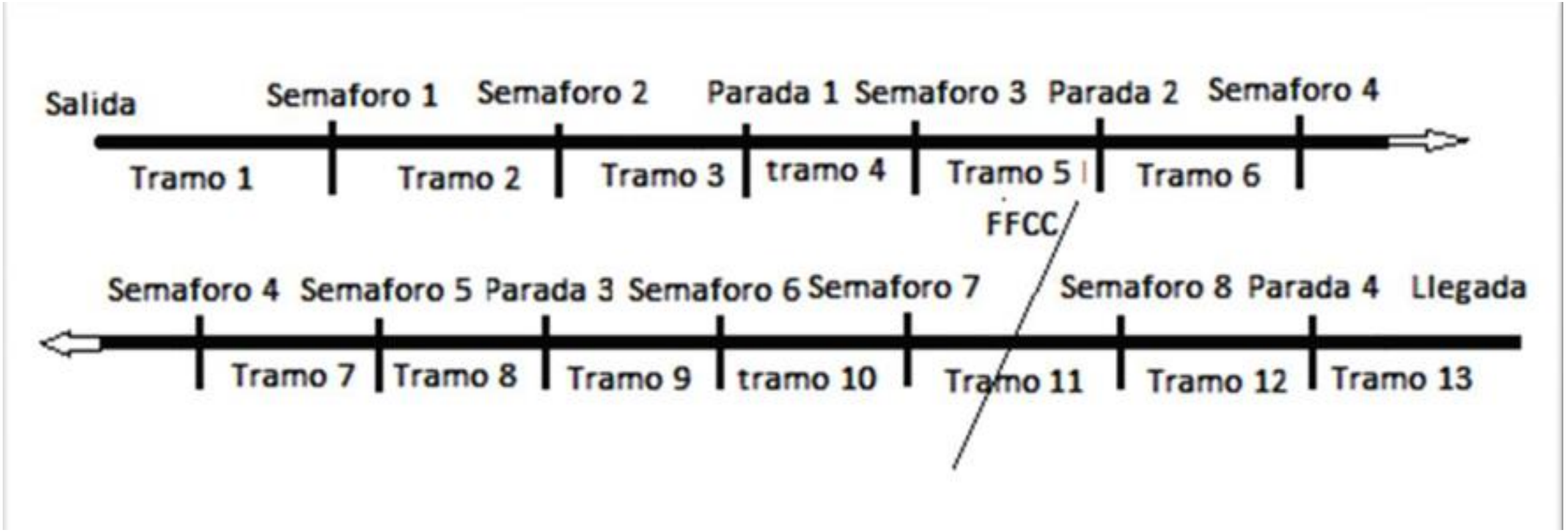
Trabajo Práctico Integrador
Modelos, Simulación y Teoría de la Decisión

ALUMNO : PONTI MATEO

Consigna “ Los ómnibus “

- Una empresa de viajes interurbanos tiene un recorrido como el siguiente:



- Un Viaje está definido por una serie de eventos , los cuales pueden o no suceder.

- Simulando 1000 viajes estableciendo un horario inicial para el servicio determinar con un alfa de 0,05 los tiempos más cortos y más largos del trayecto.

0 . Creación del Modelo

$$HoraInicial = X$$

$$MinutosViaje = \sum TiempoTramo + \sum Tiempo\ \ Semaforo + \sum TiempoParada + \sum TiempoFerrocarril$$

$$HoraFinal = HoraInicial + \frac{MinutosViaje}{60}$$

```
def simularUnViaje(horaInicial,tramos,paradas,semaforos,ferrocarril,recorrido):
  tiempo=0

  r=Ruta(tramos,paradas,semaforos,ferrocarril,recorrido)

  c = r.obtenerCamino()
  total= len(c)



  for i in range(0,total):
      tiempo+= c[i].calcularTiempo()

  horaFinal = (datetime.combine(datetime.today(), horaInicial) + timedelta(minutes=int(tiempo))).time()

  return horaFinal
```

Minutos del Viaje

Hora Final

1. Generación de los Números aleatorios y 2. Conversión en una entrada Válida para el Sistema

- Tramos
- Semáforos
- Ferrocarriles
- Paradas

Tramos

Cada Tramo está Definido por :

1. Distancia del Tramo.
2. La velocidad Esperada en el Tramo y su Probabilidad.
3. La velocidad Máxima Esperada y su Probabilidad.
4. La velocidad Mínima Esperada y su Probabilidad.

$$TiempoTramo = \frac{Velocidad}{Distancia} * 60$$

Tramo	Distancia (Km)	Velocidad Esperada (Km/H)	Velocidad Esperada Máxima Prob=0.2	Velocidad Esperada Min Prob=0.1
1	3	30	40	20
2	5	40	45	30
3	5	40	50	40
4	1	30	40	10
5	4	40	40	40
6	2	10	30	40
7	5	40	45	30
8	10	30	40	60
9	20	50	55	40
10	5	30	30	30
11	1	30	35	30
12	1	30	35	20
13	1	30	35	10

```
def calcularTiempo(self):
    r = random.random()
    velocidad = 0
    if r<=self.probE:
        velocidad=self.vE
    else:
        if r<=self.probE+self.probMin:
            velocidad=self.vMin
        else:
            velocidad=self.vMax
    tiempo= ( self.kmRecorrer / velocidad ) *60
```

- Los Semáforos tienen demora según si se encuentran en verde, rojo o amarillo:
  * Verde : No Hay Demora
  * Rojo: Demora 1 minuto
  * Amarillo : 50 % de Demora de 2 minutos y que no haya demora

	% Tiempo Verde	% Tiempo Rojo	% Tiempo Amarillo
A(1,2,3)	40	50	10
B(4,6,8)	30	60	10
C(5,7)	60	30	10

```
// Calcular el estado del Semáforo

def calcularSemaforo(self):
            r = random.random()
            if r <= self.ptV:
                return "V"
            if r <= self.ptV + self.ptR:
                return "R"
            return "A"


// Calcular Tiempo de Demora

def calcularTiempo(self):
        tiempo= self.tSemaforos[0]
        resultadoSemaforo = self.calcularSemaforo()

        if (resultadoSemaforo=="R"):
            tiempo = self.tSemaforos[1]
        if (resultadoSemaforo=="A"):
            r = random.random()
            tiempo=0
            if r<=0.5:
                tiempo=self.tSemaforos[2]
        return tiempo
```

- En el cruce de ferrocarril el conductor tiene una probabilidad del 20% de encontrar la barrera baja, y en ese caso los tiempos de espera son de 3 minutos el 60% de las veces, de 1,5 minutos el 30% de las veces y de 5 minutos el 10% de las veces

```
def calcularTiempo(self):
    r = random.random()
    tiempo = 0
    if r<=self.pBarrBaj:
        r = random.random()
        if r<=self.p1:
            tiempo=self.t1
        else:
            if r<=self.p1+self.p2:
                tiempo=self.t2
            else:
                tiempo=self.t3
```

Las paradas tienen las siguientes demoras, en función de la probabilidad de personas que estén esperando y/o que necesiten bajar en ese lugar:

Parada	P = 0,4	P = 0,3	P = 0,2	P = 0,2
1	1 min	2 min	3 min	4 min
2	0,5 min	2 min	3 min	4 min
3	0 min	1 min	2 min	3 min
4	3 min	4 min	5 min	6 min

Dado que:

a. No hay un evento asociado a cada probabilidad.
b. La Suma de las probabilidades no da 1 .
c. Intentamos asignar un significado a los eventos.

Los tiempos van incrementando de izquierda a derecha.

Suponiendo que el tiempo que una Persona tarda en bajar es menor al tiempo que tarda en subir (comprobar boleto , asignar espacio , etc) .

Teniendo en cuenta que queremos los horarios finales mínimos y máximos se estableció la **siguiente conclusión**

La tabla hace referencia a **Cuatros eventos** , **Dos Pares** son eventos **mutuamente excluyentes**.

Los eventos son :

I. La Probabilidad de que haya un flujo Bajo de gente que suba (Pocas personas bajan)

II. La Probabilidad de que haya un flujo Bajo de gente que suba (Pocas personas suben)

III. La Probabilidad de que haya un flujo Alto de gente que suba (Muchas personas bajan)

IV. La Probabilidad de que haya un flujo Alto de gente que suba (Muchas personas suben)

¿ Pero si Dos son mutuamente excluyentes no deberían sumar Uno?

- La Razón por la que la suma no da uno , es que nosotros solo tomamos en cuenta el mejor y el peor de los casos (los flujos Altos y Bajos) , y no se toma en cuenta los del Medios.

```
def calcularTiempo(self):
    r = random.uniform(0,self.pBB+self.pBA)
    tiempo=0

    //Calcular Si El Flujo de las personas que bajan es Bajo o Alto
    if (r<=self.pBB):
        tiempo+=self.tBB
    else:
        tiempo+=self.tBA


    r = random.uniform(0,self.pSB+self.pSA)

    //Calcular Si El Flujo de las personas que suben es Bajo o Alto

    if (r<=self.pSB):
        tiempo+=self.tSB
    else:
        tiempo+=self.tSA

    return tiempo
```

3. Iterar N veces

```
def simularNViajes(horaInicial,recorrido,paradas,tramos,semaforos,ferrocarril,cantidadIteraciones):
  tiemposFinales=[]
  for i in range(0,cantidadIteraciones):
    horaFinal = simularUnViaje(horaInicial,tramos,paradas,semaforos,ferrocarril,recorrido)
    tiemposFinales.append(horaFinal)

  return tiemposFinales
```

4. Obtener Resultados

```
def simularViajes(cantidadIteraciones):
    semaforos,ferrocarril,tramos,paradas,alpha,horaInicial,recorrido= getParametrosViaje()

    horariosFinales= sorted(simularNViajes(horaInicial,recorrido,paradas,tramos,semaforos,ferrocarril,cantidadIteraciones))


    horasYfrecuencias = Counter(horariosFinales)
    horas, frecuencias = zip(*horasYfrecuencias.items())

    mitadAlpha=alpha/2
    porcentajeHoraMin= mitadAlpha*100
    porcentajeHoraMax= (1 - mitadAlpha)*100


    cantidadHorasDiferentes= len(horas)
    posicionHMax = round((porcentajeHoraMax*cantidadHorasDiferentes)/100)-1
    posicionHMin= round((porcentajeHoraMin*cantidadHorasDiferentes)/100)

    horaMax=horas[posicionHMax]
    horaMin=horas[posicionHMin]

    convertirAExcel(cantidadIteraciones,list(horas),list(frecuencias),horaInicial,horaMax,horaMin,alpha)
```

Conclusión

- A modo de conclusión queremos aclarar como ya se ha dicho que los tiempos máximos y mínimos se han calculado en base a un Alfa = 5% , el cual representa el porcentaje de error que aceptamos . Ya que a la hora de simular no solo obtuvimos datos aleatorios los cuales no son determinísticos, sino que también al calcular los tiempo de parada se asumieron solo los tiempos de los flujos altos y bajos (el peor y mejor de los casos respectivamente).