, 0.30862925]), array([0.25806683,
0.30860074]), array([0.25809946, 0.30883577]), array([0.25798331, 0.30846504]), array([0.2
580588 , 0.30941801]), array([0.25817636, 0.30854543]), array([0.25775306, 0.30849563]), a
rray([0.25784533, 0.30860413]), array([0.25807492, 0.3092606 ]), array([0.2579676 , 0.3084
5934]), array([0.25855211, 0.30831241]), array([0.25807248, 0.30885981]), array([0.2581133
4, 0.30876289]), array([0.25776541, 0.30765637]), array([0.25771353, 0.30883516]), array
([0.25816674, 0.30849534]), array([0.25791856, 0.30811621]), array([0.25767299, 0.3080968
9]), array([0.25776521, 0.30803557]), array([0.25837674, 0.31025118])]
[array([0.4879153 , 0.49148161]), array([0.45797622, 0.47342522]), array([0.39981477, 0.43
611971]), array([0.30424169, 0.38270937]), array([0.26667585, 0.31797082]), array([0.24464
899, 0.2974024 ]), array([0.23927634, 0.27682116]), array([0.23358782, 0.26581606]), array
([0.2322938 , 0.26494815]), array([0.23254892, 0.26512858]), array([0.23290853, 0.2649893
7]), array([0.2327168 , 0.26509176]), array([0.23254783, 0.26518764]), array([0.23256604,
0.26563155]), array([0.2329067 , 0.26549682]), array([0.23239928, 0.26492548]), array([0.2
3306898, 0.26530447]), array([0.23329464, 0.26492079]), array([0.23296151, 0.26530188]), a
rray([0.23325197, 0.2653234 ]), array([0.23258265, 0.26542048]), array([0.23330855, 0.2656
2946]), array([0.23324826, 0.26533048]), array([0.23296111, 0.26556924]), array([0.2332023
2, 0.26517015]), array([0.23274864, 0.2652598 ]), array([0.23234379, 0.26530501]), array
([0.23316425, 0.26529843]), array([0.23288835, 0.26515298]), array([0.23287252, 0.2646282
3]), array([0.23294812, 0.26541347]), array([0.2326001 , 0.26526309]), array([0.23301471,
0.26474778]), array([0.2328318 , 0.26537319]), array([0.23288755, 0.2649637 ]), array([0.2
3313313, 0.26498869]), array([0.23277916, 0.26506284]), array([0.23241961, 0.26518396]), a
rray([0.23274111, 0.26522998]), array([0.23264229, 0.26529438]), array([0.23288465, 0.2653
3477]), array([0.23317042, 0.26522419]), array([0.23346639, 0.26499905]), array([0.2329041
3, 0.26518551]), array([0.23261563, 0.26547005]), array([0.23279906, 0.26472607]), array
([0.23297902, 0.26563597]), array([0.23311483, 0.26489612]), array([0.23281825, 0.2654966
7]), array([0.23307136, 0.26566579]), array([0.2329462 , 0.26528941]), array([0.23337904,
0.2650455 ]), array([0.23308801, 0.26502405]), array([0.23280008, 0.26450032]), array([0.2
3334807, 0.26567405]), array([0.23312379, 0.2654542 ]), array([0.23264839, 0.26491692]), a
rray([0.23246914, 0.26528231]), array([0.23318508, 0.26534639]), array([0.23327838, 0.2649
7384]), array([0.2329268 , 0.26575049]), array([0.23367254, 0.26519758]), array([0.2327463
3, 0.26536748]), array([0.23303462, 0.26484061]), array([0.23332683, 0.26511642]), array
([0.23270476, 0.26537926]), array([0.23388339, 0.26545243]), array([0.23310644, 0.2656163
3]), array([0.23276545, 0.26532963]), array([0.23266376, 0.2649972 ]), array([0.23272641,
0.26557865]), array([0.23358531, 0.26511374]), array([0.23240478, 0.26526988]), array([0.2
323045, 0.2650449]), array([0.23290254, 0.26516138]), array([0.23293072, 0.26543591]), arr
ay([0.23337161, 0.26540065]), array([0.23375841, 0.26535569]), array([0.23302923, 0.265595
73]), array([0.23325324, 0.26487934]), array([0.23273326, 0.26484128]), array([0.23237153,
0.26505646]), array([0.23289996, 0.26521245]), array([0.23296623, 0.2651286 ]), array([0.2
3277501, 0.2650909 ]), array([0.23278471, 0.2652178 ]), array([0.23294422, 0.26552759]), a
rray([0.23282787, 0.26520845]), array([0.23307621, 0.26532556]), array([0.23289507, 0.2648
9327]), array([0.23261874, 0.26539564]), array([0.23333228, 0.26554536]), array([0.2328822
6, 0.26473427]), array([0.23294358, 0.26544247]), array([0.23337919, 0.26516696]), array
([0.23256329, 0.26477449]), array([0.2332568 , 0.26509467]), array([0.23299085, 0.265326
]), array([0.23303673, 0.26514853]), array([0.23299147, 0.26520165])]




## Q6b: Analysis

What happened? Did it work better or worse? Is this what you expected (and why)?

**It looks like we end up with less diversity and fitness in the end than when comparing against 5 neighbors. This is interesting as well. It doesn't seem to have changed much. It isn't what I expected. I would have expected higher numbers of diversity since we are taking novelty as the hamming distance between the individual's fitness and every other fitness in the solution archive instead of just 5. This could be due to the fact that we are measuring diversity and novelty very differently as stated above. The algorithm is going to try to find unique fitness values, but that won't necessary satisfy diversity.**

## Q7: Mixed Fitness and Novelty

As suggested in class, perhaps the best version of an evolutionary algorithm is not one that selects just for fitness or one that selects just for novelty, but one that considers both in an attempt to carefully tradeoff

exploration and exploitation.

We may not be the most careful and inentional with our tradeoffs here, but let's start with perhaps the simplest combination of selecting for both novelty and fitness one could think of. Let's select some of the individuals in our population on the basis of novelty and some on the basis of fitness.

In particular, pleae define a new parameter `novelty_selection_prop` that defines what proportion of the parents for the next generation will be selected by novelty (and choose the most novel solutions to occupy that portion of the new generation) while the remainder of the new population ( `1-novelty_selection_prop` ) gets selected on the basis of fitness -- resulting in the same `num_parents` as before heading into the next generation.

In [17]:
```
def evolutionary_algorithm(fitness_function=None, total_generations=100, num_parents=10, r
    """ Evolutinary Algorithm (copied from the basic hillclimber in our last assignment)

        parameters:
        fitness_funciton: (callable function) that return the fitness of a genome
                          given the genome as an input parameter (e.g. as defined in Lanc
        total_generations: (int) number of total iterations for stopping condition
        num_parents: (int) the number of parents we downselect to at each generation (mu)
        num_childre: (int) the number of children (note: parents not included in this cour
        bit_string_length: (int) length of bit string genome to be evoloved
        num_elements_to_mutate: (int) number of alleles to modify during mutation (0 = no
        crossover (bool): whether to perform crossover when generating children

        returns:
        fitness_over_time: (numpy array) track record of the top fitness value at each ger
    """

    # initialize record keeping
    solution = None # best genome so far
    solution_fitness = -99999 # fitness of best genome so far
    fitness_over_time = np.zeros(total_generations)
    solutions_over_time = np.zeros((total_generations,bit_string_length))
    diversity_over_time = np.zeros(total_generations)
    solution_archive = []
    max_archive_length = 100

    # the initialization proceedure
    population = [] # keep population of individuals in a list
    for i in range(num_parents): # only create parents for initialization (the mu in mu+la
        population.append(Individual(fitness_function,bit_string_length)) # generate new i

    # get population fitness
    for i in range(len(population)):
        population[i].eval_fitness() # evaluate the fitness of each parent
        if len(solution_archive) < max_archive_length:
            solution_archive.append(population[i])
        else:
            population[i].eval_novelty(solution_archive, novelty_k)
            solution_archive = update_archive(solution_archive, population[i], max_archive



    for generation_num in range(total_generations): # repeat

        # the modification procedure
        new_children = [] # keep children separate for now (lambda in mu+lambda)
        while len(new_children) < num_children:

            # inheretance
            [parent1, parent2] = np.random.choice(population, size=2) # pick 2 random pare
```

```
            child1 = copy.deepcopy(parent1) # initialize children as perfect copies of the
            child2 = copy.deepcopy(parent2)

            # crossover
            if crossover:
                [crossover_point1, crossover_point2] = sorted(np.random.randint(0,bit_stri
                child1.genome[crossover_point1:crossover_point2+1] = parent2.genome[crossc
                child2.genome[crossover_point1:crossover_point2+1] = parent1.genome[crossc

            # mutation
            for this_child in [child1,child2]:
                elements_to_mutate = set()
                while len(elements_to_mutate)<num_elements_to_mutate:
                    elements_to_mutate.add(np.random.randint(bit_string_length)) # random
                for this_element_to_mutate in elements_to_mutate:
                    this_child.genome[this_element_to_mutate] = (this_child.genome[this_el

            new_children.extend((child1,child2)) # add children to the new_children list

        # the assessement procedure
        for i in range(len(new_children)):
            new_children[i].eval_fitness() # assign fitness to each child
            if len(solution_archive) < max_archive_length:
                solution_archive.append(new_children[i])
            else:
                new_children[i].eval_novelty(solution_archive, novelty_k)
                solution_archive = update_archive(solution_archive, new_children[i], max_a

        # selection procedure
        population += new_children # combine parents with new children (the + in mu+lambda
        for i in range(len(population)):
            population[i].eval_novelty(solution_archive, novelty_k)
        new_population = sorted(population, key=lambda individual: individual.novelty, rev
        new_population = population[:num_parents*novelty_selection_prop] # perform truncat
        population = sorted(population, key=lambda individual: individual.fitness, reverse
        population = population[:num_parents*(1-novelty_selection_prop)] # perform truncat
        population.extend(new_population)
        for i in range(len(population)):
            if len(solution_archive) < max_archive_length:
                solution_archive.append(population[i])
            else:
                solution_archive = update_archive(solution_archive, population[i], max_arc

        # record keeping
        population = sorted(population, key=lambda individual: individual.fitness, reverse
        if population[0].fitness > solution_fitness: # if the new parent is the best found
            solution = population[0].genome                 # update best solution records
            solution_fitness = population[0].fitness
            solution_generation = generation_num
        fitness_over_time[generation_num] = solution_fitness # record the fitness of the c

        solutions_over_time[generation_num,:] = solution

        genome_list = np.array([individual.genome for individual in population])
        diversity = np.mean(genome_list.std(axis=0))
        diversity_over_time[generation_num] = diversity

    return fitness_over_time, solutions_over_time, diversity_over_time
```

## Q8: Experimentation

Let's try running this mixed selection criteria for a 50/50 split between suvivors/parents for the next generation selected via novelty vs. fitness. Let's do this with our original novelty neighbhorhood of size 5, and all other

parameters the same.

As usual, please plot fitness and diversity afterwards.

In [18]:

```
num_runs = 20
total_generations = 100
num_elements_to_mutate = 1
bit_string_length = 15
num_parents = 20
num_children = 20
crossover=False

novelty_k = 5
novelty_selection_prop = 0.5
max_archive_length = 100

n = bit_string_length
k = bit_string_length-1

for run_name in ["novelty_k=5,p=.5"]:
    experiment_results[run_name] = np.zeros((num_runs, total_generations))
    solutions_results[run_name] = np.zeros((num_runs, total_generations, bit_string_length
    diversity_results[run_name] = np.zeros((num_runs, total_generations))
    for run_num in range(num_runs):
        landscape = Landscape(n=n, k=k)
        start_time = time.time()
        fitness_over_time, solutions_over_time, diversity_over_time = evolutionary_algorit
        experiment_results[run_name][run_num] = fitness_over_time
        solutions_results[run_name][run_num] = solutions_over_time
        diversity_results[run_name][run_num] = diversity_over_time
        print(run_name, run_num, time.time()-start_time, fitness_over_time[-1])
```

```
novelty_k=5,p=.5 0 1.890132188796997 0.6995050106947066
novelty_k=5,p=.5 1 3.524043560028076 0.7310517062845779
novelty_k=5,p=.5 2 3.3874258995056152 0.7267657757110775
novelty_k=5,p=.5 3 1.846595048904419 0.7089544307444736
novelty_k=5,p=.5 4 2.188390016555786 0.7241594093941116
novelty_k=5,p=.5 5 3.7122063636779785 0.6828387306816893
novelty_k=5,p=.5 6 2.4676313400268555 0.6880514192887834
novelty_k=5,p=.5 7 1.954688549041748 0.7830663724826945
novelty_k=5,p=.5 8 2.6292710304260254 0.7118555086015538
novelty_k=5,p=.5 9 3.446977376937866 0.6937409390050512
novelty_k=5,p=.5 10 1.9526865482330322 0.6935523230300216
novelty_k=5,p=.5 11 2.5557074546813965 0.699289556280429
novelty_k=5,p=.5 12 1.941176414489746 0.7000571995707094
novelty_k=5,p=.5 13 3.3914294242858887 0.6982358282486344
novelty_k=5,p=.5 14 2.107820510864258 0.7229394484885391
novelty_k=5,p=.5 15 2.1383469104766846 0.7136498631482249
novelty_k=5,p=.5 16 1.9611940383911133 0.6919691581897226
novelty_k=5,p=.5 17 2.957554817199707 0.6966248993329148
novelty_k=5,p=.5 18 1.9626951217651367 0.6593381326154613
novelty_k=5,p=.5 19 3.7977800369262695 0.7071984895099825
```

In [19]:

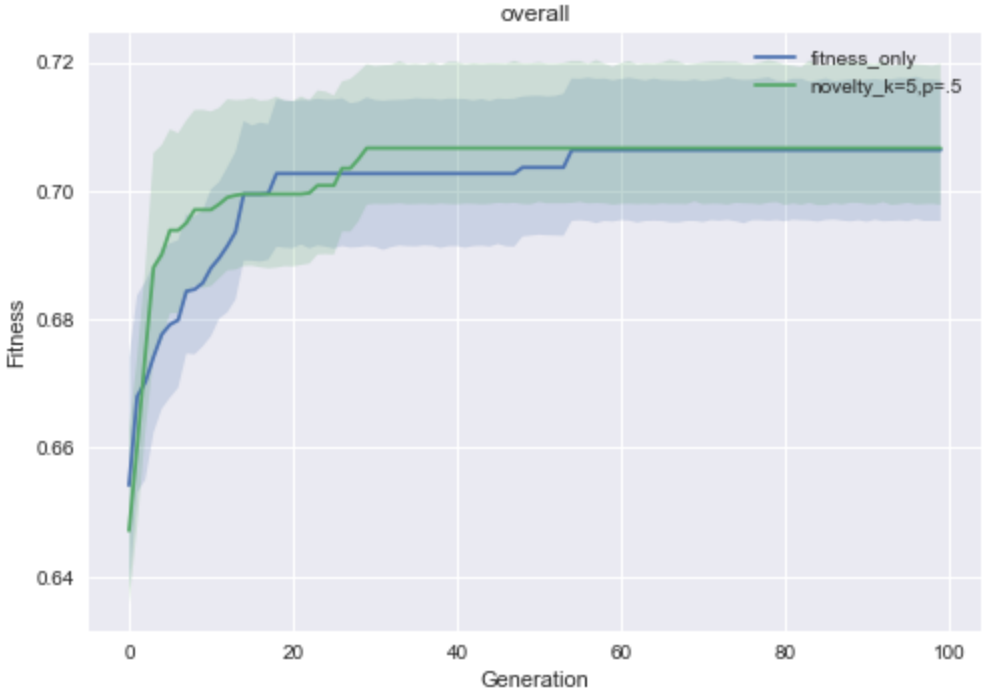
```
# plotting
plot_mean_and_bootstrapped_ci_multiple(input_data=[np.transpose(x) for k, x in experiment_
plot_mean_and_bootstrapped_ci_multiple(input_data=[np.transpose(x) for k, x in diversity_r
```

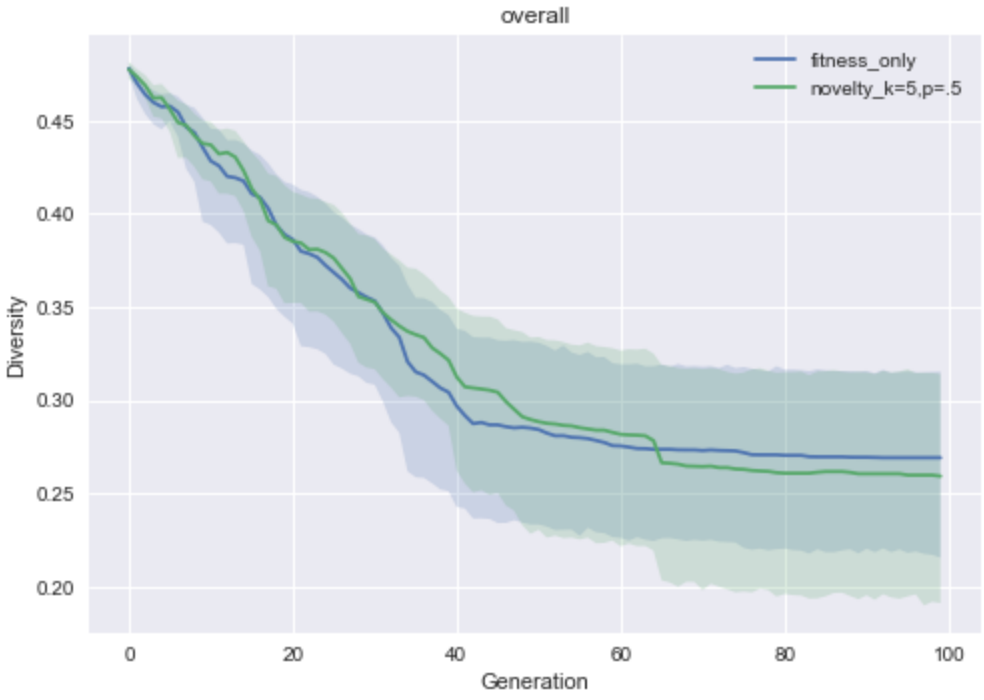
```
[array([0.63800426, 0.67397723]), array([0.65302885, 0.68372443]), array([0.65508335, 0.68
589875]), array([0.66240912, 0.68852299]), array([0.6660842 , 0.69111814]), array([0.66781
274, 0.69181633]), array([0.66935456, 0.69231612]), array([0.6746909 , 0.69635494]), array
([0.67455365, 0.69606818]), array([0.67578825, 0.69690078]), array([0.67739083, 0.7002109
7]), array([0.68010201, 0.70137469]), array([0.68110037, 0.70366298]), array([0.68310683,
0.70606705]), array([0.68914829, 0.71085256]), array([0.68912603, 0.7101579 ]), array([0.6
```

8898625, 0.71060629]), array([0.68926374, 0.71037813]), array([0.69131217, 0.71427396]), a
rray([0.69113525, 0.71405954]), array([0.69107709, 0.71394941]), array([0.69165595, 0.7141
49  ]), array([0.69103722, 0.71410392]), array([0.69149274, 0.71432483]), array([0.6913021
6, 0.71417759]), array([0.69101441, 0.71418738]), array([0.69140837, 0.71438364]), array
([0.69116547, 0.7136176 ]), array([0.69123306, 0.71382939]), array([0.69141494, 0.7145146
7]), array([0.691278  , 0.71412142]), array([0.69081548, 0.7139956 ]), array([0.69123953,
0.7141034 ]), array([0.69149175, 0.71413865]), array([0.69137662, 0.71420746]), array([0.6
9134895, 0.71439305]), array([0.69150718, 0.71430925]), array([0.69144047, 0.71432351]), a
rray([0.69112031, 0.71414518]), array([0.69139092, 0.71433429]), array([0.69139518, 0.7143
1358]), array([0.69114756, 0.71427343]), array([0.69138965, 0.71404248]), array([0.6914589
7, 0.71427484]), array([0.69131986, 0.71398459]), array([0.69172474, 0.71433677]), array
([0.6913636 , 0.71425109]), array([0.6913139 , 0.71433021]), array([0.69290076, 0.7146623
3]), array([0.69302859, 0.71475399]), array([0.69317879, 0.71479278]), array([0.6930824 ,
0.71466472]), array([0.69305207, 0.71474831]), array([0.69299597, 0.71509384]), array([0.6
9531963, 0.71747064]), array([0.69540831, 0.71741389]), array([0.69529277, 0.71760435]), a
rray([0.69496157, 0.71740563]), array([0.69531422, 0.7174542 ]), array([0.69522707, 0.7171
7824]), array([0.69543296, 0.71739146]), array([0.69537098, 0.71757687]), array([0.6950597
2, 0.71736362]), array([0.69530504, 0.71715327]), array([0.69534496, 0.71746079]), array
([0.69531045, 0.71725317]), array([0.69510286, 0.71721538]), array([0.69508875, 0.7171497
2]), array([0.69529588, 0.71708985]), array([0.69532479, 0.7173236 ]), array([0.69512931,
0.71706117]), array([0.69543723, 0.71721597]), array([0.69533838, 0.71731976]), array([0.6
954738 , 0.71765354]), array([0.69526173, 0.71736073]), array([0.69495075, 0.71723309]), a
rray([0.69541815, 0.71729122]), array([0.69542333, 0.71746784]), array([0.69532951, 0.7176
4558]), array([0.69545125, 0.71713335]), array([0.69525773, 0.71706744]), array([0.6955139
, 0.71744371]), array([0.69541918, 0.71744245]), array([0.6953887 , 0.71695579]), array
([0.69521347, 0.71720045]), array([0.69547316, 0.71750789]), array([0.6955422 , 0.7170529
5]), array([0.695366  , 0.71718387]), array([0.69524774, 0.7173744 ]), array([0.69541904,
0.71706939]), array([0.69518558, 0.71731081]), array([0.69549898, 0.7174044 ]), array([0.6
952591 , 0.71723226]), array([0.69531751, 0.71735699]), array([0.69539211, 0.71754411]), a
rray([0.69527408, 0.71748511]), array([0.6955994 , 0.71776232]), array([0.69530608, 0.7173
3831]), array([0.69532649, 0.71732928]), array([0.69529631, 0.71716077])]
[array([0.63580366, 0.66283505]), array([0.64784266, 0.67394739]), array([0.66268441, 0.68
805492]), array([0.67423118, 0.70589777]), array([0.67671799, 0.70710693]), array([0.68094
873, 0.70956637]), array([0.6810178 , 0.70896663]), array([0.68354279, 0.71094041]), array
([0.68526778, 0.7125646 ]), array([0.68489006, 0.71260038]), array([0.68518236, 0.7123258
5]), array([0.68640916, 0.7129023 ]), array([0.68761887, 0.71424585]), array([0.68774865,
0.7140914 ]), array([0.68829993, 0.71415348]), array([0.68833522, 0.71443723]), array([0.6
8810617, 0.71400236]), array([0.68792941, 0.71409358]), array([0.68804311, 0.71464888]), a
rray([0.68827694, 0.71404282]), array([0.6882232 , 0.71395862]), array([0.68847441, 0.7146
7809]), array([0.68867507, 0.71407809]), array([0.69010234, 0.71560067]), array([0.6901491
9, 0.71543171]), array([0.69007861, 0.71481801]), array([0.69367606, 0.71702739]), array
([0.69366301, 0.71738639]), array([0.6952846 , 0.71846622]), array([0.69785058, 0.7197043
4]), array([0.69798447, 0.71940803]), array([0.69792659, 0.71952275]), array([0.69795612,
0.7195984 ]), array([0.69805139, 0.7201734 ]), array([0.69790087, 0.71965507]), array([0.6
9792741, 0.71978034]), array([0.69797752, 0.71960444]), array([0.69789067, 0.71999357]), a
rray([0.69802376, 0.71950056]), array([0.69793981, 0.72007898]), array([0.6978977 , 0.7200
8165]), array([0.69800543, 0.71969093]), array([0.69798368, 0.71988946]), array([0.6978896
2, 0.71976829]), array([0.69795363, 0.71985535]), array([0.69813741, 0.71984905]), array
([0.69803718, 0.7197869 ]), array([0.69813324, 0.72020913]), array([0.69794072, 0.7198221
9]), array([0.6978984 , 0.71983684]), array([0.69828288, 0.72001367]), array([0.69805176,
0.71977781]), array([0.69805107, 0.71977769]), array([0.69805239, 0.7199053 ]), array([0.6
981217, 0.7202171]), array([0.69828536, 0.7201976 ]), array([0.6980297 , 0.71976428]), arr
ay([0.69810156, 0.7203525 ]), array([0.69803698, 0.71979415]), array([0.6978392, 0.719331
1]), array([0.69796312, 0.71988605]), array([0.69800137, 0.72020781]), array([0.69793334,
0.71966708]), array([0.69799751, 0.71983952]), array([0.6979087 , 0.71977405]), array([0.6
9804912, 0.71994419]), array([0.69795798, 0.7203762 ]), array([0.69797777, 0.72012645]), a
rray([0.6979312 , 0.71962229]), array([0.69836103, 0.71980501]), array([0.69791702, 0.7199
754 ]), array([0.69799591, 0.72010728]), array([0.69799164, 0.72015744]), array([0.6978417
1, 0.72006286]), array([0.69813991, 0.72008205]), array([0.69804833, 0.71972508]), array
([0.69800992, 0.71969446]), array([0.69789194, 0.71996512]), array([0.69816402, 0.7203511
9]), array([0.69825438, 0.72005955]), array([0.69778802, 0.71961039]), array([0.69804202,
0.71947487]), array([0.697814  , 0.71925051]), array([0.6981157 , 0.72030719]), array([0.6
9776822, 0.71947804]), array([0.69777175, 0.71949854]), array([0.69805394, 0.7197177 ]), a
rray([0.69787111, 0.72005317]), array([0.69814434, 0.72017205]), array([0.69812915, 0.7200
785 ]), array([0.69792436, 0.71991836]), array([0.69790146, 0.71971386]), array([0.6982570
6, 0.72030681]), array([0.69780377, 0.71940781]), array([0.69792556, 0.71963845]), array

([0.69812805, 0.71974081]), array([0.69788198, 0.71967836]), array([0.69792946, 0.7195703
7]), array([0.69798328, 0.71945828]), array([0.69781711, 0.71975743])]
[array([0.47481922, 0.48082097]), array([0.46164471, 0.47433208]), array([0.45365143, 0.46
901858]), array([0.44809788, 0.46619664]), array([0.44534521, 0.46409023]), array([0.44955
692, 0.46471735]), array([0.4414188 , 0.46295395]), array([0.42420192, 0.45849272]), array
([0.41717385, 0.45691533]), array([0.39578661, 0.45148831]), array([0.39400119, 0.4456770
5]), array([0.38988634, 0.44521671]), array([0.3840711 , 0.43953451]), array([0.38427291,
0.43958391]), array([0.38339232, 0.43780638]), array([0.36227419, 0.43393978]), array([0.3
5941801, 0.43165965]), array([0.35532802, 0.42707329]), array([0.34897551, 0.42122863]), a
rray([0.34473141, 0.41707984]), array([0.34109441, 0.41570092]), array([0.32891094, 0.4128
1198]), array([0.32836663, 0.41161379]), array([0.32727527, 0.409099  ]), array([0.3247687
3, 0.4055654 ]), array([0.31899929, 0.40096257]), array([0.31694887, 0.39799987]), array
([0.31418507, 0.39419809]), array([0.31277896, 0.39184928]), array([0.3092389 , 0.3889123
9]), array([0.30823487, 0.38735033]), array([0.29925001, 0.3830532 ]), array([0.28806413,
0.3764491 ]), array([0.28246454, 0.37213637]), array([0.26182488, 0.36206174]), array([0.2
5855658, 0.35486878]), array([0.25823441, 0.35465961]), array([0.25536883, 0.35265981]), a
rray([0.25205557, 0.34839616]), array([0.25117578, 0.34560312]), array([0.24261482, 0.3382
8483]), array([0.24184135, 0.33709739]), array([0.23639904, 0.33373918]), array([0.2364798
6, 0.33351136]), array([0.23620828, 0.3335443 ]), array([0.23398507, 0.33209539]), array
([0.23405735, 0.33180946]), array([0.23555566, 0.33218478]), array([0.23424469, 0.3312114
6]), array([0.23325486, 0.33139724]), array([0.23324988, 0.33080482]), array([0.2320932 ,
0.32902514]), array([0.23022126, 0.32655505]), array([0.23041908, 0.32928488]), array([0.2
3050717, 0.32557825]), array([0.22717515, 0.32420025]), array([0.23153079, 0.32583296]), a
rray([0.22905178, 0.32293589]), array([0.22859554, 0.32312547]), array([0.22648397, 0.3197
8125]), array([0.22603851, 0.3192179 ]), array([0.22462372, 0.31919688]), array([0.2264413
9, 0.31954314]), array([0.22508302, 0.3180967 ]), array([0.2243276 , 0.31815502]), array
([0.22584491, 0.31909406]), array([0.22595787, 0.31790274]), array([0.22522152, 0.3194215
]), array([0.22435667, 0.31810408]), array([0.22499141, 0.31832673]), array([0.22448637,
0.31800203]), array([0.22501543, 0.31830225]), array([0.22397709, 0.31725338]), array([0.2
2380756, 0.31897343]), array([0.22360104, 0.31721229]), array([0.22067091, 0.31776677]), a
rray([0.21967456, 0.31680032]), array([0.22034926, 0.31615564]), array([0.21932583, 0.3153
3314]), array([0.21999434, 0.31817328]), array([0.22032106, 0.31638552]), array([0.2194904
9, 0.31640565]), array([0.21926141, 0.31609353]), array([0.21760407, 0.31551022]), array
([0.21998693, 0.31652337]), array([0.21835327, 0.31595657]), array([0.21832656, 0.3155731
8]), array([0.21931246, 0.31557232]), array([0.21996254, 0.31555159]), array([0.2177214 ,
0.31536758]), array([0.22028583, 0.31565605]), array([0.21833058, 0.31442207]), array([0.2
2050441, 0.31624136]), array([0.21837965, 0.31469816]), array([0.21985769, 0.31507254]), a
rray([0.21802044, 0.31541639]), array([0.21878248, 0.31491657]), array([0.21748953, 0.3148
6581]), array([0.21697668, 0.31509907]), array([0.21546338, 0.31531707])]
[array([0.47235486, 0.48156922]), array([0.46870718, 0.4775403 ]), array([0.46235791, 0.47
446268]), array([0.4521627 , 0.46903754]), array([0.45123074, 0.46966732]), array([0.44411
784, 0.46507327]), array([0.42997812, 0.46065336]), array([0.43014164, 0.45870635]), array
([0.42523911, 0.45360598]), array([0.41848992, 0.45013696]), array([0.4173903 , 0.4492806
9]), array([0.41081038, 0.44495211]), array([0.41077672, 0.4457891 ]), array([0.4094887 ,
0.44451352]), array([0.40157145, 0.43886371]), array([0.38759623, 0.43289369]), array([0.3
8026151, 0.42821317]), array([0.36109466, 0.42151515]), array([0.35979577, 0.4202274 ]), a
rray([0.35194469, 0.41524297]), array([0.35217056, 0.41177081]), array([0.35223515, 0.4102
7805]), array([0.34747675, 0.40838061]), array([0.3476047 , 0.40804507]), array([0.3459198
, 0.40637411]), array([0.34215245, 0.40480291]), array([0.33599639, 0.40048149]), array
([0.33036226, 0.39477701]), array([0.32013695, 0.38913225]), array([0.31745811, 0.3883476
6]), array([0.31662549, 0.38641749]), array([0.311272  , 0.38133969]), array([0.30676059,
0.37811669]), array([0.30149275, 0.37497146]), array([0.30220921, 0.37107985]), array([0.3
0185675, 0.37049362]), array([0.30004191, 0.36753228]), array([0.29438336, 0.36358721]), a
rray([0.29009243, 0.35866654]), array([0.28636897, 0.35545526]), array([0.26223093, 0.3490
9729]), array([0.25522761, 0.34683031]), array([0.25050909, 0.34517912]), array([0.2517453
8, 0.34483133]), array([0.249101  , 0.34480733]), array([0.25026199, 0.34331539]), array
([0.24347242, 0.33861154]), array([0.24068161, 0.33747662]), array([0.23276783, 0.3364962
]), array([0.22826533, 0.33351356]), array([0.23044712, 0.33386306]), array([0.2261777 ,
0.33217274]), array([0.2272879, 0.3320468]), array([0.22590426, 0.3311358 ]), array([0.226
21334, 0.32974524]), array([0.22517113, 0.32932309]), array([0.22620368, 0.33083183]), arr
ay([0.22316012, 0.32894375]), array([0.22335808, 0.32848694]), array([0.22462116, 0.327819
49]), array([0.22151851, 0.32658899]), array([0.22297222, 0.32727719]), array([0.22159586,
0.32707921]), array([0.22169939, 0.32757412]), array([0.21925248, 0.32494669]), array([0.2
0321685, 0.31880635]), array([0.20214875, 0.31824989]), array([0.19980526, 0.31713178]), a
rray([0.20251803, 0.31677555]), array([0.20241487, 0.31710209]), array([0.19800409, 0.3165
7938]), array([0.20148654, 0.31717001]), array([0.19983868, 0.31661317]), array([0.1984030

6, 0.31510573]), array([0.19683207, 0.31414683]), array([0.19729563, 0.31474285]), array
([0.19680498, 0.316476  ]), array([0.19885132, 0.31703675]), array([0.19802866, 0.3157201
3]), array([0.19459095, 0.31465268]), array([0.19594385, 0.3149734 ]), array([0.1953577 ,
0.31494522]), array([0.19511202, 0.31529453]), array([0.19347828, 0.31330795]), array([0.1
9357911, 0.31535049]), array([0.19472483, 0.31544058]), array([0.19649743, 0.31611436]), a
rray([0.19665227, 0.31530454]), array([0.19483899, 0.31487467]), array([0.19316928, 0.3151
5671]), array([0.1952582 , 0.31494045]), array([0.19285065, 0.31395281]), array([0.1958581
4, 0.31569126]), array([0.19410244, 0.31473952]), array([0.1933778, 0.3153068]), array([0.
19464983, 0.3165787 ]), array([0.19573213, 0.31447043]), array([0.18979207, 0.31467966]),
array([0.19181143, 0.31458464]), array([0.19107475, 0.31397635])]




## Q9: Analysis

What happened (to both fitness and diversity)? Are you surprised? Why would this be?

**This looks very similar to the fitness based evolutionary algorithm. There is slightly higher fitness and the standard deviation seems to be a bit wider in regards to diversity. I am a bit surprised that this doesn't differ more significantly from the fitness-based evolutionary algorithm. I think that maybe it is similar because it injects the selection pressure back into the algorithm which narrows the exploration of novelty**

**search. It could also be a result of not testing novelty continuously and instead just using the initial novelty. All of these factors play a part in this outcome.**

## Q10: Balancing Novelty and Fitness

Let's run this again with a different balance of novelty vs. fitness. Please run it with 90% of survivors selected via novelty and just 10% selected via fitness, and also vise versa with just 10% novelty and 90% fitness at each generation. Which do you expect to work better?

**Judging from the previous charts, I think that the 90% fitness will work better than the 90% novelty since 100% novelty was not able to find very good solutions.**

## Q9b: Running and Visualization

Let's findout!

In [20]:

```
num_runs = 20
total_generations = 100
num_elements_to_mutate = 1
bit_string_length = 15
num_parents = 20
num_children = 20
crossover=False

novelty_k = 5
novelty_selection_prop = 0
max_archive_length = 100

n = bit_string_length
k = bit_string_length-1

for run_name in ["novelty_k=5,p=.1", "novelty_k=5,p=.9"]:
    if run_name == "novelty_k=5,p=.1":
        novelty_selection_prop = 0.1
    else:
        novelty_selection_prop = 0.9
    experiment_results[run_name] = np.zeros((num_runs, total_generations))
    solutions_results[run_name] = np.zeros((num_runs, total_generations, bit_string_length
    diversity_results[run_name] = np.zeros((num_runs, total_generations))
    for run_num in range(num_runs):
        landscape = Landscape(n=n, k=k)
        start_time = time.time()
        fitness_over_time, solutions_over_time, diversity_over_time = evolutionary_algorit
        experiment_results[run_name][run_num] = fitness_over_time
        solutions_results[run_name][run_num] = solutions_over_time
        diversity_results[run_name][run_num] = diversity_over_time
        print(run_name, run_num, time.time()-start_time, fitness_over_time[-1])
```

```
novelty_k=5,p=.1 0 2.1068193912506104 0.6945114137367722
novelty_k=5,p=.1 1 2.07879638671875 0.775179806422403
novelty_k=5,p=.1 2 1.8931350708007812 0.7148416850230094
novelty_k=5,p=.1 3 1.8210728168487549 0.7153837575965251
novelty_k=5,p=.1 4 2.359539747238159 0.6796332335668291
novelty_k=5,p=.1 5 1.8400890827178955 0.686632557636742
novelty_k=5,p=.1 6 3.948911666870117 0.6804958175325744
novelty_k=5,p=.1 7 3.286339282989502 0.7058112190602445
novelty_k=5,p=.1 8 2.0132389068603516 0.7050851081157946
novelty_k=5,p=.1 9 2.818434476852417 0.7039862336458136
novelty_k=5,p=.1 10 2.088804006576538 0.7175476509909788
novelty_k=5,p=.1 11 1.897639513015747 0.6891800369904755
novelty_k=5,p=.1 12 2.1893911361694336 0.699999079910343
```

```
novelty_k=5,p=.1 13 1.943178415298462 0.7275022872147943
novelty_k=5,p=.1 14 2.011737585067749 0.712489120238126
novelty_k=5,p=.1 15 2.5106685161590576 0.7074367359281942
novelty_k=5,p=.1 16 2.0242483615875244 0.6938856565810301
novelty_k=5,p=.1 17 1.8766210079193115 0.6980653983150764
novelty_k=5,p=.1 18 1.9226608276367188 0.6835664268578056
novelty_k=5,p=.1 19 3.9193851947784424 0.7298893075575698
novelty_k=5,p=.9 0 2.458122730255127 0.7178045970692232
novelty_k=5,p=.9 1 1.9902191162109375 0.7282373594253949
novelty_k=5,p=.9 2 3.2913432121276855 0.6974519151132089
novelty_k=5,p=.9 3 1.9226605892181396 0.7088867786331029
novelty_k=5,p=.9 4 1.8786227703094482 0.7384348188388995
novelty_k=5,p=.9 5 1.8651137351989746 0.6793159749730519
novelty_k=5,p=.9 6 1.76802659034729 0.7371026009665701
novelty_k=5,p=.9 7 1.9436790943145752 0.7032003436024279
novelty_k=5,p=.9 8 1.971703052520752 0.7142991929606789
novelty_k=5,p=.9 9 1.8741185665130615 0.7401877699435516
novelty_k=5,p=.9 10 2.8219375610351562 0.7135566190074145
novelty_k=5,p=.9 11 1.9421775341033936 0.6838074498769809
novelty_k=5,p=.9 12 2.25895094871521 0.6701175498979993
novelty_k=5,p=.9 13 2.5216782093048096 0.7379307446611443
novelty_k=5,p=.9 14 2.01173734664917 0.7054346476764376
novelty_k=5,p=.9 15 2.116328001022339 0.696307163734428
novelty_k=5,p=.9 16 4.227150917053223 0.6829174057828042
novelty_k=5,p=.9 17 2.164870262145996 0.7259241970534844
novelty_k=5,p=.9 18 2.489150047302246 0.7085636061342998
novelty_k=5,p=.9 19 2.577226161956787 0.7064426587535727
```

In [21]:

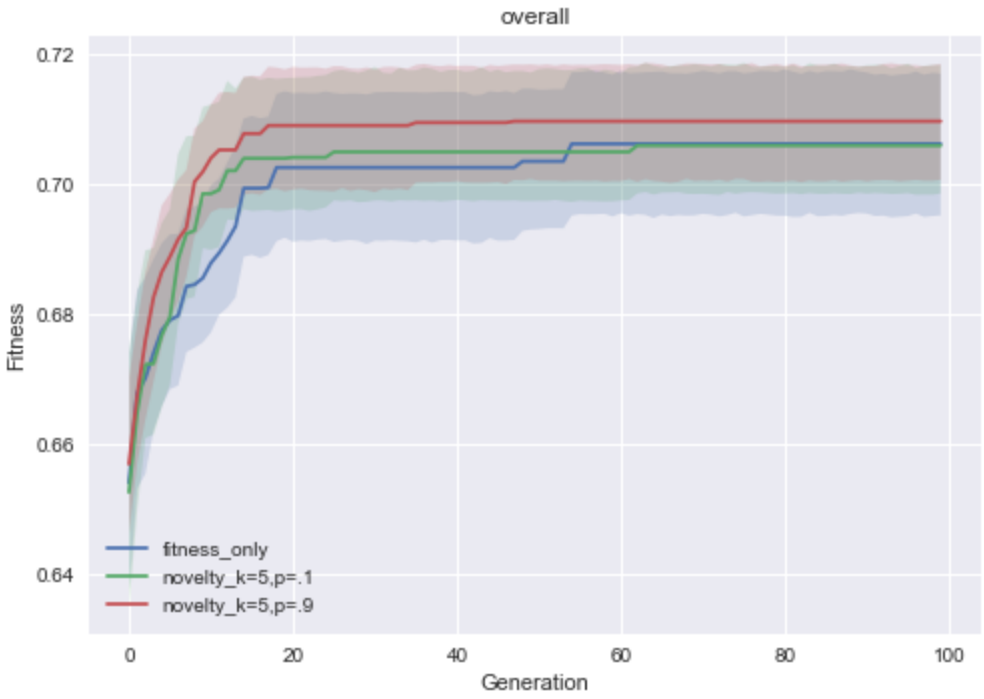
```
# plotting
plot_mean_and_bootstrapped_ci_multiple(input_data=[np.transpose(x) for k, x in experiment_
plot_mean_and_bootstrapped_ci_multiple(input_data=[np.transpose(x) for k, x in diversity_r
```

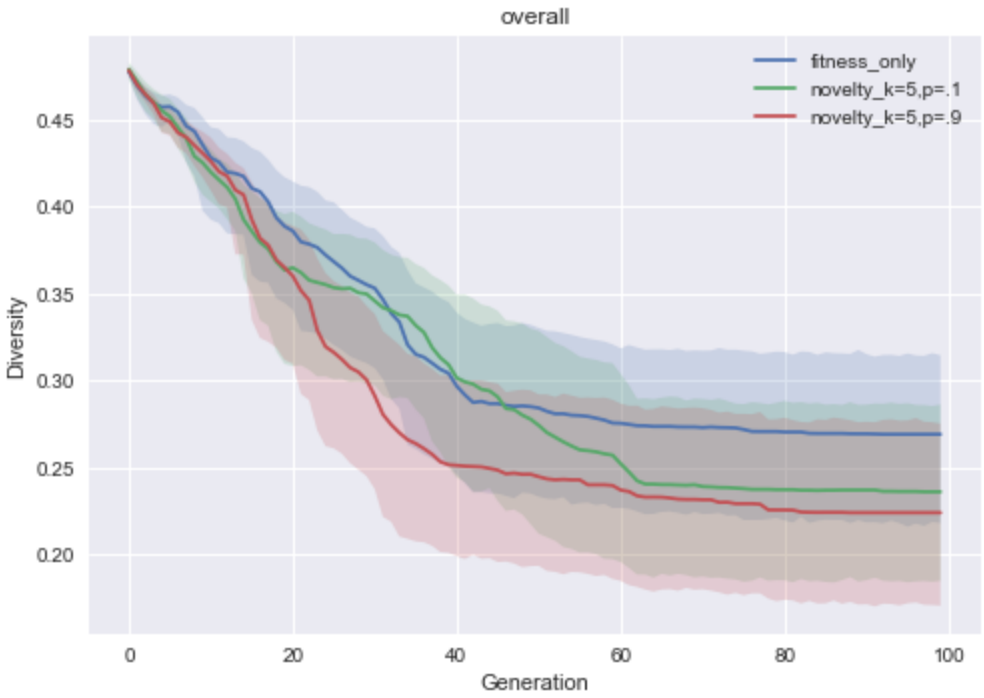
```
[array([0.63786504, 0.67405941]), array([0.6532683, 0.6838509]), array([0.65552689, 0.6858
1473]), array([0.66226739, 0.68858105]), array([0.66592399, 0.69100598]), array([0.6685593
4, 0.69238693]), array([0.66911393, 0.69246526]), array([0.67427599, 0.69627404]), array
([0.67496283, 0.69665519]), array([0.67575451, 0.69726221]), array([0.67731192, 0.6999559
8]), array([0.68005025, 0.70096579]), array([0.68112076, 0.70345583]), array([0.68272525,
0.70597895]), array([0.68895126, 0.7102251 ]), array([0.68923978, 0.71060782]), array([0.6
8877584, 0.71025253]), array([0.68922355, 0.71047899]), array([0.69134927, 0.71401215]), a
rray([0.69187021, 0.71439196]), array([0.69126535, 0.71411592]), array([0.69145139, 0.7141
6134]), array([0.69135901, 0.71410697]), array([0.69158166, 0.71411902]), array([0.6916768
7, 0.71436928]), array([0.69120861, 0.71412264]), array([0.69140125, 0.71400585]), array
([0.69154065, 0.71407529]), array([0.69138799, 0.71405536]), array([0.69091604, 0.7139009
9]), array([0.69142893, 0.71422879]), array([0.69139351, 0.71426033]), array([0.69148411,
0.71429733]), array([0.69139578, 0.71425823]), array([0.69132735, 0.71423824]), array([0.6
9113516, 0.71423955]), array([0.69156513, 0.71420232]), array([0.69110598, 0.71423059]), a
rray([0.69151108, 0.71430414]), array([0.69155588, 0.71431506]), array([0.69143356, 0.7142
763 ]), array([0.69147385, 0.71426194]), array([0.69098761, 0.71377841]), array([0.6910898
4, 0.71386796]), array([0.69180649, 0.7144041 ]), array([0.69142804, 0.7142255 ]), array
([0.69142854, 0.71444896]), array([0.69154014, 0.71412298]), array([0.69292996, 0.7148087
]), array([0.6931321 , 0.71480941]), array([0.69319516, 0.7146488 ]), array([0.69332446,
0.71465455]), array([0.6933417 , 0.71473954]), array([0.69332008, 0.71472769]), array([0.6
9529594, 0.71731162]), array([0.69550143, 0.71734539]), array([0.6954636 , 0.71753724]), a
rray([0.69549945, 0.71760803]), array([0.69556164, 0.71728828]), array([0.69513857, 0.7169
4954]), array([0.69564998, 0.71761445]), array([0.69526608, 0.71717783]), array([0.6955107
, 0.71723862]), array([0.69538073, 0.71730989]), array([0.69490373, 0.71738455]), array
([0.69543061, 0.71743678]), array([0.69519626, 0.71717255]), array([0.69536182, 0.7171837
2]), array([0.69527422, 0.7171066 ]), array([0.69531628, 0.71729612]), array([0.69557426,
0.71756712]), array([0.69541893, 0.71737462]), array([0.69514906, 0.71716863]), array([0.6
9536512, 0.71749474]), array([0.6957285 , 0.71750873]), array([0.69514158, 0.71716432]), a
rray([0.69533933, 0.71729555]), array([0.69555978, 0.71754705]), array([0.69541703, 0.7175
2604]), array([0.69527442, 0.71739143]), array([0.69525479, 0.71776852]), array([0.6950662
9, 0.7174137 ]), array([0.6950865 , 0.71709658]), array([0.69569761, 0.71752971]), array
([0.69517504, 0.71722145]), array([0.69521293, 0.71752744]), array([0.6953705 , 0.7174942
7]), array([0.69524952, 0.71743295]), array([0.69553835, 0.71777627]), array([0.69568972,
```

0.71748619]), array([0.69559198, 0.71727938]), array([0.69521469, 0.71693842]), array([0.6
9527988, 0.71708316]), array([0.69500741, 0.71681565]), array([0.69510514, 0.71698191]), a
rray([0.69520791, 0.71707773]), array([0.69559572, 0.71744507]), array([0.69506736, 0.7169
499 ]), array([0.69521783, 0.71724228]), array([0.69530897, 0.71711622])]
[array([0.63513096, 0.67410133]), array([0.65001367, 0.68304224]), array([0.66105035, 0.68
990553]), array([0.66159614, 0.69028482]), array([0.66577122, 0.69412715]), array([0.66923
27 , 0.69623974]), array([0.67869084, 0.70477867]), array([0.68241121, 0.70748246]), array
([0.68279366, 0.70758322]), array([0.69026552, 0.711961 ]), array([0.69001116, 0.7123631
7]), array([0.69045422, 0.71278167]), array([0.69467963, 0.71601514]), array([0.69454181,
0.71482961]), array([0.69626752, 0.71653352]), array([0.69599267, 0.71664146]), array([0.6
9607493, 0.7161816 ]), array([0.69614751, 0.71647208]), array([0.69596728, 0.71630747]), a
rray([0.69605562, 0.71641761]), array([0.69625591, 0.71648482]), array([0.69626855, 0.7165
5072]), array([0.69599725, 0.71656767]), array([0.69623142, 0.71677261]), array([0.6965462
8, 0.71690146]), array([0.6976371 , 0.71750783]), array([0.6975179 , 0.71767867]), array
([0.69762755, 0.71747414]), array([0.69733514, 0.71723899]), array([0.69742081, 0.7176138
1]), array([0.69762073, 0.71815986]), array([0.69761124, 0.71724984]), array([0.69747053,
0.71714269]), array([0.6976466 , 0.71801511]), array([0.69759433, 0.717596 ]), array([0.6
9764767, 0.71763025]), array([0.69755066, 0.7180722 ]), array([0.69760933, 0.71784817]), a
rray([0.69747733, 0.71738432]), array([0.69757316, 0.71765289]), array([0.69770393, 0.7177
1626]), array([0.69763532, 0.7174248 ]), array([0.69749493, 0.71755467]), array([0.6976589
2, 0.7181187 ]), array([0.69760328, 0.71742189]), array([0.69759932, 0.71772579]), array
([0.69757806, 0.71731728]), array([0.69764063, 0.71752397]), array([0.69752613, 0.7175264
7]), array([0.69753865, 0.7172755 ]), array([0.69756349, 0.71780248]), array([0.69755796,
0.71791043]), array([0.69770342, 0.71781678]), array([0.69752844, 0.71762414]), array([0.6
9739654, 0.71767658]), array([0.69759307, 0.71794048]), array([0.69753284, 0.71797504]), a
rray([0.69750614, 0.71727856]), array([0.69745765, 0.71775713]), array([0.69748469, 0.7175
3379]), array([0.69761325, 0.71714251]), array([0.69768579, 0.71811658]), array([0.6986870
7, 0.71821843]), array([0.69890331, 0.71878608]), array([0.69853171, 0.71834031]), array
([0.6988208 , 0.71810084]), array([0.69883143, 0.7182632 ]), array([0.69888658, 0.7183960
8]), array([0.69864918, 0.71839181]), array([0.69860493, 0.71858323]), array([0.69864269,
0.71841593]), array([0.69855143, 0.71822659]), array([0.69846757, 0.71858855]), array([0.6
9890488, 0.71851685]), array([0.69865021, 0.71844262]), array([0.69883086, 0.7187581 ]), a
rray([0.69888114, 0.7181941 ]), array([0.69883644, 0.71815257]), array([0.69874399, 0.7182
8144]), array([0.69866212, 0.71829193]), array([0.69870115, 0.71815997]), array([0.6987464
4, 0.71842867]), array([0.69857224, 0.71794741]), array([0.6987596, 0.7187215]), array([0.
69863895, 0.71837018]), array([0.69861565, 0.71834154]), array([0.69871972, 0.71868347]),
array([0.69863311, 0.71822235]), array([0.69860361, 0.71827589]), array([0.6988097 , 0.718
16444]), array([0.69865822, 0.7183909 ]), array([0.6987654 , 0.71845311]), array([0.698709
01, 0.71847744]), array([0.69860347, 0.71797186]), array([0.69864865, 0.71829724]), array
([0.69875075, 0.71865086]), array([0.69848196, 0.71798543]), array([0.69852257, 0.7183047
3]), array([0.69849736, 0.7184291 ]), array([0.6986902 , 0.71843497])]
[array([0.64639976, 0.67025713]), array([0.657195 , 0.6786516]), array([0.66386216, 0.6865
2476]), array([0.66974133, 0.691651 ]), array([0.67548733, 0.69687841]), array([0.6792551
, 0.69864386]), array([0.68200154, 0.70083518]), array([0.685821 , 0.70227247]), array
([0.69219224, 0.708699 ]), array([0.6936051 , 0.70981956]), array([0.69580349, 0.7124299
4]), array([0.69638855, 0.71416216]), array([0.69635324, 0.71429899]), array([0.69649903,
0.71413478]), array([0.69847952, 0.71663368]), array([0.69870607, 0.71680365]), array([0.6
9846317, 0.71670655]), array([0.69932365, 0.7181808 ]), array([0.69940853, 0.71805926]), a
rray([0.69900846, 0.71777914]), array([0.69927884, 0.71816144]), array([0.69940897, 0.7180
3662]), array([0.69919297, 0.71806874]), array([0.69911676, 0.7180818 ]), array([0.6990984
3, 0.717877 ]), array([0.69899525, 0.71827361]), array([0.69949996, 0.71821903]), array
([0.69906453, 0.71792587]), array([0.699335 , 0.71784876]), array([0.69917265, 0.7180931
1]), array([0.69888591, 0.71776975]), array([0.69941401, 0.71813426]), array([0.69926845,
0.71817544]), array([0.6994338 , 0.71805319]), array([0.69918406, 0.71789959]), array([0.7
0035328, 0.71846048]), array([0.70039894, 0.7186229 ]), array([0.70037916, 0.71838883]), a
rray([0.70046309, 0.71825548]), array([0.70044345, 0.71844129]), array([0.70057151, 0.7183
0415]), array([0.70038891, 0.71836098]), array([0.70034214, 0.71831973]), array([0.7003332
1, 0.71803169]), array([0.70037707, 0.71845797]), array([0.700456 , 0.71830421]), array
([0.70013077, 0.71842381]), array([0.7007758 , 0.71853117]), array([0.70071159, 0.7184932
2]), array([0.70092367, 0.71854681]), array([0.70066358, 0.71842889]), array([0.7006555 ,
0.71855248]), array([0.70066552, 0.71836538]), array([0.70056541, 0.71854524]), array([0.7
0064245, 0.71843462]), array([0.70065675, 0.71847697]), array([0.70061862, 0.71843339]), a
rray([0.70068699, 0.71850403]), array([0.7005056 , 0.71858626]), array([0.70060748, 0.7185
1979]), array([0.70058478, 0.71846468]), array([0.7008968 , 0.71860248]), array([0.7004819
4, 0.71844053]), array([0.7008829 , 0.71876454]), array([0.70061685, 0.71869324]), array
([0.70050196, 0.71852638]), array([0.70031121, 0.71819295]), array([0.70047576, 0.7184141

5]), array([0.70094508, 0.71868791]), array([0.70061312, 0.71866724]), array([0.70068657, 0.71860208]), array([0.70062955, 0.71855078]), array([0.70082192, 0.71860237]), array([0.7 005199 , 0.71830542]), array([0.70094254, 0.7183217 ]), array([0.7009095, 0.7185591]), arr ay([0.70051093, 0.71873095]), array([0.70047842, 0.71822414]), array([0.70056616, 0.718405 46]), array([0.70044805, 0.71861884]), array([0.7005414 , 0.71829516]), array([0.70066568, 0.71869166]), array([0.70039532, 0.71825199]), array([0.70069546, 0.71829259]), array([0.7 0053655, 0.71846642]), array([0.70056352, 0.718487 ]), array([0.70074125, 0.71857211]), a rray([0.70048689, 0.71868171]), array([0.70064561, 0.71837886]), array([0.70074961, 0.7182 0049]), array([0.70054913, 0.71827625]), array([0.70072214, 0.71830416]), array([0.7007669 3, 0.71860134]), array([0.70072766, 0.71872004]), array([0.70056316, 0.71852952]), array ([0.70066718, 0.71848276]), array([0.70079425, 0.71841163]), array([0.7006612 , 0.7182702 1]), array([0.70065129, 0.71843216]), array([0.70079938, 0.71862807])]
[array([0.47479355, 0.48074938]), array([0.4618793 , 0.47428798]), array([0.45354577, 0.46 916733]), array([0.44778776, 0.4660896 ]), array([0.44592224, 0.46417293]), array([0.44945 561, 0.46481239]), array([0.44151854, 0.46296784]), array([0.42572513, 0.45871511]), array ([0.41766973, 0.45706659]), array([0.39785893, 0.45160375]), array([0.39266338, 0.4458418 9]), array([0.3906243 , 0.44489241]), array([0.38471089, 0.44010197]), array([0.38426943, 0.43966607]), array([0.38400681, 0.43772928]), array([0.36092656, 0.43394125]), array([0.3 5699882, 0.43189394]), array([0.35637327, 0.42724766]), array([0.34730616, 0.42116452]), a rray([0.34476989, 0.41706361]), array([0.34079457, 0.41462304]), array([0.32860461, 0.4121 8755]), array([0.32809948, 0.41139722]), array([0.32714656, 0.40869603]), array([0.3240399 6, 0.40374637]), array([0.32051977, 0.40266044]), array([0.3176219 , 0.39835644]), array ([0.31389943, 0.39384668]), array([0.31183605, 0.39139288]), array([0.31001062, 0.3885192 ]), array([0.30722614, 0.38764454]), array([0.300762 , 0.38322217]), array([0.28944912, 0.37707179]), array([0.28393823, 0.37353244]), array([0.26307986, 0.36355462]), array([0.2 605388 , 0.35595434]), array([0.25740021, 0.35361927]), array([0.25493215, 0.35123648]), a rray([0.25239602, 0.34751584]), array([0.25123039, 0.34515832]), array([0.2459339, 0.33911 99]), array([0.24089332, 0.33542446]), array([0.23550975, 0.33337771]), array([0.23567217, 0.33153913]), array([0.2350802 , 0.33257488]), array([0.23490399, 0.3331394 ]), array([0.2 3359381, 0.33129894]), array([0.23227524, 0.33167686]), array([0.23547257, 0.33310432]), a rray([0.23496302, 0.33211905]), array([0.2331176, 0.3303163]), array([0.23202033, 0.328621 94]), array([0.23168116, 0.32833556]), array([0.23295484, 0.32733262]), array([0.22859214, 0.32630131]), array([0.22947164, 0.32510034]), array([0.22973935, 0.32412654]), array([0.2 2830385, 0.32333887]), array([0.22903262, 0.32244758]), array([0.22661611, 0.32082814]), a rray([0.22666957, 0.31854393]), array([0.226192 , 0.32044921]), array([0.2246645 , 0.3190 2891]), array([0.22299969, 0.31773624]), array([0.22323119, 0.31793366]), array([0.2243956 2, 0.31796672]), array([0.22462938, 0.31750756]), array([0.22382192, 0.3182486 ]), array ([0.22268629, 0.31699535]), array([0.22430724, 0.31735366]), array([0.22309644, 0.3171845 1]), array([0.22328089, 0.31727398]), array([0.22382704, 0.3186522 ]), array([0.22223788, 0.31770157]), array([0.22247783, 0.31776255]), array([0.2196794, 0.3171571]), array([0.219 70961, 0.31675928]), array([0.21976691, 0.31540775]), array([0.22017051, 0.3159423 ]), arr ay([0.22090991, 0.31663806]), array([0.21942888, 0.316365 ]), array([0.21826998, 0.314851 32]), array([0.22156404, 0.3167201 ]), array([0.21796175, 0.31645661]), array([0.21930389, 0.31478345]), array([0.21991961, 0.31623998]), array([0.21969825, 0.31572886]), array([0.2 1972331, 0.3155788 ]), array([0.21796481, 0.3153142 ]), array([0.21961141, 0.3176172 ]), a rray([0.21832256, 0.31510574]), array([0.21843808, 0.31374297]), array([0.21782676, 0.3147 0458]), array([0.21935602, 0.31477104]), array([0.21787916, 0.31418888]), array([0.2158511 , 0.31558775]), array([0.21769009, 0.314412 ]), array([0.21626943, 0.31422935]), array ([0.21861776, 0.31575647]), array([0.21799485, 0.3145339 ])]
[array([0.47350516, 0.48298598]), array([0.46293113, 0.47778479]), array([0.45521131, 0.47 209728]), array([0.45045072, 0.46783303]), array([0.44319632, 0.46459547]), array([0.44200 334, 0.46215444]), array([0.43377943, 0.45641015]), array([0.42585578, 0.45108024]), array ([0.41468779, 0.44255648]), array([0.40701622, 0.43823066]), array([0.40302024, 0.4332110 2]), array([0.39910234, 0.42778328]), array([0.39285116, 0.42508228]), array([0.38635467, 0.41942375]), array([0.35867061, 0.41279331]), array([0.35020102, 0.40873576]), array([0.3 3482614, 0.40516214]), array([0.33164435, 0.40184144]), array([0.31656878, 0.39779561]), a rray([0.3095216 , 0.39585195]), array([0.30840141, 0.39717803]), array([0.30882973, 0.3943 7943]), array([0.30615128, 0.39055968]), array([0.30234025, 0.38863577]), array([0.3024245 5, 0.38699818]), array([0.30077169, 0.38427455]), array([0.3009464 , 0.38446971]), array ([0.29911286, 0.38400619]), array([0.2994599 , 0.38187282]), array([0.29836999, 0.3816825 4]), array([0.29728035, 0.37949147]), array([0.29121197, 0.37516283]), array([0.28922785, 0.37243121]), array([0.29052618, 0.37109913]), array([0.28996947, 0.3707004 ]), array([0.2 8342144, 0.36702749]), array([0.27829337, 0.36484886]), array([0.26492349, 0.36055818]), a rray([0.25874278, 0.35791228]), array([0.25030201, 0.35466794]), array([0.24409934, 0.3493 1563]), array([0.24144379, 0.34949697]), array([0.23614034, 0.34723595]), array([0.236205 , 0.34381433]), array([0.23806566, 0.34265974]), array([0.23068928, 0.34019166]), array

([0.22492982, 0.33461193]), array([0.22574695, 0.33409925]), array([0.22291389, 0.3328715
5]), array([0.21734422, 0.32985279]), array([0.21179835, 0.32853452]), array([0.20981497,
0.32505907]), array([0.20777868, 0.32161037]), array([0.20416381, 0.31976614]), array([0.2
0302979, 0.31685173]), array([0.20094346, 0.31370764]), array([0.20102301, 0.31268216]), a
rray([0.20005776, 0.31185801]), array([0.19739245, 0.31045633]), array([0.19715982, 0.3099
905 ]), array([0.19548345, 0.30378552]), array([0.19215267, 0.29737065]), array([0.1883211
8, 0.28944327]), array([0.18823207, 0.28968552]), array([0.18651283, 0.28918568]), array
([0.18874133, 0.29013722]), array([0.18768635, 0.2907482 ]), array([0.18869796, 0.2904762
]), array([0.18790996, 0.28845671]), array([0.18665285, 0.28959304]), array([0.1871957 ,
0.28930919]), array([0.18640554, 0.28881244]), array([0.18658924, 0.28958604]), array([0.1
8591971, 0.28819513]), array([0.18546875, 0.28670045]), array([0.18416518, 0.28641028]), a
rray([0.18473502, 0.28750777]), array([0.18659212, 0.28704743]), array([0.18472938, 0.2862
1952]), array([0.18510235, 0.28713889]), array([0.18496359, 0.28827535]), array([0.1839013
5, 0.28709526]), array([0.18572962, 0.2875609 ]), array([0.18445902, 0.28670468]), array
([0.18473349, 0.28718231]), array([0.18398396, 0.28606985]), array([0.18370312, 0.2864320
1]), array([0.18508961, 0.28589593]), array([0.18685228, 0.28748474]), array([0.18421586,
0.28594408]), array([0.18511166, 0.28611461]), array([0.18646698, 0.28676876]), array([0.1
8474805, 0.28656911]), array([0.18441652, 0.28725069]), array([0.18461483, 0.28651474]), a
rray([0.18381941, 0.28692609]), array([0.18430935, 0.28595285]), array([0.1845765 , 0.2858
3723]), array([0.18416386, 0.28531971]), array([0.18473309, 0.28613189])]
[array([0.47593805, 0.4808333 ]), array([0.46649215, 0.47419738]), array([0.46048528, 0.46
907528]), array([0.45341185, 0.46477281]), array([0.44276928, 0.4585629 ]), array([0.44110
685, 0.45660133]), array([0.43396714, 0.45181939]), array([0.43090378, 0.44974652]), array
([0.42361642, 0.44593525]), array([0.4170489 , 0.44327469]), array([0.40947994, 0.4399025
9]), array([0.40145888, 0.43400372]), array([0.39966304, 0.43054218]), array([0.37267557,
0.42535067]), array([0.37254929, 0.422307  ]), array([0.33497524, 0.41402661]), array([0.3
2450727, 0.4058949 ]), array([0.3223224 , 0.40188384]), array([0.31889051, 0.39520971]), a
rray([0.31186889, 0.39081054]), array([0.30972646, 0.38903314]), array([0.29227551, 0.3870
5275]), array([0.28874728, 0.38119661]), array([0.27117212, 0.3702346 ]), array([0.2621666
1, 0.36185245]), array([0.26095703, 0.35721929]), array([0.25785111, 0.3544275 ]), array
([0.25241822, 0.35070505]), array([0.24868564, 0.34853039]), array([0.24496033, 0.3429441
6]), array([0.23750423, 0.33829793]), array([0.22151164, 0.32843134]), array([0.21867927,
0.32454797]), array([0.21021015, 0.32048921]), array([0.20886413, 0.31528974]), array([0.2
0738748, 0.31407678]), array([0.20686709, 0.30987804]), array([0.20481147, 0.30674021]), a
rray([0.20123036, 0.30294375]), array([0.20076912, 0.30338334]), array([0.19886313, 0.3003
0054]), array([0.19723719, 0.2994766 ]), array([0.19967297, 0.3002523 ]), array([0.1973804
2, 0.30016981]), array([0.19768727, 0.2991132 ]), array([0.19545888, 0.29823802]), array
([0.19579706, 0.29331984]), array([0.19462329, 0.29415273]), array([0.19249924, 0.2947560
1]), array([0.19312668, 0.29435621]), array([0.19330096, 0.29198716]), array([0.19037567,
0.29201576]), array([0.18982537, 0.29051323]), array([0.1884551 , 0.29245271]), array([0.1
8978063, 0.29134333]), array([0.1895401 , 0.29010415]), array([0.18866131, 0.29024344]), a
rray([0.18741228, 0.28993708]), array([0.18802885, 0.29022432]), array([0.18577856, 0.2889
3455]), array([0.18460842, 0.28668097]), array([0.18219663, 0.28545998]), array([0.1811268
8, 0.28556495]), array([0.1796245 , 0.28325783]), array([0.17845163, 0.28365534]), array
([0.17966749, 0.28512242]), array([0.17987978, 0.28457407]), array([0.17895475, 0.2838592
1]), array([0.18054654, 0.28424889]), array([0.17950721, 0.28446775]), array([0.17982128,
0.28403913]), array([0.17901437, 0.28311159]), array([0.17801244, 0.28145677]), array([0.1
766851 , 0.28210456]), array([0.17776143, 0.28079714]), array([0.17633515, 0.28083001]), a
rray([0.17744283, 0.28058213]), array([0.17746266, 0.28183745]), array([0.17441457, 0.2775
8896]), array([0.17295954, 0.2776947 ]), array([0.17376948, 0.2785934 ]), array([0.1733970
1, 0.27717567]), array([0.17163733, 0.27607872]), array([0.17314335, 0.27583271]), array
([0.17160545, 0.27763528]), array([0.17171242, 0.27733889]), array([0.17336847, 0.2786420
3]), array([0.1714186 , 0.27798777]), array([0.17252195, 0.27711803]), array([0.17099209,
0.27592618]), array([0.17120775, 0.27624447]), array([0.16990845, 0.27594204]), array([0.1
715142 , 0.27732532]), array([0.17131058, 0.27769026]), array([0.1705013 , 0.27686371]), a
rray([0.1717381 , 0.27698886]), array([0.17223049, 0.2783845 ]), array([0.17086613, 0.2762
8767]), array([0.17069886, 0.27596188]), array([0.17034395, 0.27518206])]




## Q10: Analysis

Did the experiment turn out the way you thought it would? Why or why not? What does this imply about the use of novelty vs. fitness in expoitation vs. exploration? Do the diveristy plots support this idea?

**It did not go as I thought it might. The 90% novelty (10% fitness), actually did quite well and better than any of the experiments so far. It had lower levels of diversity, but it had higher fitness than any of the previous cases. This implies that novelty can help exploit in certain situations. It might also need some some attention with regards to how novelty is injected to boost diversity in a population. It does okay at exploring the landscape and better at exploiting to do better than the rest of the cases in fitness. The diversity plots above are difficult to interpret, but I believe this may be one thing they portray.**

## Q11: The Effect of Ruggedness

How much do you think the conclusions you came to above are the result of the particular (maximally rugged) fitness landscape we experiemented with? What would happen if we used a much smoother landscale (e.g. a NK landscape with `K=0`)

**I think it is very dependent on the landscape. If we used something more smooth, a selection process with more fitness bias might do better. I think that the novelty bias above does well because of the ruggedness of the landscape.**

## Q12: Experiementation

Let's find out! Please pick your best ratio of novelty vs. fitness selection, and compare it to purely fitness and purely novelty selection on a NK landscale with `K=0`. Please plot your results.

In [22]:

```
num_runs = 20
total_generations = 100
num_elements_to_mutate = 1
bit_string_length = 15
num_parents = 20
num_children = 20

novelty_k = 5
novelty_selection_prop = 0
max_archive_length = 100

n = bit_string_length
k = 0

for run_name in ["novelty_k=0", "novelty_k=0,p=.9", "fitness_k=0"]:
    if run_name == "novelty_k=0,p=.9":
        novelty_selection_prop = 0.9
    elif run_name == "novelty_k=0":
        novelty_selection_prop = 1
    else:
        novelty_selection_prop = 0
    experiment_results[run_name] = np.zeros((num_runs, total_generations))
    solutions_results[run_name] = np.zeros((num_runs, total_generations, bit_string_length
    diversity_results[run_name] = np.zeros((num_runs, total_generations))
    for run_num in range(num_runs):
        landscape = Landscape(n=n, k=k)
        start_time = time.time()
        fitness_over_time, solutions_over_time, diversity_over_time = evolutionary_algorit
        experiment_results[run_name][run_num] = fitness_over_time
        solutions_results[run_name][run_num] = solutions_over_time
        diversity_results[run_name][run_num] = diversity_over_time
        print(run_name, run_num, time.time()-start_time, fitness_over_time[-1])
```

```
novelty_k=0 0 0.307265043258667 0.7226598855742812
novelty_k=0 1 0.23270106315612793 0.6232785786905399
novelty_k=0 2 0.31277012825012207 0.6132617575964402
novelty_k=0 3 0.20017313957214355 0.669446471889682
novelty_k=0 4 0.20467686653137207 0.7017557270315685
novelty_k=0 5 0.1991720199584961 0.657482901519484
novelty_k=0 6 0.19466757774353027 0.6408114471614857
novelty_k=0 7 0.26372766494750977 0.7407483639507569
novelty_k=0 8 0.26372814178466797 0.6969967900443694
novelty_k=0 9 0.2682313919067383 0.7066916404344601
novelty_k=0 10 0.3092670440673828 0.5815655266414337
novelty_k=0 11 0.2532191276550293 0.7399312846194778
novelty_k=0 12 0.2261950969696045 0.6057963704694306
novelty_k=0 13 0.21968984603881836 0.6938081075258454
novelty_k=0 14 0.3117692470550537 0.6115423292102486
novelty_k=0 15 0.30876636505126953 0.5493551586059253
```

```
novelty_k=0 16 0.2877485752105713 0.7214114898474361
novelty_k=0 17 0.30776548385620117 0.7699956855671544
novelty_k=0 18 0.21168279647827148 0.6425245402952999
novelty_k=0 19 0.30626463890075684 0.819831295643823
novelty_k=0,p=.9 0 0.2306993007659912 0.6801590337334176
novelty_k=0,p=.9 1 0.2382056713104248 0.6259778732930381
novelty_k=0,p=.9 2 0.3187754154205322 0.5512841648496085
novelty_k=0,p=.9 3 0.19767045974731445 0.7252213902252896
novelty_k=0,p=.9 4 0.19767117500305176 0.733756210630818
novelty_k=0,p=.9 5 0.2537193298339844 0.6101248768976948
novelty_k=0,p=.9 6 0.20017290115356445 0.628176611707067
novelty_k=0,p=.9 7 0.19466829299926758 0.6892621663986639
novelty_k=0,p=.9 8 0.21468544006347656 0.7381106877063536
novelty_k=0,p=.9 9 0.31577301025390625 0.7406019354482336
novelty_k=0,p=.9 10 0.20918059349060059 0.6038944467057343
novelty_k=0,p=.9 11 0.20217442512512207 0.6436624321844313
novelty_k=0,p=.9 12 0.19266605377197266 0.666112203933422
novelty_k=0,p=.9 13 0.25672197341918945 0.6498945092683021
novelty_k=0,p=.9 14 0.19867181777954102 0.5940729938961345
novelty_k=0,p=.9 15 0.24270939826965332 0.6596672986034233
novelty_k=0,p=.9 16 0.1986713409423828 0.6446758957284356
novelty_k=0,p=.9 17 0.19717049598693848 0.751780505499758
novelty_k=0,p=.9 18 0.22269225120544434 0.6384004274027771
novelty_k=0,p=.9 19 0.21018171310424805 0.6804562349794154
fitness_k=0 0 0.19867157936096191 0.571383154448381
fitness_k=0 1 0.26923274993896484 0.5546200895567241
fitness_k=0 2 0.20017266273498535 0.6979381832802946
fitness_k=0 3 0.30826640129089355 0.7317855607546077
fitness_k=0 4 0.2161865234375 0.665767646533822
fitness_k=0 5 0.32027673721313477 0.7113638359082362
fitness_k=0 6 0.1986713409423828 0.6764170917615215
fitness_k=0 7 0.25872325897216797 0.5946058018802453
fitness_k=0 8 0.2747373580932617 0.6702907351195043
fitness_k=0 9 0.20217442512512207 0.6405172585503139
fitness_k=0 10 0.24521183967590332 0.6797005874943967
fitness_k=0 11 0.24721360206604004 0.6730146214646057
fitness_k=0 12 0.201674222946167 0.5693877440627462
fitness_k=0 13 0.312269926071167 0.7018423412038707
fitness_k=0 14 0.24220919609069824 0.666045357524359
fitness_k=0 15 0.27423667907714844 0.6524631746313759
fitness_k=0 16 0.3237795829772949 0.6592816943183818
fitness_k=0 17 0.22169160842895508 0.6992135259573431
fitness_k=0 18 0.2977571487426758 0.7138273364947811
fitness_k=0 19 0.30326223373413086 0.5139220055737458
```

In [23]:

```
# plotting
plot_mean_and_bootstrapped_ci_multiple(input_data=[np.transpose(x) for k, x in experiment_
plot_mean_and_bootstrapped_ci_multiple(input_data=[np.transpose(x) for k, x in diversity_r
```

```
[array([0.63782975, 0.6738947 ]), array([0.65285318, 0.68359681]), array([0.65514393, 0.68
584905]), array([0.66226606, 0.68852986]), array([0.66571016, 0.69096825]), array([0.66796
669, 0.69193232]), array([0.66894709, 0.69224606]), array([0.67440422, 0.69602974]), array
([0.67507375, 0.69681935]), array([0.67588232, 0.69716916]), array([0.67746045, 0.7003233
7]), array([0.68012929, 0.70100822]), array([0.68103913, 0.70380722]), array([0.68320983,
0.70594352]), array([0.68884447, 0.71044474]), array([0.68917116, 0.71045808]), array([0.6
8896975, 0.71034678]), array([0.6893287 , 0.71032476]), array([0.69149761, 0.71448747]), a
rray([0.6910168 , 0.71395956]), array([0.69157175, 0.71424193]), array([0.69157387, 0.7141
1638]), array([0.69139766, 0.7144014 ]), array([0.69146088, 0.71410388]), array([0.6909766
1, 0.71394774]), array([0.69118187, 0.71430234]), array([0.69144891, 0.71469185]), array
([0.69133398, 0.71440555]), array([0.69159823, 0.71425792]), array([0.69141161, 0.7139608
9]), array([0.69123843, 0.71429145]), array([0.69156508, 0.71450701]), array([0.69133144,
0.71427278]), array([0.69147267, 0.71456343]), array([0.69131445, 0.71413227]), array([0.6
9161092, 0.71446984]), array([0.69149947, 0.71388892]), array([0.69172144, 0.71482643]), a
rray([0.6917458 , 0.71437212]), array([0.69126688, 0.71436939]), array([0.69136529, 0.7143
3577]), array([0.69138795, 0.71400188]), array([0.69153525, 0.71429133]), array([0.6911503
```

7, 0.71409439]), array([0.69161541, 0.71438515]), array([0.69165625, 0.7143183 ]), array
([0.69145041, 0.71398149]), array([0.69120577, 0.7141758 ]), array([0.69283432, 0.7147482
1]), array([0.69323059, 0.71501588]), array([0.69287551, 0.71498461]), array([0.69345088,
0.71478004]), array([0.69331234, 0.71480203]), array([0.6931489 , 0.71477659]), array([0.6
9557534, 0.71702477]), array([0.69538778, 0.71730159]), array([0.69520066, 0.71744663]), a
rray([0.69562209, 0.71766614]), array([0.69490885, 0.71723829]), array([0.69507792, 0.7172
6651]), array([0.69523578, 0.71715105]), array([0.69542472, 0.71725299]), array([0.6950262
9, 0.71717696]), array([0.6950959 , 0.71740803]), array([0.69567627, 0.71742392]), array
([0.69565254, 0.71731827]), array([0.69546017, 0.71727123]), array([0.69547823, 0.7175173
]), array([0.69551073, 0.7173662 ]), array([0.69548542, 0.71748338]), array([0.69534847,
0.71736538]), array([0.6954467 , 0.71739658]), array([0.69504419, 0.71714342]), array([0.6
9517213, 0.71705932]), array([0.69538268, 0.71714864]), array([0.6954339 , 0.71760874]), a
rray([0.69551377, 0.71732055]), array([0.69554951, 0.71745224]), array([0.6953063 , 0.7171
9847]), array([0.69541268, 0.71734363]), array([0.69563327, 0.71761122]), array([0.6954737
8, 0.71740609]), array([0.69513276, 0.71739688]), array([0.69535748, 0.71747671]), array
([0.69565274, 0.71754842]), array([0.69514207, 0.71689852]), array([0.6953775 , 0.7174065
2]), array([0.69548371, 0.71760274]), array([0.69558573, 0.71728262]), array([0.69535659,
0.71730441]), array([0.69528811, 0.71742463]), array([0.69529542, 0.7172005 ]), array([0.6
9536828, 0.71724239]), array([0.69516659, 0.71751136]), array([0.69521264, 0.71729117]), a
rray([0.6952322 , 0.71729634]), array([0.69544819, 0.71705135]), array([0.69546845, 0.7175
9149]), array([0.69546749, 0.71741482]), array([0.69540954, 0.71741069])]
[array([0.57449973, 0.63400701]), array([0.58503842, 0.64757253]), array([0.59684906, 0.65
621956]), array([0.60360423, 0.66421401]), array([0.61211253, 0.67200839]), array([0.61965
052, 0.67962357]), array([0.6239682 , 0.68369908]), array([0.62707847, 0.68851769]), array
([0.63409533, 0.6955533 ]), array([0.63667328, 0.69759185]), array([0.6377132 , 0.6988948
8]), array([0.64163913, 0.70157318]), array([0.64281449, 0.70108479]), array([0.64389534,
0.70265298]), array([0.64500982, 0.70451278]), array([0.64548364, 0.70339587]), array([0.6
467611 , 0.70450225]), array([0.64675762, 0.70550034]), array([0.64684388, 0.70492906]), a
rray([0.64735864, 0.70525725]), array([0.646913  , 0.70484378]), array([0.64672714, 0.7051
4587]), array([0.6472185 , 0.70537953]), array([0.64722008, 0.70587092]), array([0.6466031
, 0.70460231]), array([0.6477439 , 0.70644983]), array([0.64717461, 0.70454753]), array
([0.64707525, 0.70470679]), array([0.64729673, 0.70480421]), array([0.64685953, 0.7055519
7]), array([0.64671033, 0.704486  ]), array([0.6472621 , 0.70486547]), array([0.64740477,
0.70503622]), array([0.64718254, 0.70555505]), array([0.64710167, 0.70497475]), array([0.6
4666174, 0.70441486]), array([0.64689891, 0.70479613]), array([0.64725758, 0.70480303]), a
rray([0.64608552, 0.70397113]), array([0.64729441, 0.70509627]), array([0.64768514, 0.7061
0144]), array([0.6468327 , 0.70457755]), array([0.64683121, 0.70589659]), array([0.646821
2, 0.7051268]), array([0.64745009, 0.70503847]), array([0.64699459, 0.70540545]), array
([0.64739012, 0.70473895]), array([0.64789992, 0.70471961]), array([0.64780034, 0.7054256
3]), array([0.64697331, 0.70448248]), array([0.64751978, 0.70497853]), array([0.64663608,
0.70547488]), array([0.64711137, 0.70508247]), array([0.64714716, 0.70547231]), array([0.6
4641075, 0.70421316]), array([0.64720116, 0.705153  ]), array([0.64719756, 0.7056589 ]), a
rray([0.64709059, 0.70498815]), array([0.64680624, 0.70546095]), array([0.64702555, 0.7049
9225]), array([0.64677407, 0.70483041]), array([0.64693121, 0.70526952]), array([0.6468781
7, 0.70524841]), array([0.64751222, 0.70483702]), array([0.64740806, 0.70518497]), array
([0.64725016, 0.70511133]), array([0.64664111, 0.70523608]), array([0.64736519, 0.7056132
4]), array([0.64660551, 0.70515371]), array([0.64799976, 0.70542068]), array([0.64617285,
0.70414771]), array([0.64688146, 0.70569327]), array([0.64683913, 0.70596367]), array([0.6
4774162, 0.70557213]), array([0.64703771, 0.7046133 ]), array([0.64685107, 0.70550772]), a
rray([0.64703307, 0.70487142]), array([0.64696334, 0.70530739]), array([0.6453937 , 0.7043
7151]), array([0.64671362, 0.70527418]), array([0.64758601, 0.70471955]), array([0.6468520
3, 0.70539808]), array([0.64778843, 0.70579789]), array([0.64716222, 0.70457359]), array
([0.64719944, 0.70422494]), array([0.64680311, 0.70456231]), array([0.64669496, 0.7047779
5]), array([0.64779587, 0.70571443]), array([0.6466536 , 0.70493361]), array([0.64700769,
0.70525443]), array([0.64668872, 0.70438404]), array([0.64717065, 0.70487954]), array([0.6
4725339, 0.70504747]), array([0.6472791 , 0.70574428]), array([0.64729804, 0.70539334]), a
rray([0.64657712, 0.70431014]), array([0.64628004, 0.70510583]), array([0.6466399 , 0.7051
2336]), array([0.64728622, 0.70517257]), array([0.64642938, 0.70437161])]
[array([0.58063876, 0.62380866]), array([0.58520547, 0.63164369]), array([0.59816552, 0.64
758245]), array([0.60504281, 0.65233959]), array([0.60754013, 0.65416025]), array([0.61295
5  , 0.66175851]), array([0.61954204, 0.66691335]), array([0.62413031, 0.66957235]), array
([0.62861127, 0.67531936]), array([0.63105184, 0.67678824]), array([0.63204718, 0.6778635
4]), array([0.63505165, 0.68111224]), array([0.6360792 , 0.68320514]), array([0.6372543 ,
0.68416796]), array([0.63717986, 0.68476971]), array([0.63713334, 0.68502387]), array([0.6
3913572, 0.68485735]), array([0.63804315, 0.68595684]), array([0.63902436, 0.68600433]), a
rray([0.6394394 , 0.68657187]), array([0.63979662, 0.68682318]), array([0.64011949, 0.6865

877 ]), array([0.63959359, 0.68696972]), array([0.63986441, 0.68608586]), array([0.6397651
9, 0.68707104]), array([0.63873083, 0.68628298]), array([0.63951366, 0.68590813]), array
([0.63945888, 0.68647469]), array([0.63848123, 0.68600914]), array([0.63962346, 0.6864372
9]), array([0.63977515, 0.68632738]), array([0.64001143, 0.68657182]), array([0.63957865,
0.68673829]), array([0.64003303, 0.68550533]), array([0.63947418, 0.68666573]), array([0.6
3965946, 0.68651562]), array([0.63924576, 0.68674121]), array([0.63987139, 0.68558241]), a
rray([0.63918059, 0.68620636]), array([0.63860596, 0.68611204]), array([0.63973978, 0.6866
6928]), array([0.63914506, 0.68579236]), array([0.63916812, 0.68601676]), array([0.6386789
9, 0.68670415]), array([0.63965484, 0.68675377]), array([0.6389348 , 0.68671104]), array
([0.63909462, 0.68667732]), array([0.63899425, 0.68583846]), array([0.63887574, 0.6858273
]), array([0.63892231, 0.68564651]), array([0.63888685, 0.68678945]), array([0.63898816,
0.68601761]), array([0.63955853, 0.68642808]), array([0.63990651, 0.68656992]), array([0.6
3904957, 0.68653083]), array([0.63876307, 0.68627716]), array([0.63949757, 0.68641226]), a
rray([0.63944856, 0.68570647]), array([0.63895252, 0.68642804]), array([0.63901796, 0.6860
6739]), array([0.6391303 , 0.68617317]), array([0.64015573, 0.68602015]), array([0.6399099
2, 0.6860891 ]), array([0.63869796, 0.68546363]), array([0.63955157, 0.68595463]), array
([0.63940874, 0.68671682]), array([0.63934225, 0.68591168]), array([0.639611  , 0.6856813
7]), array([0.6395667 , 0.68587025]), array([0.63903431, 0.6866209 ]), array([0.64005449,
0.68643768]), array([0.63954292, 0.68647042]), array([0.6394404 , 0.68649437]), array([0.6
3976469, 0.68631672]), array([0.63980287, 0.68670298]), array([0.63905187, 0.68613947]), a
rray([0.64017628, 0.68680748]), array([0.63979461, 0.68587124]), array([0.63915969, 0.6863
7944]), array([0.63993034, 0.68604366]), array([0.63916936, 0.68692948]), array([0.6386310
5, 0.68673132]), array([0.64010013, 0.68647303]), array([0.63941173, 0.6866853 ]), array
([0.6399671 , 0.68630016]), array([0.63949624, 0.68728106]), array([0.63951333, 0.6866502
3]), array([0.63894739, 0.6857148 ]), array([0.63964309, 0.6863458 ]), array([0.63924919,
0.68632899]), array([0.63933272, 0.68590237]), array([0.63889413, 0.68641858]), array([0.6
3878614, 0.68666067]), array([0.63887266, 0.685841  ]), array([0.63957453, 0.68615207]), a
rray([0.63860175, 0.68597819]), array([0.63943737, 0.68622677]), array([0.63920102, 0.6858
6966]), array([0.63857117, 0.68586098]), array([0.63966911, 0.68629518])]
[array([0.56140715, 0.60975052]), array([0.56881483, 0.62094915]), array([0.5774264 , 0.63
174001]), array([0.58584121, 0.63569451]), array([0.59372939, 0.64695168]), array([0.59936
755, 0.65263193]), array([0.60492581, 0.65796938]), array([0.60907732, 0.66160807]), array
([0.60986549, 0.66397634]), array([0.61406697, 0.66765704]), array([0.6170605 , 0.6691105
6]), array([0.61789972, 0.67046033]), array([0.61958032, 0.67078899]), array([0.62087795,
0.67279621]), array([0.62231795, 0.67333501]), array([0.62120001, 0.67365291]), array([0.6
2155028, 0.67400485]), array([0.62262505, 0.6745609 ]), array([0.62268222, 0.67485262]), a
rray([0.62196745, 0.67392756]), array([0.62270002, 0.67510345]), array([0.62201193, 0.6738
5206]), array([0.62230695, 0.67469361]), array([0.62257172, 0.6748764 ]), array([0.6224156
2, 0.67427455]), array([0.62200297, 0.67475028]), array([0.62260022, 0.674814  ]), array
([0.62260307, 0.6736008 ]), array([0.62186886, 0.67429788]), array([0.62200149, 0.6741420
5]), array([0.62276206, 0.67492908]), array([0.62133812, 0.67450536]), array([0.62208004,
0.67405976]), array([0.6227696 , 0.67450381]), array([0.62237462, 0.6746365 ]), array([0.6
2215955, 0.67416617]), array([0.62179194, 0.6742693 ]), array([0.62208941, 0.67437129]), a
rray([0.62195152, 0.67405068]), array([0.62230371, 0.67464364]), array([0.62333558, 0.6748
8226]), array([0.62206948, 0.67425969]), array([0.6230322 , 0.67426355]), array([0.6229411
4, 0.67449655]), array([0.62310337, 0.67434911]), array([0.62246984, 0.6745141 ]), array
([0.62259479, 0.67479944]), array([0.62121374, 0.67407486]), array([0.62200178, 0.6740454
8]), array([0.62356728, 0.67476639]), array([0.62179377, 0.67432812]), array([0.62152297,
0.67451217]), array([0.62184272, 0.67476711]), array([0.62197343, 0.67433811]), array([0.6
2218513, 0.67427534]), array([0.62245826, 0.67437356]), array([0.62239952, 0.67404439]), a
rray([0.62237402, 0.67426288]), array([0.62233765, 0.6746992 ]), array([0.62204938, 0.6746
2421]), array([0.62263387, 0.67454892]), array([0.62244529, 0.6736993 ]), array([0.6229135
9, 0.67454848]), array([0.62258137, 0.67501674]), array([0.62238093, 0.67403306]), array
([0.62193305, 0.67447433]), array([0.62272027, 0.67466041]), array([0.62283518, 0.6743800
7]), array([0.62252462, 0.67458484]), array([0.62236004, 0.67410218]), array([0.62223464,
0.675402  ]), array([0.62288754, 0.67515208]), array([0.62315678, 0.67488555]), array([0.6
2234078, 0.67471848]), array([0.62277189, 0.67476219]), array([0.62246554, 0.67496437]), a
rray([0.62236872, 0.6746348 ]), array([0.62191767, 0.67392923]), array([0.62334124, 0.6748
7682]), array([0.62327497, 0.67462659]), array([0.62346416, 0.67438938]), array([0.6216221
8, 0.67464386]), array([0.62303895, 0.67423428]), array([0.62105144, 0.67366248]), array
([0.62212288, 0.67447173]), array([0.62324275, 0.6748436 ]), array([0.62222769, 0.6741925
5]), array([0.62183453, 0.67425341]), array([0.62240057, 0.67383372]), array([0.6221492 ,
0.67427701]), array([0.62149415, 0.6740645 ]), array([0.62214322, 0.67453766]), array([0.6
2314786, 0.67460987]), array([0.62298391, 0.67440613]), array([0.62257886, 0.67434644]), a
rray([0.62196958, 0.67426998]), array([0.62217342, 0.67480445]), array([0.62251444, 0.6743
1624]), array([0.62265328, 0.67432145]), array([0.62329953, 0.67435273])]

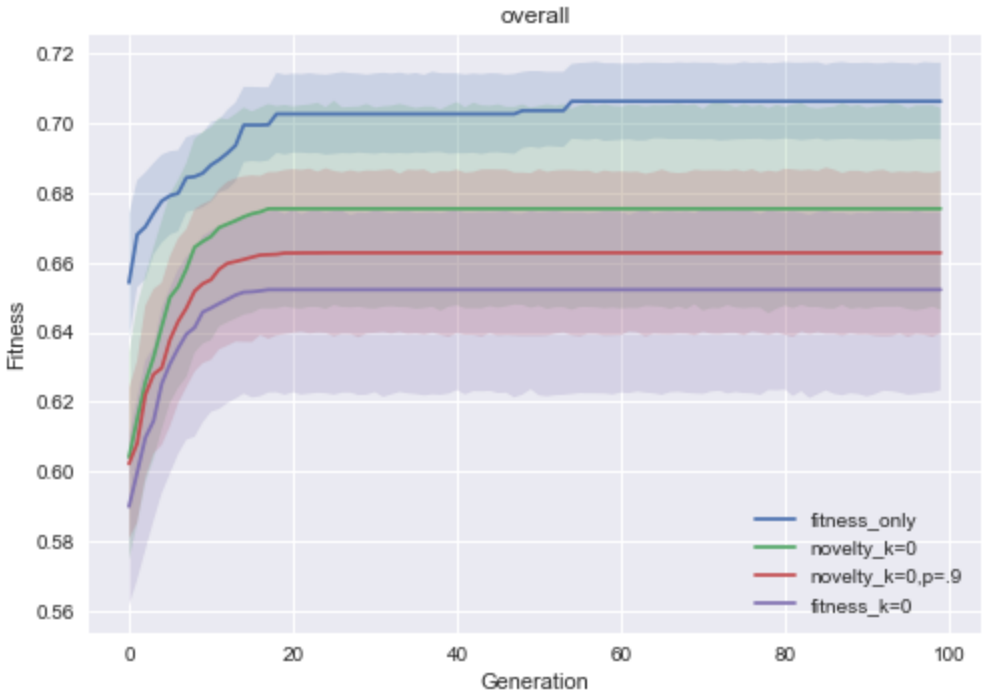
[array([0.47486487, 0.48084629]), array([0.4617868 , 0.47425032]), array([0.45343754, 0.46
92311 ]), array([0.44827288, 0.46616996]), array([0.44515705, 0.46412543]), array([0.44964
832, 0.46489255]), array([0.4417089 , 0.46300701]), array([0.42377669, 0.45848239]), array
([0.41675579, 0.45691256]), array([0.39553151, 0.45173466]), array([0.39327277, 0.4457742
3]), array([0.39039863, 0.44498183]), array([0.38402242, 0.44011115]), array([0.38430373,
0.44010384]), array([0.38444823, 0.43798412]), array([0.36125918, 0.43375982]), array([0.3
5968947, 0.43128429]), array([0.35634136, 0.42766142]), array([0.34707591, 0.42099961]), a
rray([0.34419105, 0.41655519]), array([0.34132766, 0.41533868]), array([0.33079399, 0.4121
8242]), array([0.32761681, 0.41124319]), array([0.32770912, 0.40962902]), array([0.3250095
9, 0.40431874]), array([0.3208054 , 0.40131833]), array([0.31896653, 0.3980009 ]), array
([0.31297199, 0.3938874 ]), array([0.31251484, 0.39168495]), array([0.30997891, 0.3886333
7]), array([0.30642915, 0.38730118]), array([0.30057099, 0.38265244]), array([0.29014455,
0.37802061]), array([0.28201991, 0.37197764]), array([0.26354601, 0.36145408]), array([0.2
5935609, 0.35590299]), array([0.25822912, 0.35478663]), array([0.25337883, 0.35260276]), a
rray([0.25366237, 0.34818242]), array([0.25152164, 0.34540296]), array([0.24561544, 0.3380
7148]), array([0.24047764, 0.33473684]), array([0.2352207 , 0.33272179]), array([0.2358530
5, 0.33444482]), array([0.23520886, 0.33239751]), array([0.23584057, 0.3331917 ]), array
([0.23508048, 0.33133862]), array([0.23375793, 0.33094773]), array([0.23328388, 0.3314733
]), array([0.23483463, 0.33251138]), array([0.23366458, 0.33022926]), array([0.2325097 ,
0.32819572]), array([0.22961119, 0.32650549]), array([0.23065515, 0.32793106]), array([0.2
3007624, 0.32653872]), array([0.23094607, 0.32730832]), array([0.23032317, 0.32608256]), a
rray([0.22937922, 0.32435605]), array([0.22829977, 0.32316981]), array([0.22867439, 0.3206
218 ]), array([0.22798706, 0.32055171]), array([0.22611221, 0.31955363]), array([0.2256534
1, 0.31802443]), array([0.22462235, 0.31950444]), array([0.22374681, 0.31643118]), array
([0.22434361, 0.31840738]), array([0.2229969 , 0.31782448]), array([0.22379818, 0.3181108
2]), array([0.22322215, 0.31825599]), array([0.22446651, 0.31765405]), array([0.22276184,
0.3171164 ]), array([0.22402707, 0.31791714]), array([0.22302186, 0.31683166]), array([0.2
2303012, 0.31764447]), array([0.22180265, 0.31868885]), array([0.2206833 , 0.31611311]), a
rray([0.21879175, 0.31603166]), array([0.21877441, 0.31681864]), array([0.2185935 , 0.3156
8182]), array([0.21833026, 0.31655671]), array([0.21871986, 0.31489696]), array([0.2194303
6, 0.3167506 ]), array([0.21852962, 0.31629952]), array([0.21806531, 0.31567319]), array
([0.21921642, 0.31546586]), array([0.21960616, 0.31669313]), array([0.217753  , 0.3150145
5]), array([0.21841279, 0.31466174]), array([0.21946804, 0.31708886]), array([0.2170171 ,
0.31413069]), array([0.21966311, 0.31556515]), array([0.21831501, 0.31574969]), array([0.2
1763206, 0.31506438]), array([0.21939193, 0.31470001]), array([0.21830566, 0.3143261 ]), a
rray([0.21811193, 0.3157325 ]), array([0.2202319 , 0.31533439]), array([0.21839655, 0.3144
8396]), array([0.21857155, 0.31482833]), array([0.21856382, 0.31485376])]
[array([0.45521453, 0.46589514]), array([0.41232105, 0.44024033]), array([0.36272316, 0.39
794129]), array([0.32977617, 0.36426593]), array([0.30010434, 0.33722949]), array([0.27348
733, 0.30385221]), array([0.25242026, 0.28404477]), array([0.23492016, 0.27386582]), array
([0.21336509, 0.24564704]), array([0.19484184, 0.22550981]), array([0.18995121, 0.2160033
7]), array([0.17657273, 0.20554324]), array([0.16381292, 0.19242824]), array([0.15129688,
0.1851779 ]), array([0.14428477, 0.174844  ]), array([0.13605112, 0.16458328]), array([0.1
2835589, 0.15738849]), array([0.11719722, 0.1440607 ]), array([0.10186257, 0.12959298]), a
rray([0.09583653, 0.12293099]), array([0.08318715, 0.11009835]), array([0.07177735, 0.0982
5742]), array([0.06766648, 0.09065316]), array([0.06199928, 0.08569888]), array([0.0537846
1, 0.07741394]), array([0.04828847, 0.07003181]), array([0.04398087, 0.06416247]), array
([0.04116635, 0.0596447 ]), array([0.03862719, 0.05278989]), array([0.03479749, 0.0465812
6]), array([0.03348444, 0.04534383]), array([0.03257092, 0.04411466]), array([0.03178555,
0.04245302]), array([0.03071429, 0.0411488 ]), array([0.02903761, 0.03845201]), array([0.0
2809268, 0.03574753]), array([0.02763386, 0.03557488]), array([0.02412868, 0.03306829]), a
rray([0.02330192, 0.0316388 ]), array([0.02297659, 0.03113072]), array([0.02237239, 0.0302
7458]), array([0.02241742, 0.03021078]), array([0.02085084, 0.02797438]), array([0.0204807
5, 0.02759084]), array([0.01995022, 0.02693208]), array([0.01970223, 0.02641509]), array
([0.01863989, 0.02538852]), array([0.01818005, 0.02476881]), array([0.01575346, 0.0232633
3]), array([0.01506794, 0.02220708]), array([0.01328641, 0.02107873]), array([0.01275205,
0.02015599]), array([0.01233544, 0.01958361]), array([0.01185901, 0.01881187]), array([0.0
1152567, 0.01833544]), array([0.01153835, 0.01806192]), array([0.01019234, 0.01710906]), a
rray([0.00819234, 0.01555124]), array([0.00819234, 0.01556203]), array([0.00653835, 0.0142
8851]), array([0.00555993, 0.01358361]), array([0.0053589, 0.0130021]), array([0.00536969,
0.01301289]), array([0.00490593, 0.01254914]), array([0.00390593, 0.01163452]), array([0.0
0317945, 0.01046586]), array([0.00245297, 0.00972859]), array([0.00217945, 0.00853835]), a
rray([0.00217945, 0.00863242]), array([0.00217945, 0.0087178 ]), array([0.00217945, 0.0085
3835]), array([0.00217945, 0.00881187]), array([0.00217945, 0.0087178 ]), array([0.0014529
7, 0.00781187]), array([0.00172648, 0.00808538]), array([0.00172648, 0.00799131]), array
([0.00172648, 0.00808538]), array([0.00145297, 0.00808538]), array([0.00145297, 0.0072648

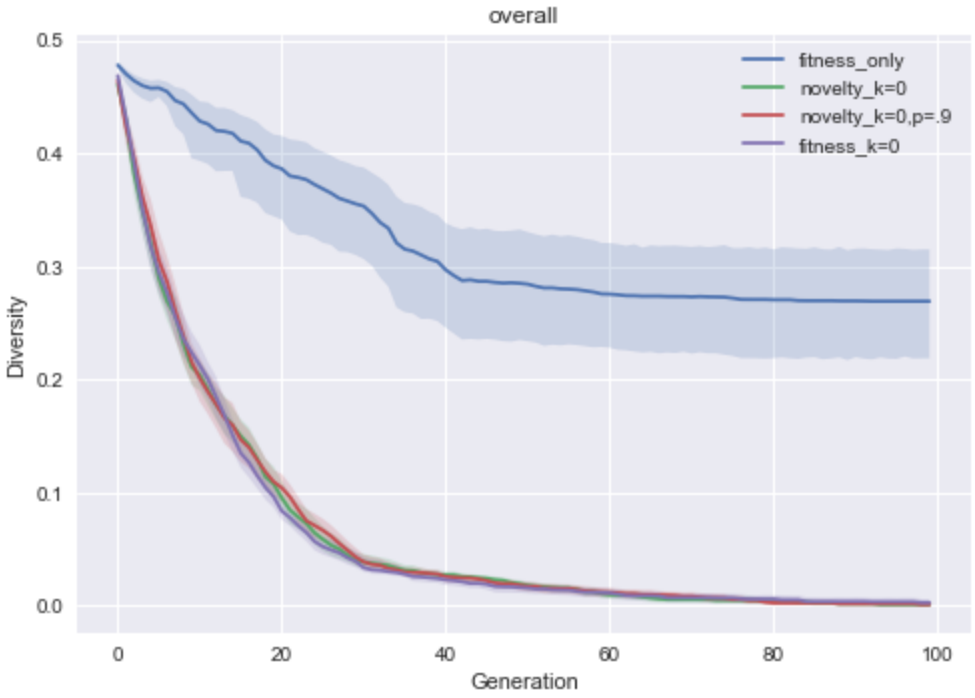
3]), array([0.00145297, 0.00653835]), array([0.00145297, 0.00653835]), array([0.00072648, 0.00581187]), array([0.00072648, 0.00581187]), array([0.00072648, 0.00508538]), array([0.00072648, 0.00508538]), array([0. , 0.0043589]), array([0. , 0.0043589]), array([0. , 0.0043589]), array([0. , 0.00363242]), array([0. , 0.00363242]), array([0. , 0.00290593]), array([0. , 0.00363242]), array([0. , 0.00363242]), array([0. , 0.00217945]), array([0. , 0.00217945]), array([0. , 0.00217945]), array([0. , 0.00217945]), array([0. , 0.00217945])]

[array([0.45710432, 0.46933652]), array([0.40850443, 0.4359832 ]), array([0.37833574, 0.41080882]), array([0.34392601, 0.37954616]), array([0.31080661, 0.35933851]), array([0.2800885 , 0.33084235]), array([0.26175652, 0.30890005]), array([0.23584424, 0.28435284]), array([0.21484597, 0.25730834]), array([0.19675763, 0.23140962]), array([0.1803452 , 0.21872255]), array([0.16906468, 0.20529621]), array([0.15656477, 0.1964861 ]), array([0.14324125, 0.18408433]), array([0.13595352, 0.17911583]), array([0.12684171, 0.16259019]), array([0.12132249, 0.15477382]), array([0.11257825, 0.14300342]), array([0.10507515, 0.13037532]), array([0.09781727, 0.12000972]), array([0.09139628, 0.11616233]), array([0.08440268, 0.10714917]), array([0.07201783, 0.09603826]), array([0.06356512, 0.08528567]), array([0.05947728, 0.08116553]), array([0.05609482, 0.07661516]), array([0.05186535, 0.07112153]), array([0.04642119, 0.06424157]), array([0.04152128, 0.05725723]), array([0.0363122 , 0.05095465]), array([0.03347716, 0.04490172]), array([0.03180935, 0.04279062]), array([0.03116126, 0.0414863 ]), array([0.02940432, 0.03845154]), array([0.0285242, 0.0364543]), array([0.02524351, 0.03364353]), array([0.02487477, 0.03323762]), array([0.0245533 , 0.03299895]), array([0.02383813, 0.03183355]), array([0.02341038, 0.03141212]), array([0.02063384, 0.02775139]), array([0.02029265, 0.02701521]), array([0.02014934, 0.026549 ]), array([0.01997266, 0.02641776]), array([0.01923738, 0.0254803 ]), array([0.01805982, 0.02407465]), array([0.01576679, 0.02280665]), array([0.01496341, 0.02217711]), array([0.01504274, 0.02215197]), array([0.0133298 , 0.02137769]), array([0.01222968, 0.02087363]), array([0.01211963, 0.02039953]), array([0.0105726 , 0.01916678]), array([0.0105726 , 0.01907271]), array([0.01050011, 0.01863053]), array([0.01028641, 0.01855993]), array([0.00947665, 0.0178698 ]), array([0.00836969, 0.01702368]), array([0.0080021 , 0.01639148]), array([0.00769223, 0.01582265]), array([0.00754914, 0.01547853]), array([0.0071452, 0.0147884]), array([0.0063589 , 0.01419234]), array([0.00636969, 0.01426483]), array([0.00563242, 0.01366877]), array([0.00570302, 0.01372859]), array([0.0053589 , 0.01292961]), array([0.00517945, 0.01253835]), array([0.00490593, 0.01208538]), array([0.0043589 , 0.01153835]), array([0.0043589, 0.011717 8]), array([0.00372648, 0.01108538]), array([0.00317945, 0.0103589 ]), array([0.00317945, 0.0103589 ]), array([0.00290593, 0.00999131]), array([0.00217945, 0.00881187]), array([0.00217945, 0.00881187]), array([0.00145297, 0.00653835]), array([0.00145297, 0.00653835]), array([0.00072648, 0.00508538]), array([0. , 0.0043589]), array([0. , 0.0043589]), array([0. , 0.0043589]), array([0. , 0.0043589]), array([0. , 0.0043589]), array([0. , 0.0043589]), array([0. , 0.0043589]), array([0. , 0.00363242]), array([0. , 0.00363242]), array([0. , 0.00363242]), array([0. , 0.00290593]), array([0. , 0.00363242]), array([0. , 0.00363242]), array([0. , 0.00363242]), array([0. , 0.00363242]), array([0. , 0.00363242]), array([0. , 0.00217945]), array([0. , 0.00217945])]

[array([0.46369541, 0.47291864]), array([0.42219126, 0.43976399]), array([0.37100047, 0.40536667]), array([0.33157293, 0.36712502]), array([0.2972351 , 0.33885196]), array([0.27549565, 0.31543517]), array([0.25801046, 0.29865487]), array([0.23741035, 0.27540482]), array([0.21967565, 0.25775493]), array([0.21097431, 0.2418451 ]), array([0.19426475, 0.23032637]), array([0.18172341, 0.21515356]), array([0.16548376, 0.20219274]), array([0.15228687, 0.18278938]), array([0.1362346 , 0.16441616]), array([0.12189969, 0.14714373]), array([0.11320254, 0.13969226]), array([0.10358403, 0.12630317]), array([0.09374632, 0.11567246]), array([0.08673358, 0.1064785 ]), array([0.07576376, 0.0921772 ]), array([0.07036385, 0.08543352]), array([0.0632079 , 0.07873345]), array([0.05663789, 0.0722208 ]), array([0.04881115, 0.06445439]), array([0.0450239 , 0.06072205]), array([0.04254389, 0.05806519]), array([0.03982911, 0.05553231]), array([0.0363915 , 0.05118762]), array([0.03367307, 0.04791 3]), array([0.02978802, 0.04237163]), array([0.02813589, 0.0404633 ]), array([0.02762048, 0.0397728 ]), array([0.02701939, 0.03707466]), array([0.02606841, 0.03595715]), array([0.02498572, 0.03227745]), array([0.02140137, 0.03103368]), array([0.02093058, 0.03063117]), array([0.02037321, 0.03024168]), array([0.01997663, 0.02951351]), array([0.01797333, 0.0277947 ]), array([0.01743709, 0.02666299]), array([0.01652996, 0.02544948]), array([0.01533578, 0.02272277]), array([0.01527562, 0.02278294]), array([0.01395308, 0.02220627]), array([0.01156388, 0.02107873]), array([0.01141683, 0.02054644]), array([0.01137364, 0.02033957]), array([0.01102368, 0.01990822]), array([0.0106237 , 0.01933355]), array([0.01008538, 0.01815122]), array([0.01015994, 0.01819755]), array([0.00936969, 0.01774144]), array([0.00903635, 0.01734434]), array([0.00883344, 0.01713064]), array([0.00790593, 0.01643863]), a

rray([0.00717945, 0.01595119]), array([0.00710696, 0.01555993]), array([0.00709617, 0.0153
1009]), array([0.00682265, 0.01501289]), array([0.00582265, 0.01410906]), array([0.0049059
3, 0.01290593]), array([0.00463242, 0.01245507]), array([0.00463242, 0.01246586]), array
([0.00490593, 0.01255993]), array([0.00463242, 0.01218155]), array([0.00336969, 0.0109188
2]), array([0.00317945, 0.01085712]), array([0.00336969, 0.01082265]), array([0.00336969,
0.01092961]), array([0.00336969, 0.01092961]), array([0.00317945, 0.01092961]), array([0.0
0317945, 0.0103589 ]), array([0.00317945, 0.0103589 ]), array([0.00317945, 0.0103589 ]), a
rray([0.00245297, 0.00981187]), array([0.00245297, 0.00908538]), array([0.00217945, 0.0090
8538]), array([0.00245297, 0.0093589 ]), array([0.00245297, 0.0093589 ]), array([0.0021794
5, 0.00881187]), array([0.002     , 0.00863242]), array([0.002     , 0.00863242]), array
([0.00072648, 0.00717945]), array([0.00072648, 0.00717945]), array([0.00072648, 0.0069059
3]), array([0.00072648, 0.00690593]), array([0.00072648, 0.00708538]), array([0.00072648,
0.00690593]), array([0.00072648, 0.00717945]), array([0.00072648, 0.00690593]), array([0.0
0072648, 0.0063589 ]), array([0.00072648, 0.0063589 ]), array([0.00072648, 0.00663242]), a
rray([0.00072648, 0.0063589 ]), array([0.00072648, 0.00663242]), array([0.00072648, 0.0065
3835]), array([0.        , 0.00563242]), array([0.        , 0.00563242])]




## Q12b: Analysis

Were you right? Was novelty, or novelty+fitness helpful? Was it harmful? What was the effect on diversity?

**The fitness-based approach does seem to work best here with regards to overall fitness in the 100th generation. All of the charts for fitness and diversity look very similar with diversity almost 0 by the 80th generations and fitness trending the same but reaching different values across all of the cases. It seems that injecting novelty in this situation is slightly harmful (at least in the manner that we are doing so). This is what I predicted above with an easier landscape, fitness-based solutions do better.**

## Q13: Future Work

In this assignment we explored just one (very simple) way to combine novelty and fitness, how else might you want to do this that could be more effetive (and why)?

**We could maintain novel subpopulations that compete based on fitness within their own groups. This could help by maintaining several highly fit, yet very diverse subpopulations of solutions.**

## Congratulations, you made it to the end!

Nice work -- and hopefully you're starting to get the hang of these!

Please save this file as a .ipynb, and also download it as a .pdf, uploading **both** to blackboard to complete this assignment.

For your submission, please make sure that you have renamed this file (and that the resulting pdf follows suit) to replce `[netid]` with your UVM netid. This will greatly simplify our grading pipeline, and make sure that you receive credit for your work.

### Academic Integrity Attribution

During this assignment I collaborated with:

**Just me**

In [ ]: