Proyecto Final Inteligencia Artificial - Simulación

Traffic Lights Optimization

Integrantes:

Leandro Rodríquez Llosa

Laura V. Riera Pérez

Marcos M. Tirador del Riego

Grupo: C-311

Tercer año. Ciencias de la Computación.

Facultad de Matemática y Computación, Universidad de La Habana, Cuba

Enero 2023

I. Repositorio del proyecto

https://github.com/science-engineering-art/traffic-lights

II. Descripción

En todo el mundo, la congestión del tráfico sigue siendo un problema importante en la mayoría de las ciudades, debido al creciente número de vehículos privados, de mercancías y de transporte público. Este fenómeno afecta, sobre todo en horas pico, a los usuarios de la red vial, los cuales pierden mucho tiempo en la carretera; además de incidir de manera negativa en el medio ambiente pues los carros se encuentran más tiempo encendidos liberando gases a la atmósfera.

Se puede pensar en varias soluciones para este problema:

1. Construcción de nuevas carreteras: Muchas veces esto no es posible debido a las condiciones geográficas, y más importante aún, es muy costoso, por lo que en general no es una solución viable.

2. Mejora del sistema de señalización vial: Es más sensata pues se relaciona inteligentemente con la infraestructura existente. Es de especial interés la mejora de los semáforos ya que estos controlan el flujo de la red vial de la ciudad. En estos podemos tener:

   - Plan de luces fijo (Estático): se fijan los tiempos de verde y rojo en cada línea de luces de una intersección así como su secuencia una sola vez teniendo en cuenta las previsiones de tráfico, y estas no cambian.
   - Controladores de tiempo real (Dinámicos): en técnicas de tiempo real, el sistema debe ser capaz de adaptarse inmediatamente (o muy brevemente) a las condiciones del tráfico. Dicho sistema posee algoritmos que permiten controlar el tráfico, los cuales reciben información sobre el estado del tráfico, que ha sido recolectada por los sensores colocados en cada carril, y recalculan la duración y la sincronización de la luces para minimizar la congestión, es decir, para minimizar el tiempo promedio de espera en las luces, y la duración de colas.

1. Objetivo

Creación de un algoritmo de control que determine el tiempo de luz verde óptimo en los semáforos de las intersecciones, con el fin de hacer más fluido el tráfico y minimizar las colas.

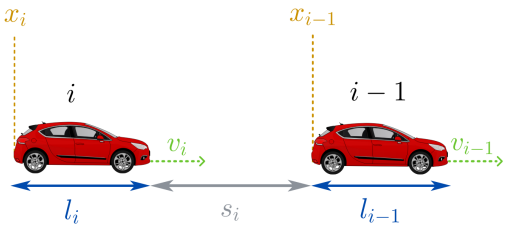
III. Simulación

I. Modelo mesoscópico

Para modelar el flujo del tráfico se utiliza un modelo mesoscópico, modelo híbrido que combina las características de los modelos microscópico y macroscópico.

Como en el modelo microscópico, se representan los vehículos de forma independiente y se intenta replicar el comportamiento de un conductor. En consecuencia, es un sistema multiagente, es decir, cada vehículo opera por sí mismo utilizando información de su entorno.

En cada calle (el fragmento delimitado por dos intersecciones, llamado también cuadra), cada vechículo está identificado por un número i. El i-ésimo vehículo sigue al (i-1)-ésimo vehículo. Para el i-ésimo vehículo, se denota por x_i su posición a lo largo del camino, v_i su velocidad y l_i su longitud. Sean, además, s_i la distancia de parachoque a parachoque y Δv_i la diferencia de velocidad entre el i-ésimo vehículo y el vehículo que le precede (vehículo número i-1).



$$s_i = x_i - x_{i-1} - l_i$$
$$\Delta v_i = v_i - v_{i-1}$$

Por otro lado, como característica del modelo macroscópico, se tienen en cuenta factores globales que describen el movimiento de vehículos como un todo, en términos de densidad de tráfico (vehículos por km) y flujo de tráfico (vehículos por minuto), que sirven para la generación de vehículos en la simulación, que más adelante se abordará en mayor profundidad.

II. Modelo Conductor Inteligente

El Modelo de Conductor Inteligente[1] describe la aceleración del i-ésimo vehículo en función de sus variables y las del vehículo que le precede. La ecuación dinámica se define como:

$$\frac{dv_i}{dt} = a_i \left(1 - \left(\frac{v_i}{v_{0,i}}\right)^{\delta} - \left(\frac{s^*(v_i, \Delta v_i)}{s_i}\right)^2\right)$$

$$s^*(v_i, \Delta v_i) = s_{0,i} + v_i T_i + \frac{(v_i \Delta v_i)}{\sqrt{(2 a_i b_i)}}$$

donde:

- $s_{0,i}$: es la distancia mínima deseada entre el vehículo i el i-1.
- $v_{0,i}$: es la velocidad máxima deseada del vehículo i.
- δ : es el exponente de la aceleración y controla la "suavidad" de la aceleración.
- T_i : es el tiempo de reacción del conductor del i-ésimo vehículo.
- a_i : es la aceleración máxima del vehículo i.
- b_i : es la desaceleración cómoda para el vehículo i.
- s^* : es la distancia real deseada entre el vehículo i e i-1.

Interpretando los términos en s^*:

- $v_i T_i$: es la distancia de seguridad del tiempo de reacción. Es la distancia que recorre el vehículo antes de que el conductor reaccione (frene). Dado que la velocidad es la distancia entre el tiempo, la distancia es la velocidad por el tiempo.
- $(v_i \Delta v_i)/\sqrt{(2 a_i b_i)}$: es una distancia de seguridad basada en la diferencia de velocidad. Representa la distancia que tardará el vehículo en reducir la velocidad (sin chocar con el vehículo de enfrente), sin frenar demasiado (la deceleración debe ser inferior a b_i).

III. Modelo de red vial de tráfico

Para modelar la red vial se utiliza un grafo dirigido, donde las aristas representan las

[1] Tomado de [1]

calles y los vértices las intersecciones. Cada vehículo tiene un camino que consta de múltiples calles. Se aplica el Modelo Conductor Inteligente para vehículos en la misma calle. Cuando un vehículo llega al final de la calle, se retira de la misma y es añadido a la siguiente.

IV. Generación de vehículos

Los vehículos se generan en los extremos del mapa. Cada calle tiene un λ asignado que representa la cantidad de carros por segundo que pasan por ella, el cual corresponde a un valor entre 70 y 150 carros por hora. A las calles que se determinan como principales corresponden mayores valores de λ.

Se usa λ para saber el tiempo que se debe esperar para que pase otro carro por esta calle. Para ello se genera una variable aleatoria que cumple con la distribución exponencial de parámetro lambda ($X \sim Exp(\lambda)$), cuya función de densidad es

$$f_X(x) = \lambda e^{-\lambda x}.$$

Cuando haya pasado el tiempo determinado por la variable, se genera un nuevo carro en esta calle y se vuelve a generar la variable aleatoria.

Al generar un carro se establece su destino y su ruta. El destino se selecciona aleatoriamente de acuerdo a una probabilidad que tiene cada calle de que un carro vaya hacia ella, asignándole mayor peso a las calles más transitadas. Cuando se ejecuta por primera vez una simulación en un mapa, se utiliza Floyd-Warshall para hallar los caminos de menor distancia entre todo par de calles, y se guardan los mismos. La ruta de cada carro será entonces el camino de menor distancia de su calle de salida a su calle de destino, el cual está precalculado.

V. Semáforos

v.1. Turnos

Un turno es un conjunto de semáforos que se encuentran en verde en un mismo período de tiempo. Cada semáforo de la intersección puede existir en más de un turno, siendo posible así establecer distintas combinaciones para las direcciones factibles (seguir recto, doblar a la derecha o doblar a la izquierda) tal que no haya choques y haya la menor cantidad de turnos posibles.

v.2. Zonas

Los semáforos tienen dos zonas en los que los vehículos se comportan de forma diferente:

- Zona de ralentización: Zona en la que los vehículos reducen su velocidad máxima utilizando un factor de ralentización.

$$v_{0,i} := \alpha v_{0,i} \text{ donde } \alpha < 1$$

- Zona de parada: Zona en la que se detienen los vehículos. Esto se logra utilizando una fuerza de amortiguamiento a través de la siguiente ecuación dinámica:

$$\frac{dv_i}{dt} = -b_i \frac{v_i}{v_{0,i}}$$

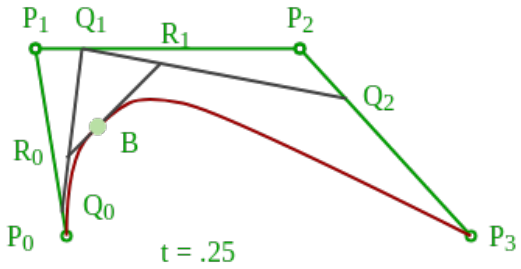
VI. Curvas de Bézier

Una curva de Bézier es una curva polinomial que aproxima a una serie de puntos llamados puntos de control. Está definida por un conjunto de puntos de control P_0 a P_n donde n es su grado. Se dice que una curva de grado n aproxima a $n+1$ puntos de control. El primer y el último punto de control son siempre los puntos extremos de la curva; sin embargo, los puntos de control intermedios (si los hay) por lo general no se encuentran en la misma.

$$P(u) = \sum_{i=0}^{n} P_i B_i^n(u)$$

$$B_i^n(u) = \binom{n}{i}(1-u)^{n-i}u^i$$

donde P_i es el conjunto de puntos, $B_i^n(u)$ representa los polinomios de Bernstein y u toma valor entre 0 y 1.



Para producir una curva en nuestro mapa se crean las calles (rectas) y se utilizan las curvas de Bézier como un spline de suavizado. Este mismo enfoque se utiliza para crear visualmente las intersecciones, de modo que se pueda observar el giro que hace el vehículo al doblar. Sin embargo si no se quiere visualizar, el vehículo simplemente saldrá de una calle y entrará en la otra, optimizando así la complejidad algorítmica de la simulación.

IV. Inteligencia Artificial

I. Algoritmo genético

i.1. Optimización

Mediante el uso de un algoritmo genético se intenta minimizar la demora de los carros en las intersecciones.

i.2. Proceso

El algoritmo genético consiste de varias iteraciones o generaciones. En cada iteración se tiene una población, o conjunto de individuos cada uno de los cuales representa una posible solución del problema. Cada población es el resultado de un proceso de adaptación basado en la población anterior. Se espera que a lo largo de las generaciones se vayan obteniendo mejores soluciones, más aptas.

i.3. Solución

La solución (individuo) será un vector que contiene enteros correspondientes a los tiempos de luz verde de cada turno de la intersección, para todas las intersecciones del mapa. Nótese que los tiempos de luz roja de los turnos quedarán determinados por la suma de los tiempos de luz verde de los restantes turnos. Se considera en este caso que la secuencia de turnos es fija y viene dada por quien provea el mapa. El vector contiene además un valor por cada intersección que corresponde a un tiempo de desplazamiento que tendrán los turnos de la misma. De esta forma se optimiza también la sincronización entre semáforos.

i.4. Población inicial

Para la población inicial se genera un número fijo de individuos que componen a la misma. Cada individuo se hallará de manera aleatoria, en donde cada uno de los valores de tiempo de luz verde estará entre el tiempo promedio que le toma a un carro pasar por la intersección y el tiempo máximo que un vehículo puede estar esperando, ambos configurables.

i.5. Nueva población

En cada iteración la nueva población tendrá la misma cantidad de individuos que la anterior. Para su creación se seleccionan algunos pares de individuos de la población anterior que se cruzarán para crear nuevas soluciones. Estas soluciones también están sujetas a mutaciones. Además, los dos individuos más aptos de cada generación pasarán para la siguiente.

i.6. Función de evaluación *fitness*

Para evaluar qué tan buena es una solución, se configuran los semáforos de acuerdo a la misma y se ejecuta con ella un número de simulaciones (por lo general al menos 30) y se saca información como promedio de las mismas. Luego, con los resultados arrojados por esta en un período determinado y bajo el criterio escogido, se le da una puntuación a dicho individuo.

Se consideraron tres formas de calcular el fitness:

- Según el máximo, tomado sobre todas las calles, del tiempo promedio que demoran

los carros en transitar una calle (dígase el tiempo que demora en recorrerla más el que espera en el semáforo, si es el caso).
- Según el tiempo total de los recorridos de todos los carros.
- Según una media ponderada, tomada sobre todas las calles, del tiempo promedio que toma a los carros transitar una calle. En este caso el peso asignado será el cuadrado de la cantidad de carros que pasan por la calle, de modo que se intenta dar más importancia en la optimización, a las calles más transitadas. La media ponderada en cuestión está dada por la fórmula

$$\frac{a_1 w_1^2 + a_2 w_1^2 + \cdots + a_n w_n^2}{w_1^2 + \cdots + w_n^2},$$

donde a_i es el tiempo que toma como promedio a los carros en la simulación transitar la calle i y w_i es la cantidad de carros que la cruzan durante el tiempo de observación.

Nótese que se quieren minimizar los tiempos de los criterios anteriores, para obtener mejores soluciones. Sin embargo, dado que el algoritmo fue implementado en un primer momento buscando maximización del fitness, se utilizan los opuestos de las métricas calculadas según dichos criterios.

i.7. Selección de padres

Una de las formas de seleccionar los padres implementada fue la selección proporcional al fitness. O sea se le asigna a cada individuo una probabilidad de procrear proporcional a su fitness.

Sin embargo se optó mejor por la utilización de la selección basada en el rango (*rank selection*), en la cual se ordena según el valor de fitness y se da una probabilidad de selección a cada individuo.

La selección basada en el rango reduce los efectos potencialmente dominantes de individuos de alto fitness, comparativamente, en la población, estableciendo una cantidad predecible y limitada de presión de selección a favor de tales individuos. Al mismo tiempo, exagera la diferencia entre valores de fitness agrupados de forma cercana para que los mejores se puedan muestrear más.

Finalmente para seleccionar los padres que van a procrear se generan aleatoriamente según la probabilidad descrita dos listas de padres, donde las parejas son los individuos en igual posición de cada lista. Se tiene en cuenta que un individuo no se aparee consigo mismo. Nótese que un individuo puede aparecer múltiples veces en las listas si su probabilidad es alta, significando que deja varios descendientes. Los tamaños de las listas son calculados de antemano de forma que la nueva población sea del mismo tamaño que la anterior.

i.8. Cruzamiento

La función de cruzamiento genera dos individuos para añadir a la nueva población a partir de cada apareamiento de los padres. Se consideraron tres formas de realizar el cruzamiento:

- **Cruzamiento por puntos múltiples:** Se seleccionan p puntos random y los segmentos (resultado de picar los individuos por dichos puntos) alternos de los individuos se intercambian para obtener nuevos descendientes.
- **Cruzamiento geométrico:** Se emparejan los genes que tienen la misma posición en los padres, y en esta posición del hijo se pone la media geométrica entre ellos. Además de este descendiente, se devuelve uno de los padres, escogido de forma aleatoria, para que no disminuya el tamaño de la población.
- **Cruzamiento intermedio:** Se emparejan los genes que tienen la misma posición en los padres, y en esta posición del hijo se pone la media aritmética entre ellos. Además de este descendiente, se devuelve uno de los padres, escogido de forma aleatoria, para que no disminuya el tamaño de la población.

i.9. Mutación

La función de mutación se aplica sobre una población, dada una probabilidad de mutación, en este caso igual al inverso del tamaño del vector solución. Por cada gen de cada individuo, se escoge un número m aleatorio entre 0 y 1, y dicho gen es mutado si m es menor que la probabilidad de mutación. De esta forma resulta mutado, como promedio, un gen por cada individuo. La forma de mutar es sustituir el gen en cuestión por un número aleatorio entre el tiempo promedio que le toma a un carro pasar por la intersección y el tiempo máximo que un vehículo puede estar esperando, de igual forma que para crear la población inicial.

II. A-star (A^*)

La búsqueda A^* es un algoritmo de búsqueda *best-first* informada que utiliza la función de evaluación:

$$f(n) = g(n) + h(n)$$

donde $g(n)$ es el costo de alcanzar el nodo n, $h(n)$ es una heurística del costo estimado del camino de menor costo desde n hasta el objetivo, siendo $f(n)$ entonces el costo estimado de la mejor solución que pasa por n.

Se concibieron dos algoritmos A^*, un bastante clásico para hallar la distancia mínima entre dos calles, y otro para hallar el camino que menor tiempo le tomará a un vehículo para llegar de una calle a otra. En nuestro caso a cada nodo corresponde una esquina o intersección y a cada arista corresponde una calle que une dos intersecciones.

ii.1. A^* para hallar la distancia mínima entre dos calles

$g(n) =$ *distancia recorrida hasta el momento.*

$h(n) =$ *distancia en línea recta desde el punto actual hasta el objetivo.*

ii.2. A^* para hallar el camino que menor tiempo le tomará a un vehículo para llegar de una calle a otra

$g(n) =$ *tiempo que demoró el vehículo en llegar desde el estado inicial hasta el estado intermedio n.*

En este caso los valores de $g(n)$ se calculan mediante una simulación del tránsito de la ciudad. Cada vez que se expande un nodo m, para cada hijo n correspondiente, se genera un carro que transita la calle correspondiente al nodo n, calculando el tiempo tomado por el mismo. Luego el valor de $g(n)$ lo podemos calcular mediante $g(n) = g(m) + e$, donde e es el valor que se obtuvo antes.

$h(n) =$ *aproximado del tiempo requerido para llegar del estado intermedio n hasta el estado final.*

Se precalcula antes de correr el algoritmo sobre un mapa dado, ejecutando Disjktra desde el punto de destino, de forma que se calculen para cada punto del mapa, el tiempo estimado que tomará a un vehículo llegar desde este hasta el destino. Para este cálculo se tienen en cuenta los semáforos, distancia de las calles, velocidad de los carros y tráfico. Luego, para obtener $h(n)$ simplemente se busca el tiempo precalculado correspondiente al punto del mapa relativo al nodo n.

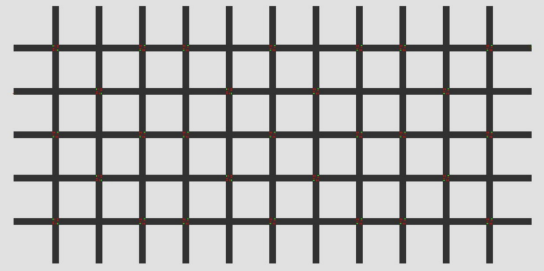
V. Tests

Se ejecutaron 53 pruebas con los siguientes parámetros para la optimización:

- **tamaño de la población:** 30,
- **cantidad de iteraciones:** 50,
- **tiempo de observación de cada individuo en la simulación:** 10s,
- **velocidad de la simulación:** 30, que significa que por cada segundo en tiempo real, habrán pasado 30s en el entorno simulado,
- **tipo de cruzamiento:** cruzamiento por puntos múltiples (2),

- **cálculo de fitness:** según la media ponderada, calculada sobre todas las calles, del tiempo promedio que toma a los carros cruzar una calle,
- **tiempo promedio que le toma a un carro pasar por la intersección:** 3s,
- **tiempo máximo que un vehículo puede estar esperando:** 90s,

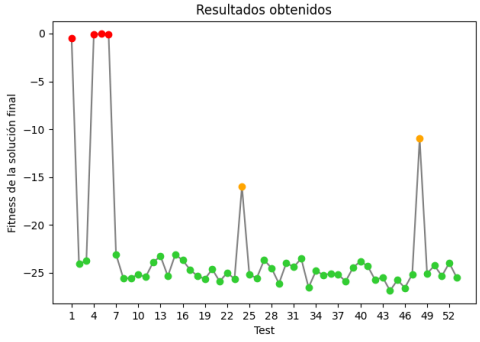
sobre el mapa, de 12×6 cuadras y 29 intersecciones semaforizadas, que se muestra a continuación:



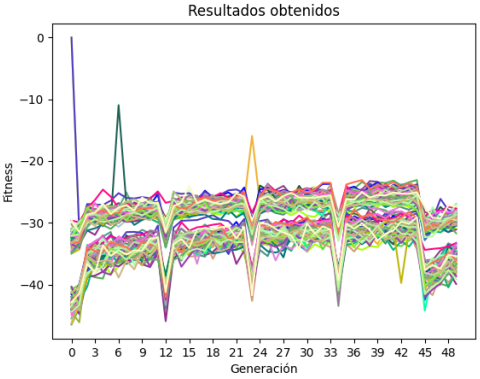
Dichas pruebas se pueden encontrar en la carpeta *tests* del repositorio.

1. Resultados

La siguiente gráfica muestra los valores de fitness de las 53 pruebas corridas del algoritmo genético. Se puede observar que el valor de fitness en la mayoría de los tests, para la mejor solución obtenida, se mantiene en el rango de 23 a 26. Esto podría interpretarse como que los carros (sobre todo aquellos en las calles más transitadas porque les fueron asignados mayores pesos) se demoran como promedio 25*s* en cruzar una calle. Hay que notar que hubo algunos tests cuyas mejores soluciones dieron valores demasiado buenos para ser reales, los cuales fueron marcados en rojo y naranja en la gráfica. Los resultados marcados en naranja es probable que se deban a que de la aleatoriedad, de generación de vehículos, hayan resultado simulaciones con eventos muy favorables, mientras que los marcados en rojo deben haber sido errores ocurridos durante la ejecución del algoritmo.



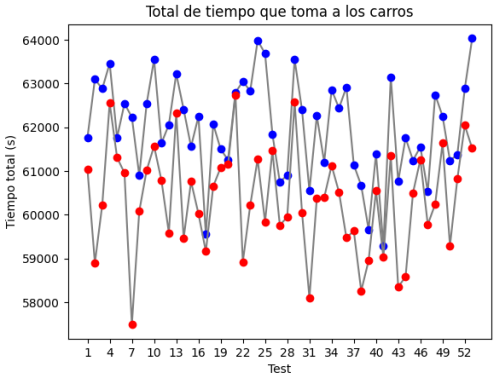
Observando los valores de fitness obtenidos para cada test, a lo largo de todas las generaciones, se puede notar que estos tienen siempre tendencia a mejorar. Para apreciar mejor esto, se provee junto con este trabajo un gráfico por cada test[2]. En estos se muestran para cada generación, la mejor y peor solución, donde se aprecia claramente como la curva que describen es creciente. La siguiente gráfica muestra una unión de todas las gráficas antes descritas, para todos los tests. En ella se puede ver bien como los valores de fitness van aumentando según pasan las generaciones.



Finalmente, se seleccionaron por cada test las mejores soluciones y alguna solución arbitraria de la primera generación. Para estas se calcularon nuevamente, mediante algunas ejecuciones más de la simulación, el tiempo total que toma a todos los vehículos en completar su recorrido en el mapa. En la gráfica a continuación se muestran en puntos azules las

[2] dirección: src/graphics/graphics.zip

mejores soluciones, y en rojo las de la primera generación. Se puede apreciar que los puntos azules se encuentra siempre por encima de los rojos, llegando a encontrarse diferencias de hasta más de una hora (entre todos los carros). Teniendo en cuenta que las simulaciones fueron realizadas durante 5 minutos solamente, una mejoría de una hora en los pocos carros que pueden haber circulado el mapa en este tiempo, nos da una medida de qué tanto se podría llegar a optimizar si se ejecuta por el tiempo suficiente.



Como conclusión podemos decir que el algoritmo arroja resultados que optimizan en cierta medida el tiempo de demora de los carros en su trayecto. Hay que tener en cuenta que dada la limitada capacidad de cómputo con que se contaba, y el tiempo del que se dispuso para obtener los resultados, se tuvieron que relajar mucho los parámetros utilizados. Idealmente deberían haberse corrido simulaciones más largas y con un tamaño de población más acorde al número de coordenadas de los vectores solución.

VI. Recomendaciones

Se recomienda, con más tiempo y capacidad de cómputo, correr el algoritmo genético con parámetros más grandes. Pudiera también escalarse el mapa para simular una ciudad más grande. Otra recomendación es probar con otras métricas para optimizar necesidades más específicas de un posible cliente.

Como trabajo futuro sería interesante hacer que la distribución de los turnos formase parte de la solución, de forma que una distribución dada pueda contribuir a un mejor flujo del tráfico. También se pudiera optimizar la posición de las intersecciones semaforizadas (añadir, quitar o reacomodar intersecciones en el mapa), para contribuir a hacer el tráfico más fluido, objetivo perseguido en este proyecto.

Además, se recomienda quitar algunas de las relajaciones realizadas al problema para simular un ambiente más parecido a uno real. Por ejemplo, incluir el tráfico de peatones o permitir que los carros cambien de senda a antojo del conductor.

La generación de mapas, se hace de una manera determista, y por cada mapa se tiene que implementar un template que lo construya. Este proceso de generación se puede automatizar utilizando una grámatica libre del contexto con algún tipo de conocimiento asociado a la modelación de un mapa. Por ejemplo se conoce que si se tiene una calle principal, la calle que le sigue forma parte de la calle principal y conserva el mismo número de carriles, y las calles que la cruzan por lo general no son avenidas principales ya que estas se encuentran dispersas. Pero como las producciones de una gramática libre del contexto son determinitas quizás puedan caer en el mismo problema que se tenía inicialmente, o sea, construir mapas de un tipo específico. Entonces se puede pensar en en añadir a cada producción una probabilidad asociada que le de mayor peso a producciones correspondientes a distribuciones de calles que tiene más ocurrencia en la vida real y menos a las que no, como en el ejemplo descrito. Finalmente, lo que se está proponiendo es utilizar una gramática libre del contexto probabilística para poder generar mapas arbitrarios sin tener que generar un template por cada mapa distinto que se quiera obtener.

Referencias

[1] Martin Treiber, Ansgar Hennecke, and Dirk Helbing: *Congested traffic states in empirical observations and microscopic simula-*

tions. Phys. Rev. E 62, 1805 – Published 1 August 2000.

[2] Stuart Russell & Peter Norvig: *Artifitial Intelligence. A Modern Approach.* Fourth Edition. Pearson Series In Artifitial Intelligence. 2020.