Ingeniare. Revista Chilena de Ingeniería
ISSN: 0718-3291
facing@uta.cl
Universidad de Tarapacá
Chile

Ramírez, Pablo; Leger, Paul; Vallone, Andrés
Un modelo flexible para la simulación de distribución de ciudades
Ingeniare. Revista Chilena de Ingeniería, vol. 22, núm. 3, septiembre, 2014, pp. 363-373
Universidad de Tarapacá
Arica, Chile

Un modelo flexible para la simulación de distribución de ciudades

A flexible model for city distribution simulations

Pablo Ramírez[1] Paul Leger[1] Andrés Vallone[1]

Recibido 25 de febrero de 2014, aceptado 24 de abril de 2014
Received: February 25, 2014 Accepted: April 24, 2014

RESUMEN

El cumplimiento de la ley de Zipf es ampliamente reconocido en la literatura de economía urbana, al punto de utilizar su coeficiente como medida de concentración espacial de la población. Mientras el exceso de concentración presenta problemas al bienestar social debido a la existencia de disparidades regionales y congestión de algunas ciudades, la dispersión no aprovecha, por ejemplo, los beneficios en la disminución de los costos de producción producidos por las economías de escalas.
Diversos trabajos se han realizado a fin de dar una explicación al cumplimiento a la ley de Zipf, donde los modelos *top-down*, que fuerzan el cumplimiento de la ley de Zipf, dominan la literaria. Pocos trabajos han intentado explicar el cumplimiento de Zipf usando autómatas celulares. Estas propuestas, llamadas *bottom-up*, generan de manera emergente la distribución y los tamaños de las ciudades. Sin embargo, estos modelos son poco flexibles y no pueden ser adaptados a distintos sectores geográficos.
Este trabajo propone una implementación inicial de un modelo extensible basado en autómatas celulares, llamado CityCA, para intentar explicar el coeficiente de Zipf. Se evaluó y adaptó CityCA a un escenario real, Chile. Los resultados de la simulación son los esperados. Muestra que el coeficiente de Zipf se acerca más a la sobreconcentración que a la dispersión, igual que en la realidad. Con este modelo, los economistas urbanos no requieren conocimientos de programación avanzada, pues este modelo especializado ya contiene el conocimiento. Además, surgen nuevos requerimientos que se incorporarán en un trabajo futuro de CityCA.

Palabras clave: Ley de Zipf, distribución de ciudades, autómatas celulares, modelo flexible.

ABSTRACT

The fulfillment of the Zipf's law is widely recognized in urban economics literature. Indeed, its coefficient is used as a measure of spatial concentration of a population. Whereas the concentration excess has problems associated with social welfare due to the existence of regional disparities and the congestion in some cities, the dispersion does not take advantage of, for example, the benefits of reducing costs of production generated by economies of scale.
Several studies have been performed in order to give an explanation to the fulfillment of Zipf's law, where the top-down models, that force the fulfillment of Zipf's law, have dominated the literature. Some proposals have attempted to explain the fulfillment of Zipf using cellular automata. These bottom-up proposals generate the distribution and sizes of cities in an emergent way. However, these models are not flexible and extensible, therefore, these proposals cannot be adapted to different geographical areas. This paper proposes an initial implementation of a flexible and extensible model based on cellular automata, called CityCA, which tries to explain the Zipf coefficient. CityCA was evaluated and adapted to a real scenario, Chile. The simulation results were as expected, since they show that the ratio is closer to Zipf over-concentration than the dispersion, as in reality. With this model and implementation, urban

[1] Universidad Católica del Norte. Coquimbo, Chile. E-mail: prl012@alumnos.ucn.cl; pleger@ucn.cl; avallone@ucn.cl

economists do not require advanced programming knowledge, since CityCA already integrates specific knowledge.

Keywords: Zipf's law, cities distribution, cellular automata, flexible model.

INTRODUCCIÓN

La observación empírica muestra que las ciudades difieren en cuanto al tamaño y a las actividades que realizan. La distribución de las ciudades se forma con la aglomeración y desaglomeración de las personas, quienes eligen un lugar para establecerse según los beneficios que le otorga aquel lugar.

A medida que un país se desarrolla y disminuyen los costos de transporte, la población tiende a concentrarse espacialmente a fin de aprovechar los beneficios inherentes a la cercanía (ej. la mejora en los encadenamientos productivos y el incremento en los tamaños de los mercados). Sin embargo, la excesiva aglomeración genera efectos negativos (ej. congestión, contaminación). Por otro lado, la dispersión no permite obtener los beneficios de la concentración espacial.

Una de las medidas de concentración espacial de la actividad es el coeficiente del Zipf, el que es derivado de la ley de Zipf [19] o regla rango-tamaño. La ley de Zipf se cumple si al ordenar todas las ciudades de mayor a menor según su tamaño, la ciudad de mayor tamaño tendrá aproximadamente el doble de población que la segunda, el triple que la tercera y así sucesivamente. Según esta regla, el coeficiente de Zipf será igual a uno si se cumple plenamente la ley, menor a uno si existe mayor dispersión y mayor a uno si hay más concentración en torno a la ciudad de mayor tamaño.

Al cumplirse la ley de Zipf, las ciudades se distribuyen de tal forma que se logra sacar provecho de los beneficios de la concentración espacial, si no se cumple, se pueden desaprovechar estos beneficios (dispersión) o generar problemas de exceso de concentración.

Actualmente no se dispone de una teoría que explique de manera concluyente la ley de Zipf. Las principales líneas de investigación tienden a explicar la ley con modelos matemáticos de tipo *top-down*, las cuales generalizan el comportamiento de distribución de los tamaños de las ciudades partiendo con el condicionante cumplimiento de la ley de Zipf. Es decir, estos modelos no explican cómo se genera la ley Zipf, sino que ellos asumen que Zipf se cumple y buscan qué comportamientos de la población satisfacen esta ley.

Algunos trabajos [13, 24] usan modelos *bottom-up* para tratar de explicar la ley de Zipf mediante una conducta emergente y no impuesta. Estos modelos usan autómatas celulares (ACs) [14]. Los agentes de un AC representan empresas/personas que se relacionan entre ellas con el fin de reducir sus costos y maximizar sus beneficios. Sin embargo, la implementación de estos modelos es poco flexible y no permite adaptar el modelo a diferentes escenarios con el fin de capturar territorios geográficos reales.

Nosotros proponemos una implementación inicial en Python acerca de un modelo flexible (y abierto) que permite explicar el tamaño de las ciudades y su distribución espacial. En este modelo, basado en autómatas celulares, el coeficiente de Zipf resulta de una propiedad emergente y no de una propiedad impuesta (es decir, *top-down*). La implementación de este modelo, llamada CityCA, absorbe los conceptos asociados al área de la economía urbana y permite a los usuarios adaptar flexiblemente su implementación para ajustarla a diferentes escenarios, por lo que no necesitan saber mucho sobre programación.

Estructura del artículo. La sección *Distribución de ciudades* introduce brevemente los problemas asociados a los tamaños de ciudades y cómo Zipf es usado para medir el nivel de concentración espacial de estas en un país. La sección *Autómatas celulares* explica los principales conceptos de estos. En la sección *City-CA: un AC flexible para determinar la distribución espacial de ciudades*, se presenta este modelo. La sección *Chile: caso de estudio de City-CA* presenta el uso de City-CA para tal caso de estudio. A continuación la sección *Trabajo relacionado* presenta estudios anteriores sobre ACs

aplicados a la simulación de modelos urbanos. Por último la sección *Conclusiones y trabajo futuro* finalizan el trabajo.

DISTRIBUCIÓN DE CIUDADES

La elección de localización de las personas se basa en que el lugar escogido le otorga más beneficios que cualquier otro lugar. Este proceso tiende a generar naturalmente concentraciones de población en lugares geográficos determinados. Al concentrarse la población se genera eficiencia productiva debido al surgimiento de economías de escala y también se incrementan los mercados, ya que al aumentar la población, habrá más empresas debido a una mayor demanda de productos.

La concentración espacial genera beneficios, sin embargo, un exceso de esta provoca un conjunto de efectos negativos. Los excesos de concentración de la población pueden presentar problemas tanto a nivel de eficiencia en el crecimiento de las economías [3, 8-9, 17] como problemas en el bienestar social debido a la existencia de disparidades regionales [1]. Una sobreconcentración de la población produce un aumento en el precio del suelo debido a la escasez de terrenos y lugares para vivir [7], además causa congestión que se traduce en una pérdida de bienestar [2].

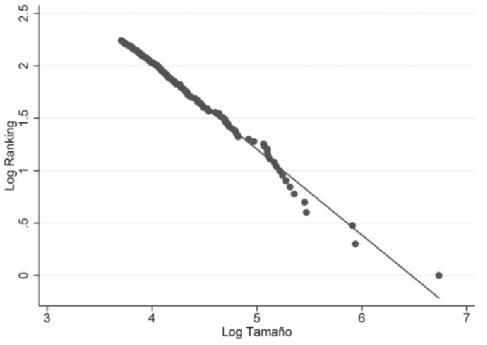


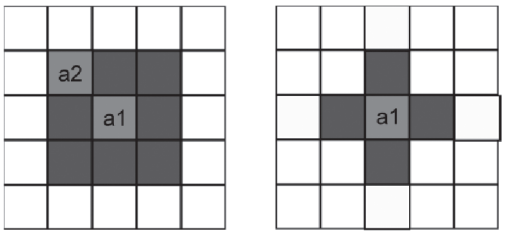
Figura 1. Gráfico del logaritmo del ranking (por población) v/s el logaritmo del tamaño de la población. Chile, Censo 2002.

En este proceso de agrupación y dispersión se construye el sistema urbano de un territorio, y es aquí donde se puede observar una regularidad típica de distribución urbana que es la ley de Zipf [19]. Este hecho estilizado postula que la relación entre el ranking y el tamaño de las ciudades sigue una distribución de probabilidad de Pareto, y su versión linealizada puede representarse en la ecuación .

$$\log y = \log A - \alpha \log x \qquad (1)$$

Donde x es el número de habitantes de una ciudad; y es el ranking de la ciudad según el número de habitantes ordenado de mayor a menor; A y α son constantes. El coeficiente α, conocido como el coeficiente de Zipf, mide el grado de concentración de la población. Un $\alpha > 1$ indica que hay una dispersión tal que no permite acceder a los beneficios de la aglomeración. En cambio un $\alpha < 1$ indica una alta concentración de la población. Para visualizar la ley de Zipf se puede tomar el caso de Chile y ordenar las ciudades por número de habitantes[2]. Al graficar los resultados se identificará claramente una tendencia lineal como se muestra en la Figura 1.

Al estimar la regresión se obtiene la ecuación .



(a) (b)

Figura 2. a) Dos agentes, usando la vecindad de Moore, en un AC. b) Un agente en un AC, el cual usa la vecindad de Neumann.

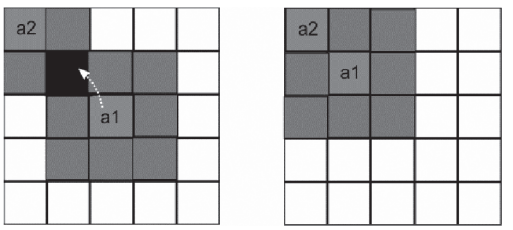
$$\log y = 5{,}167753 - 0{,}7969709 * \log x \qquad (2)$$

Donde el coeficiente $\alpha \cong 0{,}8$, mostrando una concentración de población excesiva. Sin embargo, Gabbaix e Ioannides [6] sostienen que empíricamente se cumple la ley de Zipf para un coeficiente α entre 0,8 y 1,2, situando a Chile en el límite inferior del cumplimiento. La Ley de Zipf se ha transformado en un condicionante de los modelos de crecimiento urbano, ya que en última instancia independientemente del factor asociado al crecimiento de las ciudades, la distribución final del tamaño de las ciudades debe seguir la ley de rango-tamaño [5]. En

[2] Se consideraron ciudades mayores a 5.000 habitantes según lo considerado por el INE como ciudad con los datos del Censo 2002.

los modelos de crecimiento urbano que contemplan el surgimiento de la ley de Zipf, predominan aquellos basados en procesos estocásticos [18]. Sin embargo, en todos estos modelos el surgimiento de la ley es impuesto por la construcción del modelo desde una perspectiva *top-down*, es decir, van desde lo general a lo específico y ello requiere de una gran cantidad de parámetros para representar el comportamiento dinámico de los individuos. Con el tiempo estos modelos han intentado reflejar la realidad de manera más precisa y por lo tanto se han hecho más complejos aún.

Una forma alternativa de estimar un modelo tan complejo es simular el comportamiento individual de los actores participantes en él, es decir, desde una perspectiva *bottom-up*, donde se tiene más información del comportamiento individual que del general. Algunos trabajos [10, 14] usan instrucciones simples y cortas para cada individuo (o agente), en vez de tener que formular un modelo complejo desde el inicio. Por esto, un modelo basado en comportamiento individual se hace más conveniente al momento de simular comportamientos dinámicos complejos.



a) Tic de reloj t b) Tic de reloj $t+1$

Figura 3. a) Un agente a1 en el tic de reloj t decide moverse cerca de a2. b) En el tic de reloj $t+1$, el agente a1 se encuentra más cerca del agente a2.

AUTÓMATAS CELULARES

John von Neumann [14] propuso Autómatas Celulares (ACs) como un modelo matemático discreto con un poder de computabilidad equivalente a una máquina de Turing. Los ACs han sido usados en varios campos de las ciencias (ej. vida artificial [10], sistemas complejos [12, 20], criptografía [16]). El Juego de la Vida[3] es un AC ampliamente conocido.

[3] http://es.wikipedia.org/wiki/Juego de la vida. Consultado el 20 de mayo de 2013.

Tal como muestran las Figuras 2a y 2b, un AC se compone de una grilla, la que contiene un conjunto de agentes que interactúan con sus agentes vecinos con acciones simples. A continuación describimos cada componente de un AC.

Grilla. Este componente define el sector donde los agentes se albergan. Este sector está dividido en locaciones (ej. 25 locaciones en la Figura 2a), las que contiene uno o más agentes. Una grilla puede ser unidimensional, bidimensional, incluso hasta tridimensional [10]. Además, una grilla puede tener características especiales, por ejemplo, puede ser finita, circular o infinita.

Agente. La Figura 2a muestra dos agentes (a1 y a2) en una grilla, donde cada uno es un cuadrado azul. En AC, un agente es una entidad que puede contener un estado y objetivo. El agente busca alcanzar su objetivo por las acciones que él puede tomar.

Vecindad. Cada agente tiene una vecindad. Esta vecindad determina cuáles son los elementos que afectan al estado y objetivo de un agente. Hay varios tipos de vecindades como Moore (Figura 2a), la que incluye las ocho locaciones más cercanas a un agente más la locación del agente. Otro tipo de vecindad es la definida como Neumman, involucrando las cuatro celdas verticales y horizontales más cercanas además de la locación del agente (Figura 2b). El número de locaciones que considera una vecindad depende de su radio. Por ejemplo, estas figuras muestran un radio de largo 1.

Acciones. En cada instante de tiempo discreto (conocido como "tic de reloj") los agentes pueden ejecutar una acción dentro de un conjunto de potenciales acciones. La elección de una acción busca alcanzar su objetivo y obedecer las reglas del entorno. Por ejemplo, si el objetivo de un agente es estar cerca de otros agentes, un agente puede tomar la acción de moverse desde una locación x a una y si esta última se encuentra más cerca de otros agentes. En AC se usa una función de *fitness* para determinar qué acción en un tic de reloj ayuda más a un agente a alcanzar su objetivo. Por ejemplo, la función de *fitness* que determina cuál locación ayuda más a un agente a estar cerca de otros se expresa en la ecuación (3).

$$nextLoc_a = max_{[loc \in neighbors(radio_a)]}\, count(loc) \quad (3)$$

Donde se muestra que para cada locación de su vecindad la función de *fitness* del agente a cuenta el número de agentes cercanos. Luego compara estos valores y selecciona la locación que contiene el mayor número de agentes cercanos.

La Figura 3 ejemplifica el uso de la función de *fitness* presentada en la ecuación anterior. La Figura 3a muestra que un agente a_1 decide tomar la acción de moverse a la locación negra. La razón de esta decisión es debido a que su función de *fitness* entrega que la locación negra (debido a su radio de locaciones disponibles) ayuda a alcanzar su objetivo en el tic de reloj *t*. En la Figura 3b se puede apreciar que el agente a_1 se encuentra más cerca del agente a_2 en el tic de reloj $t+1$.

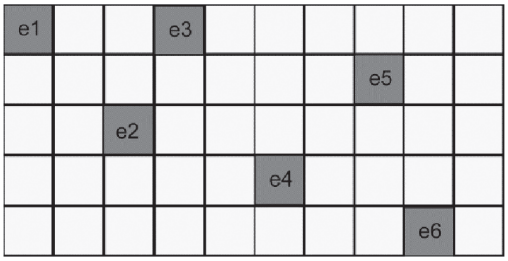


Figura 4. Una grilla de CityCA de cincuenta locaciones con seis empresas.

CITY-CA: UN AC FLEXIBLE PARA DETERMINAR LA DISTRIBUCIÓN ESPACIAL DE CIUDADES

Dentro de las aplicaciones que tiene un modelo de AC se encuentra la simulación de interacciones dinámicas entre empresas (o personas). Con AC es posible estimar un punto de convergencia tras varias iteracciones, donde cada empresa se encontrará en un lugar del cual no quiera moverse. Así se puede determinar cuál será la distribución espacial y los tamaños de las ciudades de un sector geográfico en un país. Usando los tamaños de las ciudades, es posible calcular el coeficiente de Zipf (sección *Distribución de ciudades*).

CityCA es una implementación extensible de un modelo basado en AC CityCA es actualmente implementado en Python, un lenguaje orientado a objetos, dinámicamente tipeado y con soporte a funciones de primera clase, parámetros opcionales y metaprogramación (como en Lisp). CityCA es usado para determinar el tamaño y distribución de las ciudades en un sector geográfico (ej. un país). Donde el tamaño de una ciudad es medido por el número de empresas en una celda. CityCA engloba los conceptos de economías urbanas en ciudades (ej. Zipf), dejando configurables (o abiertos al programador) los aspectos necesarios para adaptarse a características específicas de un sector geográfico. Usando la misma estructura de la sección anterior, nosotros explicamos cada componente de CityCA.

Grilla

La idea de la grilla de este modelo es que represente un sector geográfico real. Por tanto, cada casilla representa una cantidad específica de kilómetros cuadrados. La Figura 4 muestra una grilla cuadrada de 50 locaciones. Por ejemplo, si cada locación representa un espacio de 0,1 [km] de largo por 1 [km] de alto, el sector geográfico total representado será de 5 [km^2], ya que (0,1 [km]*10)*(1[km]*5)=5[km^2]. Es importante mencionar que el tamaño de una locación no es un elemento a considerar mientras no se analicen los procesos de formación de las ciudades. Solo ha de considerarse que el tamaño de una locación en relación con el tamaño total de la grilla no sea excesivo al punto de perder representatividad del espacio geográfico a estudiar.

La grilla de CityCA es bidimensional y su tamaño puede ser configurado al momento de crearse un objeto del tipo CityCA. El tamaño de una grilla se configura indicando la cantidad de filas y de columnas. Por ejemplo, si se quiere una grilla de 30 filas por 25 columnas se define:

```
chile = CityCA(30,25)
```

Aunque CityCA está orientado a simular sectores geográficos reales, estamos conscientes que ciertos experimentos buscan explorar comportamientos de formación de ciudades en escenarios imaginarios [13]. Por esta razón, la configuración de una grilla por omisión (finita y no circular) puede ser modificada:

```
chile = CityCA(30,25,CityCA.CIRCULAR_GRID)
```

CityCA ofrece un conjunto de estrategias para administrar una grilla (ej. circular, infinita).

Agentes

Dentro de la economía urbana y en este artículo se plantea como supuesto que la relación empresa-persona es 1:1, es decir, una empresa equivale a una persona para generar modelos consistentes con la distribución de ciudades [11, 4].

En este modelo, cada agente representa una empresa. En un sector geográfico real, una o más empresas pueden estar en la misma locación, donde la concentración de una locación es medida por el número de empresas que contiene. En CityCA las empresas no contienen estados, pero sí tienen objetivos. El objetivo de cada empresa es disminuir sus costos y mejorar sus ingresos. Por ejemplo, el siguiente código crea e inserta 15.000 empresas en un objeto CityCA.

```
chile.create(15000)
```

Algunas características de las empresas (ej. función de *fitness*) pueden ser personalizadas al momento de crearlas. Estas características las explicamos en las próximas dos secciones.

Vecindad

En la práctica, una empresa decide moverse con el objetivo de mejorar sus utilidades. En general, la mayoría de las empresas no pueden moverse rápidamente en una gran distancia por razones de infraestructura. Este límite de movimiento es el radio de la empresa, el que indica donde se puede mover en la siguiente iteración. El radio mínimo es 0, ya que se considera que una empresa podría no moverse debido a los costos implicados. El radio máximo posible es equivalente a la longitud máxima de la grilla[4]. En CityCA cada empresa puede tener un radio distinto. Los radios pueden ser distribuidos en forma aleatoria entre un radio mínimo y un radio máximo. Por ejemplo, el siguiente código crea 15.000 agentes con un radio al azar usando la distribución uniforme entre 1 y 15:

```
uniformRadium = CityCA.Radium.uniform(1,15)
cityCA.create(15000,uniformRadium)
```

```
#Donde uniform de 'CityCA.Radium' es
def uniform(rmin,rmax):
  def radium():
    return random.uniform(rmin,rmax)
  return radium
```

La variable `uniformRadium` es una función que retorna un valor entero al azar entre 1 y 15 cada vez que es evaluada[5]. Cuando las empresas son creadas, cada una tiene asociada esta función de radio. Notar que `create` es la misma función del código mostrado en la sección anterior, pero ahora con un argumento adicional, el cual es opcional. Por omisión, todas las empresas se pueden mover a cualquier locación (es decir, tienen un radio del máximo de la grilla).

Para los programadores, una serie de funciones de librería para establecer radios están disponibles (ej. `normal`, la cual asigna radios usando la distribución normal). Adicionalmente, un programador puede definir su propia función para asignar radios. Por ejemplo, el siguiente código crea 15.000 agentes con un radio fijo de 15:

```
def createFixedRadium(radio):
  def radium():
    return radio
  return radium

fixedRadium = createFixedRadium(15)
chile.create(15000,fixedRadium)
```

Acciones

Las empresas deben tomar acciones que mejoren sus utilidades. En CityCA las empresas usan la función de *fitness* para decidir qué acción tomar, la que básicamente es moverse a un lugar. Un programador de CityCA puede personalizar las funciones de *fitness* de acuerdo con los requerimientos de su experimento. Esta función es personalizada al momento de crear las empresas:

```
chile.create(15000,
  fitness=CityCA.fitness.betterToBeTogether)

#Donde 'betterToBeTogether' es:
def betterToBeTogether(self,location):
  return location.countCompanies()
```

[4] En el caso de que sea una grilla circular, el radio máximo sería: $r_{max} = Z/2$, donde Z equivale a la longitud máxima de la grilla.

[5] Notar que la evaluación de `uniform` retorna otra función (funciones de primera clase en Python), la que es la pasada por parámetro a `createCompanies`.

CityCA también ofrece a los programadores una librería de funciones de *fitness*. Por ejemplo, en el código de arriba podemos ver que al momento de crear las compañías, la función `betterToBeTogether` es asignada como función de *fitness*[6]. Con esta función, las empresas buscarán estar en locaciones con más empresas. La función `betterToBeTogether` sigue el principio que cuando hay más empresas juntas, los costos de transportes/producción disminuyen. Como en la mayoría de las opciones de CityCA, los programadores pueden definir sus propias funciones de *fitness* para las empresas. Por ejemplo, si deseamos que las empresas consideren los costos asociados a la concentración y congestión de empresas de una ciudad, un programador puede definir la siguiente función:

```
def itIsNotSoGoodToBeTogether(self,location):
  companiesNumber=location.countCompanies()
  factor = 0.25
  deglomeration=(factor*companiesNumber)**2
  return companiesNumber - deglomeration

#Usando una funcion personalizada de fitness
chile.create(15000,
  fitness=itIsNotSoGoodToBeTogether)
```

La variable `deglomeration` representa el costo asociado a la concentración de empresas, y su factor de influencia (0,25) ha sido usado en otros trabajos (ej. [13]) para indicar cómo afecta la economía de desaglomeración.

Ejecutando una simulación en CityCA

Luego que se tiene una grilla con un conjunto de empresas distribuidas, donde cada una posee un determinado radio y una función de *fitness*, CityCA permite a los programadores ejecutar una simulación. En esta simulación las empresas comienzan a interactuar en cada iteración moviéndose al lugar que mejor evalúe según su función de *fitness*.

Siguiendo los principios de un CA, las empresas se pueden mover una vez por cada tic de reloj. Para definir el número de iteraciones en CityCA, basta con especificar la cantidad deseada en el método `run` de un objeto `CityCA`. Por ejemplo, si se requiere de 20 iteraciones:

```
chile.run(20)
```

Usando la característica de *dynamic typing* de Python, un programador puede pasar a una función (en vez de un número entero) que señale cuándo se debe detener la simulación. Por ejemplo, si deseamos detener la simulación solamente cuando ninguna empresa se mueve en un tic de reloj:

```
def anyAgentMoved(previousComps,currentComps):
  boolean moved = False
  for pAgent in previousComps:
   for cAgent in currentComps:
    if pAgent == cAgent: #same companies?
     if not sameLocation(pAgent,cAgent):
       moved = True
  return moved

#Simulando hasta que una condicion se cumpla
chile.run(anyAgentMoved)
```

La función `anyAgentMoved` recibe la lista de las empresas del tic de reloj t–1 y la lista de las empresas del tic de reloj t. Luego, empresa por empresa, se verifica si alguna ha cambiado de locación. Si ninguna ha cambiado de locación, la simulación se detiene, entregando un reporte gráfico acerca de las distribuciones de las ciudades y el coeficiente de Zipf asociado.

CHILE: CASO DE ESTUDIO DE CITYCA

En este caso de estudio se valida la flexibilidad de CityCA para adaptarse al territorio chileno y a las configuraciones que determinan las normas de comportamiento de los agentes. Estas configuraciones determinan el tamaño y la distribución de las ciudades a la que converge luego de n iteraciones.

Usando los conceptos de la sección anterior, nosotros mostramos el código necesario para llevar a cabo la simulación realizada en el caso de estudio.

```
1 uniformRadium = CityCA.Radium.uniform(0, 537)
2 fitness = CityCA.fitness.betterToBeTogether
3 chile = CityCA(22,537)
4 chile.create(20000, uniformRadium,fitness)
5 chile.run(50)
```

A continuación se explica la configuración de CityCA para el territorio chileno usando el código de arriba:

[6] En Python es posible mencionar qué argumento opcional se desea pasar como parámetro a una función.

Vecindad. La distribución del radio se define en la línea de código 1. El radio se fija con un mínimo de 0 y un máximo equivalente a la longitud máxima (537), ya que es una malla finita, mientras que su distribución es aleatoria y uniforme.

Acciones. La función de *fitness* se define en la línea de código 2. La función asignada es `betterToBeTogether`, con ella las empresas buscarán estar en locaciones que tengan la mayor cantidad de empresas.

Grilla. El tamaño de la grilla se define en la línea de código 3. Para que los parámetros dimensionales se ajusten a la realidad geográfica de un país, es necesario saber el área de su superficie y la distribución del largo y ancho de esta. En este caso, Chile cuenta con 756.102,4 km^2 de superficie, mide entre 4.270 km y 4.329 km de largo y tiene un ancho promedio de 177 km distribuidos entre 90 km y 435 km[7].

Según estos datos, la configuración que difiere menos de la superficie real es una malla de la Figura 5. Resultado de la simulación. 22*537 locaciones de 8 km^2 cada una, lo que da un total de 756.096 km^2, que representan el 99,9992% del tamaño total real.

El número de locaciones en una grilla solo influirá en el grado de especificación de los resultados, ya que mientras más subdividida sea la grilla, la información resultante será a un nivel más local. Sin embargo, este número se vuelve irrelevante ya que no se puede considerar una locación como un espacio geográfico equivalente a una ciudad, porque esta última puede ocupar varias casillas a la vez. Por último, los límites de la grilla se consideran finitos para que corresponda a un límite fronterizo real.

Agentes. El número de empresas se define en la línea de código 4, donde además se especifica la distribución y función de *fitness* definida en las líneas anteriores. El número de agentes corresponde al número de empresas que tiene el país. En Chile existen aproximadamente 800.000 empresas: 8.000 grandes, 12.000 medianas, 120.000 pequeñas y 660.000 microempresas. Se considera la suma de las grandes y medianas empresas (20.000) que son las que poseen una mayor cuota de mercado en la economía y que atraen a más personas.

[7] https://es.wikipedia.org/wiki/Chile

Iteraciones. Por último, en la línea de código 5 se define cuándo se detiene la simulación, la que puede ser un número o una función. En este escenario se usó 50 como máximo número de iteraciones.

Con solo estas cinco líneas de código es posible configurar el modelo para obtener los resultados de una simulación de las distribuciones de ciudades adaptadas a un territorio geográfico real como Chile. Para cualquier adaptación solo se necesita cambiar algunas de estas líneas. Por ejemplo, si deseamos cambiar la función de *fitness* a `itIsNotSoGoodToBeTogether` solamente debemos modificar la línea 2:

```
2 fitness = itIsNotSoGoodToBeTogether
```

Resultados

Los resultados de la simulación se muestran en la Figura 5. La estimación de la regresión que resultó se muestra en la ecuación

$$\log y = 1{,}957183 - 0{,}5504229 * \log x \qquad (4)$$

Las diferencias de intercepto son irrelevantes en términos de distribución espacial, ya que solo pueden indicar el tamaño de la población más grande estimada. El coeficiente de Zipf es $\alpha \cong 0{,}55$ e indica concentración espacial. Al comparar esta interpretación con la de la sección *Distribución de ciudades*, observamos que a pesar de que en ambos casos existe sobreconcentración, el coeficiente calculado anteriormente es mayor ($\alpha \cong 0{,}8$). La diferencia en la estimación se debe a los parámetros escogidos para la simulación. La función de *fitness* utilizada es `betterToBeTogether`, la que permite a

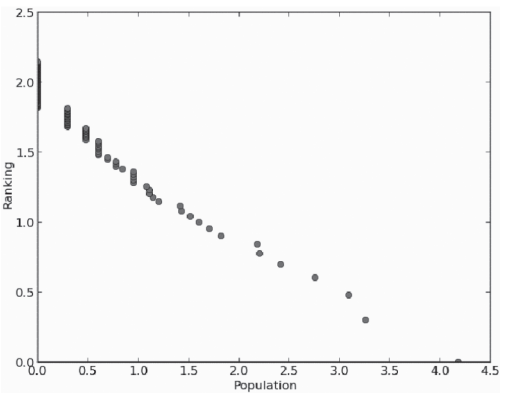


Figura 5. Resultado de la simulación.

las empresas concentrarse en las locaciones con más empresas sin que exista un costo asociado a esta congestión. Naturalmente, esta configuración genera la sobreconcentración y explica el valor del coeficiente de Zipf.

TRABAJO RELACIONADO

Desde que Von Neumann publicó sobre AC [14], su uso se expandió a simulaciones en diversas áreas. Dentro del área urbana, un artículo pionero sobre AC aplicado a la simulación de espacios geográficos fue de Tobler [21] en 1979, tema que se expandió en los 80 y 90. Al propagarse su uso se programaron diversos *software* relacionados a esta área.

CityCA es 1) una implementación flexible y adaptable para la 2) distribución del tamaño de las ciudades, pero además podría ser usado para ver el 3) crecimiento poblacional de una sola ciudad. Por esto, clasificamos los distintos trabajos acerca de programas computacionales de simulación relacionados con el área urbana en tres tipos:

***Softwares* generalizados adaptables**. Permiten configurar diversas características para su uso en distintas áreas. Por ejemplo *NetLogo* [22, 23, 15], es un entorno de modelado programable para simular fenómenos naturales y sociales. La principal diferencia con CityCA es que NetLogo es un lenguaje de dominio específico para sistemas de agentes en general. En NetLogo, para implementar un AC, es necesario codificar la lógica de implementación del AC. En CityCA, la lógica de un AC está inmersa en nuestra propuesta. Además, CityCA incluye la lógica propia del área de economía urbana.

Distribución de población en ciudades. *Softwares* basados en modelos para simular la distribución de la población dentro de un territorio interpretable como un país. Estos se enfocan en la concentración espacial de las actividades económicas, que se interpreta por medio del coeficiente de Zipf. Mansury y Gulyas [13] desarrollan un modelo basado en AC para simular un sistema de ciudades que exhibe las propiedades estadísticas de la ley de Zipf. Las diferencias con CityCA es que simulan sobre espacios geográficos no interpretables como reales y su modelo está construido en un *software* particular sin libre acceso. Otra propuesta es la realizada por Andersson, Frenken y Hellervik [24], que postula el uso de redes complejas para el modelado geográfico y demuestran cómo una aplicación de este tipo puede ser combinado con un modelo de AC para producir resultados consistentes con las regularidades a gran escala, como leyes de potencia y fractalidad. Su modelo, igual que el de Mansury y Gulyas [13], está elaborado en un *software* particular.

Crecimiento de población en metrópolis. Están basados en modelos para simular crecimiento de población dentro de un territorio interpretable como una metrópolis. *iCity* [25] incorpora una estructura espacial irregular, crecimiento urbano asincrónico, y una variada configuración espacio-temporal para ayudar en la toma de decisiones para la planificación urbana espacial. Mientras que su objetivo es simular sobre un territorio semejante a una ciudad, CityCA es configurable al punto de adaptar el territorio, y la función de *fitness* necesaria, tanto a una ciudad como a un país. Además CityCA permite agregar nuevas reglas que impliquen inclusión de códigos mediante funciones de primera clase, y no solo de parámetros, de manera sencilla.

CONCLUSIONES

La distribución de los tamaños de las ciudades en un país es fundamental para evitar problemas de concentración y dispersión. Comprender las razones asociadas a esta distribución es todavía un problema no completamente resuelto [5]. En este trabajo hemos propuesto CityCa, un modelo flexible y abierto basado en autómatas celulares para intentar explicar esta distribución. A diferencia de la mayoría de los modelos propuestos [5, 18], la distribución de los tamaños de las ciudades en CityCA es una propiedad emergente y no impuesta. Además, CityCA puede ser fácilmente personalizado para adaptarse a nuevos espacios geográficos (por ejemplo, países). Además se utilizó CityCA para determinar la distribución de los tamaños y espacial de las ciudades de Chile.

Este trabajo presenta una implementación inicial de CityCA, la que puede dejar fuera requerimientos asociados a una mayor complejidad de estudios a base de territorios. Por ejemplo, los escenarios de CityCA no consideran la existencia de recursos naturales existentes en cada locación (ej. cobre en Chile), que afectan de forma notable las decisiones de localización de las empresas. Además, CityCA

aún no considera el estado interno de cada empresa (ej. estabilidad), lo que también podría afectar la decisión de acción de una empresa. Otra cualidad que no se incorporó en la versión actual de CityCA es la explicación del proceso evolutivo de la generación de ciudades, ya que hasta ahora solo se considera la distribución final. En un futuro trabajo planeamos incorporar en CityCA estas opciones, de manera de ser configurables por el investigador.

Disponibilidad. Esta versión inicial de CityCA está disponible para su descarga en https://github.com/pleger/CityCA, un repositorio Git público para control de versiones de documentos/código fuentes.

REFERENCIAS

[1] H. Amstrong and J. Taylor. "Regional Economics and Policy". Wiley-Blackwell. Third edition. August, 2000.

[2] J.K. Brueckner. "Urban sprawl: Diagnosis and remedies". International Regional Science Review. Vol. 23, Issue 2, pp. 160-171. 2000.

[3] M. Brülhart and F. Sbergami. "Agglomeration and growth: Cross-country evidence". Journal of Urban Economics. Vol. 65, Issue 1, pp. 48-63. January, 2009.

[4] M. Fujita, P. Krugman and A.J. Venables. "The Spatial Economy: Cities, Regions, and International Trade". MIT Press Books. Vol. 1. June, 2001.

[5] X. Gabaix. "Zipf's law for cities: An explanation". The Quarterly Journal of Economics. Vol. 114, Issue 3, pp. 739-767. August, 1999.

[6] X. Gabaix and Y.M. Ioannides. "The evolution of city size distributions". Handbook of Regional and Urban Economics. Vol. 4, Issue 53, pp. 2341-2378. April, 2004.

[7] E. Helpman. "The size of regions". Working Papers. Vol. 14, Issue 95. 1995.

[8] J.V. Henderson. "The effects of urban concentration on economic growth". National Bureau of Economic Research Working Papers. Vol. 7503. January, 2000.

[9] V. Henderson. "How urban concentration affects economic growth". The World Bank Policy Research Working Paper Series. Vol. 2326. April, 2000.

[10] J. Kari. "Theory of cellular automata: a survey". Theoretical Computer Science. Vol. 334, Issue 1-3, pp. 3-33. April, 2005.

[11] P. Krugman. "Increasing returns and economic geography". Journal of Political Economy. Vol. 99, Issue 3, pp. 483-99. June, 1991.

[12] C.G. Langton. "Studying artificial life with cellular automata". Physica D: Nonlinear Phenomena. Proceedings of the Fifth Annual International Conference. Vol. 22, Issue 1-3, pp. 120-149. 1986.

[13] Y. Mansury and L. Gulyas. "The emergence of Zipf's Law in a system of cities: An agent-based simulation approach". Journal of Economic Dynamics and Control. Vol. 31, Issue 7, pp. 2438-2460. 2007.

[14] J. Von Neumann. "Theory of Self-Reproducing Automata". University of Illinois Press. 1966.

[15] S. Tisue and U. Wilensky. "NetLogo: Design and Implementation of a Multi-Agent Modeling Environment". International Conference on Complex Systems. 2004.

[16] M. Tomassini and M. Perrenoud. "Cryptography with cellular automata". Applied Soft Computing. Vol. 1, Issue 2, pp. 151-160. 2001.

[17] J.G. Williamson. "Regional inequality and the process of national development: A description of the patterns". Economic Development and Cultural Change. Vol. 13, Issue 4, pp. 1-84. July, 1965.

[18] Z. Xu and R. Harriss. "A Spatial and Temporal Autocorrelated Growth Model for City Rank Size Distribution". Urban Studies. Vol. 47, Issue 2, pp. 321-335. 2010.

[19] G.K. Zipf. "Human behavior and the principle of least effort". Addison-Wesley Press. 1949.

[20] S. Wolfram. "Cellular Automata as models of complexity". Nature. Vol. 311, Issue 5985, pp. 419-424. October, 1984.

[21] W.R. Tobler. "Cellular Geography". Philosophy in Geography. 1979.

[22] U. Wilensky. "NetLogo (and NetLogo User Manual)". Center for Connected Learning and Computer-Based Modeling. 1999.

[23] S. Tisue and U. Wilensky. "NetLogo: A Simple Environment for Modeling Complexity". International Conference on Complex Systems. 2004.

[24] C. Andersson, K. Frenken and A. Hellervik. "A complex network approach to urban growth". Environment and Planning. Vol. 38, Issue 10, pp. 1941-1964. 2006.

[25] D. Stevens and S. Dragicevic. "iCity: A GIS-CA modelling tool for urban planning and decision making". Environmental Modelling and Software. Vol. 22, pp. 761-773. 2007.