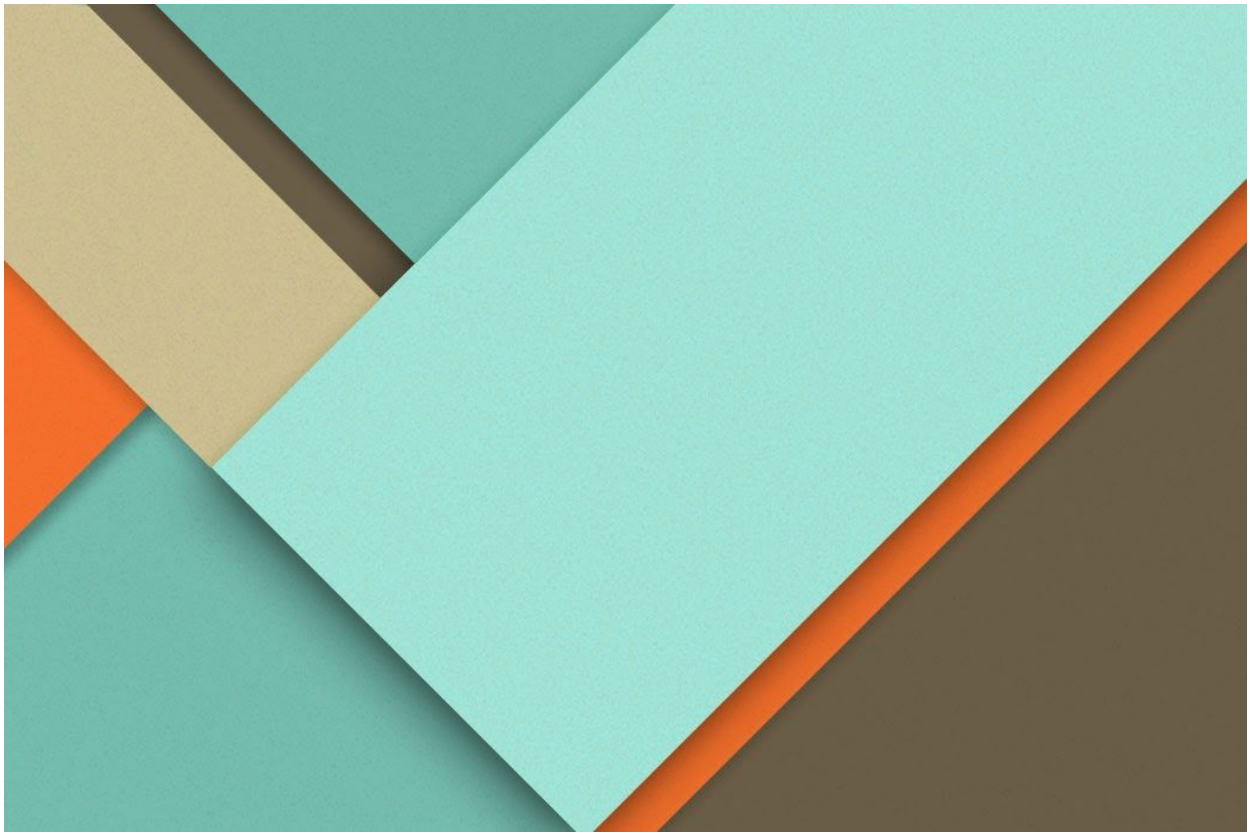

Reporte Barcos

14/06/2020

José Pablo Sandoval de la Cruz
Universidad Politécnica de Chiapas

Visión general

El propósito de este algoritmo es visualizar cómo se genera el mejor tiempo de respuesta cuando se tiene un cierto número de barcos, esto es posible gracias a hacer el proceso de permutación que nos da el número posible de formas de combinar un número de individuos.

Configuración:

1. Python 3.6.9
2. pip 20.1
3. matplotlib 3.2.1

Población

Para poder realizar esta actividad se utilizó la función `def Boat()` en el cual se generan los barcos que se van a analizar en el algoritmo, para ello se hizo uso de la variable `boat_N` la cual contiene la cantidad de barcos que van a existir en todo el algoritmo, para evitar que un barco se repita se fue agregando los datos a una lista y en ella se preguntaba si este dato existe, si no se agrega a la lista pero si llegase a existir se cambiara por otro dato, cabe recalcar que los barcos se van creando aleatoriamente de acuerdo a la velocidad mínima y máxima que está establecido en las variables `velMin` y `velMax`. Por otra parte tenemos las distancias que van a existir entre cada puerto, para ello se utilizó la misma dinámica que la de los barcos pero se omitió la parte donde se evita que las distancias sean diferentes; al momento de crear las distancias estas se hacen aleatoriamente dentro del margen establecido de las variables `distMin` y `distMax`.

Fitness

Para calcular el Fitness se utilizaron varias funciones tales como `def Port(order, distance1, item, ship_datas)`, `def room_Controller(outs1, arrivals1, ship_datas)`, `def time_skip(time, less_time, ship_datas, opcion):`, la función principal de que se encarga de calcular el fitness es `Port(order, distance1, item, ship_datas)` la cual hace las operaciones de salida de cada puerto y la hora de llegada; para hacer las salidas de los puertos se definió que se cada barco tardará 15 minutos en salir con respecto al último, posteriormente pasamos a hacer el tiempo de llegada al siguiente

puerto calculando el tiempo que le tomó recorrer la distancia que hay entre el puerto N y el siguiente puerto; la función `def room_Controller(outs1, arrivals1, ship_datas)` no ayuda a definir cuanto tiempo se va a quedar un barco esperando en el puerto, el tiempo de espera se hace aleatoriamente tomando los valores definidos en las variables *min_Time* y *max_Time* después de hacer el tiempo que van a esperar los barcos en el puerto se pasa a la función `def time_skip(time, less_time, ship_datas, opcion)` donde vamos verificando si más de 2 barcos están en el mismo lugar y si llegase a ocurrir el caso en que se encuentren más de dos barcos en ese tiempo se va a quitar el último barco que llegó al puerto.

Selección

El proceso de selección está definido en la función `def selection(datas)` la cual ejecuta las funciones de `def crossover(boats)` que posteriormente ejecuta la función de `def crossover(boats)` con esto tenemos los datos que vamos a evaluar para ver si están dentro de una lista llamada *Boats_gen* al principio metemos un datos cualquiera que nos salga después de ejecutar las funciones antes mencionadas, pero para las demás vueltas vamos analizando dato por dato para que no se generen combinaciones duplicadas, si una combinación no está dentro de la lista se toma para evaluar dentro del proceso de fitness pero si se encuentra dentro de la lista se evalúa la siguiente por último si los datos generados dentro de las anteriores funciones se encuentran dentro de la lista se sigue analizando hasta que surja un individuo que no esté dentro de la lista para posteriormente pasar a evaluar con las funciones correspondientes.

Mutación

El proceso de mutación está definido dentro de la función `def mutation(Sons, boats):` el cual hace uso de los hijos que se crearon en la función `def crossover(boats):` en esta función primero se crea una lista con la posibilidad de que un individuo pueda mutar, posteriormente se pasa a analizar si esta posibilidad es menor a la posibilidad que se definió en la variable *MutationIndividue* si la posibilidad que mute es menor a la establecida dentro de la variable se seleccionan dos barcos los cuales se van a cambiar de posición entre sí y así obtenemos un individuo que muto intercambiando el orden que se tenía del individuo .

Cruza

El proceso de cruza está definido dentro de la función `def crossover(boats):` la cual hace uso de los botes que se crearon en la función `def Boat()`, para hacer posible el proceso de cruza primero creamos N individuos los cuales salen de la función `def generateShips(boats):`, posteriormente pasamos a elegir dos individuos aleatoriamente y si no es el mismo pasamos a escoger un punto de cruza dentro del margen que está establecido en la variable *boat_N*, después de hacer la cruza entre estos individuos pasamos a ver cuales son los datos duplicados y así vamos escogiendo los datos que le faltan al otro, después de sacar los datos que le faltan al otro pasamos a meter los datos que le faltan a cada hijo y así hacemos que cada hijo esté completo con todos los barcos.

Ejecución

El número de barcos será de: 5.

El número de puertos por los cuales van a pasar los barcos es de: 5

Probabilidad de mutación: 0.15.

El número de vueltas que van a ejecutarse será de : 120, esto de acuerdo a la fórmula de permutación $nPr = \frac{n!}{(n-r)!}$.

La velocidad mínima de un barco será de : 20

La velocidad máxima de un barco será de : 100

La distancia mínima entre los puertos será de : 59

La distancia Máxima entre los puertos será de : 200

Variables utilizadas

```
boat_N = 5 # Define el número de barcos que quieres validar pero que sean menor que 8
velMin = 20 # Define la velocidad mínima que puede tener un barco
velMax = 100 # Define la velocidad máxima que puede tener un barco
```

```
N_Ports = 4 # Define el número de puertos que se pueden pasar

min_Time = 0.50

max_Time = 1

distMin = 59 # Define la distancia mínima que puede haber entre los puertos

distMax = 200 # Define la distancia máxima que puede haber entre los puertos

MutationIndividue = 0.15 # probabilidad de que un individuo mute
```

Gráfica