Proyecto 1 de Simulación
Poblado en Evolución

Dalianys Pérez Perera
Grupo C-411

Índice

1. Principales ideas de la solución del problema

La solución ofrecida al problema fue construida a partir de la idea de eventos discretos observados en un modelo durante un tiempo T, que en este caso es de 100 años. Para obtener los resultados de la simulación, se mantiene un seguimiento discretizado por años y meses de ciertas variables de conteo que nos permiten describir el estado de la población en cada año. Entre estas variables se encuentra la cantidad de mujeres, hombres, bebés nacidos, fallecidos y parejas existentes en el instante t.

Siempre que ocurre un evento los valores de estas variables se actualizan permitiendo reunir los datos de interés. Para determinar la ocurrencia de los eventos se mantiene una cola con prioridad que enumera los próximos eventos y establece en qué momento ocurrirá cada uno de los ya generados. La prioridad en la cola la determina primero el año y después el mes en que ocurrirá el evento.

Para resolver el problema fue necesario la simulación de variables aleatorias con distribución de probabilidad Uniformes, Bernoullis y Exponenciales, todas generadas a partir de Uniforme(0, 1).

A partir de estas variables se generó el resto de las variables características del problema como son el año de fallecimiento de una persona según su edad y sexo, la probabilidad de embarazarse una mujer, el número de hijos que cada persona desea tener, la probabilidad de querer tener pareja y de establecerla porteriormente, el tiempo que necesita una persona estar soltera luego de una ruptura o enviudar, la probabilidad de que ocurra una ruptura, la cantidad de bebés a tener en un embarazo y por último el sexo de los bebés.

2. Modelación del problema

La simulación comienza con la creación de un mundo que tiene M mujeres y H hombres con edades iniciales que distribuyen Uniformes(0, 100) y se ejecuta como primer evento aquel que establece el año de muerte de cada una de estas personas. Se definen los eventos siguientes:

- $StartYear_{(i,0)}$: inicia el año i en el mes 0
- $Matching_{(i,j)}$: establece parejas entre personas que desean tenerla durante el año i en el mes j
- $Break_{(i,j)}$: establece rupturas entre las parejas existentes durante el año i en el mes j
- $Pregnancy_{(i,j)}$: determina las candidatas a embarazarse por cada pareja durante el año i en el mes j
- $BornBaby_{(i,j)}$: nacimiento de una nueva persona en el año i en el mes j
- $Dead_{(i,p)}$: muerte de la persona p en el año i
- $EndTimeAlone_{(i,p)}$**:** en el año i la persona p finaliza su tiempo de querer estar sola pasando entonces a estar disponible

- $Stop_i$: termina la ejecución de la simulación en el año i

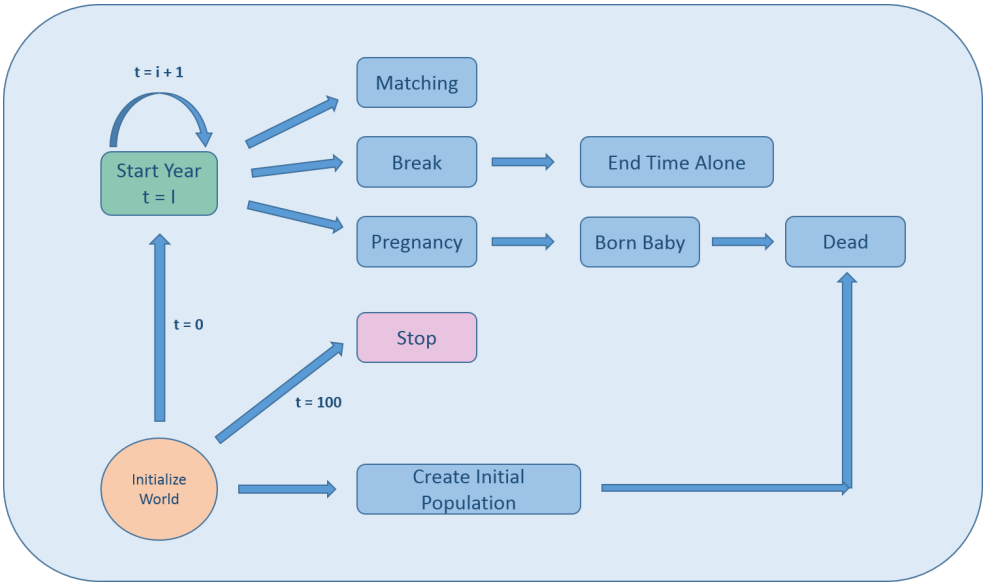

Figura 1: Modelación del problema.

El evento más importante en la línea temporal es $StartYear_{(i,0)}$ es el encargado de generar los eventos principales que tendrán lugar cada año: $Matching$, $Break$, $Pregnancy$ y por último se genera a sí mismo $StartYear_{(i+1,0)}$.

Existen además eventos secundarios que son generados por los anteriores, pues el evento $Break$ da lugar al evento $EndTimeAlone$ para aquellas personas que rompieron su relación, el evento $Pregnancy$ genera por cada embarazo el evento $BornBaby$ para cada uno de los bebés contemplados. $BornBaby$ tendrá lugar si la madre del bebé correspondiente se encuentra viva en ese momento y en tal caso generá el evento $Dead$ de dicho bebé.

Una vez transcurrida la cantidad de años deseada o la población haya sido extinguida, se activará el evento $Stop$ el cual finalizará la ejecución de la simulación.

3. Consideraciones en los datos del problema

Durante la modelación e implementación del problema se analizaron las variables aleatorias a simular y los datos correspondientes brindados en la orientación, lo que llevó a considerar la realización de algunos cambios sobre los mismos. A continuación se ofrece la explicación y justificación de las modificaciones realizadas:

- **Probabilidad de fallecer:** como ya se ha venido explicando una vez que las personas comienzan a existir en el mundo se establece en qué año morirán. Sea k la edad de una persona p y $[a, b]$ el intervalo de edades tal que $k \in [a, b]$, la edad conque muere p se encuentra en uno de los intervalos $[a, b], [b, c], ..., [100, 125]$, o sea, puede ser cualquier año mayor o igual a k. Sea $[p_i, p_{i+1}]$ el peso de probabilidad asociada al intervalo $[a, b]$, primeramente se genera una variable aleatoria $U(p_i, 1)$ para decidir en qué intervalo de edades se encuentra la edad conque morirá p, sea $[c, d]$ dicho intervalo, el valor resultante de la generación de una variable Uniforme discreta en $[min(k, c), d]$ será la edad de muerte de p.

  Los valores asociados a la generación antes planteada se muestran a continuación, notar que las mujeres tienen mayor probabilidad de morir entre $[12, 45]$ que los hombres debido a que este período contempla el embarazo para la gran mayoría de estas en cual son más suceptibles.

Edad	Hombre	Mujer
0-12	$0 < u <= 0{,}15$	$0 < u <= 0{,}15$
12-45	$0{,}15 < u <= 0{,}3$	$0{,}15 < u <= 0{,}5$
45-76	$0{,}3 < u <= 0{,}7$	$0{,}5 < u <= 0{,}7$
76-100	$0{,}7 < u <= 0{,}95$	$0{,}7 < u <= 0{,}95$
100-125	$0{,}95 < u <= 1$	$0{,}95 < u <= 1$

- **Número de hijos deseados:** los valores de esta variable fueron modificados en correspondencia al peso probabilístico que se le daba en la orientación, teniendo mayor probabilidad que una persona desee tener 1 o 2 hijos, brindando un mayor acercamiento a la realidad.

Cantidad	Probabilidad
1	0,25
2	0,35
3	0,25
4	0,05
5	0,05
¿5	0,05

- **Número de bebés en un embarazo :** la modificación realizada en este caso no es tan relevante, sino un pequeño ajuste para que la suma de las probabilidades de los intervalos fuera 1, notar que ahora el peso probabilístico para que un embarazo tenga 2 bebés es de 0.16 mientras que en la orientación es de 0.18.

Cantidad	Probabilidad
1	0,7
2	0,16
3	0,08
4	0,04
5	0,02

4. Análisis de la ejecución de las simulaciones

A continuación se visualiza el comportamiento de algunas variables que se consideraron interesantes en cuatro diferentes escenarios del proceso. Las letras H, M y A de las gráficas se refieren a la cantidad de hombres, mujeres y años respectivamente.

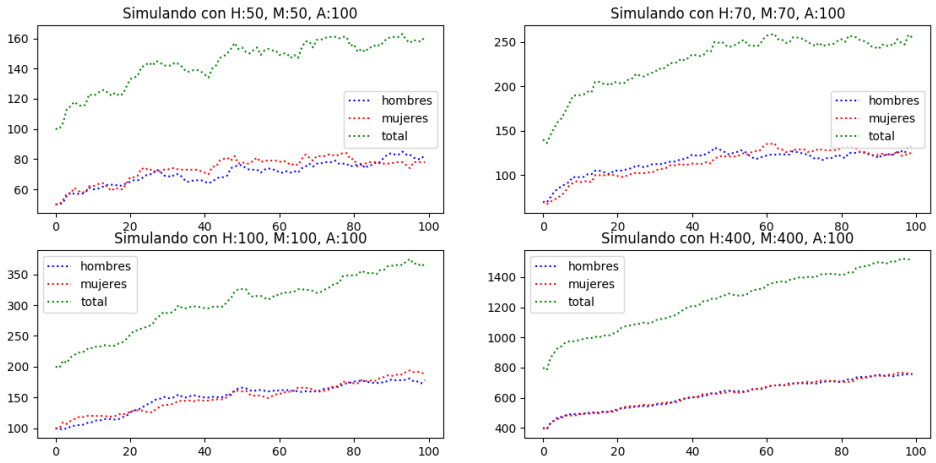

Figura 2: Comportamiento de mujeres y hombres vivos por años

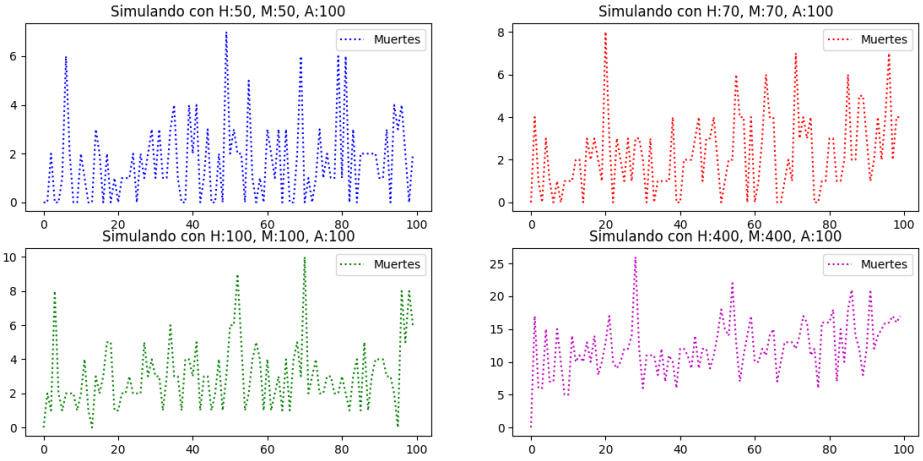

Figura 3: Comportamiento de las muertes por años

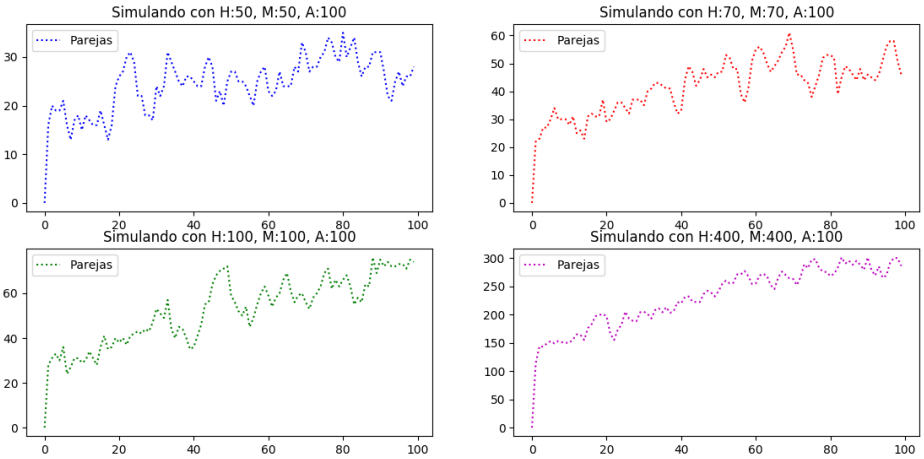

Figura 4: Comportamiento de la cantidad de parejas por años

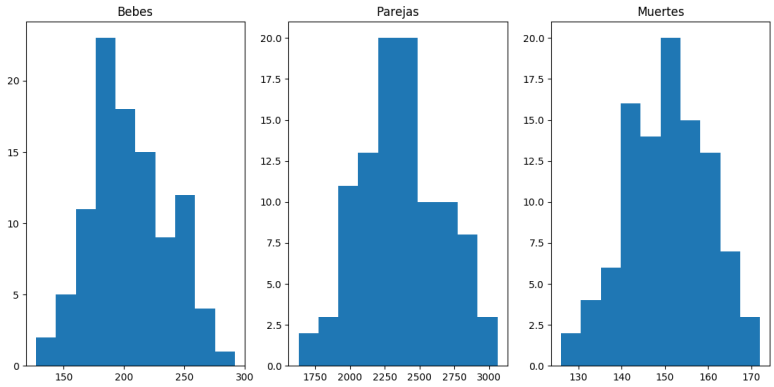

Figura 5: Histogramas de frecuencias en una muestra de 50 hombres y 50 mujeres iniciales

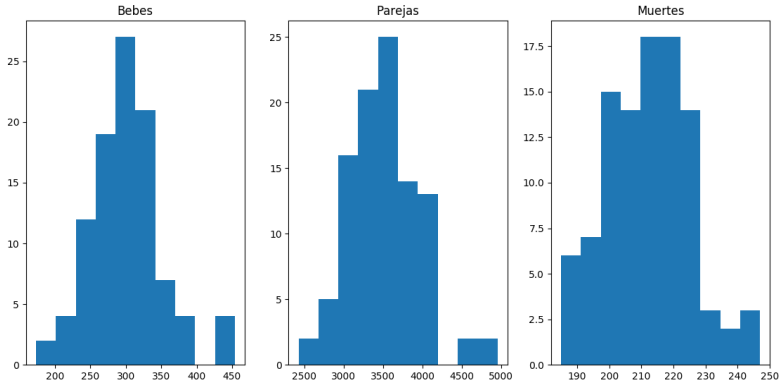

Figura 6: Histogramas de frecuencias en una muestra de 70 hombres y 70 mujeres iniciales

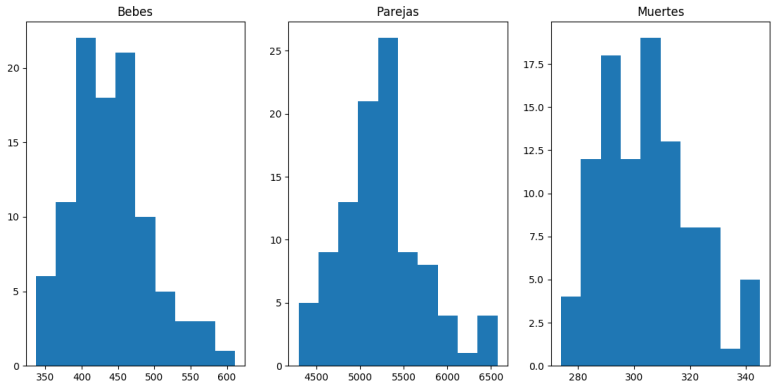

Figura 7: Histogramas de frecuencias en una muestra de 100 hombres y 100 mujeres iniciales

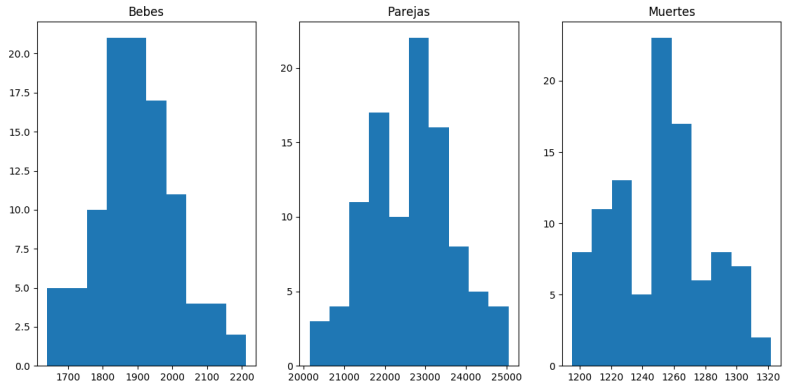

Figura 8: Histogramas de frecuencias en una muestra de 400 hombres y 400 mujeres iniciales

5. Detalles de implementación

La solución implementada en el lenguaje Python está contemplada por los siguientes archivos:

- probdata.py: contiene las tablas con los datos probabilísticos de cada una de las variables aleatorias del problema.
- generation.py: en este se encuentra implementada toda la generación de variables aleatorias.
- person.py: contiene la clase Person con los datos de cada persona en el mundo y la clase CensusAnnual la cual resume toda la información necesaria del mundo en un año.

```
class Person:
       def __init__(self, id, sex, age):
              self.id = id
              self.sex = sex
              self.age = age
              self.partner = None
              self.pregnancy = 0 # num of babys in actual pregnancy
              self.max_child = max_child_wished()
              self.available = True

class CensusAnnual:
       """
       Contain the population info in one year
       """
       def __init__(self, w, m, aw, am, nb, d, c, wdp):
              self.womans = w
              self.mens = m
              self.alive_woman = aw
```

```
                self.alive_men = am
                self.new_babys = nb
                self.dead = d
                self.couples = c
```

- world.py: contiene la implementación de las acciones que se realizan en cada uno de los eventos, además de un diccionario con las personas vivas que tiene como llave el *id* de la persona y una lista que almacena un CensusAnnuual por cada año de la simulación, entre otras variables.

```
class World:
        def __init__(self, w, m):
                self.count = w + m
                self.census = Census(w, m)
                self.year = 0
                self.persons = {}
                self.alive = []
                self.couples = []
                self.create_population(w, m)
                self.census_x_year = []

        def create_population(self, W, M):

        def create_baby(self, info_parents = None):

        def update_alive(self):

        def update_dead(self, id):

        def update_ages(self):

        def extract_info_census(self):

        def alone_persons(self):

        def matching(self):

        def break_couple(self):

        def pregnancy(self):

        def define_time_alone(self, id):
```

- events.py: contiene la clase Event definiendo lo que sería un evento como tal, una

función que se ejecuta en un mes y un año. Además de la clase Simulator la cual viene siendo la máquina de estados para la simulación.

```
class Event:
	def __init__(self, year, function, month = 0, args = None):
		"""
		year: year the event occurs
		month: priority between months
		function: action that occurs in the event
		args: arguments of function

		"""
		self.year = year
		self.month = month
		self.function = function
		self.args = args

	def do(self):
		if self.args != None:
			self.function(*self.args)
		else:
			self.function()

class Simulator:
	def __init__(self, iter, world):

	def stop(self):

	def simulate(self):

	def move(self):

	def add_event(self, e):

	def update_initial_population_event(self):

	def start_year_event(self):

	def matching_event(self):

	def end_timealone_event(self, id):

	def break_event(self):
```

```
        def pregnancy_event(self):

        def born_baby_event(self, info):

        def dead_person_event(self, person_id):
```

- simulation.py: aquí es donde se mandan a ejecutar la cantidad de simulaciones deseadas con cualquier cantidad de años y personas iniciales, generando gráficos que permiten visualizar el resultado de estas mucho mejor.