

# Primeros pasos en simulaciones con agentes: Simulando una ciudad

Aramayo Martín

Instituto Balseiro

# Introducción

Objetivos:

- Obtener información demográfica.
- Reproducir resultados de modelos existentes.
- Acelerar una simulación existente.

# Introducción

El modelo (simplificando)


- Cuento con un número fijo de viviendas.
- Los agentes tienen edad, género, parejas, hijos y viven en una casa.
- Forman parejas, envejecen, se emancipan y tienen hijos.
- ◦ Esto ocurre cuando se cumplen determinadas condiciones:
- ◦     — Determinada edad, tener o no una pareja, etc.

## Implementación: El modelo



Figura1: Modelo de casa

## Implementación: El modelo





numeroHabitante( $n_H$ )	12
numeroCasa( $n_C$ )	85
Edad	42
Genero	♂
Emancipado	V
MiddleLife	V
Pareja	V
 numeroPareja( $n_P$ )	12
 ParejaFertil	V

Figura2: Modelo de habitante

## Implementación: El modelo

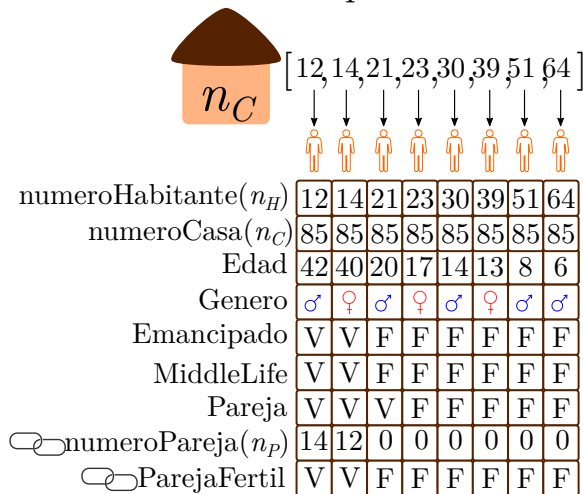


Figura3: Ejemplo completo de una casa

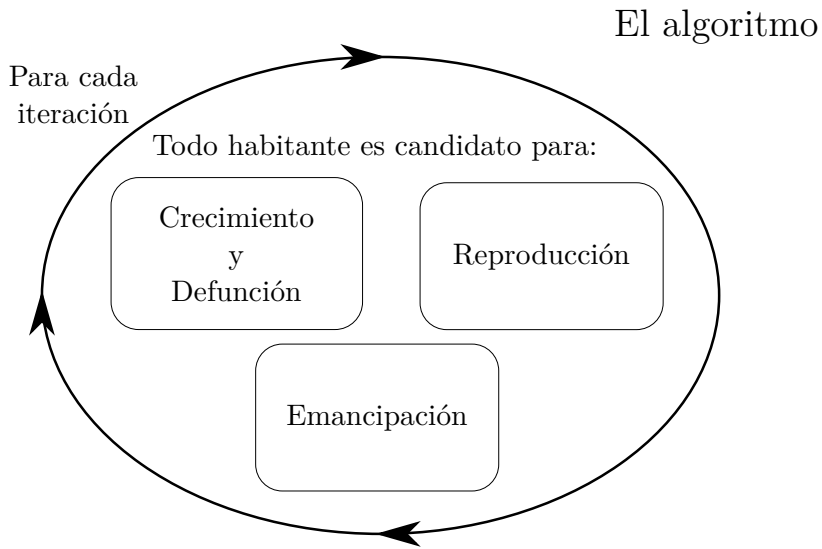


Figura4: Funcionamiento global

# El algoritmo

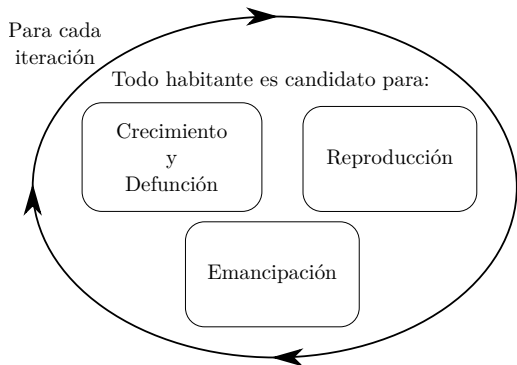


Figura5: Funcionamiento global

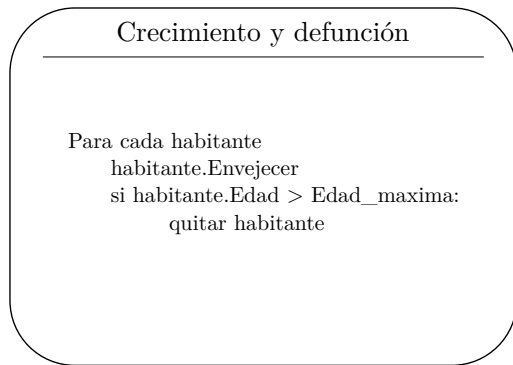


Figura6: Crecimiento y defunción



# El algoritmo

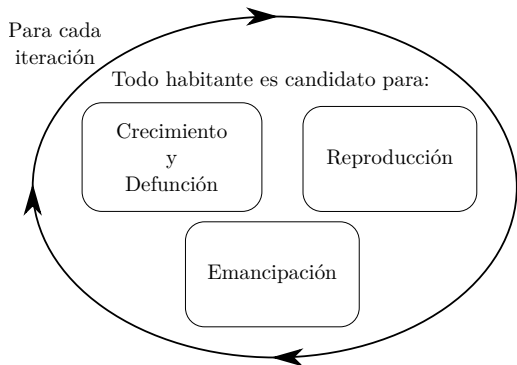


Figura7: Funcionamiento global

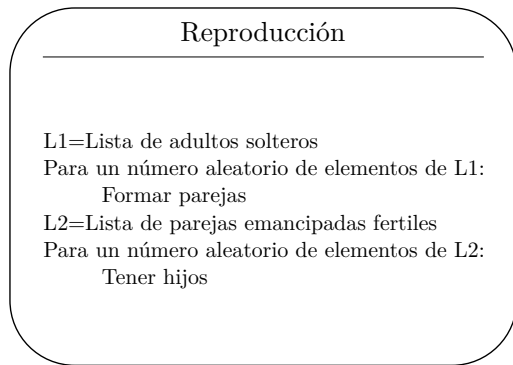


Figura8: Reproducción

## El algoritmo

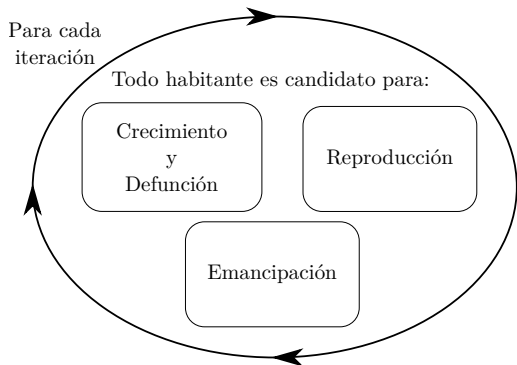


Figura9: Funcionamiento global

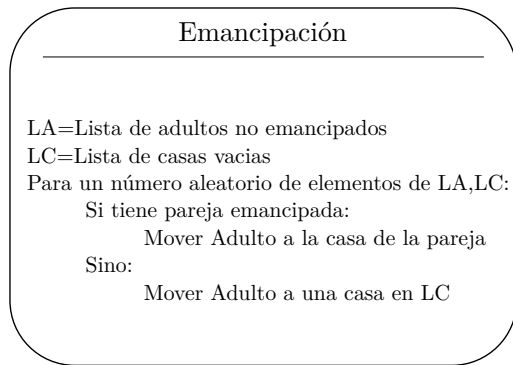


Figura10: Emancipación

## Metodología: “El número aleatorio”

Controla cuantas veces un determinado evento  $E$  de probabilidad  $p_E$  ocurre sobre un subconjunto de los habitantes de la simulación.

## Metodología: “El número aleatorio”

Controla cuantas veces un determinado evento  $E$  de probabilidad  $p_E$  ocurre sobre un subconjunto de los habitantes de la simulacion.

**La idea:** Dada una probabilidad de entrada  $p_E$ , tiro “monedas” (True, False) por cada habitante en el subconjunto afectado por el evento  $E$ . Sumando los True obtengo el resultado.

**La forma rápida:** Generar el número con una distribución binomial de probabilidad  $p$  y con su parámetro de “tiradas” dado por el número de elementos en la lista.

## Metodología: Configuración y parámetros

Un archivo yaml contiene, previo a la simulación:

- Número de iteraciones
- Probabilidades de los eventos
- Condiciones iniciales

Posterior a la simulación:

- Tiempo de ejecución
- Fecha de ejecución
- Iteraciones reales (si se extinguieron antes)

# Metodología: Configuración y parámetros

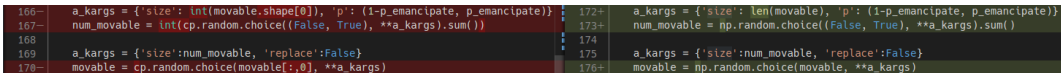
```
sauer@hal /m/5/D/m/m/r/h/experiments/experimentoNumeroHabitantes/experimento05.yaml
```

testIni~	experimento10	2	n_iteraciones: 2000
experim~	experimento09	2	p_emancipate: 0.3
tim~.pdf	experimento08	2	p_partner: 0.3
fe~.yaml	experimento07	2	p_child: 0.1
fe~.yaml	experimento06	2	p_premature_death: 0.001
fe~.yaml	experimento05	2	initial_condition:
	experimento04	2	num_houses: 15000
	experimento03	2	ratio: 0.5
	experimento02	2	my_mu:
	experimento01	2	house_size: 3
	experimentoFertil~.py	1.08 K	comment:
	experimentosFertil~.sh	895 B	Baja fertilidad y muerte prematura no nu
	timeVsSize.pdf	11.5 K	execution_time_sec: 661.3760406970978
	experimento06.yaml	418 B	num_iterations_real: 2000
	experimento05.yaml	417 B	MIDDLE_LIFE_START: 20
	experimento04.yaml	417 B	MIDDLE_LIFE_END: 50
	experimento03.yaml	416 B	LIFE_EXPECTANCY: 85
	experimento02.yaml	416 B	INFANCY: 14
	experimento01.yaml	413 B	HOUSE_CAP: 10
			execution_time: Fri Jul 2 08:07:29 2021

```
-rwxrwxrwx 1 root root 417B 2021-07-02 06:12 15.9K sum, 103G free 15/19 All
```

Figura11: Archivo de configuración de un experimento

CuPy, sutiles diferencias:



166- a_kargs = {'size': int(movable.shape[0]), 'p': (1-p_emancipate, p_emancipate)}	172+ a_kargs = {'size': len(movable), 'p': (1-p_emancipate, p_emancipate)}
167- num_movable = int(cp.random.choice((False, True), **a_kargs).sum())	173+ num_movable = np.random.choice((False, True), **a_kargs).sum()
168	174
169 a_kargs = {'size': num_movable, 'replace': False}	175 a_kargs = {'size': num_movable, 'replace': False}
170- movable = cp.random.choice(movable[:,0], **a_kargs)	176+ movable = np.random.choice(movable, **a_kargs)

Figura12: Diferencias en las dos implementaciones

- Las funciones que reducen (sum, mean, etc.) **No** retornan un escalar, retorna un CuPy array.

## Resultados

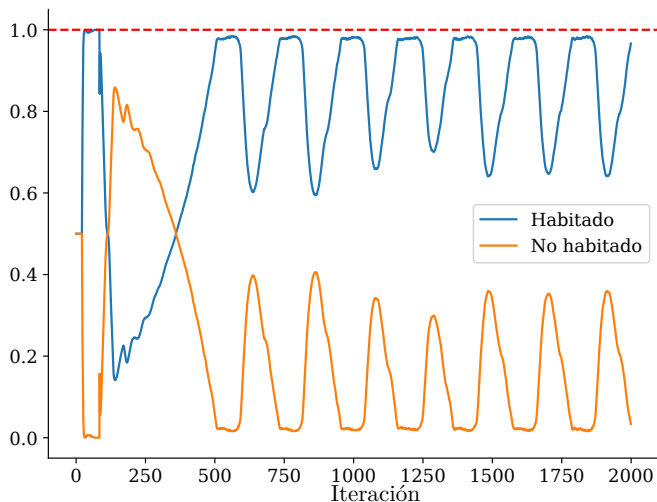


Figura13: Estado de ocupación de viviendas ( $1.5E4$  casas de condición inicial)



## Resultados

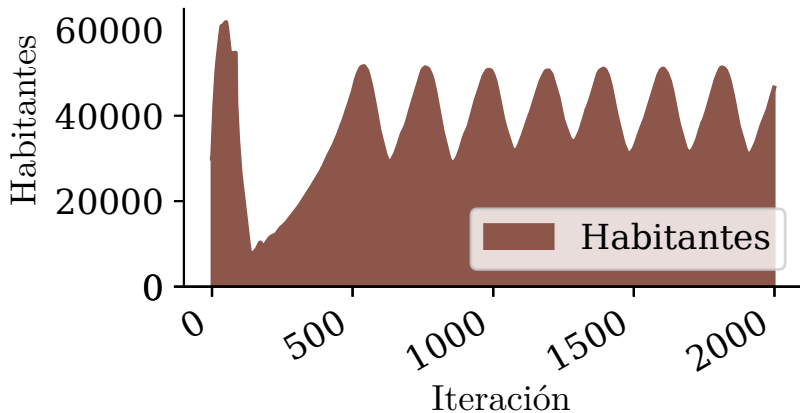


Figura14: Evolución de los habitantes con las iteraciones(1.5E4 casas de condición inicial)

## Discusión: GPU Vs CPU

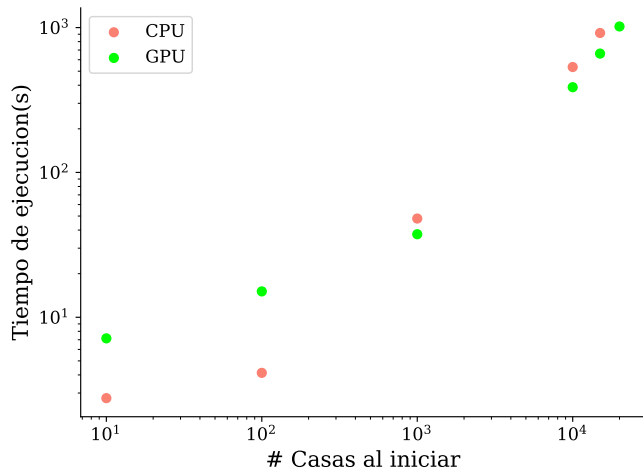


Figura15: Rendimiento GPU Vs CPU

# Discusión: Posibles mejoras

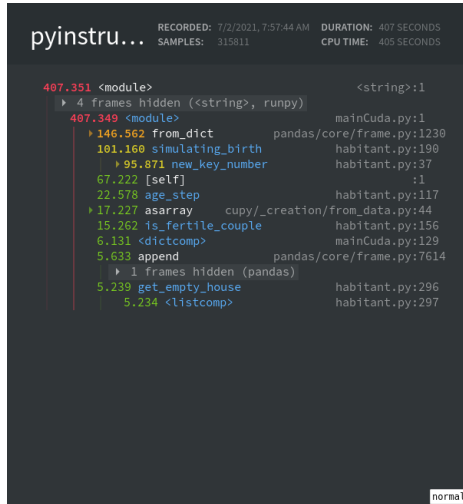


Figura16: Análisis de pyinstrument (1E4 casas de condición inicial)

## Conclusiones

- Es posible portear código python a código CuPy con mínimos cambios en el código
- Se obtuvo para una simulación de  $1.5E4$  casas una reducción del 28 % del tiempo de ejecución usando CuPy.

Muchas gracias

QR al Github del proyecto ↓



Figura17:  
MartinAramayo/habitantSimulationBeta