Evolutionary Permutations

Miles Jones

Permutation Patterns 2023, Dijon

Outline

- Motivation
- Optimization problems in Permutation Patterns
- Evolutionary Algorithms in general
- Application
- Examples

Motivation

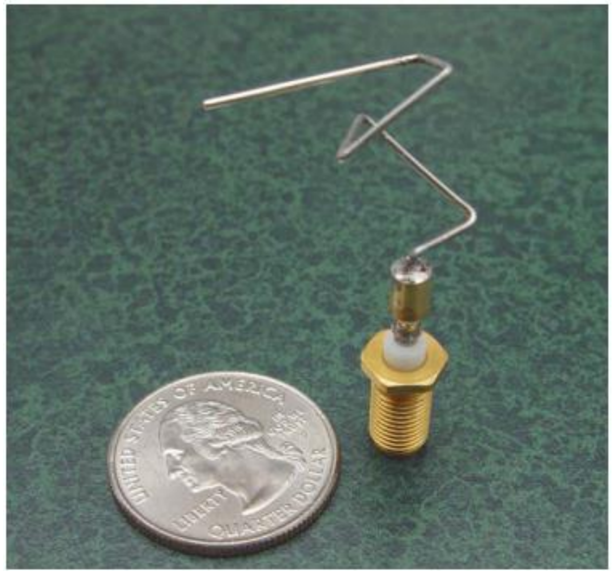
- Permutation Patterns with Jeff Remmel
- Teaching Computer Science and Exploring Evolutionary Algorithms
- Idea to apply evolutionary algorithms to optimization problems involving permutation patterns.

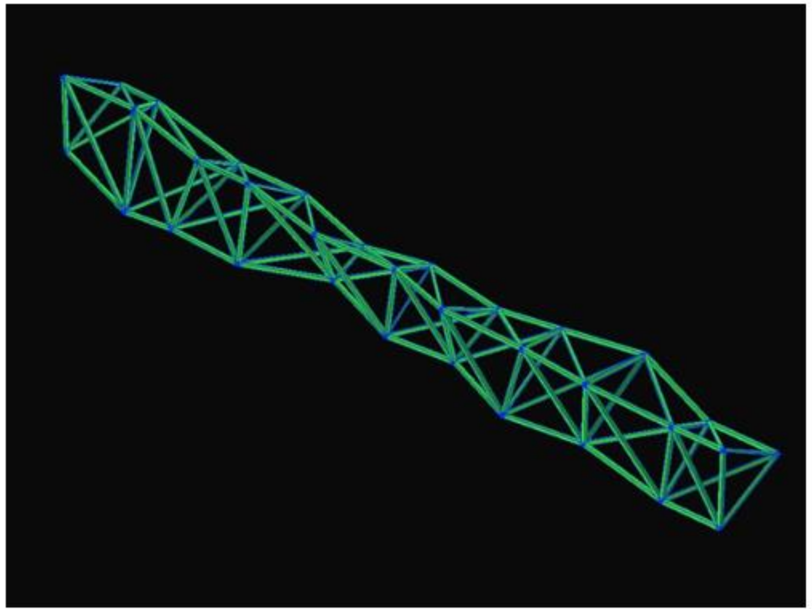
Applications of Evolutionary Algorithms

- Often Evolutionary Algorithms are used to give approximate answers to optimization problems that are not easily solvable using other methods, e.g., NP-Hard optimization problems.
  - Traveling Salesperson
  - Maximal Independent Set,…
- They work especially well with combinatorial optimization problems when you wish to find the optimal solution from a set of combinatorial objects.

Applications of Evolutionary Algorithms

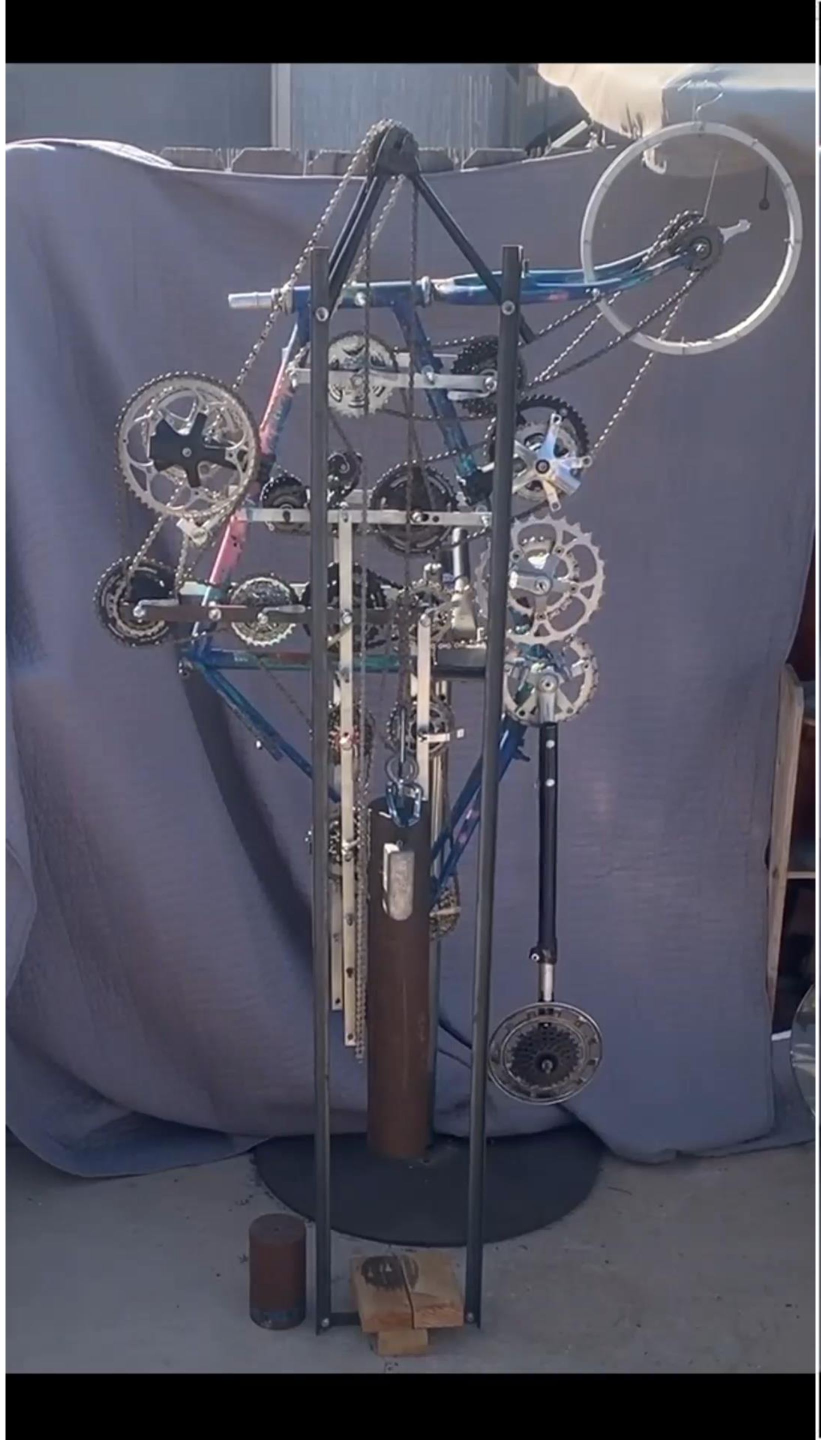
- Neural networks
    - Back-propagation can be accomplished using evolutionary algorithm crossover.
- Engineering structures
    - NASA’s antenna
    - Vibration-free bridges

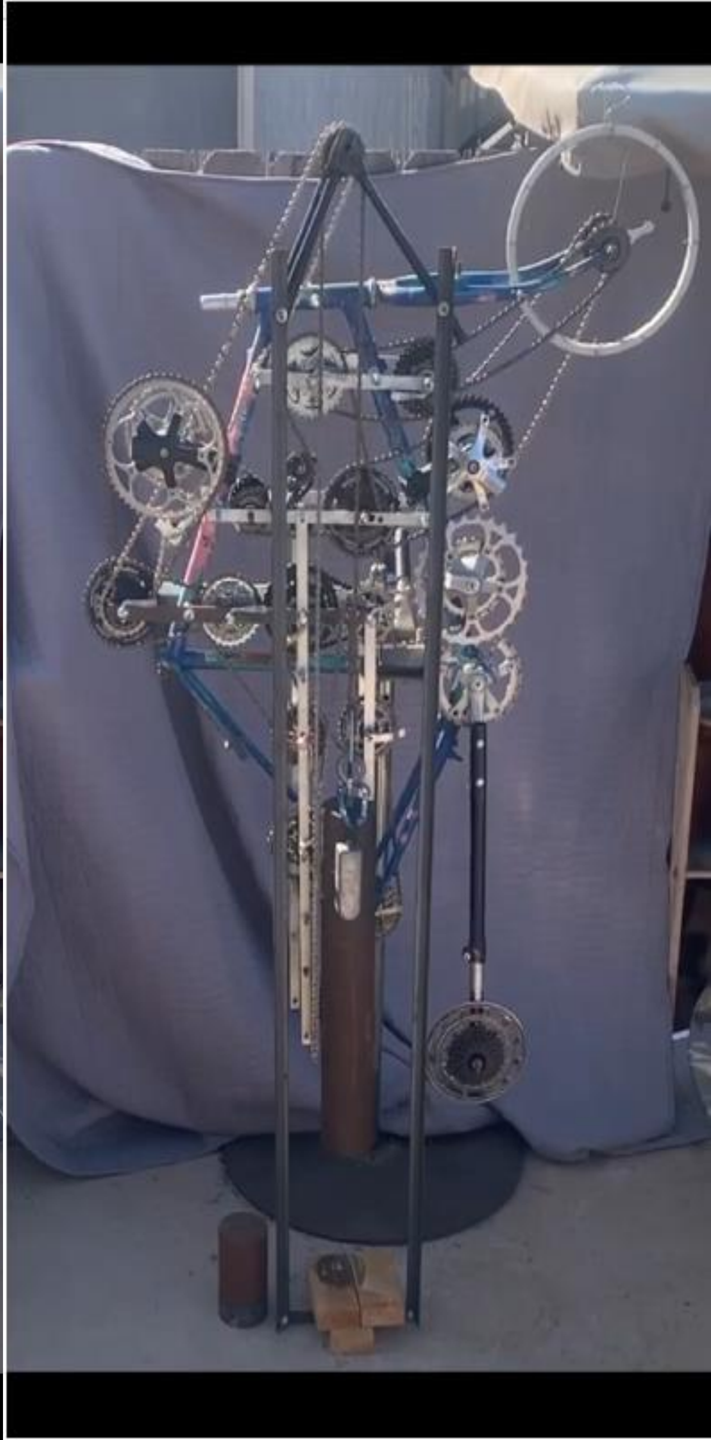




Applications

- I got involved in experimenting with Evolutionary Algorithm as I was designing the gearing in a clock constructed from recycled bicycle parts.
- There is a combinatorial optimization problem involving which gears to link the chains so that you can approximate a given ratio.

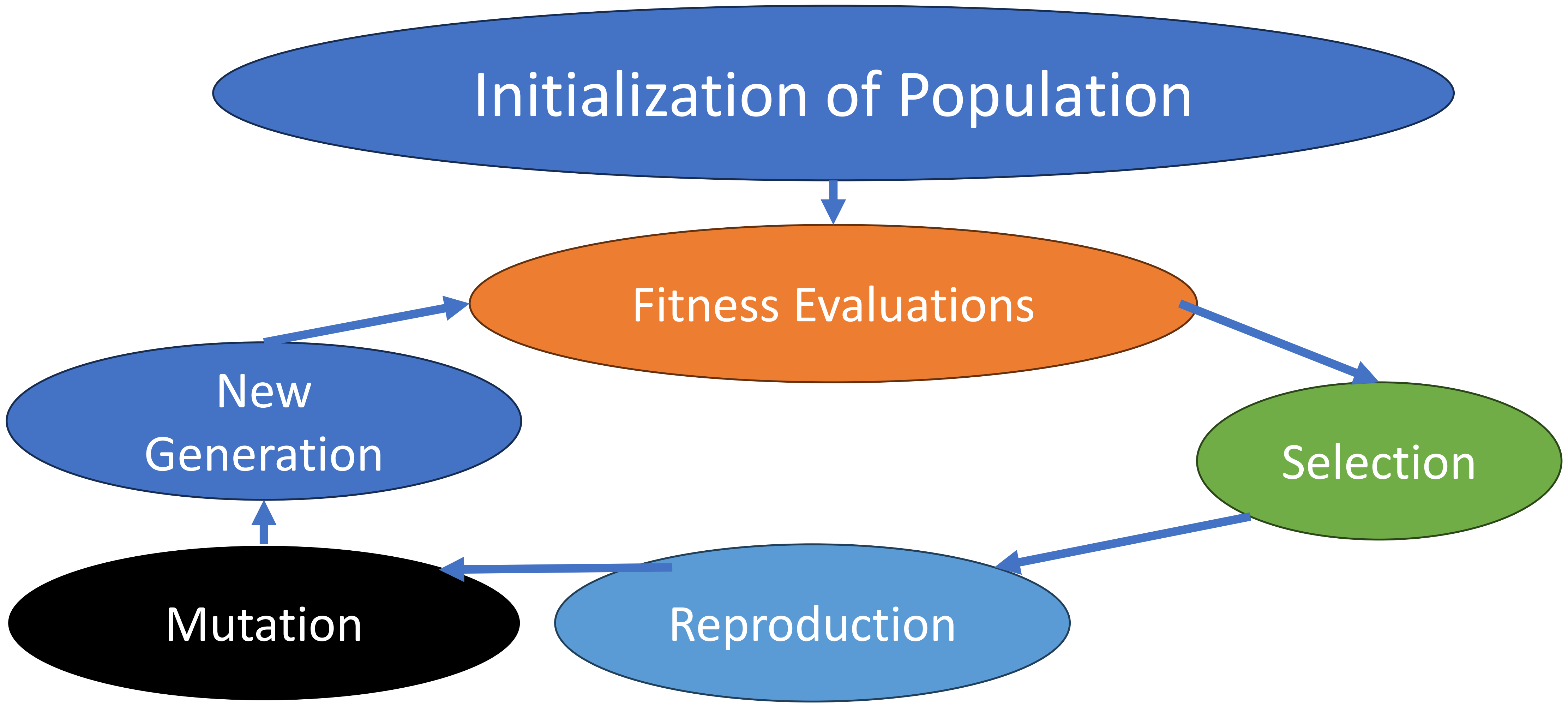




Optimization Problems in Permutation Patterns

- **Packing Density**: Given a pattern q and a length n, what is the maximum number of copies of q you can have in a permutation of length n? (Or set of patterns)
- **Restricted Packing Density:** Given two patterns, p and q, and a length n, what is the maximum number of copies of q you can have in a p-avoiding permutation of length n? (Or sets of patterns)

Evolutionary Algorithm



Packing Density

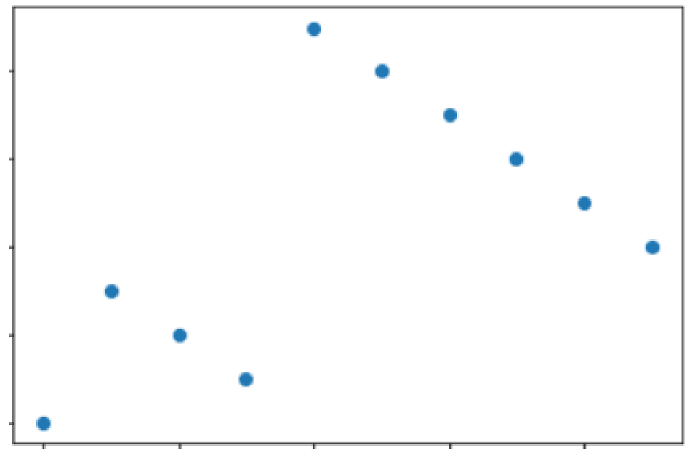
- Given a pattern q, and a length n, we wish to find a permutation of length n that is the most densely packed with the pattern q.

- For example: if the pattern is $q = (2,1)$, and $n = 10$, it should be clear that the decreasing permutation is the most densely packed permutation with the pattern (2,1): [10,9,8,7,6,5,4,3,2,1]
- In fact, there are $\binom{10}{2} = 45$ many such patterns.

Packing Density

- Given a pattern q, and a length n, we wish to find a permutation of length n that is the most densely packed with the pattern q.

- For another example: q = (1,3,2) and n = 10, the maximum number of patterns you can get (I think?) is 63 with the permutation: [1,4,3,2,10,9,8,7,6,5]



Evolutionary Algorithm components

- Representation
- Fitness
- Population size
- Parent Selection
- Variation operators
- Survivor Selector

(Taken from the book *Introduction to Evolutionary Computing* by Eiban and Smith)

Packing Density (Representation)

- The description of each organism is a member of the set that you are searhing.
- We are looking for the “best” *permutation of length n so* in a sense each *organism* is represented by a permutation of length n.
- You can think of each organism’s genetic information as a permutation rather than a DNA sequence.

Packing Density (Fitness)

- In real life, the fitness of a creature is a complicated function based on many factors (size, strength, durability, ability to reproduce, intelligence, etc.)
- For Packing Density, the fitness is much simpler: for a permutation P, the fitness of P is the number of copies of q that occur in P.
- With this fitness function, we can compare two individuals and quickly determine which one is more fit.
- (Often, fitness is just the objective function of the optimization problem.)

(Population size)

- Having a larger population means more variability which suggests that you are more likely to find individuals at or near the global optimum whereas smaller population means less variability and you may get stuck at a local optimum.
- Large populations also means more resources (memory and time.)

- (For most of the experiments that I did, I used a population of around 5000-20000)

(Parent Selection)

- Another concept borrowed from biology is to give the opportunity to reproduce to the individuals who are the most fit.
- Select a subset of size m from the population and choose the 2 fittest individuals to be the parents.

(crossover)

- Crossover is a randomized function that takes in two individuals called parents and outputs a set of individuals called offspring. The goal is for the offspring to “inheret” qualities of their parents.
- I’ll show you 3 different crossover functions that I have used.

(mutation)

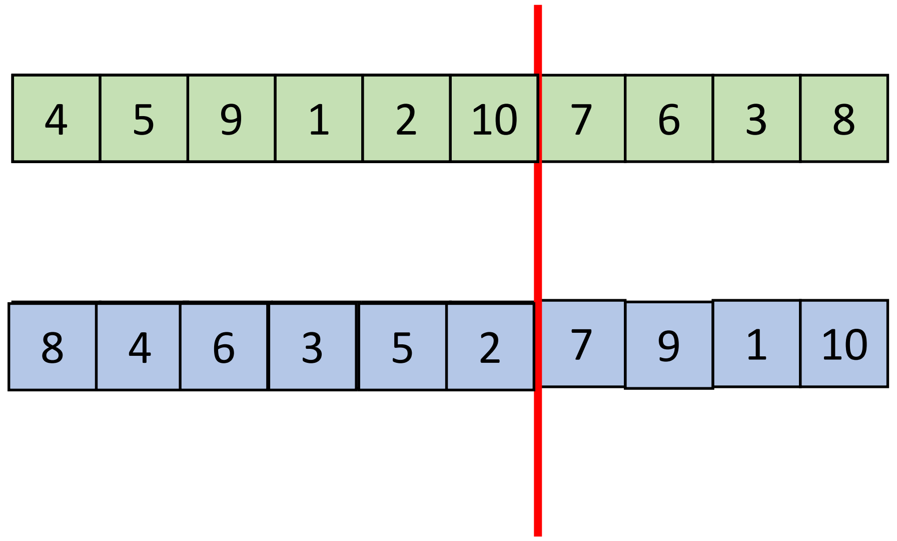
- Yet another borrowed concept from biology is mutation.
- The way it is implemented is to randomly apply a small change to an organism.
- The mutation that I use is a single random transposition.

(survivor selection)

- How do you “kill off” the less fit so that the population “improves” from generation to generation?
- I use a “fixed population size” strategy.
- After each offspring is “born”, I compute its fitness and it will replace an older individual if it is more fit in order to keep the population size fixed.

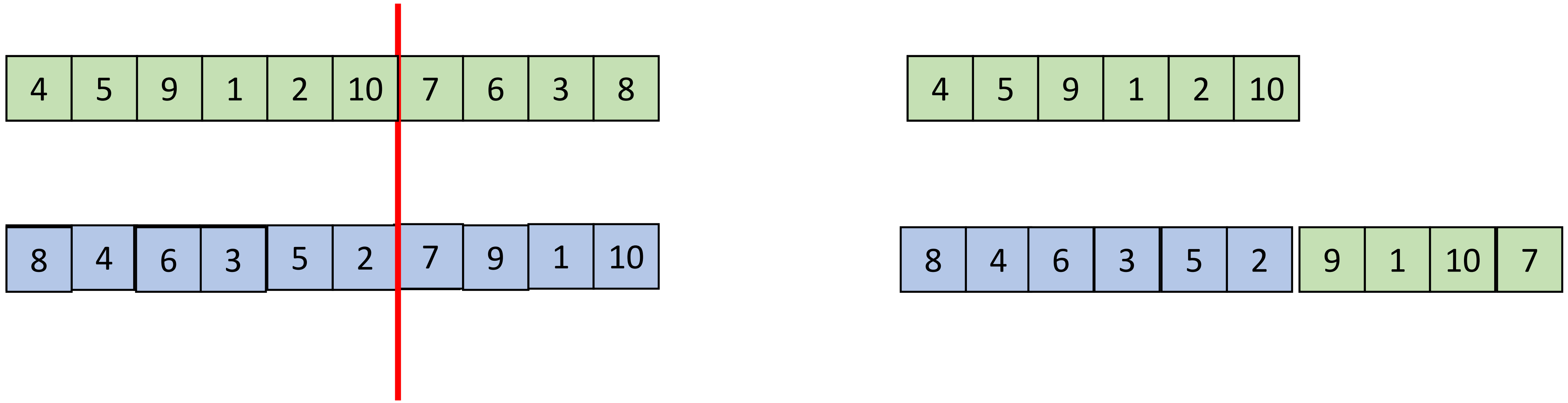
Crossover Functions:

- Crossover (Book) “Cut and crossfit”



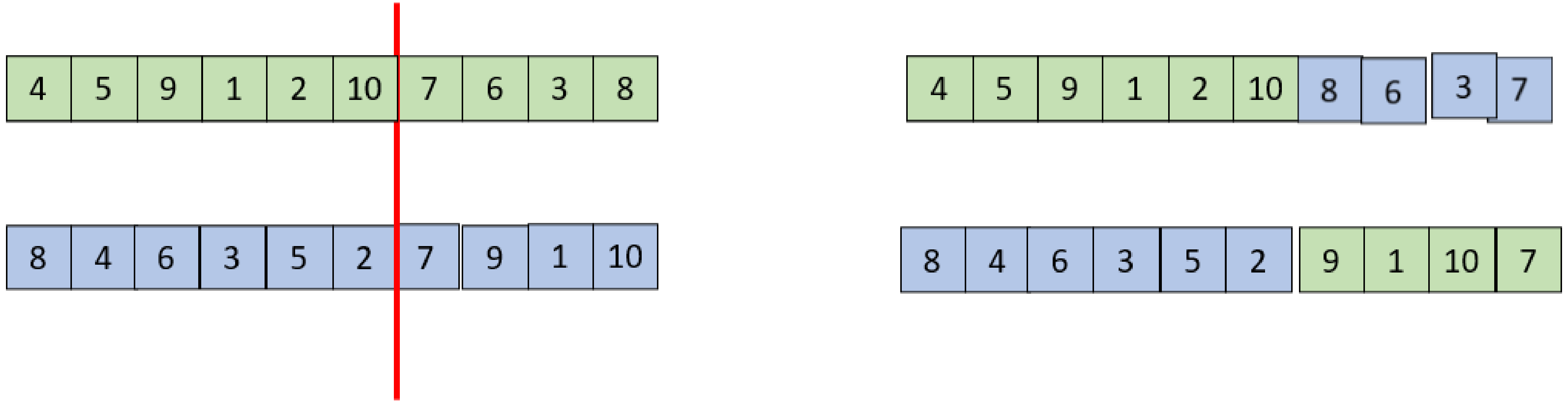
Crossover Functions:

- Crossover (Book) “Cut and crossfill”



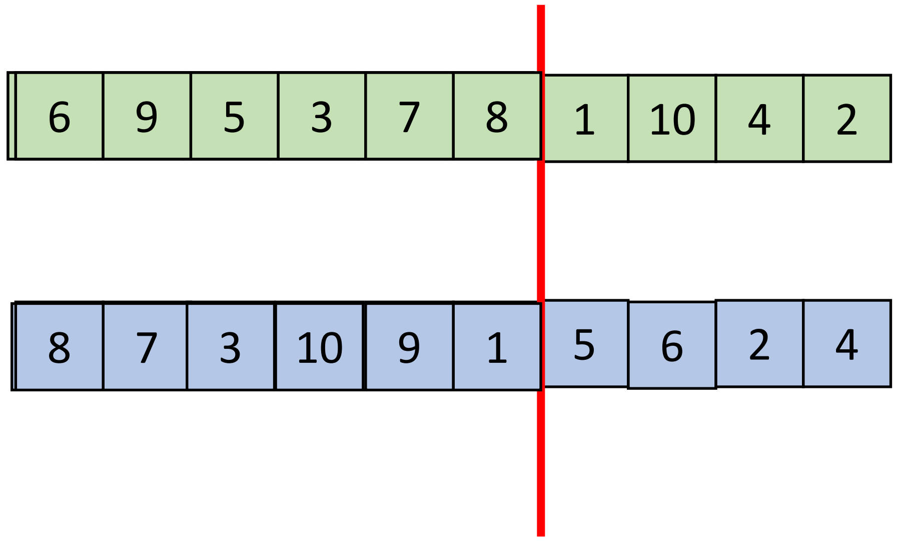
Crossover Functions:

- Crossover (Book) “Cut and crossfit”



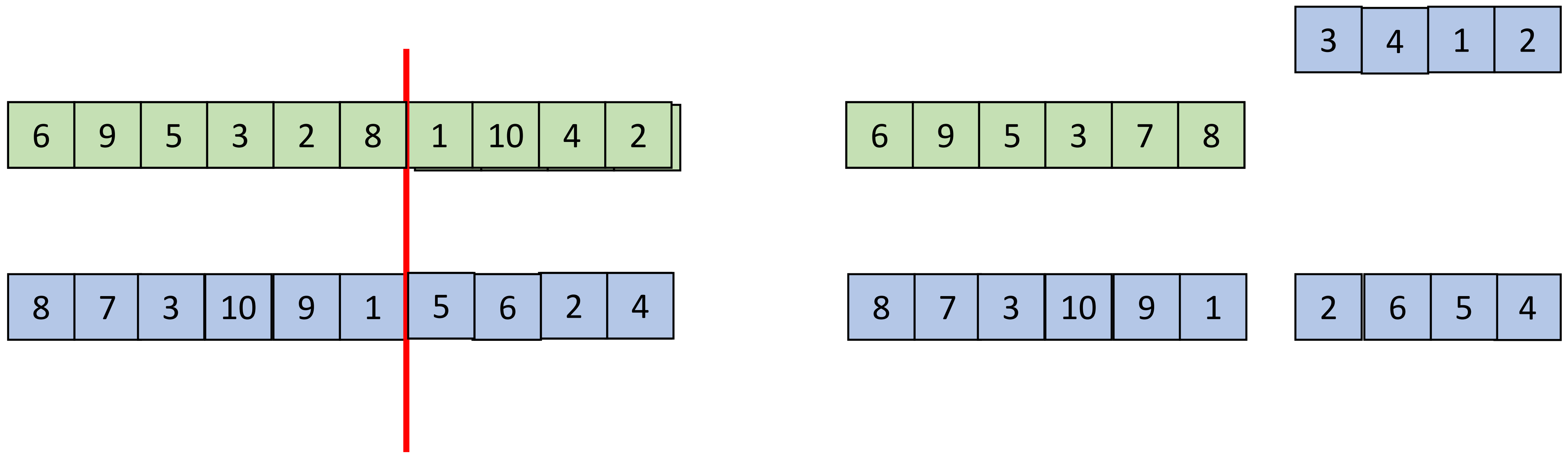
Crossover Functions:

- Crossover (Tiefenbruck) “Cut and pattern”



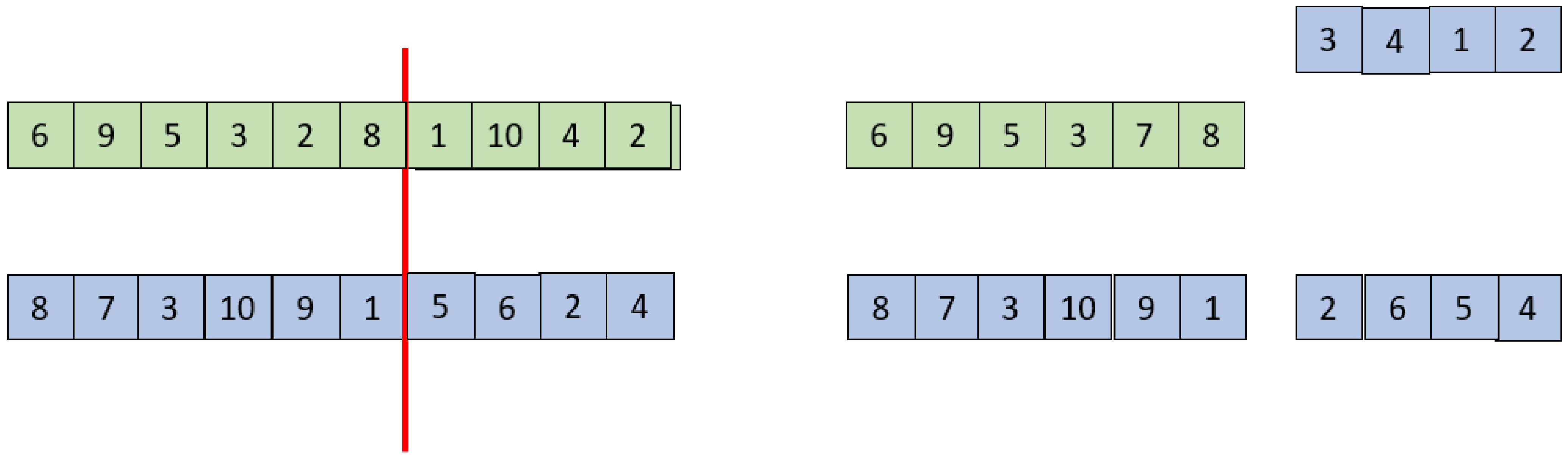
Crossover Functions:

- Crossover (Tiefenbruck) “Cut and pattern”



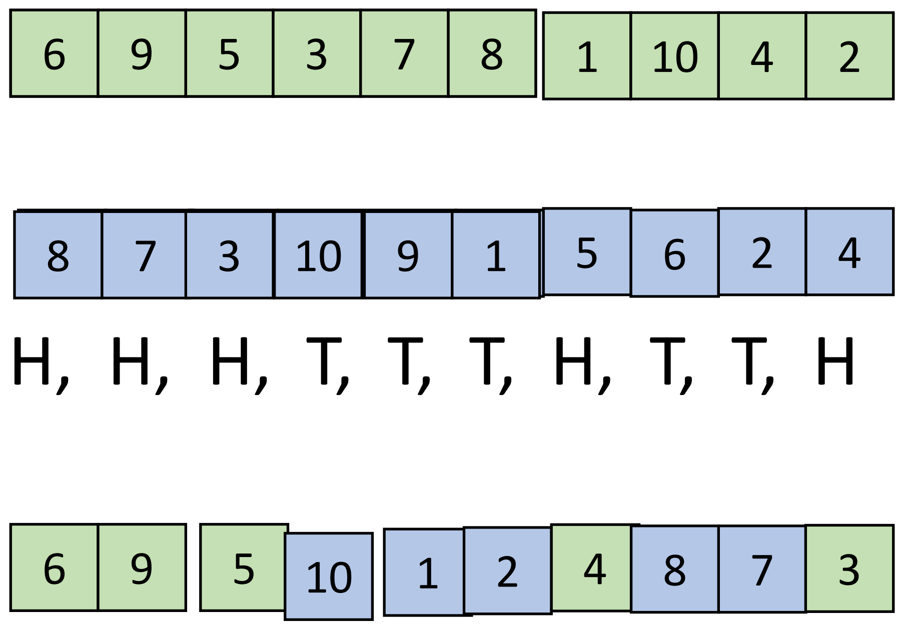
Crossover Functions:

- Crossover (Tiefenbruck) “Cut and pattern”



Crossover Function

- Crossover “flip and scan”



Mutation

- Each time an offspring is “born”, you can apply a mutation with a particular frequency.
- The mutation I have chosen is to select two random indices and transpose their values

5	3	10	1	4	8	9	2	6	7

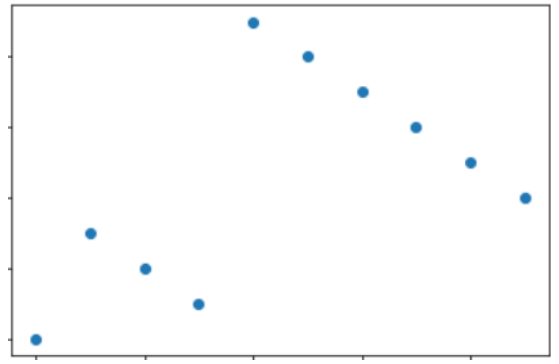
- I chose a mutation rate of 80% meaning that I picked a random number in the interval [0,1] and if it was less than 0.8 then I would apply a mutation and pick a new random number.
- Repeat this process until you select a random number greater than 0.8

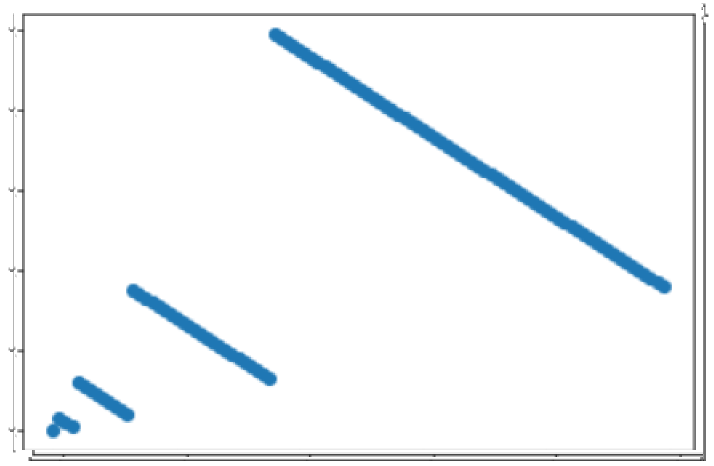
How do you know when to end?

- And, how certain are you that you are at or near an optimum value??
- I measure how long the algorithm has been running by counting fitness evaluations (in other words, counting the number of births.)
- With the parameters I have been using, you can start seeing some basic forms around 200,000 fitness evaluations but I have run some experiments for 600,000 fitness evalutaions.

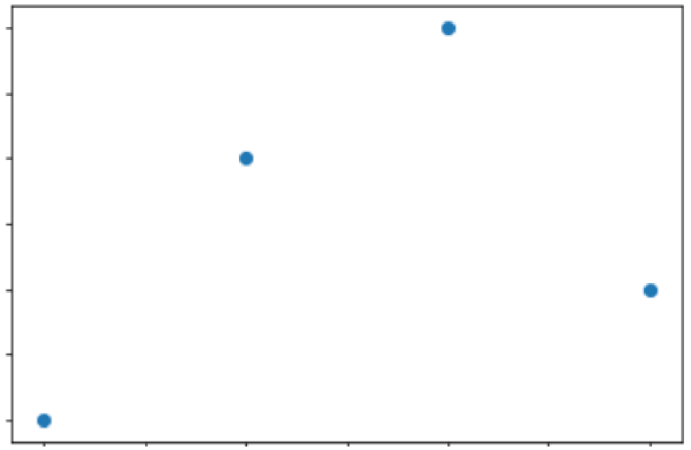
Experiments

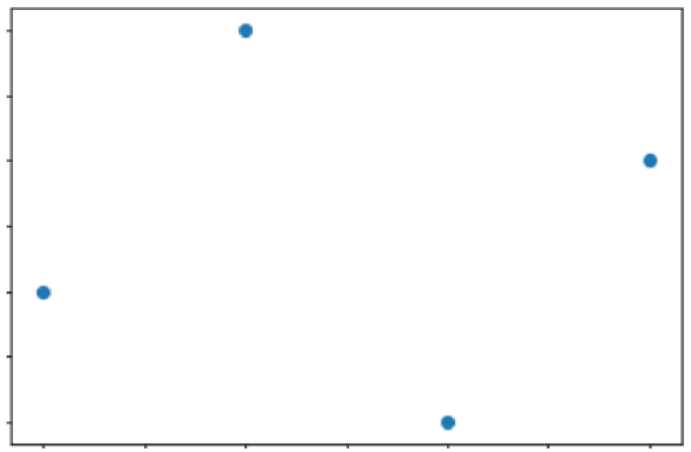
- Recall the permutation of length 10 with the maximum number of (1,3,2) occurrences:
- In what sense is this structure indicative of larger permutations with many copies of (1,3,2)?
- Here is an experiment animation of length 100 permutations that is run for 350,000 fitness evaluations on a population of 20,000, mutation rate of 0.8 and a parent selection of selecting the best 2 out of a random subset of size 60:
- (only the most fit individual in the population is shown in each frame.)
- (The final frame has 76012 copies.)

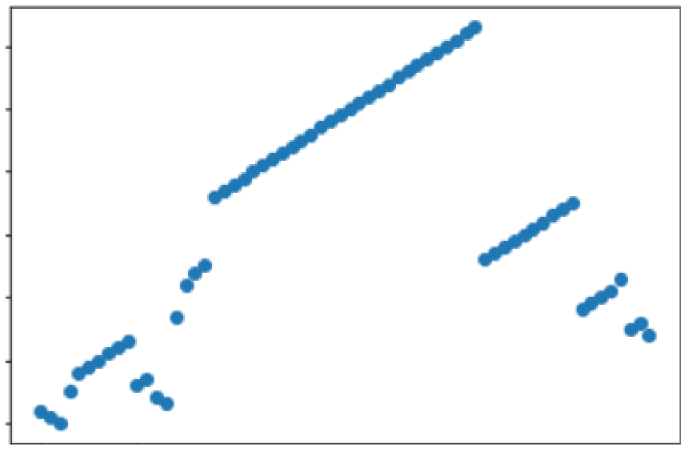


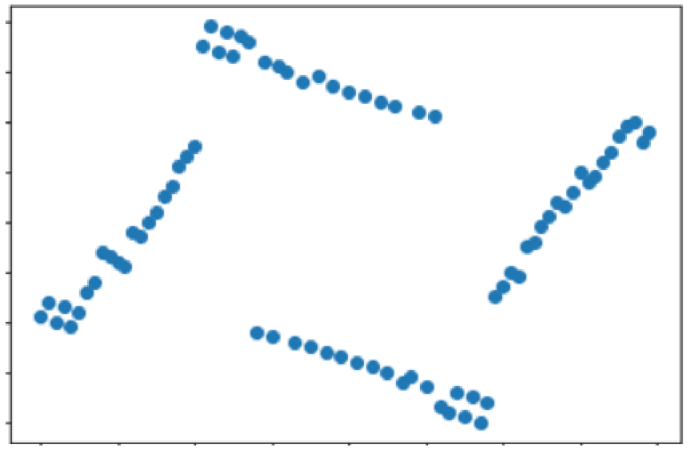


Other experiments

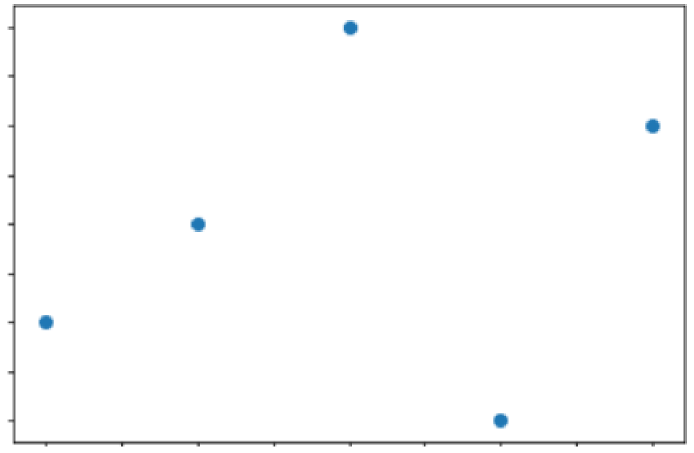

(1,3,4,2)

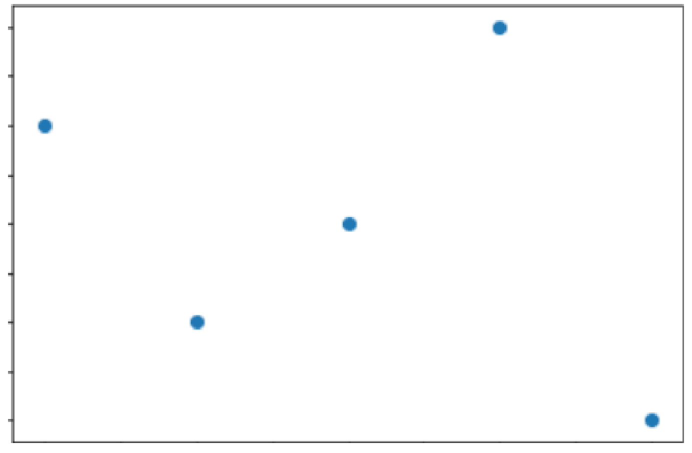

(2,4,1,3)

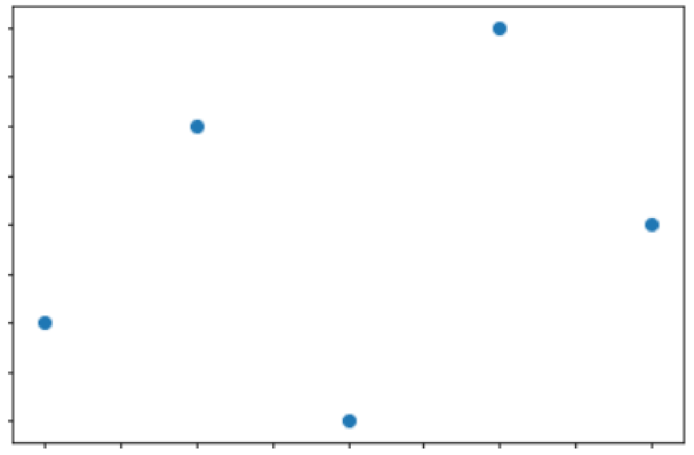

Permutation of length 64 with 133792 copies of (1,3,4,2)

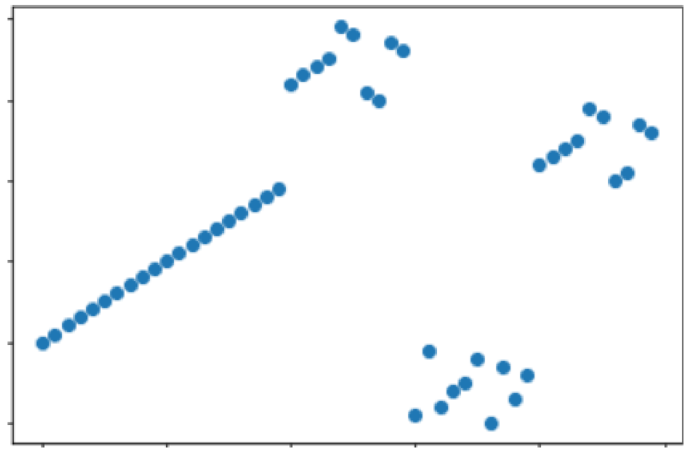

Permutation of length 80 with 178169 copies of (2,4,1,3)

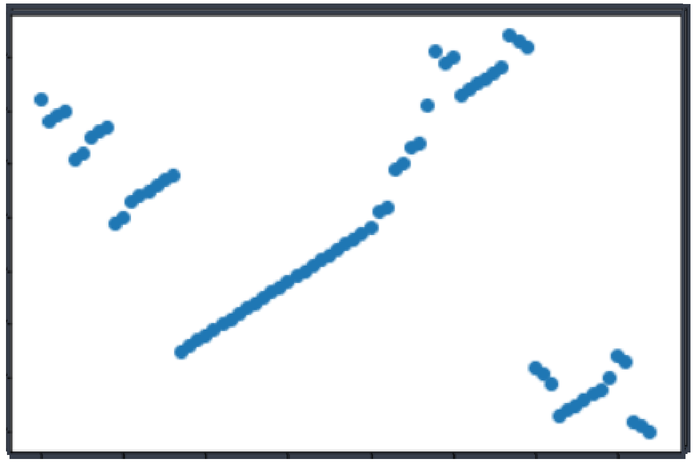
Length 5 shapes:


(2,3,5,1,4)


(4,2,3,5,1)

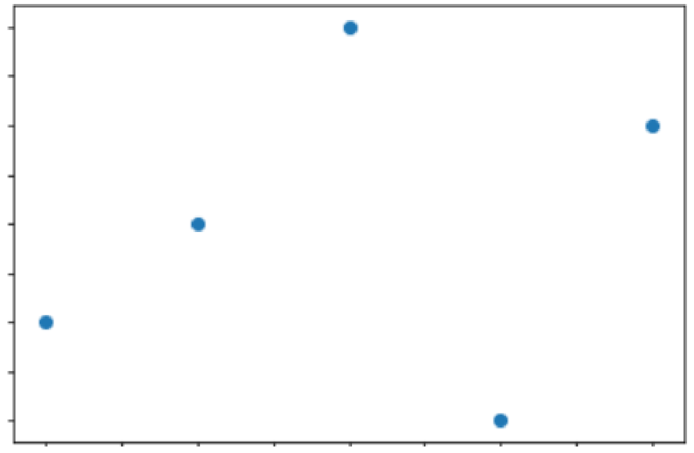

(2,4,1,5,3)

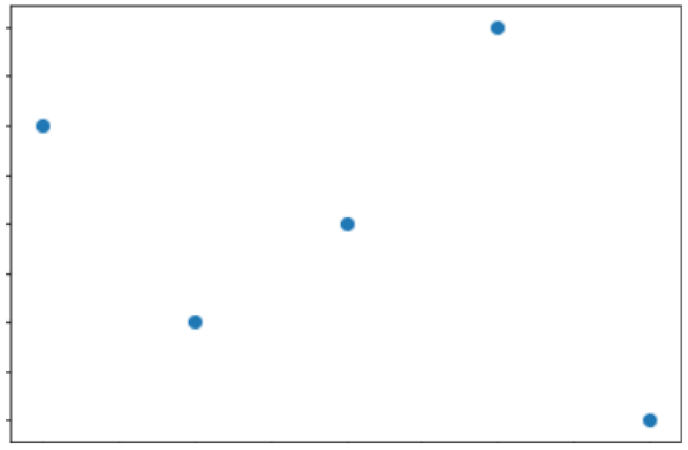

Length 50 permutation with 190113 Copies of (2,3,5,1,4)

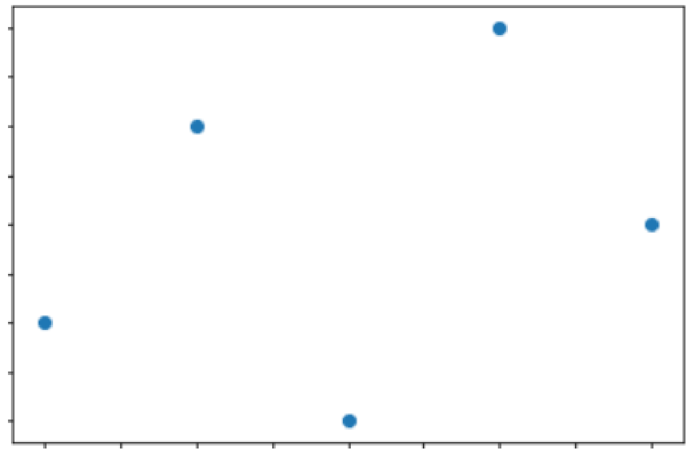

Length 75 permutation animation with many Copies of (4,2,3,5,1)

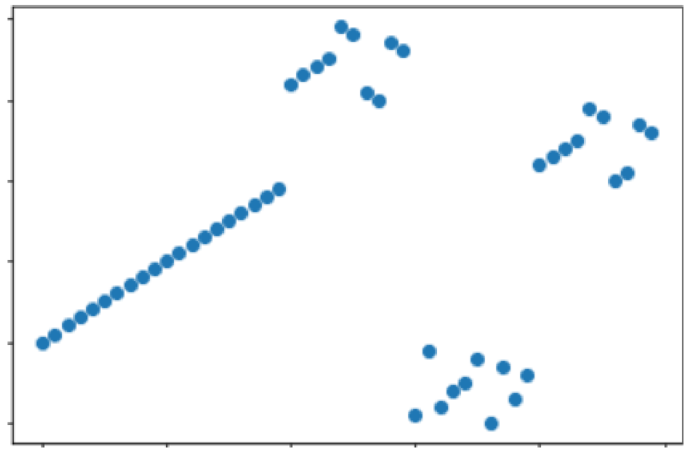
Length 100 permutation animation with many Copies of (2,4,1,5,3)

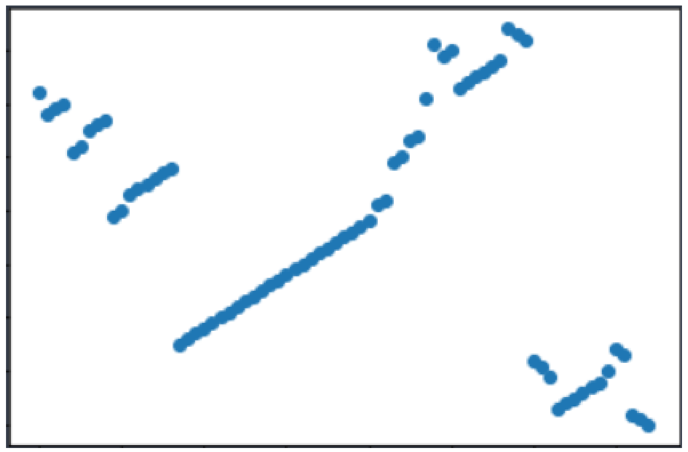
Length 5 shapes:

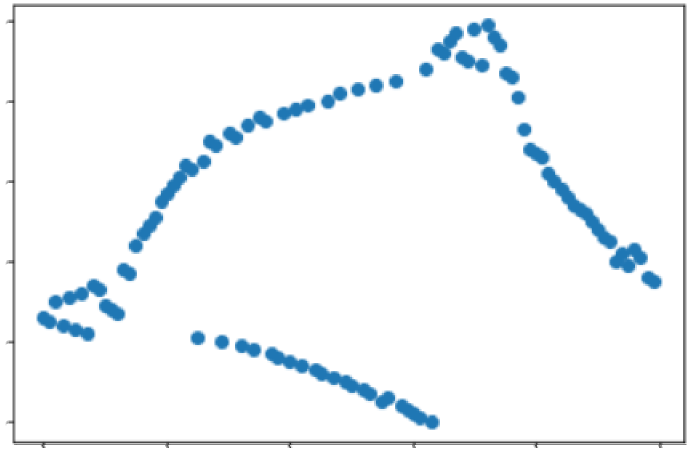

(2,3,5,1,4)

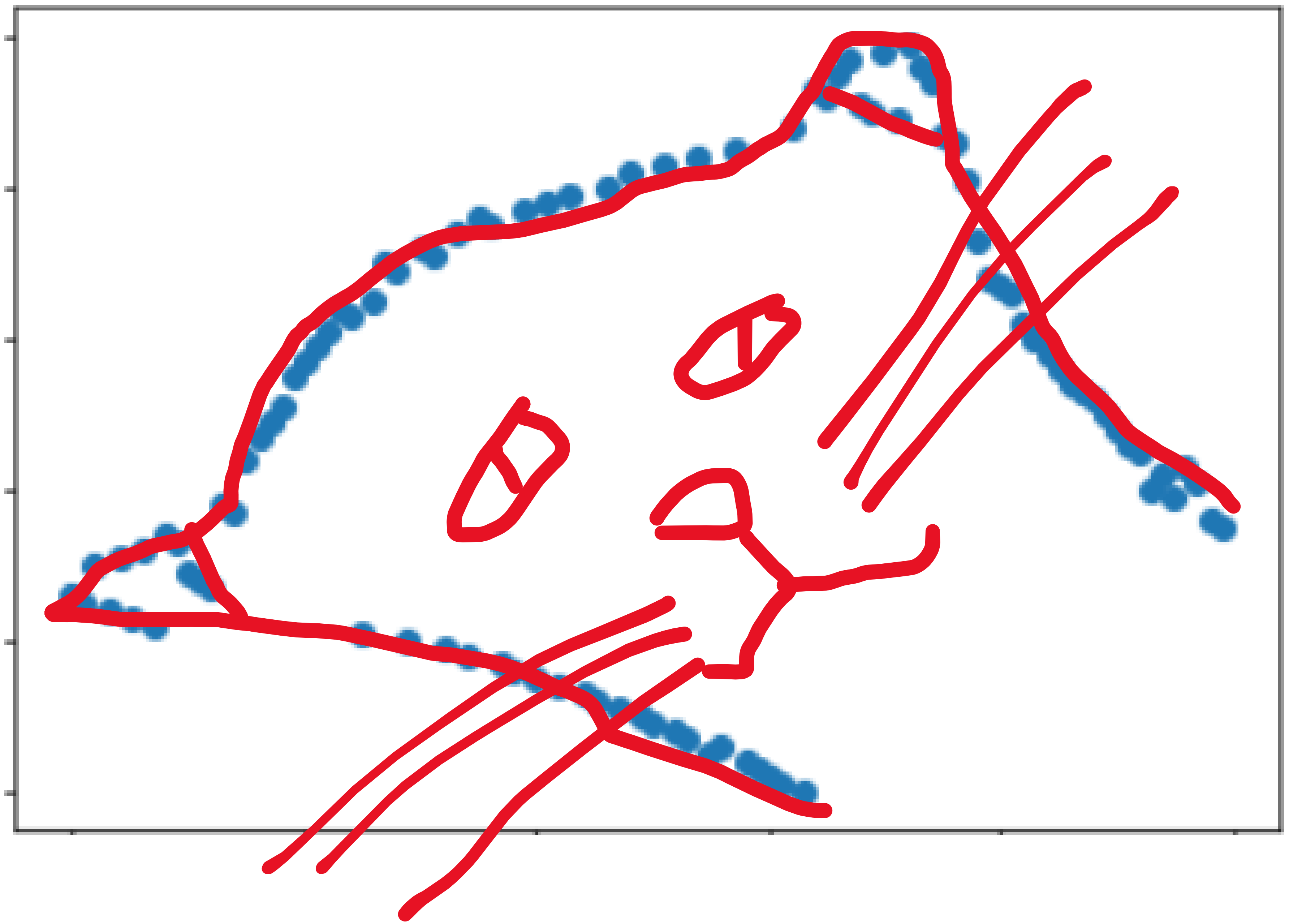

(4,2,3,5,1)


(2,4,1,5,3)


Length 50 permutation with 190113 Copies of (2,3,5,1,4)

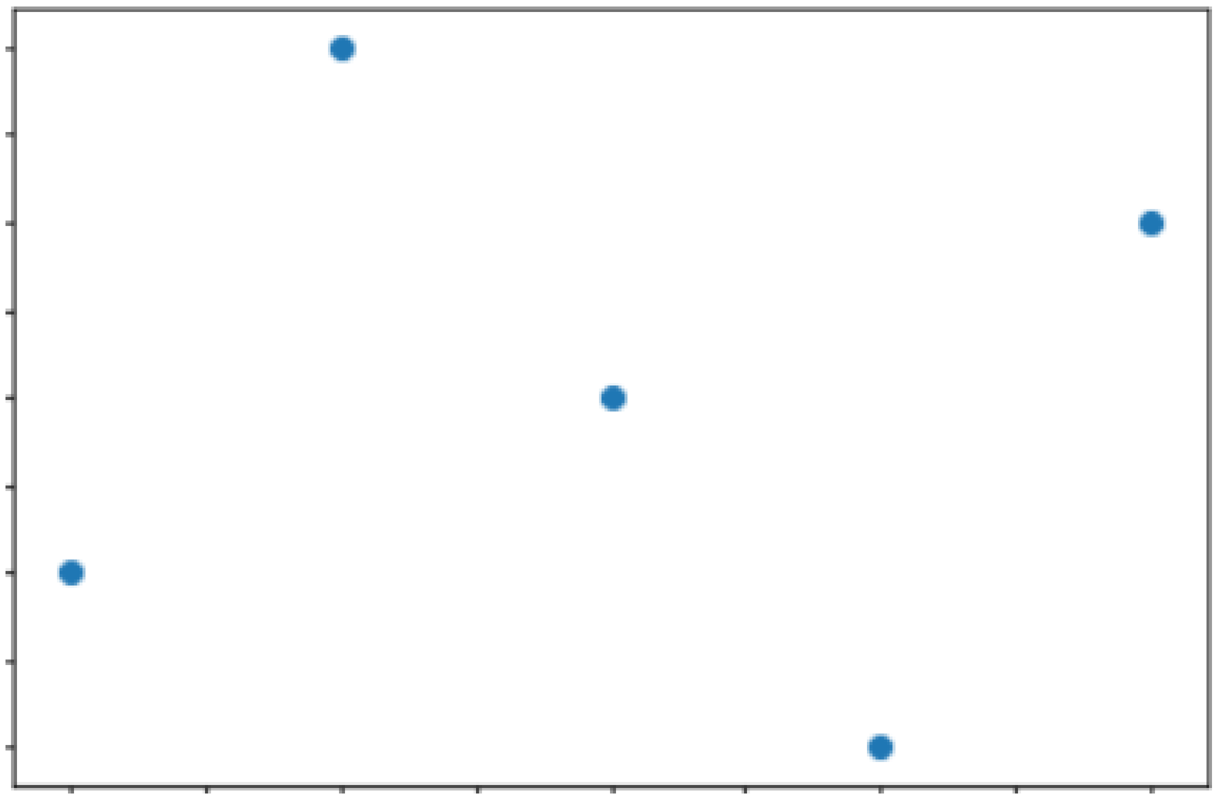

Length 75 permutation animation with many Copies of (4,2,3,5,1)

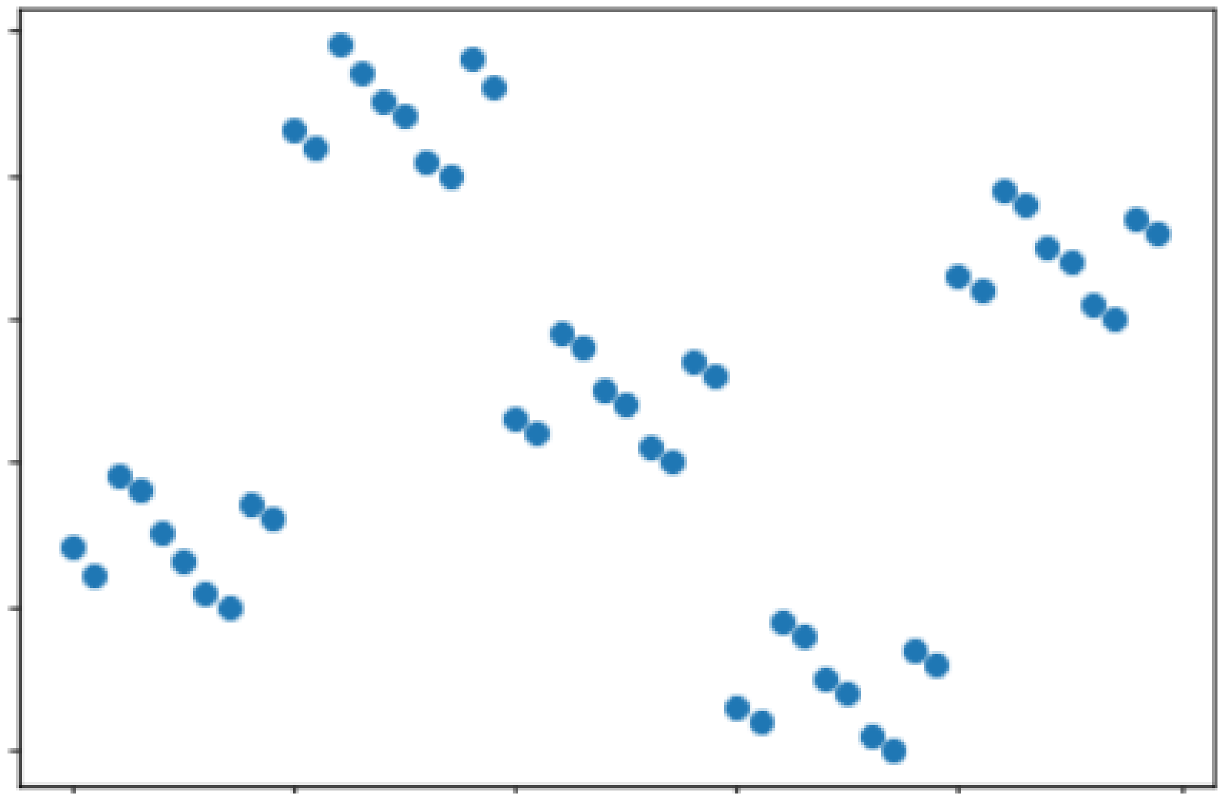

Length 100 permutation animation with many Copies of (2,4,1,5,3)



Example

(2,5,3,1,4)





Pros and Cons

- Pros:
  - Can “quickly” visualize what the optimal permutation “might” look like for “reasonably large” permutations
  - Can explore a wide variety of optimization problems and may lead to conjectures.
- Cons:
  - It often takes a while
  - It certainly is not rigorous
  - Its hard to know how close the solutions actually are to the optimum.

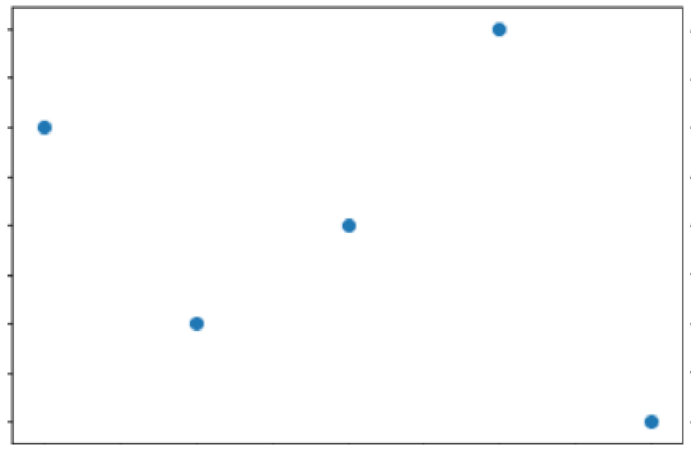
Other fitness functions:

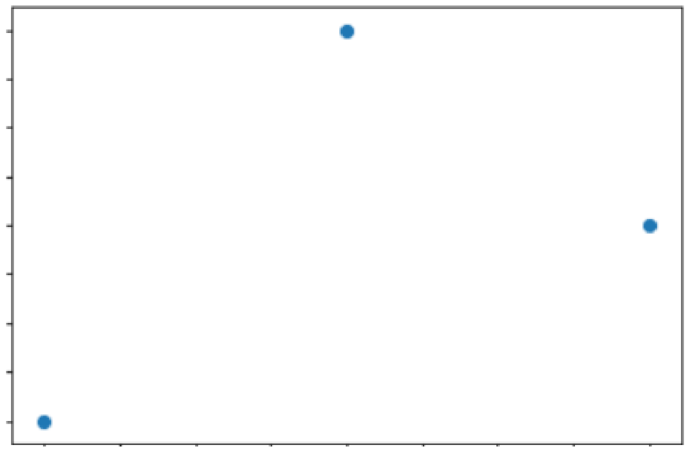
- Another optimization problem is restricted packing density where, given two patterns q and p and a length n, you want to find a permutation of length n that avoids p and is densely packed with q.

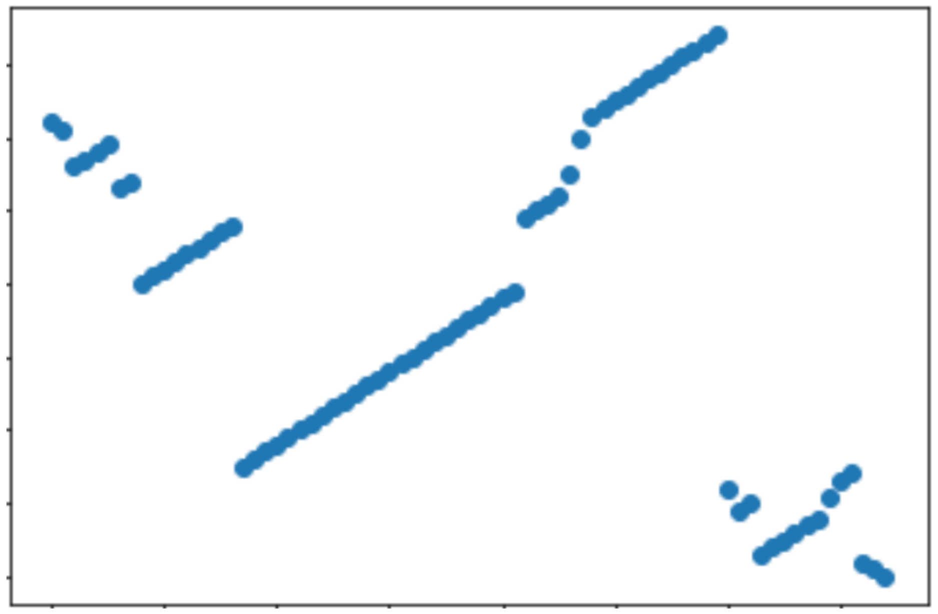
- One way to do this is to have your fitness function be:

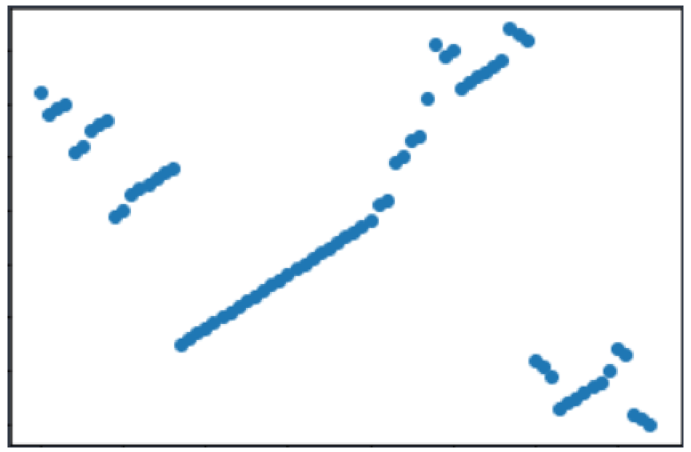
- Fitness(P) = $\frac{\text{\# of occurrences of q in P}}{\text{\# of occurrences of p in P}+0.01}$

- Trying to densely pack (4,1,3,2,5) into a (1,3,2)-avoiding permutation:


(4,2,3,5,1)


(1,3,2)





HELP!!

- I’m looking for more permutation pattern problems that would be a good fit for exploring with evolutionary algorithms
- I am hoping to improve the efficiency of the algorithms

- Thanks to everyone for listening!!

Thanks