Construyendo Simulaciones Sociales Multi-agente: Tutorial de auto-construcción, paso a paso, con NetLogo 6

Segunda Parte (2/2)

Francisco J. MIGUEL QUESADA [6]
LSDS-UAB - Dept. Sociologia UAB
(v2.2, Septiembre 2017)

Sobre este tutorial

A lo largo del presente tutorial se guía paso a paso en la construyendo de un pequeño modelo de simulación social (un prototipo de demostración), escrito y ejecutado en un entorno de programación simple llamado NetLogo. El objetivo del tutorial es mostrar, de una manera práctica e inmediata, los elementos básicos de la aproximación a la simulación de sistemas sociales denominada "modelado basado en múltiples agentes" (MABM), así como servir de iniciación a la programación usando Netlogo. Obviamente, la simulación se refiere a una sociedad extremadamente estilizada o simplificada, sin embargo resulta útil para introducir los conceptos y elementos generales de cualquier simulación social.

Una vez descargado e instalado el entorno NetLogo se puede seguir el presente tutorial tecleando, o recortando y pegando, las líneas de código informático en el lugar adecuado (la pantalla "Código" del entorno NetLogo), y se puede ejecutar la simulación (desde la pantalla "Ejecutar") para ver los efectos del código. No se requiere que quien siga el presente tutorial comprenda totalmente el código escrito, ni que aprenda a programar

[6] *Agradecimientos a las revisiones previas: Francisco Linares, José Antonio López, y Oriol Griera.*

sociedades virtuales mediante el mismo. Para profundizar en el aprendizaje de construcción y uso de simulaciones sociales se incluye en la parte final un listado de recursos adicionales.

> ***/!\ Importante:*** *Mientras se van añadiendo líneas de código, siguiendo paso a paso el presente tutorial, pueden aparecer diversos mensajes de error debido a que el interpretador del lenguaje encuentra incoherencias o partes del modelo aún sin definir. Igualmente, hasta que el modelo no está lo suficientemente desarrollado no es posible ejecutarlo.*

> ***Nota:*** *Las imágenes de este tutorial corresponden a la versión 5.3.1. Pueden existir pequeñas diferencias en el interfaz gráfico de versiones posteriores.*

Segunda parte del tutorial paso a paso

Seguir el primer tutorial es requisito imprescindible para continuar con este. En esta segunda parte del tutorial continuarás construyendo la pequeña sociedad virtual demostrativa de la metodología de "simulación social" que se inició a lo largo del primer tutorial.

Podrás incrementar las capacidades cognitivas y sociales de los agentes. Crearás paso a paso versiones del modelo existente el las cuales se añaden mecanismos como la percepción del entorno *(¿qué tienda es aquella?)*, el uso de una memoria o mapa mental del mundo *(¿por qué tiendas he pasado antes, y donde están ubicadas?)*, e incluso la comunicación entre agentes *(Hola comprador, yo conozco la ubicación de estas tiendas ¿tú cuales conoces?).*

Cada modificación del modelo será ejecutada hasta generar resultados agregados comparables con la versión inmediata anterior. La contrastación del indicador "tiempo total de compra" permitirá ejemplificar la potencia explicativa del uso de modelos de simulación social en la evaluación y el desarrollo de hipótesis relativas a cómo se generan procesos sociales.

Versión 2: "El Paraíso de la compra - Memoria"

Como siguiente paso en el desarrollo del modelo, se propone incrementar la capacidad de los agentes para percibir su entorno, y se les dotará de memoria acerca de los lugares por los que han pasado. A partir de ahora los agentes de la raza "compradores" podrán:

- "ver" cualquier tienda que se encuentre en su "vecindario de Moore" *(en alguna de las 8 parcelas inmediatamente adyacentes a aquella en la que se encuentran)*,
- caminar hacia una de ellas, si no hay tiendas en la parcela en que se encuentran, y
- recordar las coordenadas de todas las tiendas que han "visto" anteriormente.

Esta modificación implica incrementar las habilidades cognitivas de los agentes con un cierto radio de visión a distancia (a una parcela de distancia), con una lista de objetivos hacia los que orientar sus desplazamientos y con un *mapa personal* del mundo virtual. Estas mejoras hipotéticamente podría reducir el elemento azaroso del movimiento de los agentes-compradores, y de esta forma mejorar significativamente su eficiencia en el proceso agregado de compra. Tal mejora hipotética reduciría el tiempo total de compra.

2.1. Ampliación del modelo básico: Inteligencia individual.

Para trabajar con esta segunda versión o modelo se recomienda realizar una copia de la primera, esto es: 1) cargar la primera versión del modelo en NetLogo, 2) grabarla con otro nombre *Archivo / Guardar como…*, y 3) comprobar el cambio de nombre en la parte superior de la ventana de la aplicación, y seguir trabajando con la **nueva** versión para realizar sobre ella las siguientes modificaciones.

> ***/!\ Atención:*** *Para evitar cualquier riesgo de perder un modelo anterior operativo, cuando hagas modificaciones trabaja siempre con una copia del modelo anterior. Existe una convención muy extendida y muy útil para asignar numeración correlativa a*

sucesivas versiones del mismo código informático en función de la magnitud de los cambios realizados en cada nueva versión, por ejemplo, 1.0, 1.1, 1.2, 2.0, 2.0.1, 2.1...]

La mayor parte del código permanece intacto, pero se requieren algunas modificaciones y adiciones que se irán indicando a continuación.

- Modifica el procedimiento `comprar` para que contenga esta nueva versión de instrucciones:

```
to comprar
  let tienda-mas-cercana-conocida 0
  recordar productos-de-venta-aqui
  if any? tiendas-here [ compra-si-necesitas ]
  set tienda-mas-cercana-conocida buscar-memoria
  ifelse tienda-mas-cercana-conocida != "Ninguna"
      [ avanzar-hacia tienda-mas-cercana-conocida ]
      [
       avanzar-al-azar
      ]
end
```

El procedimiento `comprar` ahora es más complejo e incorpora nuevas líneas de código. Cada agente tiene que explorar los alrededores para comprobar qué tiendas están en las cercanías y además tiene que recordar lo que ha visto. Por otro lado, cuando se mueve, 1) en primer lugar revisa su memoria en busca de la ubicación de la tienda más cercana a su localización actual y 2) se encamina hacia esa dirección.

La primera acción es llevada a cabo mediante la orden `recordar productos-de-venta-aqui` compuesta por un procedimiento (`recordar`) que se aplica a una función (`productos-de-venta-aqui`).

Para añadir el siguiente procedimiento y función al código:

- Teclea, o copia, al final del código ya existente.

```
to recordar [ localizacion-tiendas ]
  set memoria remove-duplicates sentence memoria localizacion-tiendas
end
```

```
to-report productos-de-venta-aqui
  report [ (list producto xcor ycor) ] of (tiendas-on neighbors)
end
```

- La función de exploración del entorno (`productos-de-venta-aquí`):
  El agente necesita recordar tres elementos en relación a las tiendas que percibe en su entorno: el producto vendido en cada tienda, y las "coordenadas" (X e Y) de localización de la tienda. Para realizar esta exploración del entorno inmediato se precisa de una función diseñada de tal forma que proporcione una lista con las tres informaciones mencionadas para cada tienda situada en alguna de las parcelas adyacentes a la parcela en que se encuentra el agente (i.e., `tiendas-on neighbors`). Tal información se organiza en forma de una lista de tríadas para cada ubicación desde la cual el agente realiza la exploración *(por ejemplo: ["cafe" "10" "5" "cerveza" "9" "4"])*. Cada vez que un agente activa esta función se genera una lista con las localizaciones de las tiendas "visibles" a una parcela de distancia desde su ubicación actual.
- El procedimiento de memorización (`recordar`):
  Recordar los datos sobre las tiendas visitadas implica añadir las nuevas localizaciones, proporcionadas por la función `productos-de-venta-aquí`, a las previamente existentes en la memoria de cada agente (la palabra clave `sentence` unifica los contenidos de dos listas en una sola nueva lista), pero además se debe descartar cualquier duplicidad, en el caso de que la misma tienda ya haya sido "detectada" anteriormente (`remove-duplicates`).

  > *[Fíjate en la diferencia esencial entre los procedimientos (`to...end`) y las funciones (`to-report...end`). Los primeros dan lugar a la ejecución de acciones, mientras que las segundas simplemente proporcionan un resultado. Ambos elementos son definidos por el usuario, contrariamente a sus correlatos predefinidos en el lenguaje NetLogo: las instruciones (e.g., `clear-all`) y las primitivas (e.g., `random A`).]*

Tras la fase de evaluación, o percepción del entorno, el procedimiento `comprar` continúa con indicaciones sobre cómo proceder con respecto al desplazamiento por el mundo simulado.

Cuando un agente-comprador se desplaza (ya no de forma aleatoria, como en el primer modelo) primero revisará su propia memoria para encontrar la localización de una tienda "detectada" anteriormente. Esta revisión de memoria se lleva a cabo en dos fases, mediante la función `buscar-memoria`.

- Añade el siguiente código al ya existente:

```
to-report buscar-memoria
  let tienda-por-visitar 0
  set tienda-por-visitar filter [ [?1] -> member? (first ?1) lista-de-compra ] memoria
  ifelse empty? tienda-por-visitar
        [ report "Ninguna" ]
        [ report first (sort-by
          [ [?1 ?2] -> distancexy (last butlast ?1) (last ?1) < distancexy (last butlast ?2) (last ?2) ]
          tienda-por-visitar)
        ]
end
```

1. Primero, los registros actualmente en memoria son filtrados para extraer tan sólo aquellos que se relacionan con tiendas que vendan productos que todavía están en la lista de la compra del agente, esto es, que aún no han sido visitadas/comprados. Estos registros pasan a constituir la lista de datos PRIORITARIA que el agente tiene como objetivo inmediato `tienda-por-visitar`.
2. Si no hay ninguna tienda conocida que venda productos de los que desea el agente, la función `buscar-memoria` devuelve "`Ninguna`".
3. En caso contrario, el agente ordena los registros de las tiendas en función de la distancia a que se encuentran de su posición actual, y la función `buscar-memoria` devuelve el primer registro de la lista ordenada. Esto es, la tienda más cercana al agente de entre las que contienen productos que en este momento desea comprar.

Recuerda que el registro de memoria de cada tienda se almacena como una lista de tríadas: el primer elemento de estas tríadas es el nombre del producto, el segundo la coordenada X de la tienda y el tercero la coordenada Y. Estos elementos se pueden recuperar usando tres órdenes de NetLogo, `first` (que devuelve el primer elemento de una lista, en este caso el producto), `last` (que devuelve el último elemento de una lista, en este caso la coordenada y),

y `butlast` (que devuelve toda la lista excepto el último elemento). Fíjate cómo la combinación `last butlast`, aplicada a una lista de tres elementos como la aquí utilizada, devuelve el "penúltimo" elemento (esto es, el segundo de tres).

El procedimiento de ordenación, `sort-by`, por su parte realiza una comparación de cada tríada de la lista con todas las demás, para determinar en qué posición se debe colocar en la lista ordenada definitiva. Las comparaciones se realizan repetidamente sobre pares de tríadas mediante la expresión incluida en corchetes, sustituyendo una tríada por `?1` y la otra por `?2`. Así, la expresión `[ distancexy (last butlast ?1) (last ?1) < distancexy (last butlast ?2) (last ?2) ]` compara dos distancias:

1. entre la posición actual del agente y la ubicación definida por las coordenadas X e Y de uno de los registros de memoria `(last butlast ?1) (last ?1)`,y
2. entre el agente y el punto definido por las coordenadas X e Y extraídas de otro registro `(last butlast ?2) (last ?2)`.

Al proceder a una ordenación recursiva en función de este criterio el conjunto de tríadas que integra la lista de memoria de cada agente contendrá:

1. la totalidad de tríadas correspondientes a sus exploraciones anteriores,
2. ordenadas de menor a mayor distancia respecto a la posición actual de dicho agente.

> *[En este caso se utiliza una técnica clásica de computación que usa la ordenación recursiva sobre un conjunto de datos. Se basa en la idea de que ciertos elementos "flotan" sobre otros, y acaban situándose en el inicio de una lista ordenada, o en la superficie de un apilamiento de elementos.]*

Una vez que un agente-comprador ha revisado la totalidad de su memoria, y ha seleccionado uno de los agentes-tienda, de "regreso" al procedimiento `comprar`, el agente se mueve hacia la posición de tal tienda usando el nuevo procedimiento `avanzar-hacia`, que debe ser incorporado al código ya existente.

- Añade el siguiente procedimiento al código del modelo.

```
to avanzar-hacia [ localizacion-de-la-tienda ]
  if not (xcor = (last butlast localizacion-de-la-tienda) and ycor = (last localizacion-de-la-tienda))
    [ set heading towardsxy (last butlast localizacion-de-la-tienda) (last localizacion-de-la-tienda)
```

```
        avanzar ]
end
```

Este nuevo procedimiento `avanzar-hacia` comprueba primero si el agente se encuentra ya -casualmente- exactamente donde quiere ir.

1. Si es así no se mueve.
2. Si no está dónde quiere ir, gira orientando su rumbo hacia la localización de la tienda deseada y avanza.

El procedimiento `avanzar`, escrito ya en la primera versión del modelo, hace que el agente se desplace una parcela hacia adelante, acercándose hacia su objetivo.

> *[El procedimiento `avanzar` implica tan sólo el movimiento hacia adelante. En el primer modelo estaba vinculado a un procedimiento que seleccionaba aleatoriamemente la orientación de los agentes antes del avance, mientras que en este segundo modelo está vinculado a un procedimiento que establece la orientación a partir de un "objetivo" fijado por el agente previamente.*
>
> *Resulta especialmente útil codificar cada proceso como distintos procedimientos, vinculados entre sí; de esta forma se dispone de un conjunto de módulos que permiten modificar un modelo añadiendo pocos procedimientos nuevos y cambiando los vínculos con los anteriores. Mediante este tipo de programación "modular" se facilita: 1) la ampliación o mejora sucesiva del modelo, 2) la comprensión analítica del mismo y 3) el aprovechamiento de muchas líneas de código anterior.]*

- Almacena los últimos cambios con Menú / Archivo / Guardar.

2.2. Evaluación de la eficiencia del modelo ampliado con cognición.

En este segundo modelo los agentes no se desplazan aleatoriamente por el mundo simulado hasta que tropiezan ciegamente con una tienda. Su movimiento está vinculado a un procedimiento que establece la orientación de cada comprador hacia un objetivo fijado "intencionalmente" por el agente tras un proceso deliberativo de evaluación de informaciones memorizadas previamente. Este segundo modelo ampliado mediante cognición se inspira en el modelo teórico de la acción humana recogido por las "arquitecturas de software BDI", así

denominadas por sus componentes fundamentales: *Belief, Desire, Intention*.[7] Tales componentes pueden reconocerse (junto con otro componente añadido: las Oportunidades de acción) en el conjunto de ampliaciones al modelo básico realizadas en esta segunda versión:

- Creencias: Cada agente examina el mundo simulado, captando y memorizando información sobre otros agentes (tiendas y sus coordenadas). Esta información es parcial, hace referencia tan sólo a ciertos aspectos relevantes para los objetivos últimos desde la perspectiva de cada agente, pero resulta suficiente para constituir los *"mapa personales del mundo"* que constituye la memoria de cada agente.
- Deseos: Cada agente está dotado de un sistema de objetivos finales (su lista de la compra) que se utiliza como criterio para evaluar su situación actual y sus acciones (orientación y movimiento) posteriores.
- Oportunidades: Como resultado de sus acciones, cada agente recorre el mundo simulado, y el conjunto y la secuencia de sus posiciones sucesivas le coloca en situación de examinar diferentes entornos. Cada agente dispone, en cada posición, de diferentes y diversas oportunidades u opciones de acción entre las que escoger. Si bien al inicio todos los agentes estaban en la misma situación (memoria vacía), con el paso del tiempo cada uno de ellos genera un "mapa mental" diferente en función de sus movimientos (su "biografía"), y esto genera diversidad de oportunidades de cumplir con los propios objetivos.
- Intención: En una posición concreta, cada agente selecciona una única opción de acción (su orientación del movimiento, o `heading`) antes de avanzar, y lo hace realizando una deliberación en la que aplica sus deseos sobre sus creencias dentro del límite de su oportunidades. Este tipo de acción, propia del modelo de acción humana BDI, puede denominarse "intencional" aún cuando no se refiera a personas físicas sino a "agentes de software" muy simples.

Proporcionar a los agentes la capacidad de memorizar las localizaciones de las tiendas por

[7] Sobre el modelo BDI: Bratman, M. (1987) *Intention, Plans, and Practical Reason*, Cambridge, MA, USA: Harvard University Press, 1987.

las que han pasado posiblemente acelerará su proceso de compra. Es posible confirmar esta hipótesis ejecutando repetidamente el código ampliado: para ello es preciso introducir, en la pantalla [Ejecución] del entorno NetLogo, las modificaciones indicadas a lo largo del presente apartado, realizar diversas simulaciones y registrar la media de tiempo.

Después de 100 ejecuciones de prueba el promedio de `ticks` tras los cuales todos los agentes han completado sus compras desciende desde unos 14.000 para el modelo básico hasta sólo **6.983 (desviación estándar de 2.007)**, esto es, aproximadamente la mitad.

Los agentes-compradores son ahora menos “estúpidos” que los desmemoriados paseantes aleatorios de la primera versión del modelo. Aunque, obviamente, todavía son muy simples, en este segundo modelo ampliado tienen la habilidad de percibir el entorno y de reaccionar al mismo. Pero estos dos primeros modelos carecen de procesos de “interacción social”, esto es, los agentes-compradores operan de forma individual, y sin consideración hacia el resto de agentes compradores. El desarrollo siguiente del modelo continuará programando procedimientos diseñados para modelizar los mecanismos mediante los que los agentes-compradores se encuentran, se reconocen e intercambian entre sí información sobre las tiendas que han encontrado previamente en su camino.

> *[La metodología de Simulación Social, como variante del experimentalismo manipulacionista, busca estudiar el efecto de determinados mecanismos, o la sensibilidad de un sistema social a determinadas combinaciones de condiciones iniciales, mediante la comparación de resultados entre el modelo de referencia y el modelo de control. Como se ha visto a lo largo del tutorial, esto implica comparar resultados agregados entre versiones sistemáticamente diferentes del mismo modelo. Ninguno de los modelos de simulación tiene validez por sí mismo, sino como parte del proceso experimental junto con alguna variante.]*

Versión 3: “Hablando en el Paraíso de la Compra”

Un modelo con agentes *socialmente* más avanzados debe incorporar necesariamente la capacidad adicional de “hablar” con otros agentes con los que se cruzan en sus paseos de compra por el Centro Comercial. Para diseñar un tercer modelo más social crearás nuevos procedimientos para que los agentes que se encuentran en la misma parcela se reconozcan

e intercambien sus respectivos conocimientos acerca del mundo *(i.e., las localizaciones de las tiendas ubicadas sus respectivas memorias).* Este modelo ampliado comunicativo es excepcionalmente simple:

1. el agente A obtiene toda la información del agente B para añadirla a la suya propia, y
2. el agente B obtiene todo el conocimiento que tiene el agente A.

Lejos de cualquier consideración de realismo en sí mísmo, este modelo comunicativo tan simple pretende servir:

1. como contraste para experimentar con los anteriores modelos no-sociales, y
2. como base para experimentar posteriormente sobre factores tales como el efecto de introducir errores en la transmisión de información (ruido), o el efecto de intercambiar sólo una parte del conocimiento (sesgo comunicativo social).

3.1. Ampliación del modelo cognitivo: Inteligencia social.

Para poder implementar las características adicionales de comunicación social, es necesario añadir una nueva línea al procedimiento `actuar` para hacer que los agentes hablen (`hablar`) si coinciden con otros agentes en la misma parcela. La conversación, analíticamente considerada, consta de los siguientes procesos:

1. seleccionar a uno de los otros agentes en la parcela con el que hablar (pareja),
2. entonces copiar su memoria sobre la memoria propia, y
3. pedirle que copie la memoria propia sobre la suya.

Sobre una copia del anterior modelo, renombrada como versión 3, hay que **modificar** el procedimiento `actuar`, añadiendo una primera linea con una “llamada” condicional a un nuevo procedimiento `hablar`.

```
to actuar
  if any? other compradores-here [ hablar ]
  ifelse not empty? lista-de-compra
   [ comprar ]
   [ set color sky ]
end
```

Además hay que teclear, o pegar, el **nuevo** procedimiento `hablar` añadiéndolo al código ya existente.

```
to hablar
  let pareja 0
  set pareja one-of other compradores-here
  recordar [memoria] of pareja
  ask pareja [ recordar [memoria] of myself ]
end
```

Recuerda que cada agente dispone de un "mapa personal" del mundo simulado, formado por listas de tríadas que registran las ubicaciones de tiendas y los productos ofrecidos. Pero ahora este conocimiento puede diseminarse entre toda la población, ya que cuando dos agentes se encuentran en la misma parcela cada uno de ellos proporciona toda la información contenida en su mapa (sus creencias sobre el mundo) al otro.

Para modelizar este "intercambio de información" se aprovecha el procedimiento anteriormente existente de construcción de un mapa por incorporación de conjuntos de información `recordar`: originalmente se utilizó para "recibir" en la memoria información no redundante de observaciones del entorno, y ahora se usa para "recibir" de la misma forma información de otros agentes-compradores.

Fíjate en el curioso uso de la instrucción `of myself` que hace referencia, no a la propia memoria (la del comprador considerado "pareja") , sino a la del comprador que inicia la "conversación". En este contexto *el comprador que inicia la conversación recibe la memoria de otro, ubicado en la misma parcela, y a continuación le pide a este último que reciba la memoria de sí mismo (es decir, del que hace la petición).*

3.2. Evaluación de la eficiencia del modelo ampliado con comunicación.

Anteriormente ya se pudo comprobar cómo el conocimiento individual sobre la localización de las tiendas reduce a la mitad el número de `ticks` requeridos para completar las compras de todos los agentes. ¿Qué diferencia puede proporcionar el intercambio social del

conocimiento que acabamos de implementar?

Para realizar simulaciones ejecutando este modelo hay que pulsar el botón [Inicia] para establecer un estado inicial del mundo simulado, y a continuación pulsar el botón [Simula] para poner en funcionamiento el proceso de compra, hasta la total satisfacción de todos los agentes.

En una serie de 1000 simulaciones se ha podido comprobar cómo, con la adición de la conversación entre agentes, el tiempo promedio de finalización de compras decae de manera drástica hasta **1,738 `ticks` (con una desviación estándar de 1,100**). En el modelo anterior, sin el procedimiento `hablar`, los agentes dedicaban un promedio de unos 7.000 `ticks` hasta completar sus compras.

Posibles ampliaciones del modelo

Esta secuencia de tres versiones del modelo de simulación, con agentes progresivamente más complejos, pretende ser de utilidad para:

1. introducir los conceptos fundamentales, la estructura sintáctica y algunas de las palabras clave más usuales del lenguaje de simulación NetLogo, y para
2. mostrar cómo se pueden modelizar agentes más “inteligentes” sobre la base de sus capacidades previas, procediendo a partir de modelos muy simples que funcionan, añadiendo complejidad teórica, o bien calibración empírica (mediante datos observacionales o experimentales).

La ampliación de complejidad teórica, mediante sucesivos modelos más desarrollados, pasa por implementar bajo la forma de código informático algunas de las características propias de cualquier sistema social.

Por ejemplo, en cuanto a los agentes, el desarrollo posterior de modelos puede ir integrando los cuatro principios básicos de autonomía, habilidad social, reactividad y

proactividad [8] dentro de un modelo de acción social como el mencionado BDI+O. La información sobre qué principios o mecanismos se asumen como fundamentales para producir la acción humana debe provenir:

- de la evidencia empírica más actual en el campo de las ciencias cognitivas, o
- de análisis cualitativo detallado de las narrativas analíticas producidas por individuos y grupos humanos, o
- de resultados de experimentación social en contextos de laboratorio o campo.

En cuanto a los aspectos relacionados con la calibración empírica, fundamentalmente se trata de introducir heterogeneidad entre los agentes en cuanto a los conjuntos de deseos, creencias y dotaciones de forma que se reproduzca aproximadamente la estructura existente en la sociedad real de referencia. La información sobre esta estructura debe provenir de los estudios empíricos adecuados.

Para finalizar este tutorial de auto-construcción de modelos de simulación social con Netlogo, y sin ánimo de exhaustividad, se apuntan una serie de ideas que te puedan orientar para la introducción de nuevas modificaciones sobre los "modelos de compra" desarrollados hasta el momento.

- Restringir el intercambio social de conocimiento sólo a aquellos agentes que se consideren "amigos" (de acuerdo con alguna definición operativa de amistad), o bien a aquellos a los que públicamente se atribuya "confianza". Esto implica enriquecer los "mapas personales" actuales (con información sobre tiendas) con información sobre otros agentes (reputación).
- Construir un modelo más realista de comunicación social o intercambio de información, en el que se supere el modelo de intercambio simbólico directo y libre de errores entre cada agente.

[8] Los agentes pueden controlar sus propias acciones y estados internos (tienen autonomía). Los agentes son capaces de interactuar con otros agentes (tienen capacidades sociales). Los agentes pueden percibir el entorno y pueden responder al mismo (tienen reactividad). Finalmente, son capaces de llevar a cabo un comportamiento orientado según fines (son proactivos).

- Añadir nuevas líneas de código para comprobar el efecto de las creencias erróneas, sobre el conjunto de tiendas, o sobre los agentes.
- Las tiendas podrían tener una provisión limitada de productos para vender, y por tanto ocasionalmente se podría agotar la disponibilidad de algún producto. Aún así los agentes no serían conscientes de ello antes de visitar la tienda o de recibir información social acerca de ello (rumor), y por tanto continuarían acudiendo a la tienda para ver si pueden comprar.
- Los agentes tienen tan sólo un único objetivo (completar su compra) y carecen de capacidad de planificar, y ambos aspectos del diseño de los agentes podrían mejorarse.

¿Puedes pensar en otras mejoras del modelo?

¿Puedes intentar seguir modificando el código para incorporar alguna de estas ideas?

Más información sobre ABM y Netlogo

Para una introducción completa al uso del entorno de simulación social basada en agentes "Netlogo" se recomienda consultar los siguientes recursos:

- Selección de recursos del LSDS *(Laboratorio de Simulación de Dinámicas Socio-Históricas)* <http://sct.uab.cat/lsds/>.
- Miguel, F. J. & Hassan, S. (2012) La investigación mediante simulación social multi-agente, en M. Arroyo & I. Sábada, (Eds.) *Metodología de la investigación social, innovaciones y aplicaciones.* Madrid, Síntesis.
- Gilbert, N. & Troitsch K. (2005) "Modelos Multi agente", en *Simulación para las Ciencias Sociales (2ª edición)*, Madrid: McGraw-Hill, 2006, Capítulo 8.
- Wilensky, U. (1999). NetLogo. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University,

Evanston, IL.

- Wilensky, U. & Rand, W. (2015). Introduction to Agent-Based Modeling: Modeling Natural, Social and Engineered Complex Systems with NetLogo. Cambridge, MA. MIT Press.