Modelación y simulación computacional basada en agente
Práctica 2: Modelos Basados en Agentes

Edgar Quiroz

17 de diciembre de 2020

Resumen

Se presenta la implementación, análsis y extensión de varios modelos de simulación basados en agentes.

Entre ellos el modelo de segregación de Shelling[Sch71], las termitas apiladores de Resnick[Res94], las hormigas caóticas de Miramontes[Mir95] y el Loop de Langton[Lan84].

Además, se propone como proyecto final implementar el modelo de colonia de hormigas [DMC96] para resolver algún problema *NP*-Completo, por ejemplo el problema de encontrar la solución óptima a un cubo de Rubik [DER18].

1. Generalidades

Se realizaron las implementaciones usando *mesa* [1], una biblioteca de Python para *MBA*.

Ésta provee código reutilizable para definir agentes y entornos, además de de visualización en vivo del modelo y múltiples herramientas para analizar múltiples ejecuciones variando parmátros de entorno.

Junto con esas herramientos, se utilizó *jupyter* [2], *pandas* [3] y *bokeh* [4] para manipular y visualizar los resultados.

2. Segregación de Schelling

El modelo propuesto originalmente por Thomas Schelling consiste en dos grupos de agentes, por ejemplo, rojos y verdes, que localmente tratan de satisfacer la necesidad de estar con los de su mismo grupo. De manera general este comportamiento es establecido con un parámetro conocido como porcentaje de similitud requerida o nivel de tolerancia.

Los agentes toman una decisión a partir de la información que tienen en su vecindad. Si el agente satisface las condiciones del entorno entonces se queda en su posición actual, de lo contrario, se mueve a una posición vacía.

Esta dinámica local genera como resultado la formación de cúmulos de agentes del mismo tipo, es decir, hay segregación.

2.1. Entorno

El sistema se compone de una retícula de $n \times n$ donde n se establece usualmente entre 50 y 100. Cada celda con posición (i, j) alberga a un agente rojo o verde. El sistema tiene un parámetro de densidad poblacional, usualmente se establece en 0.9. La mitad de la población se inicializa como rojos y la otra como verdes. Cada agente toma una celda de manera aleatoria.

2.2. Dinámica

Cada agente en la posición (i, j) se "muda" a un lugar vacío si en su vecindad de Moore (8-vecinos) no cumple con el porcentaje de similitud requerida.

[1] https://mesa.readthedocs.io
[2] https://jupyter.org/
[3] https://pandas.pydata.org/
[4] https://bokeh.org/

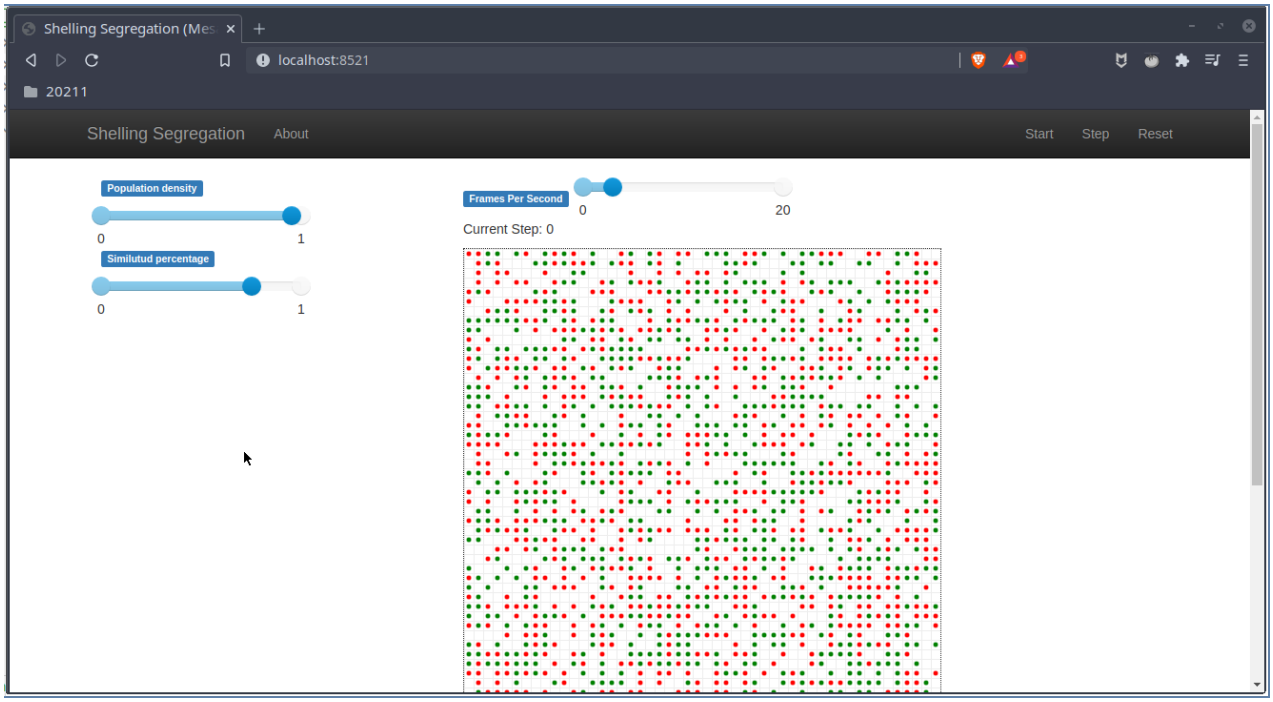


Figura 1: Interfaz gráfica del modelo de segregación de Schelling

2.3. Actividades

2.3.1. Implementación

Implemente el modelo de segregación de Schelling original [Sch71]. Puntos extra si no es en *NetLogo*.

Se realizó la implementación en *python* usando el módulo de *mesa*. La interfaz resultante se puede ver en 1.

2.3.2. Límite de similitud

Establezca el tamaño de la retícula como $n = 50$, con densidad de población 0,9. ¿Qué valor del parámetro de similitud es el límite máximo para formar dinámicas de segregación? A este valor le llamaremos S_{max}.

Determinar si hay una segregación puede ser bastante complicado, pues habría que definir un método estadísitico para medirla. Aún así, si es sufientemente fuerte, se puede apreciar a simple vista.

En la figura 2 se pueden ver dos ejecuciones del modelo. La variación de la similitud es muy pequeña pero el cambio en la dinámica es notable. Si bien en la figura 2b se podría argumentar que hay inicios de cúmulos, el hecho de que no se haya formado nada aún después de 500 pasos significa que no hay tendencia a formar cúmulos significativos.

Así que a grandes rasgos, $S_{max} \approx 0{,}75$ con los parámetros dados.

2.3.3. Límite de convergencia

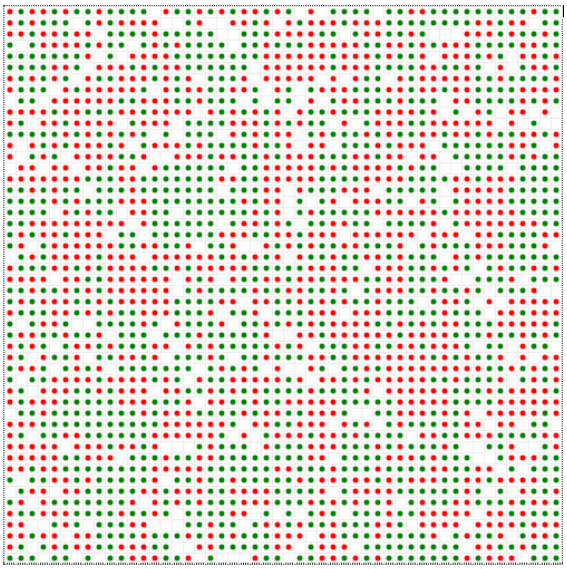
Una propuesta de medida para detectar convergencias es cuando los agentes ya no cambian de posición.

- Cuándo el parámetro de similitud es igual a S_{max}, ¿cuál es el tiempo en el que el sistema converge?

  En la figura 3 se tiene una ejecución convergente con $sim = S_{max} = 0{,}75$. Se puede ver como en la configuracón en 3a hay una frontera muy marcada entre los dos grupos. Y al final de la gráfica en 3b la cantidad de agentes satisfechos se estabiliza al rededor de 2250.

- Forme una gráfica de similitud contra tiempo de convergencia. ¿Cómo crece/decrece el tiempo de convergencia ? ¿lineal, logarítmico, exponencial?

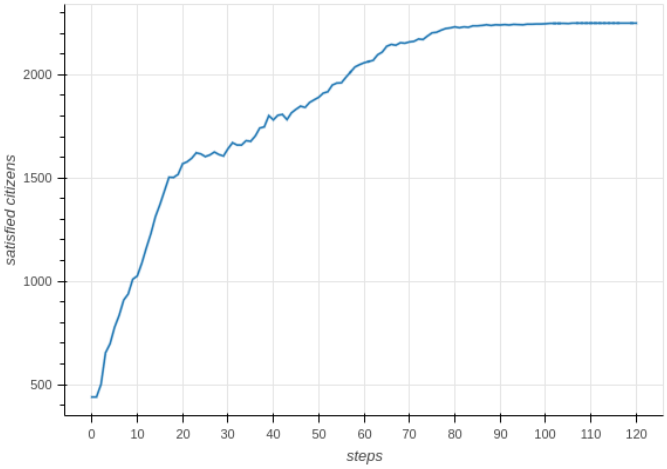
(a) $sim = 0{,}75$ después de 120 pasos



(b) $sim = 0{,}755$ después de 500 pasos

Figura 2: Estimación de S_{max}

(a) Configuración estable con $sim = S_{max}$



(b) Satisfacción con $sim = S_{max}$

Figura 3: Convergencia con $sim = S_{max}$

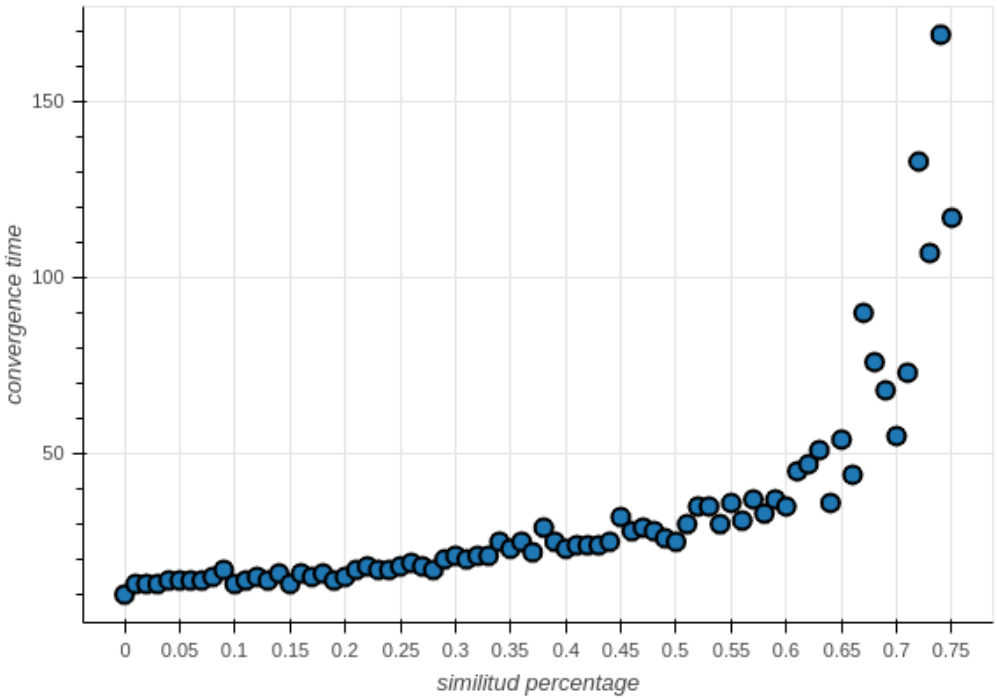


Figura 4: Tiempo de convergencia

Un agente se mueve si y sólo si no está satisfecho, así que se considera esta cantidad en lugar de contar directamente si los agentes se mueve. Pero podría ser que un agente insatisfecho, al moverse, siga insatisfechos. Así que podría ser que la cantidad de agentes satisfechos no cambie aunque sí haya movimiento.

Una manera de prevenir esto es esperar a que la cantidad de agentes satisfechos no cambie por varios turnos. Entre más turnos, menor la posibilidad de que haya movimiento a pesar de que la cantidad no cambie. En este caso, se esperó a que la cantidad se mantuviera constante por diez pasos.

Para valores de similitud mayores a S_{max}, el sistema no va a converger. Así que solo se probaron valores en $0 - 0,75$, con 400 pasos como máximo. La gráfica resultante está en la figura 4.

A simple vista, parece una función exponencial creciente.

2.3.4. Similitud normal

Establezca el parámetro de similitud como un atributo de los agentes. Inicialice la similitud requerida del agente i-ésimo a partir de una distribución normal.

- Con media 50 y desviación estándar 10.¿Cómo cambian los patrones de segregación?

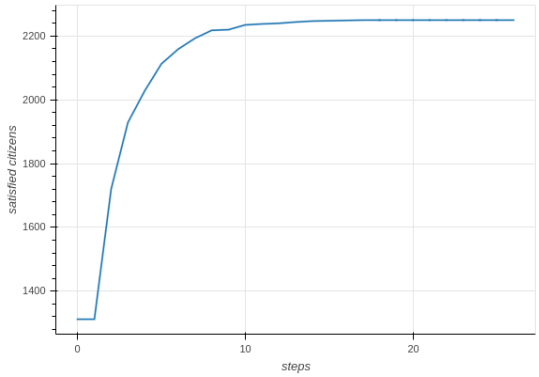
  El sistema converge estrictamente en aproximadamente 100 iteraciones, pero si se observa la figura, desde la iteración 20 se tiene un valor muy estable. Para comparar, con desviación estándar 0, el sistema converge estrictamente en 20 iteraciones.

  Este se puede deber a que una media de 50 es bastante tolerable, y la desviación es lo suficientemente pequeña para que el sistema se comporte casicomo si no hubiera desviación estándar.

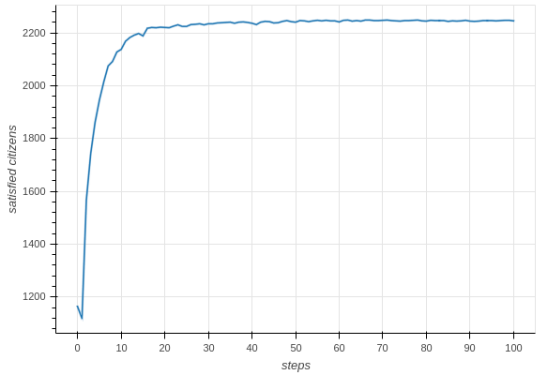
  La convergencia con desviación 0 y 10 se puede ver en la figura 6

- ¿Y con media = S_{max} y desviación estándar pequeña y grande?

  La media cerca de S_{max} hace el sistema increíblemente inestable, por lo que deja de converger inclusive con una mínima desivación, como se puede ver en la figura 6a. Con una gran desviación, los saltos son aún más erráticos, como se ve en la figura 6b.

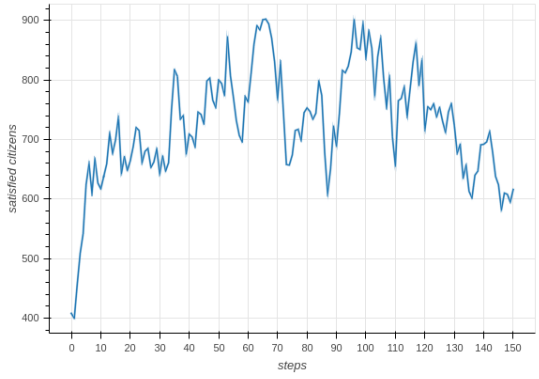


(a) Convergencia con media 50 y desviación estándar 0

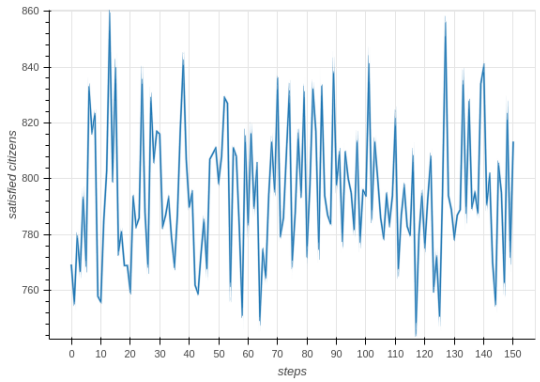


(b) Convergencia con media 50 y desviación 10

Figura 5: Convergencia con media 50



(a) Convergencia con media S_{max} y desviación estándar 0.0001



(b) Convergencia con media 50 y desviación 50

Figura 6: Convergencia con media S_{max}

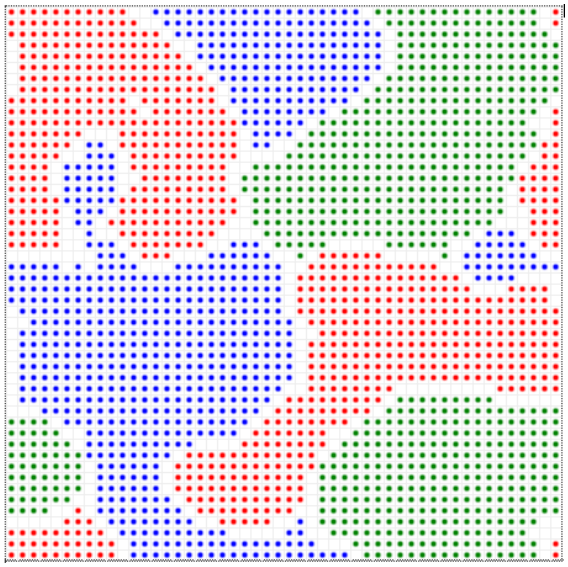
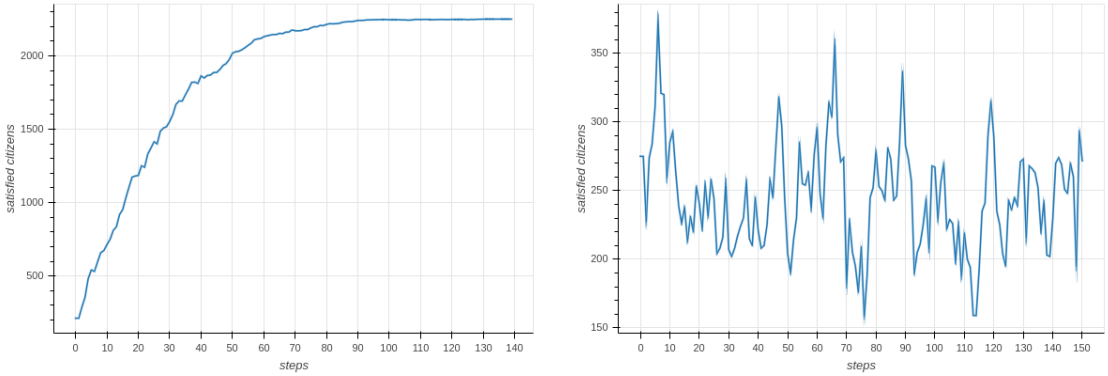


Figura 7: Convergencia con tres grupos



(a) Convergencia con $sim = 0{,}625$ (b) Divergencia con $sim = 0{,}6251$

Figura 8: S_{max} con tres grupos

2.3.5. Maś grupos

Modifique su programa previo para considerar tres tipos de agentes (rojos, verdes y azules). Inicialice cada grupo como $\frac{1}{3}$ de la población y establezca de manera global el parámetro de similitud requerida.

- ¿Se forman patrones de segregación?

  Sí, se forman cúmulos similares, solo que separados en tres tipos en lugar de solo dos. Un ejemplo de una ejecución que ha convergido se puede ver en la figura 7.

- ¿Cuál es el valor de umbral de S_{max}?

  Por prueba y error, se encontró que al rededor de 0,625 es el punto donde el sistema deja de converger. Para ilustrar esto, en la figura 8 se puede ver como un paso de tan solo 0,001 hace que el sistema ya no converga.

2.3.6. Otras propuestas

- ¿Qué otros elementos de modelación se podrían definir en el modelo de Schelling para hacerlo más realista?

  Se podría definir diferentes similitudes en función del grupo, simbolizando que algunos grupos pueden ser más tolerantes que otros.

  O también, se podría hacer que la similitud requerida cambie dependiendo de las interacciones son los otros agentes, haciéndola un valor dinámico.

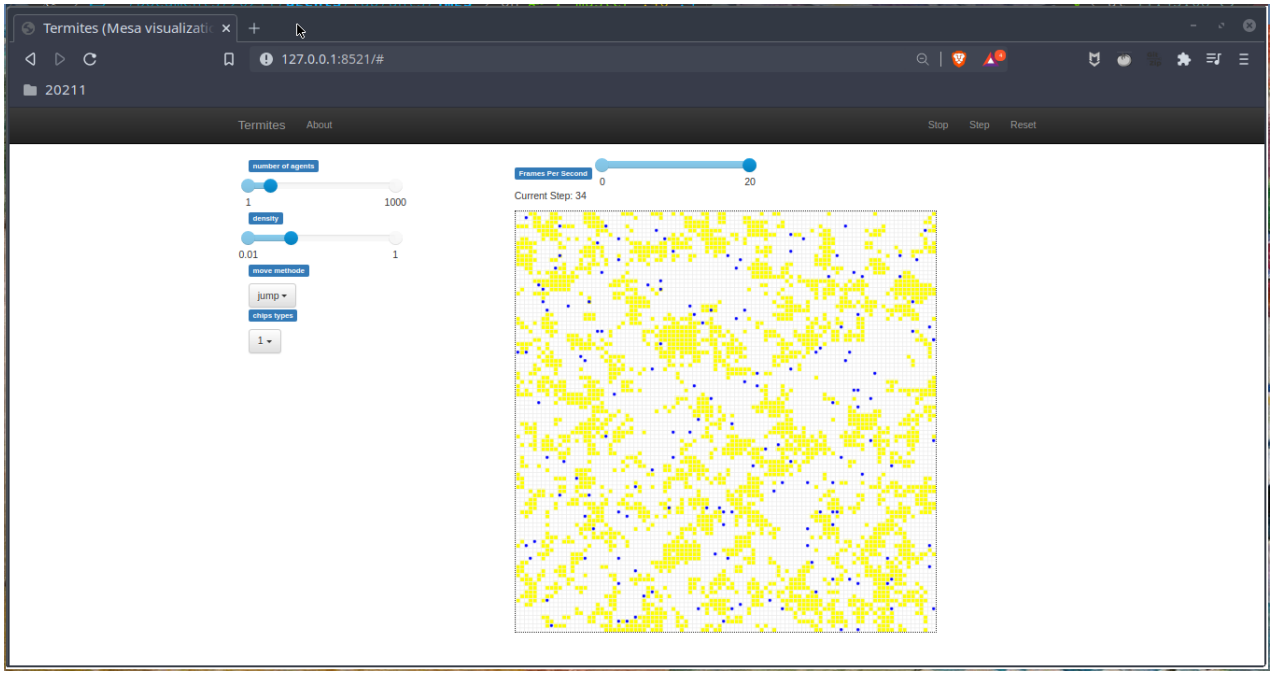


Figura 9: Interfaz gráfica del modelo de termitas

- ¿Qué otro análisis se podría implementar para explicar las dinámicas?

  Se podrían calcular varios aspectos de los cúmulos, como su tamaño promedio, o su dimensión fracta de los cúmulos. Menor dimensión y mayor tamaño significaría grupos más uniformes y cerrados.

3. Termitas apiladoras

Este modelo fue propuesto por Mitchel Resnick [Res94] como una estrategia decentralizada para apilar astillas de madera a través de simples reglas ejecutadas por termitas.

3.1. Entorno

El sistema se compone de una retícula de 100×100. Al inciciar, se colocan aleatoriamente n termintas, y se colocan astillas es una fracción d de la retícula.

3.2. Dinámica

Las termitas tienen dos reglas básicas.

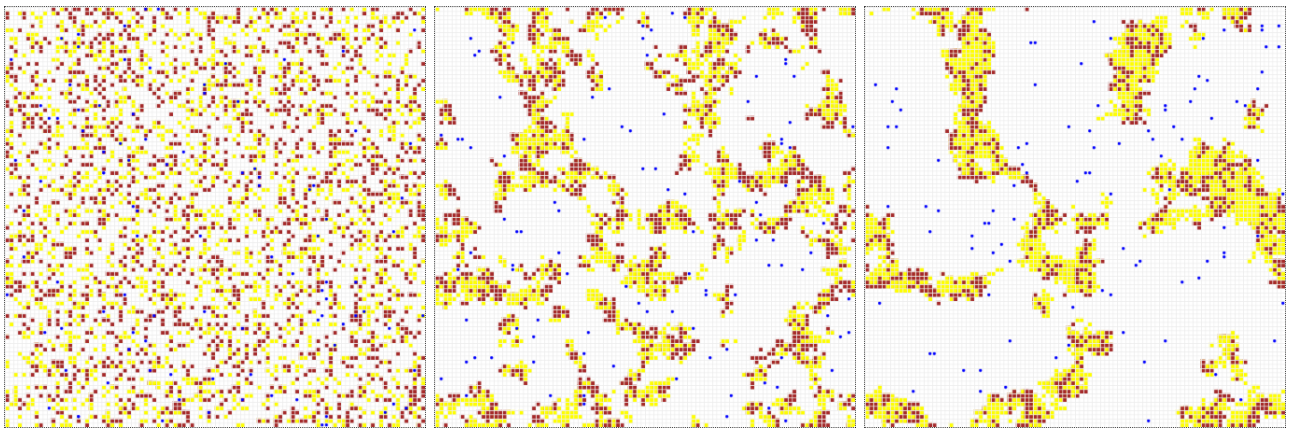
- Si la termita no está cargando nada y se encuentra una astilla, la recoge
- Si está cargando una astilla y se encuentra otra, la suelta y sigue su camino.

Las termitas realizan una caminata aleatoria con apertura de visión de -50 a 50 grados.

3.3. Actividades

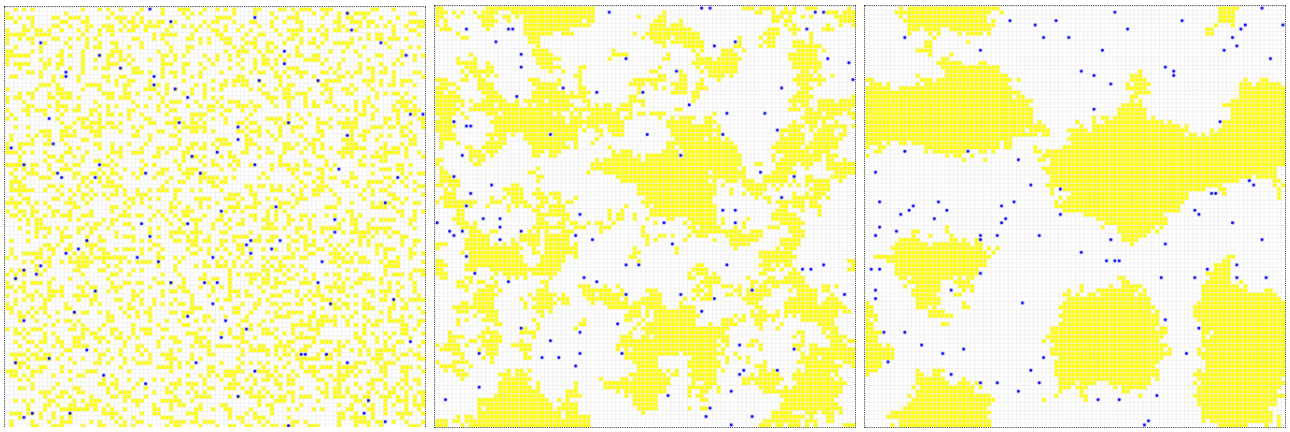
3.3.1. Implementación

Implemente el modelo de termitas apiladoras. Punto extra si no es en *NetLogo*.
Se realizó la implementación con *mesa*. La visuaización resultante se puede ver en la figura 9.



(a) 0 pasos (b) 100 pasos (c) 200 pasos

Figura 10: Evolución con dos tipos de astilla



(a) 0 pasos (b) 100 pasos (c) 200 pasos

Figura 11: Evolución de termitas con saltos

3.3.2. Recolección de datos

Implemente la recolección de varios datos del sistema al progresar el tiempo. En particular, el número de cúmulos, el tamaño promedio de los cúmulos, y el número de termitas cargando astillas. Argumente porque cada medida es útil para entender el sistema.

3.3.3. Más tipo de astillas

Extienda el modelo considerando dos tipos de astillas (amarillas y cafés). Las termintas solo sueltan las astillas al encontrar otra del mismo color. ¿Cuántos cúmulos quedan al final? Muestre la visualización de la evolución del sistema.

Una ejecución de esta variante se puede ver en la figura 10. No se forman cúmulos homogéneos. Las astilas de diferente color se “estorban” entre sí, impidiendo que se formen cúmulos grandes.

3.3.4. Caminata con saltos

En la regla original, al soltar la astilla se sigue el camino normal. En lugar de eso, haga que la termita salte a otra posición de manera aleatoria. ¿Cómo cambian los patrones formados? Muestre la visualización de la evolución del sistema y explique su comportamiento. ¿Esto se deduce a partir de las reglas locales?

Una ejecución de esta variantes se puede ver en la figura 11. Inicialmente se forman “franjas” de astillas en lugar de cúmulos, aunque eventualmente se converge a cúmulos. Una explicación es que originalmente, una termite evita adentrarse en los cúmulos. Esto genera una barrea de cúmulos que normalmente no pueden pasar, así que las termitas suelen quedarse es una misma área, lo que hace más

grandes los cúmulos existentes en esa área. Pero con el salto aleatorio, pueden viajar a cualquier parte del tablero, tomando y quitando de todos los cúmulos por igual, lo que provoca estas delgadas franjas.

Esto es un comportamiento emergente, que no se habría podido deducir de las reglas sin realizar algún tipo de simulación.

4. Hormigas al borde del caos

Las hormigas *Leptothorax* son una especia caracterizada por sus colonias pequeñas, de veinte a cien individuos, la carencia de reinas y sus patrones aparentemente aleatorios entre trabajar y descansar para alguna hormiga. Aún así, a nivel de colonial presentan gran coordinación y eficacia.

Usando información de estudios biológicos, Miramontes [Mir95] propuso un modelo para simular los periodos de trabajo reposo en colonias de hormigas artificiales. En este, se colocan hormigas artificales en una fracción d de una malla. Con una d pequeña, las hormigas no interactuan y todo su comportamiento es aleatorio. Con una d grande, las hormigas se sincronizan, trabajando y descansando exactamente en los mismo periodos.

Los experimentos mostraron que las colonias reales tienen una densidad en un punto intermedio, donde las hormigas interatúan y se sincronizan, pero aún tienen libertad suficiente para realizar acciones aleatorias. Están al borde del caos.

4.1. Entorno

La colonia está representada por una malla de 10×10. Cada casilla puede estar vacía o contener una hormiga. Al inicia, se colocan hormigas aleatoramente en una fracción d de la malla.

4.2. Dinámica

Cada hormiga puede estar activada o no. Puede pasar de inactiva a activa de dos formas. Primero, con una probabilidad p_a de forma expontánea. Luego, cuando otra hormiga activa entre a vecindad de Moore.

Una vez activa, una hormiga se mueve de manera aleatoria, una casilla a la vez. Al activarse, una hormiga adquiere una energía inicial s_a. A partir de ese momento, su energía evoluciona de la siguiente manera

$$s_i(t) = \tanh(g \cdot ((\sum_{j \in N(i)} s_i(t-1)) + s_i(t-1)) \tag{1}$$

donde g es una constante llamada ganancia y $N(i)$ es la vecindad de Moore de i. Al llegar la energía a un valor menor a ε, se considera que la energía es cero, y la hormiga se desactiva. Siguiendo los experimentos de Miramontes, se toma $\varepsilon = 10^{-16}$, $s_a = 10^{-6}$, $p_a = 10^{-2}$ y $0{,}005 \leq g \leq 0{,}5$.

4.3. Actividades

4.3.1. Implementación

Implemente un modelo basdo en agentes donde considere las siguientes caracterísiticas

- La función de activación (tangente hiperbólica)
- La vecinad de Moore
- El movimiento aleatorio de las hormigas
- El tamaño de la retícula
- El tamaño de la población

Como todas las implementaciones, se usó *mesa* para hacer la simulación. La visualización resultante se puede ver en la figura 12.

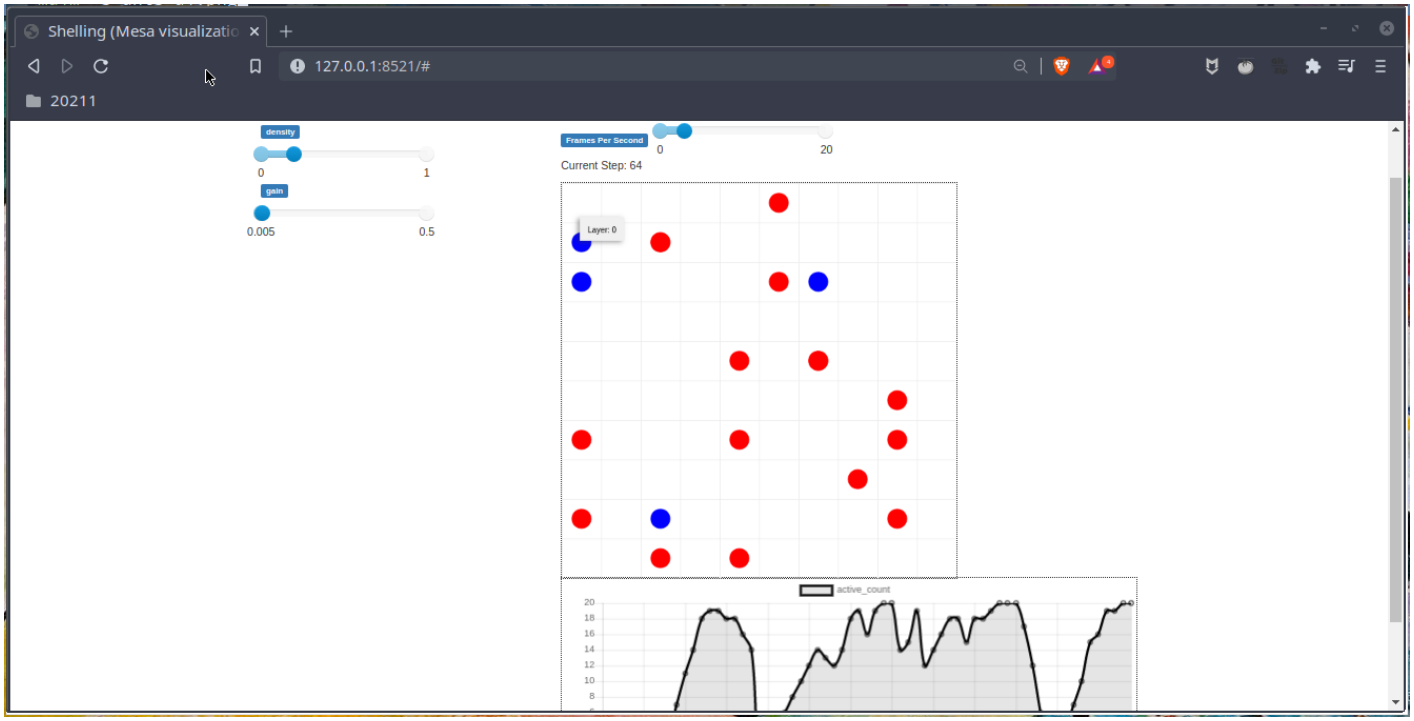


Figura 12: Interfaz gráfica de las hormigas caóticas

4.3.2. Series de tiempos de activación

Cree las series de tiempo de activación para distintas densidades de a población de hormigas. Específicamente, contar las hormigas que se encuentran en estado activo en el tiempo t, y grafica la evolución durante 600 tiempos.

Se realizaron ejecuciones con densiad de 0.01, 0.1, 0.2, 0.3, 0.5 y 0.9. En la figura 13 se puede ver dichas series.

4.3.3. Transición orden/desorden

Mostrar con varias series de tiempo que al rededor de $d = 0{,}2$ el sistema exhibe una transición de fase. Explicar a detalle.

En la figura 14 se tienen las series de tiempo para valores de d alrededor de 0,1, cuando la series es caótica, y 0,3, donde la serie es prácticamente periódica.

A partir de 1,5, se puede ver que empiezan a surgir secciones sincronizadas, entre los tiempos 75 y 125, o entre 250 y 300. Análogamente, en 0,25 aún hay periodos caóticos, como entre 400 y 450.

En 0,2 no hay nada claro. A grandes rasgos toda la serie es periódica, pero los cambios entre periodos no son tan marcados, y dentro de cada uno aún hay secciones caóticas no tan marcadas.

Como explica Miramontes, esto puede ser debido a que en $d = 0{,}1$, por ejemplo, las hormigas están lo suficientemente dispersadas para ignorarse entre sí. Esto resulta en únicamente activaciones aleatorias. Por otro lado, en $d = 0{,}3$, hay tantas hormigas que no se pueden dar espacio entre sí. Así que al activarse una, todas se activan, su energía disminuye maś o menos al mismo ritmo, por lo que las secciones de reposo también se sincronizan. En $d = 0{,}2$, hay una una cantidad de hormigas tal que pueden interatuar constantemente, pero dándonse suficiente espacio para poder tomar decisiones individuales sin afectar directamente al resto de las hormigas.

4.3.4. Compresión

Usar algún algoritmo de compresión (tar, gzip, bzi, tgz, rar, etc.) para compactar algunas de las series de tiempo. Medir la tasa de compresión para cada una de las series, variando densidades entre 0,01 y 1. Graficar los resultados.

Se computaron dos series de tiempo para cada valor de densidad entre 0 y 1 con un paso 0,01. Luego, se extrajeron los valores y se pusieron en una cadena, separados por espacios. Estas cadenas se

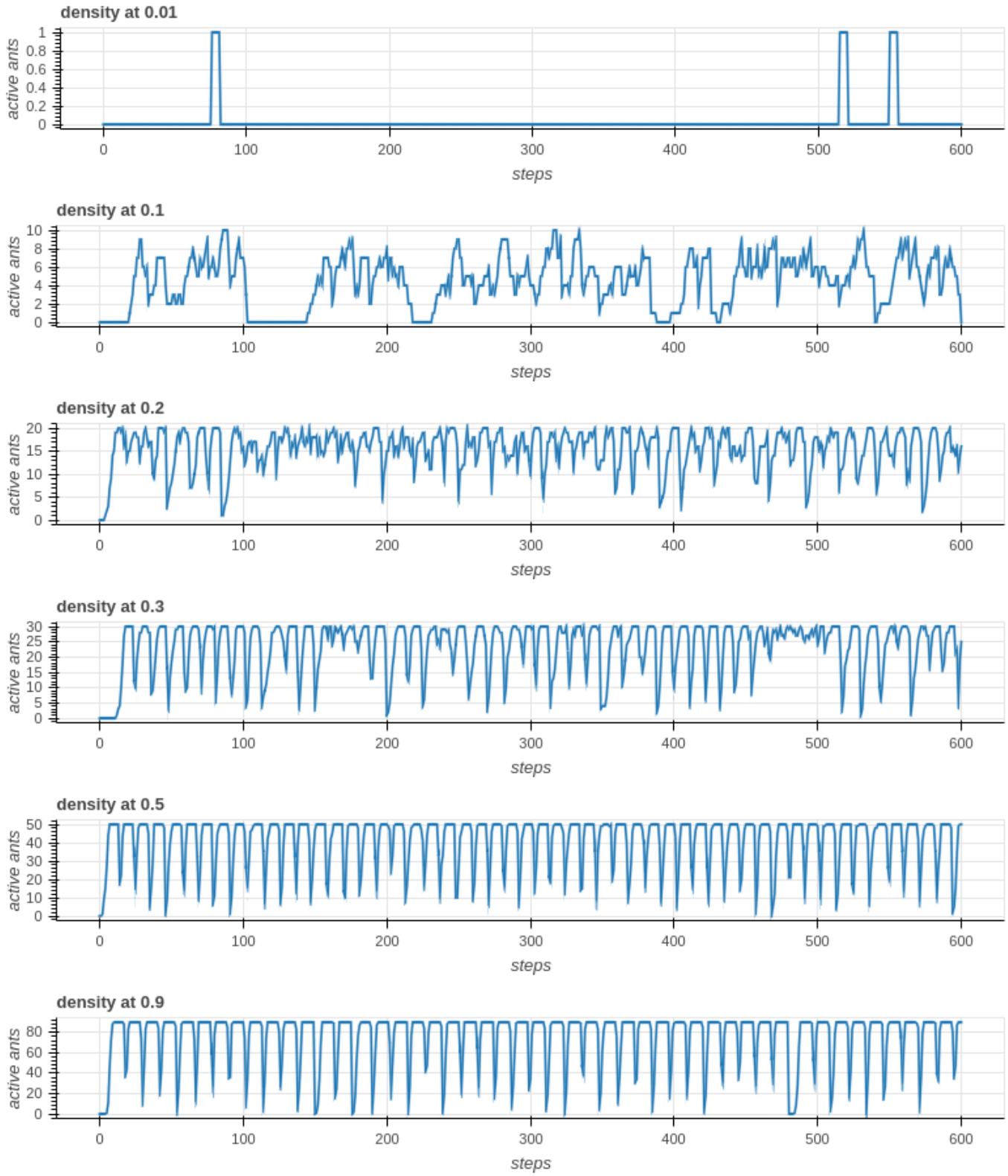


Figura 13: Series de tiempo con varias densiades

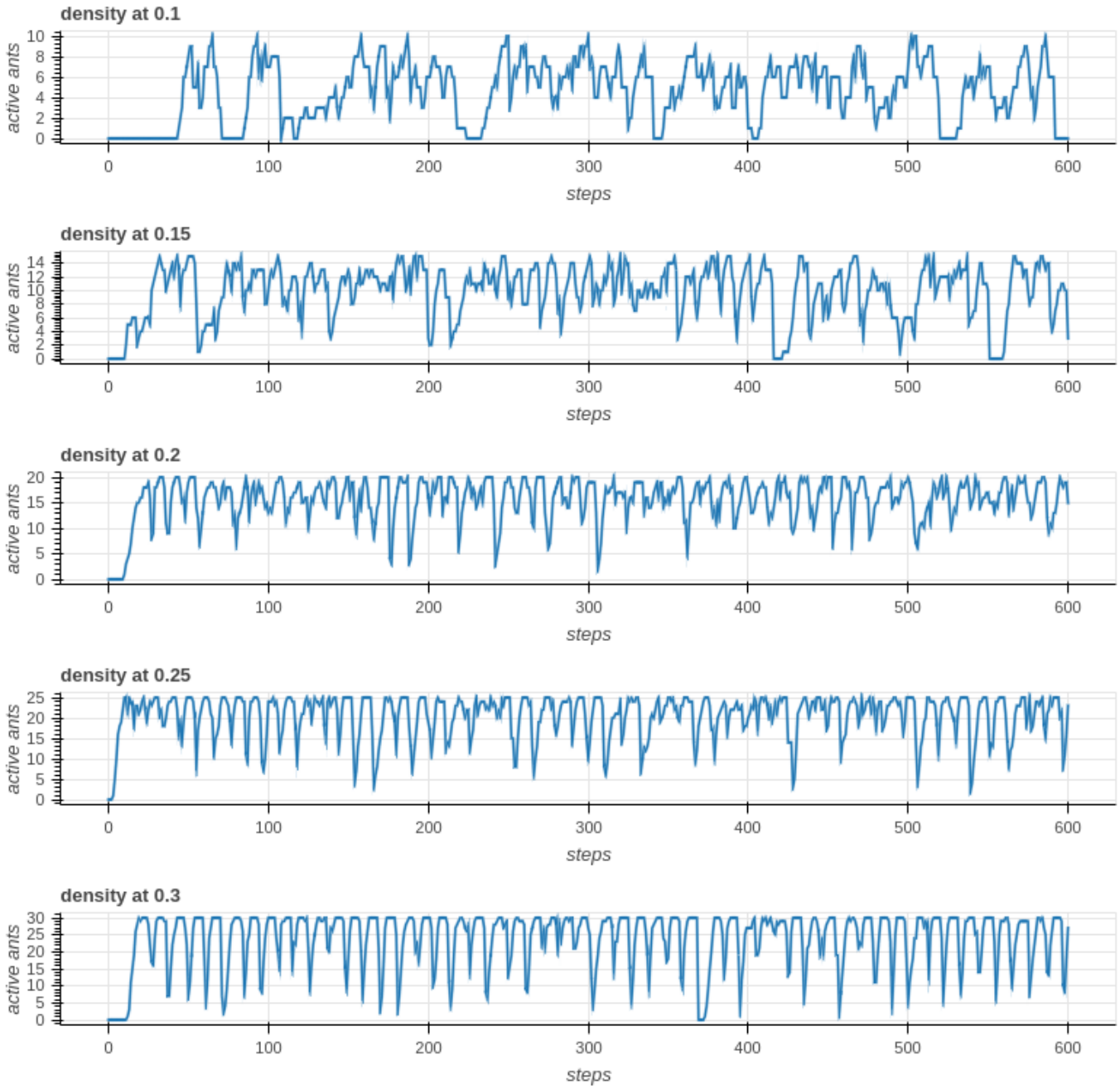


Figura 14: Series de tiempo al rededor de $d = 0,2$

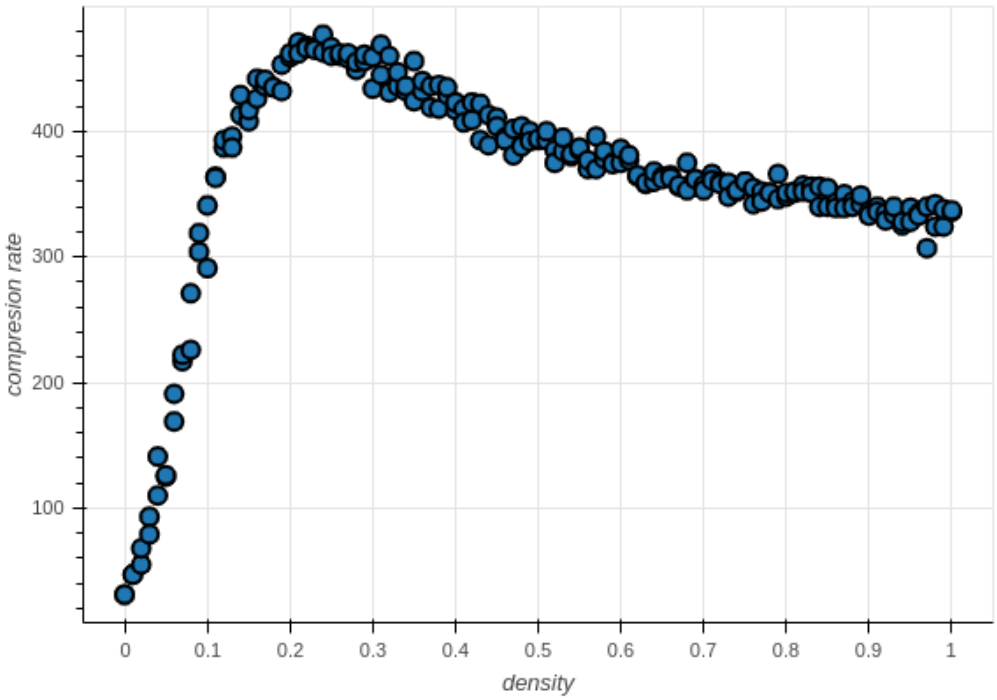


Figura 15: Tasa de compresión en función de densidad

comprimieron usando el módulo *zlib*[5] de *python*, una implementación método LZ77, el mismo usado en *gzip*.

Todas las series de tiempo tienen una longitud de 600, así que la tasa de compresión se tomó como la longitud de la cadena comprimida.

Los valores resultantes se pueden ver en la figura 15. Se puede ver un máximo aproximadamente en $d = 0{,}2$, que corresponde al borde del caos y orden observado en las series visualizadas.

5. Loop de Langton

Christopher Langton mostró la capacidad de autoreproducción de los autómatas celuladres a partir de los modelos de von Neuman y Codd [Lan84]. El loop de Langton es un modelo simple que satisface los criterios de autoreproducción. Esta estrucutura logra su simplicidad almacenando su descripción en un "loop" dinámico.

El modelo se desarrolla en un autómata celular bidimensional con ocho posibles estados en sus celdas y considerando la vecindad de Neuman para las transiciones.

5.1. Estados

- El estado cero son celdas muertas o inactivas.
- Estado 1 representa el canal interno del loop.
- Estado 2 es la coraza que protege al canal.
- 4, 5, 6, y 7 (seguidos de un 0) son codificaciones de la información que viaja por los canales.
- El estado 3 indica a la apertura o cierre de canales.

Usando los estados recien definidos, la configuración inicial se puede ver en la figura 16. Los espacios vacíos representan al estado 0.

[5] https://www.zlib.net/

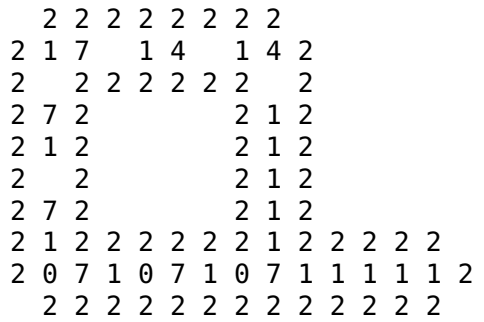


Figura 16: Configuración inicial del Loop de Langton

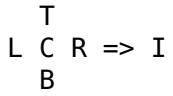


Figura 17: Nomenclatura de las vecindades

5.2. Transiciones

Como se considera una vecindad de von Neuman, entonces la función de transición debe tomar en cuenta los cuatros vecinos además de la celda acutual. Esto es $f : S^5 \to S$, con S conjunto de estados. Esto da un total de $8^5 = 32768$ entradas. Como son bastantes, Langton codificó varias de ellas. Primero, todas las reglas son invariantes ante rotaciones de la vecindad. Es decir, si (a, b, c, d, e) es una entrada de la función, entonces a (a, c, d, e, b), (a, d, e, b, c) y (a, e, b, c, d) les corresponde la misma regla. Además, no se muetran las reglas inválidas.

Aún así, el compendio de reglas es bastante amplio. Se presenta en la tabla 2. Hay que notar que el orden CTRBL → I corresponde a la denominación de la figura 17.

5.3. Actividades

5.3.1. Implementación

Implemente en el lenguaje de se preferecia el loop de Langton. Utilice coloes para representar los estados. La malla debe tener un tamaño de al menos 150×150 para obsevar adecuadamente el crecimiento de la colonia de loops.

La implementación se realizó en *mesa*. La interfaz gráfica resultante se puede ver en la figura 18. El tamaño de la malla se alteró para los diferentes ejercicios, con el fin de tener una buena visualización.

5.3.2. Segunda generación

Muestre como se ve el sistema en el tiempo 151. Este corresponde al fin de la formación de la segunda generación de loops.

La visualización del sistema se puede ver en 19. Se pueden ver dos loops idénticos perfectamente formados, con la diferencia de que el loop original está rotado.

5.3.3. Séptima generación

Muestre como se ve el sistema en la generación 7. ¿Cuántos loops activos, muertos y moribundos hay?

La visualización del sistema se puede ver en la figura 20. Tiene un total de 11 loops muertos, que tiene un núcleo sin flujo de información. Tiene 10 loops moribundos, que aún tiene flujo de información pero no canales nuevos a donde mandarla. Y tiene 18 loops activos, que tiene flujo y canales activos. En total hay 39 loops en esta generación.

5.3.4. Crecimiento

Deduzca una fórmula para el crecimiento de la colonia de loops.

Para tener una referencia, en las tablas 3 y 4 están los resultados de simular y contar directamente como crece la colonia de loops.

En cada generación, cada loop vivo se replica, así que la cantidad total de loops se podría ver como

$$\eta(t) = \begin{cases} 1, & \text{si } t = 1 \quad 14 \\ \eta(t-1) + \alpha(t-1) & \text{en otro caso} \end{cases}$$

CTRBL→I	CTRBL→I	CTRBL→I	CTRBL→I	CTRBL→I
00000→0	02527→1	11322→1	20242→2	30102→1
00001→2	10001→1	12224→4	20245→2	30122→0
00002→0	10006→1	12227→7	20252→0	30251→1
00003→0	10007→7	12243→4	20255→2	40112→0
00005→0	10011→1	12254→7	20262→2	40122→0
00006→3	10012→1	12324→4	20272→2	40125→0
00007→1	10021→1	12327→7	20312→2	40212→0
00011→2	10024→4	12425→5	20321→6	40222→1
00012→2	10027→7	12426→7	20322→6	40232→6
00013→2	10051→1	12527→5	20342→2	40252→0
00021→2	10101→1	20001→2	20422→2	40322→1
00022→0	10111→1	20002→2	20512→2	50002→2
00023→0	10124→4	20004→2	20521→2	50021→5
00026→2	10127→7	20007→1	20522→2	50022→5
00027→2	10202→6	20012→2	20552→1	50023→2
00032→0	10212→1	20015→2	20572→5	50027→2
00052→5	10221→1	20021→2	20622→2	50052→0
00062→2	10224→4	20022→2	20672→2	50202→2
00072→2	10226→3	20023→2	20712→2	50212→2
00102→2	10227→7	20024→2	20722→2	50215→2
00112→0	10232→7	20025→0	20742→2	50222→0
00202→0	10242→4	20026→2	20772→2	50224→4
00203→0	10262→6	20027→2	21122→2	50272→2
00205→0	10264→4	20032→6	21126→1	51212→2
00212→5	10267→7	20042→3	21222→2	51222→0
00222→0	10271→0	20051→7	21224→2	51242→2
00232→2	10272→7	20052→2	21226→2	51272→2
00522→2	10542→7	20057→5	21227→2	60001→1
01232→1	11112→1	20072→2	21422→2	60002→1
01242→1	11122→1	20102→2	21522→2	60212→0
01252→5	11124→4	20112→2	21622→2	61212→5
01262→1	11125→1	20122→2	21722→2	61213→1
01272→1	11126→1	20142→2	22227→2	61222→5
01275→1	11127→7	20172→2	22244→2	70007→7
01422→1	11152→2	20202→2	22246→2	70112→0
01432→1	11212→1	20203→2	22276→2	70122→0
01442→1	11222→1	20205→2	22277→2	70125→0
01472→1	11224→4	20207→3	30001→3	70212→0
01625→1	11225→1	20212→2	30002→2	70222→1
01722→1	11227→7	20215→2	30004→1	70225→1
01725→5	11232→1	20221→2	30007→6	70232→1
01752→1	11242→4	20222→2	30012→3	70252→5
01762→1	11262→1	20227→2	30042→1	70272→0
01772→1	11272→7	20232→1	30062→2	

Cuadro 2: Transiciones para el Loop de Langton

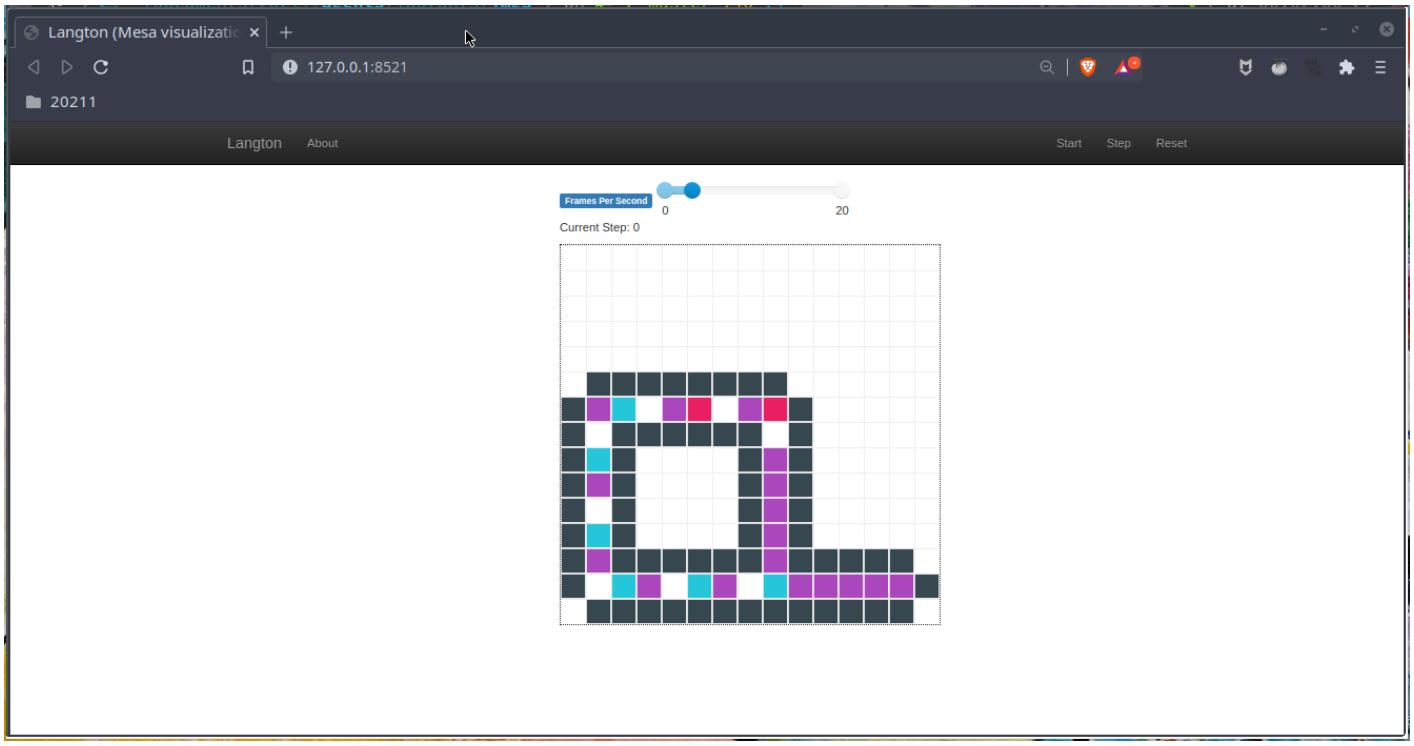


Figura 18: Visualización del loop de Langton

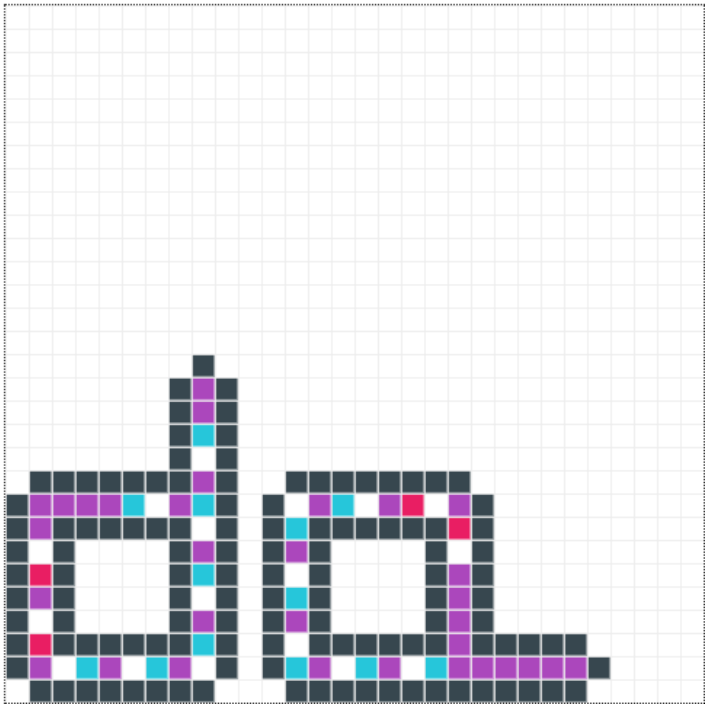


Figura 19: Loop de Lagton al tiempo 150

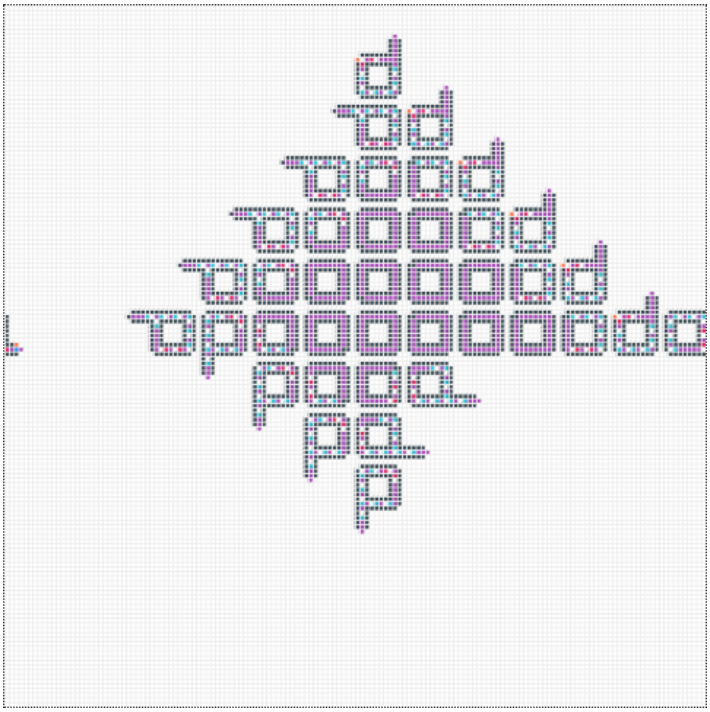


Figura 20: Loop de Lagton después de siete generaciones

Gen	Vivo	Moribundo	Muerto	Total
1	1	0	0	1
2	2	0	0	2
3	4	0	0	4
4	7	1	0	8
5	10	4	1	15
6	14	6	5	25
7	18	10	11	39
8	22	14	21	57

Cuadro 3: Evolución de la colonia de loops

Gen	Génesis	Primogénito	Otro	Total
1	1	0	0	1
2	1	1	0	2
3	1	3	0	4
4	1	5	1	7
5	0	7	3	10
6	0	8	6	14
7	0	8	10	18
8	0	8	14	22

Cuadro 4: Evolución de los tipos de loop vivos

- Génesis

  El loop inicial. Como no tiene obstáculos, tiene cuatros descendientes antes de morir.

- Primogénito

  Son los primogénitos de los cuatros descendientes del loop inicial. Al ser creados, estos crean un nuevo primogénitos siguiendo la dirección en la que fueron creados. Al siguiente turno, rotan y crean otro loop 90 grados en sentido antihorario. En el siguiente turno, vuelven a rotar, se topan con su padre y mueren

  Así que tienen dos descendientes.

- Otro

  Corresponden a los segundo hijos de los primogénitos. Estos crecen de forma perpendicular a las direcciones de los primogénitos. Al crearse, crean otro loop en la misma dirección en la que fueron credos. Luego rotan, se encuentran con otro loop y mueren. Estos tienen un descendiente.

El loop inicial tarda en crear sus descendientes, así que las cuatro primeras generaciones se pueden pensar como un periodo de estabilización. Además, cada uno de sus descendientes debe completar un ciclo suyo para terminar de estabilizar la dinámica. Como el último descendiente del loop inicial es creado en la generación 4, entonces en la generación 6 se podría decir que la dinámica global del sistema se estabiliza.

En este punto, no hay loop iniciales. Hay cuatro primogénitos en su primera fase y cuatro en su segunda fase, dando un total de ocho. Para los otros loops, cada uno se replicó y murió, asi que se mantiene esa cantidad, pero además cada primogénito en segunda fase crea uno nuevo. Hay entonces $\omega(t) = \omega(6) + 4(t-6), t \geq 6$ otros loops.

Así que para $t \geq 6$ se tiene que

$$\alpha(t) = 8 + \omega(t) = 8 + \omega(6) + 4(t-6) = 4t + \omega(6) - 16$$

De la tabla 4 se tiene que $\omega(6) = 6$, así que la fórmula queda como

$$\alpha(t) = 4t - 10 \tag{3}$$

Entonces, restrigiendo que $t > 6$, la ecuación 2 se puede reescribir como

$$\begin{aligned}\eta(t) &= \eta(6) + \sum_{k=6}^{t-1}(4k-10) \\ &= \eta(6) + 4(\sum_{k=6}^{t-1} k) - 10(t-6) \\ &= \eta(6) + 4(\sum_{k=6}^{t-1} k) - 10t + 60\end{aligned}$$

Y de la tabla 3 se puede obtener que que $\eta(6) = 25$. Por lo que

$$\begin{aligned}&= 25 + 4(\sum_{k=6}^{t-1} t) - 10t + 60 \\ &= 4(\frac{t(t-1)}{2} - \frac{6(7)}{2}) - 10t + 85 \\ &= 2t(t-1) - 60 - 10t + 85 \\ &= 2t^2 - 12t + 25\end{aligned}$$

Así que para $t > 6$, la cantidad de loops en la colonia en la generación t está dada por

$$\eta(t) = 2t^2 - 12t + 25 \tag{4}$$

En la tabla 3 se puede ver que la fórmula funciona para $t = 7, 8$.

6. Proyeto final

Marco Dorigo propuso un modelo basado en el comportamiento social de las hormigas para optimización sobre gráficas [DMC96]. Este intenta imitar los rastros de feromonas usados por las hormigas para comunicar donde se encuentran las mejores fuentes de comida. Esta técnica fue utilizada con éxito para aproximar soluciones a problemas *NP*-Duros, como el problema del agente viajero, TSP.

Debido a su eficacia, muchas variantes han surgido [DBS06]. Para proyecto final, se propone primero implementar una visualización de alguna de las variantes del modelo para resolver el problema de TSP. Luego, como extensión del modelo, adaptar algún otro problema *NP*-Completo para ser resuelto por el modelo de colonia de hormigas. Esto aún está al aire, pero una propuesta interesante sería resolver un cubo de Rubik [DER18], que recientemente se demostró *NP*-Completo.

Referencias

[DBS06] Marco Dorigo, Mauro Birattari y Thomas Stutzle. “Ant colony optimization”. En: *IEEE Computational Intelligence Magazine* 1.4 (nov. de 2006), págs. 28-39. ISSN: 1556-603X. DOI: 10.1109/mci.2006.329691. URL: http://dx.doi.org/10.1109/MCI.2006.329691.

[DER18] Erik D. Demaine, Sarah Eisenstat y Mikhail Rudoy. “Solving the Rubik’s Cube Optimally is NP-complete”. en. En: (2018). DOI: 10.4230/LIPICS.STACS.2018.24. URL: http://drops.dagstuhl.de/opus/volltexte/2018/8533/.

[DMC96] M. Dorigo, V. Maniezzo y A. Colorni. “Ant system: optimization by a colony of cooperating agents”. En: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 26.1 (feb. de 1996), págs. 29-41. ISSN: 1941-0492. DOI: 10.1109/3477.484436. URL: http://dx.doi.org/10.1109/3477.484436.

[Lan84] Christopher G. Langton. “Self-Reproduction in Cellular Automata”. En: *Physica D: Nonlinear Phenomena* 10.1-2 (1984), págs. 135-144. DOI: 10.1016/0167-2789(84)90256-2. URL: https://doi.org/10.1016/0167-2789(84)90256-2.

[Mir95] Octavio Miramontes. “Order-disorder transitions in the behavior of ant societies”. En: *Complexity* 1.3 (ene. de 1995), págs. 56-60. ISSN: 1076-2787. DOI: 10.1002/cplx.6130010313. URL: http://dx.doi.org/10.1002/cplx.6130010313.

[Res94] Mitchel Resnick. *Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds*. Cambridge, MA, USA: MIT Press, 1994. ISBN: 0262181622.

[Sch71] Thomas C. Schelling. “Dynamic models of segregation†”. En: *The Journal of Mathematical Sociology* 1.2 (jul. de 1971), págs. 143-186. ISSN: 1545-5874. DOI: 10.1080/0022250x.1971.9989794. URL: http://dx.doi.org/10.1080/0022250X.1971.9989794.