Proyecto Final de Inteligencia Artificial y Simulación

World of Needs

Autores (Grupo C-311):

David Manuel García Aguilera

Usuario en Github: @dmga44

Rolando Sánchez Ramos

Usuario en Github: @rolysr

Andry Rosquet Rodríguez

Usuario en Github: @aXoXoR2

Repositorio en Github:

`https://github.com/rolysr/world-of-needs`

Curso 2022

Resumen: En las sociedades humanas actuales se manifiestan diariamente procesos de movimientos de personas con diversos objetivos. Un ejemplo bastante importante es el que realizan algunos actores sociales con el fin de buscar víveres y realizar trámites de manera general para satisfacer una serie de necesidades personales o de su hogar. Las instituciones encargadas de velar por la satisfacción de los pedidos básicos de los habitantes de una zona determinada, invierten en recursos y medios correspondientes a estos pedidos, además de planificar la forma de acceso a los mismos por parte de los pobladores. El objetivo de este proyecto es permitir simular un sistema que defina el comportamiento de una sociedad o conjunto de personas distribuidas a lo largo de una zona acotada, donde existen una serie de centros especiales a los que estos humanos deben trasladarse para satisfacer sus necesidades, con lo cual se permitirá dar respuesta a problemáticas relacionadas con la cantidad de dinero a invertir por una entidad central a cargo de los centros de destino y el grado de satisfacción de un conjunto de clientes potenciales.

1. INTRODUCCIÓN

El problema que se abordará en este proyecto tiene inicialmente una entidad central (organización gubernamental, empresa o institución), la cual posee un conjunto de destinos (centros comerciales, tiendas, institutos) en los que se puede dar solución a una o más necesidades predefinidas (venta de alimentos, víveres del hogar y otros productos, tramitación de documentos). Se tiene además, un espacio acotado donde estarán localizados cada uno de los centros adjuntos de dicha entidad, el cual puede ser considerado como una porción de una ciudad. También, es conocido el presupuesto con que la misma cuenta para adquirir los productos o soluciones que puede brindar a manera de ofertas, de forma tal que estas puedan ser distribuídas de alguna forma en los destinos posibles. En el problema se tiene en cuenta la presencia de una serie de clientes, los cuales son personas que poseen necesidades que pueden ser ofertadas en algunos de los centros establecidos. Por lo tanto, uno de los objetivos principales de este trabajo consiste en simular el proceso mediante el cual un conjunto de personas se traslada en una región con presencia de destinos posibles, con el fin de satisfacer sus necesidades, para lo cual se propone un modelo basado en múltiples agentes.

El modelado basado en agentes es una metodología computacional que nos permite crear, analizar y experimentar con mundos artificiales poblados por agentes con disímiles comportamientos y objetivos. Existen varias áreas de investigación relacionadas con el modelado basado en agentes para la simulación del efecto de los cambios en el entorno a partir del movimiento de las personas [1] [2], las cuales pueden contar con una serie de necesidades y/o formas de actuar, como por ejemplo en modelos a microescala [3] [4], los cuales constituyen una interesante solución para agentes que poseen una planificación bien definida de la forma en que deciden satisfacer sus necesidades. Otros estudios referentes al comportamiento de agentes que se trasladan para realizar compras a nivel regional [5] resultan interesantes pues se introducen conceptos como índices de calidad de un destino en base a las necesidades, la distancia y la calidad de las ofertas, así como características propias de los agentes, como la edad, el índice de ingresos monetarios y la distribución de sus necesidades.

Finalmente, teniendo en cuenta las investigaciones en el campo de simulaciones de multiples agentes en ciudades con una planificación de compras en base a necesidades, se propone establecer un modelo computacional que permita, a partir de un conjunto de centros de destino, agentes con necesidades predefinidas, verificar interrogantes como: ¿De qué forma se pudiera distribuir un presupuesto inicial para construir lugares de destinos en un ambiente dado con el fin de satisfacer la mayor cantidad de necesidades de los agentes en este? ¿Cuánto presupuesto haría falta para satisfacer la necesidad de todos? ¿Cuál es el mínimo presupuesto para satisfacer a una cantidad especificada de ciudadanos con cierto margen de error? Por lo tanto, podemos ver inicialmente la utilidad de esta propuesta.

2. DESARROLLO

Al analizar los requerimientos del problema planteado, vemos una serie de componentes principales que resulta evidente implementar, así como un conjunto de funcionalidades que estas deberían ofrecer y las relaciones que se manifiestan entre las mismas. Por lo tanto dichas componentes serían:

• Ambiente: Representa el medio sobre el cual se llevarán a cabo los sucesos de la simulación. El mismo deberá tener una forma de almacenar o representar el estado geográfico de los componentes que este contiene y de los hechos que en él ocurren. Una forma de representación de este pudiese ser a partir de una matriz bidimensional con celdas que representen una unidad de espacio que puede ser válida o no para la presencia y recorrido de los agentes o la creación de centros posibles de destino para estos.

El ambiente debe tener funcionalidades como:

1. Añadir agentes y centros en posiciones válidas y almacenar datos referentes a estos de alguna forma.
2. Ofrecer datos suficientes del estado geográfico para ser utilizados por un framework dedicado a la visualización de mapas o zonas.
3. Calcular y llevar internamente estadísticas que permitan describir el estado de los componentes y hechos en el sistema como por ejemplo el tiempo transcurrido.

• Lugares de destino: Estos componentes pueden representar desde una tienda de productos hasta un centro para que los agentes realicen algún trámite. Los mismo deben estar ubicados en zonas específicas del mapa y ofrecer una descripción de las necesidades que pueden ser satisfechas ahí.
Para cada necesidad posible en un lugar de destino habrá una forma específica de tratar a los agentes que arriban. Un ejemplo de esto puede ser un lugar de destino $D1$, al cual se le define una necesidad $N1$ con ciertas características como límite de disponibilidad, el cual atiende a los clientes con una política de colas bien definida.
Las posibles funcionalidades de los lugares de destino son:

1. Llevar un estado interno del sistema siendo simulado y que contenga estadísticas que permitan analizar el estado actual.
2. Definir una política mediante la cual serán atendidos los clientes, por ejemplo, la cantidad de servidores de atención, si estos servidores son en serie o paralelo.
3. Definir una forma mínima de almacenamiento del estado de los clientes inmersos en el proceso de espera.

• Agentes: Son aquellos integrantes del sistema que tienen como objetivo simular el comportamiento humano y tomar decisiones en base a una serie de objetivos predefinidos en los mismo. Estos pueden tener una serie de límites como dinero disponible, cantidad de necesidades a cumplir.
Las funcionalidades que pueden tener estos son:

1. Definir límites de los mismo en cuanto a diversos parámetros como capacidad monetaria y necesidades.
2. Definir política o forma de comportamiento, lo cual influye en el proceso de toma de decisiones como decidir a dónde moverse etc.
3. Tener formas de mostrar el estado de su satisfacción en base a los objetivos cumplido y que se pueda conocer los que le faltan por cumplir.

2.1. Implementación del modelo propuesto:

Para llevar a cabo la simulación se implementaron un conjunto de módulos principales, utilizando el lenguaje de programación *Python* [6]. Dichos módulos son: *agents* ,

environments y *utils*, los cuales contienen aquellos tipos de datos y funciones necesarias para ejecutar todos los procesos pensados para este proyecto. El módulo *agents* está destinado para la implementación de los tipos de agentes que intervienen en la simulación, así como el conjunto de campos, propiedades y métodos necesarios para su correcto funcionamiento en un entorno. Por otro lado, el módulo *environments* tiene como objetivo crear tipos de datos que permitan representar un entorno de ejecución, colocar agentes, verificar el estado de la simulación, llevar a cabo un conjunto de iteraciones, entre otras funcionalidades. Finalmente, se tiene el módulo *utils*, que contiene a su vez submódulos encargados de la generación de grafos en forma de ciudades, generación de variables aleatorias que representen algunos hiperparámetros de la simulación (como por ejemplo la velocidad de desplazamiento de las personas en una ciudad), algoritmos de decisión de los agentes para determinar la próxima acción a realizar y las políticas de comportamiento de los agentes para realizar las compras o adquisición de medios en un lugar de destino.

Los módulos *agents* y *environments* dependen de implementaciones en *utils*, mientras que en un segundo plano, pero no menos importante, se tiene el módulo *testing*, el cual está orientado a la implementación de pruebas unitarias e integración entre los tres módulos principales. En la figura 1 se muestra la arquitectura general de los módulos del proyecto y la dependencia entre estos:

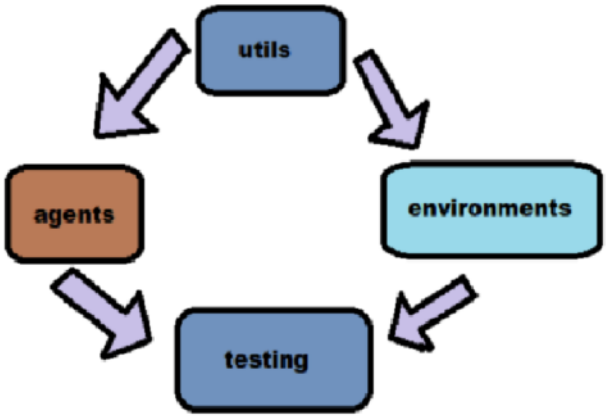

FIGURA 1. Arquitectura del proyecto.

2.1.1. Módulo agents:

El módulo *agents* contiene las especificaciones de los tipos de agentes que intervendrán durante la simulación. Inicialmente se tiene la implementación de la clase *Agent*, que representa la clase base que contiene todas las funcionalidades comunes mínimas de cualquier tipo de agente que se desee implementar para el proyecto, esta posee los campos: *id* (identificador único del agente en el entorno), *log_record* (lista de descripciones de las acciones del agente durante su ejecución), *name* (forma legible de identificar a un agente). Además, esta clase tiene el método *narrate*, el cual es utilizado para imprimir las acciones de un agente en un determinado intervalo de tiempo desde que este comenzó a ejecutarse en una simulación.

La clase *HumanAgent* hereda de *Agent*, y la misma representa a los agentes humanos. Los campos principales que contiene son: *needs* (contiene una lista de las necesidades del agente, las cuales contienen la prioridad, un identificador y la cantidad a satisfacer), *speed* (velocidad de traslación del agente en el entorno en metros por minuto), *balance* (capacidad monetaria del agente humano), *social_class* (clase social del agente humano, determina su política de compra). También, esta clase contiene el método *offers_requests*, el cual recibe un conjunto de ofertas de varias posibles necesidades y en base a su clase social, ejecuta una lógica que consiste en seleccionar un conjunto de ofertas de tal forma que satisfaga sus necesidades en base a un criterio particular. Otro de los métodos importantes es *next_destination_to_move*, con el cual el agente humano decide, dado su estado actual y su percepción del entorno, cuál sería el próximo destino al cual dirigirse para satisfacer sus requerimientos personales. Los agentes humanos poseen también una forma de determinar la calidad de las compras que realizar así como el grado de satisfacción dado el estado actual, lo cual es determinado por los métodos *purchase_dissatisfaction* y *dissatisfaction* respectivamente. En un segundo plano se encuentra el método *reset*, que es utilizado para reestablecer los valores del agente humano exactamente como los tenía al inicio de su creación.

El tipo definido *DestinationAgent* permite representar a los destinos posibles a los que pueden acudir los agentes humanos. Los campos principales de estos son: *offers* (conjunto de ofertas del destino, las cuales contienen un identificador de la necesidad que satisface, la cantidad que ofrecen y el precio al cual ofrecen cada unidad), *total_time_working* (tiempo en que está disponible un destino, dado en horas) y *queue* (forma interna de dar seguimiento al estado interno de los clientes y determinar quién es el próximo a ser atendido). Para determinar tiempo de atención a un cliente se utiliza el método *attention_time* el cual devuelve el resultado de alguna variable aleatoria que represente este fenómeno en minutos. También, esta clase contiene el método *process_offers_requests*, el cual recibe un conjunto de solicitudes de ofertas de un agente humano y se encarga de internamente actualizar el estado del destino. Análogamente a *HumanAgent*, esta clase posee el método *reset* cuya implementación sigue el mismo principio que en dicho caso.

En la figura 2 vemos un ejemplo de la manera en que se expresan las dependencias y principales características de las clases en este módulo.

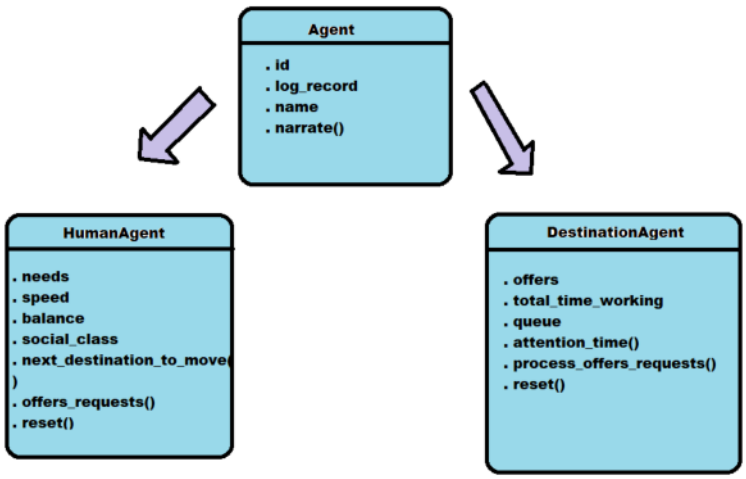

FIGURA 2. Módulo *agents*.

2.1.2. Módulo environments:

Con el módulo *environments* se logra representar los posibles entornos donde podrían intervenir los agentes implementados y realizar ejecuciones de los mismos para obtener

resultados que permitan arribar a conclusiones de una simulación particular. Para ello, se cuenta con las implementaciones de las clases *Environment* y *Experiment*.

La clase *Environment* es una implementación genérica de los posibles entornos que pueden ser probados para la ejecución. Sus campos principales son: *human_agents* (conjunto de instancias de los agentes humanos), *destination_agents* (conjunto de instancias de los agentes de destino), *graph* (grafo interno del entorno que representa una ciudad) y *schedule* (cola de prioridad interna con el conjunto de acciones que deben ocurrir en la simulación). También, unido a dichos campos se encuentran métodos como *execute* y *run*, donde el primero recibe un parámetro que representa el tiempo en minutos a simular dado el estado actual del entorno y ejecuta los procesos que pueden ocurrir en dicho intervalo de tiempo, mientras que el segundo se encarga de correr toda la simulación mientras queden acciones por realizarse o el tiempo de ejecución del sistema no haya terminado. Para determinar que el sistema ya completó todo el proceso de simulación se implementó el método *is_done*, que básicamente comprueba que el tiempo de ejecución predefinido para el entorno no se haya cumplido. Además, resulta importante destacar los métodos *negotiation* y *arrival*, los cuales son utilizados para ejecutar los dos tipos de acciones fundamentales que pueden ocurrir en el entorno, la negociación de un humano con un centro de destino y la llegada de un agente a un centro de destino. Por otro lado, al igual que en las clases del módulo *agents*, se tienen implementaciones para los métodos *narrate* y *reset*.

Por último, se tiene la implementación del tipo *Experiment*, cuya función es ejecutar repetidamente un entorno predefinido, de tal forma que en cada iteración se varíen algunos hiperparámetros escogidos de este con el fin de dar respuestas a las posibles interrogantes que se esperan solucionar con este proyecto. Básicamente esta clase recibe una instancia del tipo *Environment* y con los métodos de optimización que tiene implementado, permite definir si se quiere optimizar los parámetros del entorno en base a un factor predefinido. Estos factores de optimización están especificados en un *enum* denominado *optimization_target*, el cual contiene los factores:

1. STORE_OFFERS_DENSITY: Optimización de la densidad de las ofertas en los lugares de destino.
2. OFFERS_PRICE_FACTOR: Optimización de los precios de los productos en los destinos.
3. TOTAL_BUDGET_FACTOR: Optimización del presupuesto de la entidad central a cargo de los destinos.
4. STORE_DISTRIBUTION: Optimización de la proporción del suministro de los destinos en el entorno.

La ejecución del proyecto se basa principalmente en la clase *Experiment*, la cual depende de una implementación específica del tipo *Environment*. Aunque los entornos puedan ser ejecutados de forma independiente, es mediante los experimentos que se logra encontrar sentido a los posibles resultados que puedan derivarse de la simulación. Un ejemplo de la arquitectura de este módulo se muestra en la figura 3.

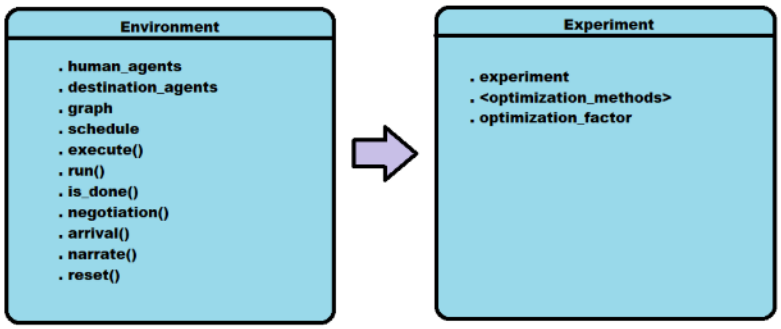

FIGURA 3. Módulo *environments*.

2.1.3. Módulo utils:

En este módulo se encuentran varios submódulos encargados de muchas de las funcionalidades añadidas a los módulos *agents* y *environments*:

1. *generator*: Contiene todos las implementaciones de los hiperparámetros, estructuras de datos y variables aleatorias generadas para la simulación.
   a) *destination_generators*: Generadores de los agentes de destino.

   b) *human_generators*: Generadores de los agentes humanos.
   c) *natural_language_generation*: Generación de los nombres de los agentes humanos y de destino.
   d) *environment_schedule_generator*: Generación del orden de las acciones iniciales en un entorno.
   e) *graph_generator*: Generación de los grafos de las ciudades donde se ejecutarán las simulaciones.
   f) *street_length_generator*: Generación del tamaño de las calles de las ciudades.
2. *graph*: Implementaciones para representar el estado de los grafos de las ciudades
   a) *algorithms*: Algoritmos para el recorrido de grafos.
   b) *Graph*: Clase utilizada para representar los grafos.
   c) *Node*: Clase utilizada para representar los nodos de los grafos generados.
3. *next_destination_logic*: Algoritmos utilizados por los agentes humanos para determinar el siguiente destino al cual moverse.
4. *offers_requests_policies*: Algoritmos utilizados por los agentes humanos para llevar a cabo la compra de productos o de trámites ofertados por los agentes de destino.

2.2. Aplicaciones de la Inteligencia Artificial:

Muchas de las acciones llevadas a cabo por los agentes humanos son potenciadas gracias a diversos algoritmos de Inteligencia Artificial (IA). La diversidad de los mismos ayuda a tener en cuenta cuáles son los mejores comportamientos que pueden ejecutar los agentes humanos para satisfacer sus necesidades. Por otro lado, el uso de los algoritmos de IA están presentes también en los procesos de optimización, los cuales tienen un fuerte uso en la ejecución de los experimentos con el fin de dar respuesta a las interrogantes planteadas inicialmente en el proyecto. Además, es importante destacar la importancia de estos algoritmos en la generación de nombres de los agentes que intervienen en la simulación.

2.2.1. Algoritmos de Búsqueda:

Los algoritmos de búsqueda son utilizados por los agentes humanos para determinar el próximo destino al cual dirigirse en un momento determinado de la ejecución de un

entorno. Esto ocurre cada vez que un humano llega a un destino y determina que este no puede satisfacer sus necesidades en dicho instante de tiempo, o justo después de que este interactúe con un agente de destino en un proceso de negociación.

Inicialmente para el proyecto se implementó un método basado en búsqueda informada mediante en el algoritmo A^* para múltiples objetivos. La intuición de este algoritmo consiste en tener como estado inicial la posición actual del agente humano y con este, un conjunto de estados finales que representen los posibles destinos al los cuales podría dirigirse. Luego el algoritmo realizaría una búsqueda de tipo *Best First Search*, donde se utilizaría además una heurística basada en la distancia estimada hacia un agente de destino más la calidad estos. Dicha calidad de un agente de destino consiste en determinar para cada uno de estos, el siguiente valor:

$quality\_coefficient(vector_1, vector_2) = \frac{1}{1+\frac{scalar\_product(vector_1, vector_2)}{vector\_norm(vector_1)^2}}$.

Dicho valor esta comprendido en el rango $[0; 1]$ y constituye un relación proporcional entre el vector de necesidades de un agente humano ($vector_1$) y el vector de ofertas de un agente de destino ($vector_2$). Mientras menor sea la capacidad de las ofertas de un destino de satisfacer las necesidades de un humano, más cercano a 0 estará dicho valor, en caso contrario el valor estaría más cercano a 1. Ideas similares a esta han sido tratadas en estudios anteriores [5]. Finalmente, para determinar el valor de la heurística en cada nodo, se corresponde a colocar en cada uno el mínimo valor de distancia de un nodo de destino hacia dicho nodo que tenemos fijado, multiplicado por el valor del coeficiente 2.2.1 y finalmente dicho valor se divide entre dos.

La heurística planteada anteriormente es admisible ya que su valor es estrictamente menor que el valor de la distancia estimada de un nodo cualquiera al destino más cercano. Por lo tanto, dicho algoritmo planteado ofrece respuestas óptimas. De esta forma, si se tiene el caso en el que un destino está más alejado del agente humano pero lo satisface mucho más que otro que tiene más cerca, entonces el algoritmo permitirá converger al lugar de mayor distancia, lo cual es un ejemplo de los resultados que se desean.

Dada la complejidad computacional que conlleva ejecutar el algoritmo A^*, se realizó una segunda implementación del algoritmo de búsqueda de los agentes humanos utilizando el algoritmo de $Dijkstra$. Por lo tanto, la calidad de un destino se calcularía directamente mutiplicando la distancia real desde el agente humano hacia este, multiplicado por el coeficiente 2.2.1. Finalmente, se logra obtener una vía con menor costo computacional y que permite obtener resultados igual de buenos que con la variante anterior.

2.2.2. Problema de Satisfacción de Restricciones:

El uso de un algoritmo de satisfacción de restricciones está presente en una de las políticas de selección de ofertas por parte de los humanos en los destinos donde estos se encuentran. La idea consiste en realizar una compra que puede ser considerada como válida apoyándose en el uso de una búsqueda de $backtracking$. Este tipo de procedimiento es realizado por algunos de los agentes humanos con mayores ingresos en la simulación, lo que alude a aquellos que pertenecen a la alta clase social (para el caso particular de este proyecto, aquellos cuyos ingresos son el doble de la media de ingresos en la población de agentes humanos). La idea del algoritmo se ve reflejada en la siguiente figura 4:

```
function BACKTRACKING-SEARCH(csp) returns a solution, or failure
  return BACKTRACK({ }, csp)

function BACKTRACK(assignment, csp) returns a solution, or failure
  if assignment is complete then return assignment
  var ← SELECT-UNASSIGNED-VARIABLE(csp)
  for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
      if value is consistent with assignment then
          add {var = value} to assignment
          inferences ← INFERENCE(csp, var, value)
          if inferences ≠ failure then
              add inferences to assignment
              result ← BACKTRACK(assignment, csp)
              if result ≠ failure then
                  return result
      remove {var = value} and inferences from assignment
  return failure
```

FIGURA 4. Pseudocódigo del algoritmo de satisfacción de restricciones implementado.

2.2.3. Metaheurísticas:

El uso de metaheurísticas es uno de los principales aspectos de este trabajo. Su uso está ligado principalmente a los proceso de optimización, que pueden incluir desde una política de negociación de un agente humano, hasta las optimizaciones llevadas a cabo en los experimentos que se realicen con un entorno.

Una de las metaheurísticas utilizadas fue una variante de algoritmo genético como una forma de modelar la política de negociación de un agente humano, específicamente aquellos que pertenecen a la clase social media, es decir aquellos cuya ganacia es menor que el doble del promedio de los ingresos de los humanos y mayor o igual que la media de dichos ingresos solamente. Específicamente en este proyecto, se utilizó dicha variante de algortimo genético para la solución del problema de la mochila con repetición [7], la cual es una forma de ver el mismo problema de elaborar una solicitud de ofertas a comprar por un humano en un destino específico. La idea con este método es que aquellos que lo ejecuten puedan optimizar principalmente en base a la prioridad de las necesidades sin tener mucho en cuenta el dinero que gastan. En la figura 5 se muestra el pseudocódigo del algoritmo.

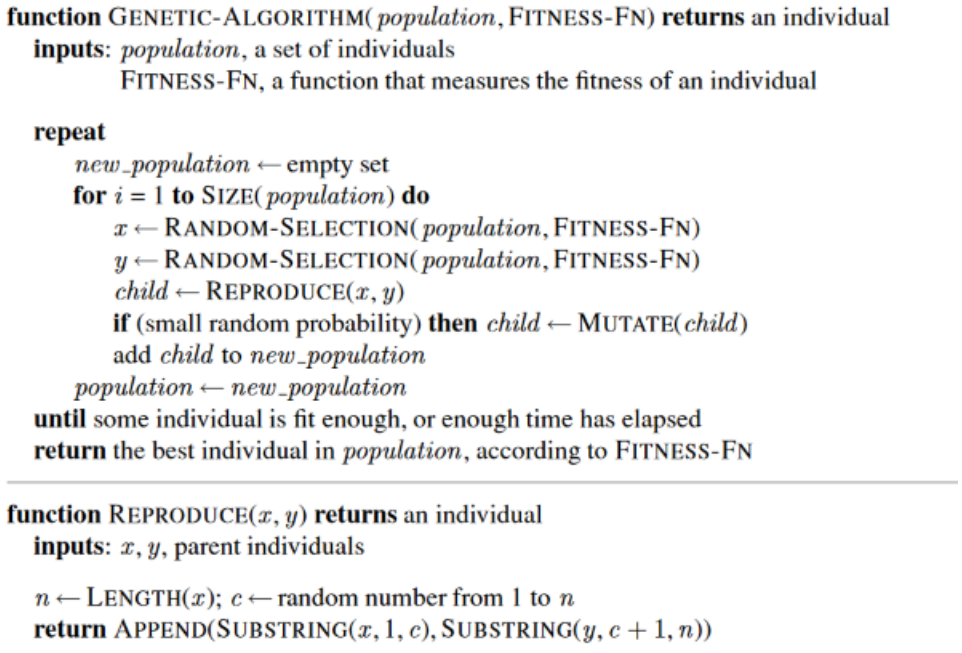

FIGURA 5. Pseudocódigo del algoritmo genético.

Las restantes metaheurísticas están relacionadas con los procesos de optimización, como por ejemplo en la optimización de los experimentos, en cuyo caso se utilizó el algoritmo de *Hill Climbing* [8] y en la optimización de las compras realizadas por un agente humano, donde se implementó el algoritmo de *Threshold Accepting* [9] para aquellos agentes humanos de la clase social más baja, es decir, aquellos cuyos ingresos son menores que la media general de ingresos en la población. La idea de estos algoritmos es poder realizar una optimización eficiente de ciertos parámetros de tal forma que se puedan acotar fácilmente los criterios de parada de los mismos, por lo que el costo computacional puede considerarse asumible.

2.2.4. Procesamiento de Lenguaje Natural:

Una de las aplicaciones más importantes del procesamiento de lenguaje natural es la generación de texto, la cual es una tarea que tiene como objetivo la producción controlada de texto que cumpla con cierta estructura y restricciones predefinidas. Para llevar a cabo este enfoque se utiliza un modelo lingüístico formal, el cual puede basado en una gramática que formalice el lenguaje que se intenta generar.

El problema que se logró solucionar con este método es el de generar nombres de lugares de destino, de forma tal que se disponga con una vía de identificar los agentes de destino con mayor legibilidad al mostrar las notificaciones de las acciones que ocurren durante una simulación. Para ello se construyó una gramática, se creó una pequeña base de nombres de dueños y tipos de destinos, y con el uso de la biblioteca $NLTK$ se logró implementar de manera sencilla dicho algoritmo de generación.

2.2.5. Modelos de Markov:

Los modelos de Markov son una forma bastante común y relativamente simple de modelar estadísticamente procesos aleatorios. Se han utilizado en muchos dominios diferentes, desde la generación de texto hasta el modelado financiero. En general, las Cadenas de Markov son conceptualmente bastante intuitivas y son muy accesibles en el sentido de que pueden implementarse sin el uso de conceptos estadísticos o matemáticos avanzados. Son una excelente manera de comenzar a aprender sobre técnicas de modelado probabilístico y ciencia de datos.

En el caso específico de este proyecto, se utilizó un modelo de Markov [10] ya empleado en otros estudios relacionados con la generación de nombres de personas. Por lo cual, este mismo resultado se utilizó para a generación de identificadores legibles para los agentes humanos que participan en la simulación.

2.3. Aplicaciones de la Simulación:

Existen una serie de elementos pertenecientes al campo de estudio de la Simulación que sería relevante resaltar, lo cual tiene como fin brindar un mayor entendimiento del modelo implementado. Esta constituye una vía de justificar en base a conocimientos bien establecidos, el porqué de varias decisiones durante la realización del proyecto. Por lo tanto, se hace necesaria la mención de algunos de los más importantes.

2.3.1. Características del sistema modelado:

1. *Dinámico no estacionario*: La estructura del sistema cambia con el tiempo y los procesos del sistema ocurren en y dependen de el tiempo.
2. *No determinista*: Conocemos la estructura del sistema y las funciones de cambio de estado y de salida son funciones no deterministas.
3. *Información completa*: Los agentes humanos tienen conocimiento total del estado del sistema. Aunque no ocurre lo mismo con los agentes destino.

2.3.2. Generación de variables aleatorias:

La generación de variables aleatorias se ve presente en varias secciones del proyecto. Los resultados de la generación de estas pueden encontrarse en varios estudios de referencia [11] [12] [13]. Algunas de las principales son:

1. Tiempo de atención (en minutos) de los lugares de destino a los agentes humanos.
2. Tiempo de funcionamiento (en horas) de los agentes de destino durante la simulación.
3. Ingresos de los agentes humanos.
4. Distribución de las necesidades de los agentes humanos.
5. Distribución de las ofertas en los agentes de destino.

6. Velocidad (en metros por segundo) de traslación de los agentes humanos.
7. Generación de la longitud de las calles de las ciudades.

2.3.3. Eventos discretos:

Los eventos discretos son utilizados específicamente en la forma en que se tratan las colas en los agentes de destino y actualizan los estados de estas a lo largo el tiempo. Como conocemos, en varias de las simulaciones basadas en eventos discretos se tienen tres tipos de variables principales: la variable de tiempo (representada por el campo *next_available_time*), las variables contadoras (campo *number_current_client*) y las variables de estado del sistema (campos *offers*, *budget* y *queue*). En el caso específico de los agentes de destino, estos contituyen un sistema de atención de un solo servidor, además, los mismos pueden ser vistos a su vez como un modelo de inventario, ya que internamente tiene el seguimiento de un conjunto de productos, los cuales son ofertados a un precio determinado por unidad.

2.3.4. Implementación de los Agentes:

En este proyecto se cuenta con la implementación de dos tipos de agentes, *HumanAgent* y *DestinationAgent*, de los cuales ya se habló en la sección 2.1.1. Ahora, resulta importante describir con más detalle las características principales de dichos agentes, la forma de ejecución de sus funcionalidades principales y las interacciones entre estos. Primeramente, cabe destacar que ambos son agentes con estados, ya que los mismos tienen un conocimiento de ciertas variables producto de un conjunto de iteraciones realizadas en el ambiente, por ejemplo, los agentes humanos en cada momento de la simulación conocen el dinero que poseen, las necesidades que les quedan por satisfacer y la posición donde se encuentran en el ambiente, mientras que por otro lado, los agentes de destino conocen el estado de las ofertas que tienen disponibles y el tiempo transcurrido desde que comenzaron a ejecutarse. Añadido a lo anterior, ambos agentes tienen la capacidad de almacenar un conjunto de acciones realizadas por estos en un

momento determinado, por lo cual es posible acceder al historial de acciones de estos.

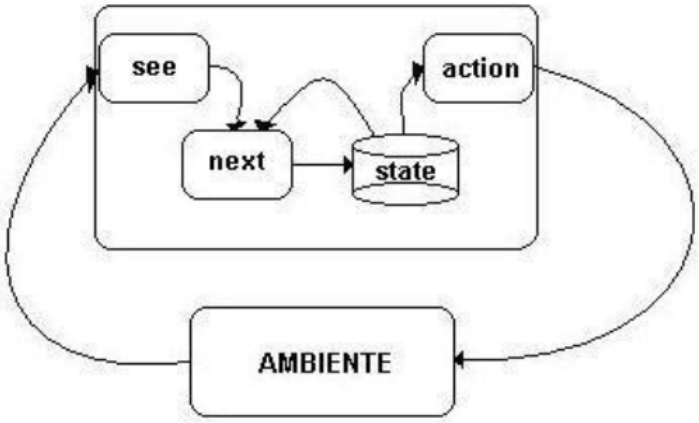

FIGURA 6. Arquitectura genérica de los agentes con estados.

En el caso de los agentes humanos, estos pueden ser clasificados como proactivos de razonamiento práctico, ya que tienen la funcionalidad de decidir, momento a momento, qué acción realizar para lograr sus objetivos (en este caso, el conjunto de necesidades que poseen). Estos agentes pueden estar solamente en tres estados para la toma de decisiones: en sus hogares (inicio de la simulación), después de haber realizado una compra en un destino o haber arribado a uno de estos. En el primer caso, la acción la lógica del agente consiste en recibir el estado actual del entorno y determinar el destino al cuál debería ir para satisfacer sus necesidad de acuerdo al criterio que este siga. En el segundo caso, el agente decidirá exactamente lo mismo que en el primero, aunque si ocurre que se haya quedado sin dinero o ya haya visitado todos los destinos posibles, este terminaría su ejecución. Finalmente, en el tercer caso, el agente recibiría del entorno el estado actual del destino en el que se encuentre, del cual podrá conocer si al momento de su llegada el mismo está lleno o ya su horario de funcionamiento expiró, en cuyo caso tratará de dirigirse a otro destino, mientras que en otro caso, este intentará satisfacer sus necesidades en dicho centro.

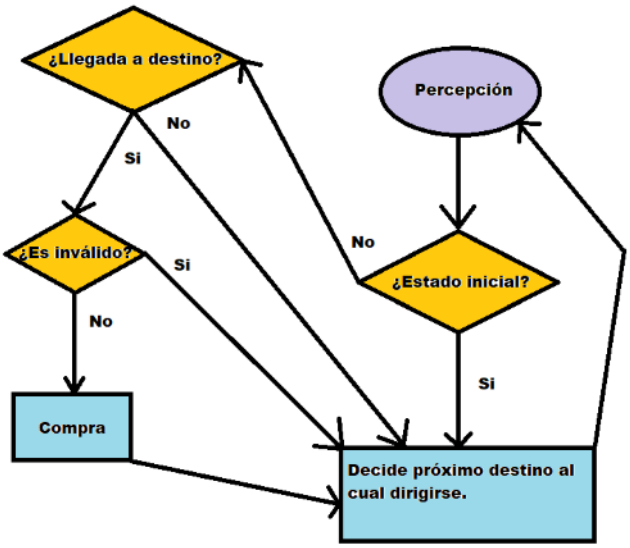

FIGURA 7. Algoritmo de comportamiento de los agentes humanos.

Por otro lado, los agentes de destino reaccionan a peticiones que reciben de compra de artículos o a la llegada de un agente humano al mismo, por lo tanto pueden ser considerados como agentes reactivos. En el primer caso, estos tratarán de validad la compra y actualizar el estado interno del humano, mientras que en el segundo caso estos tratan de encolar al agente humano en su sistema interno de atención.

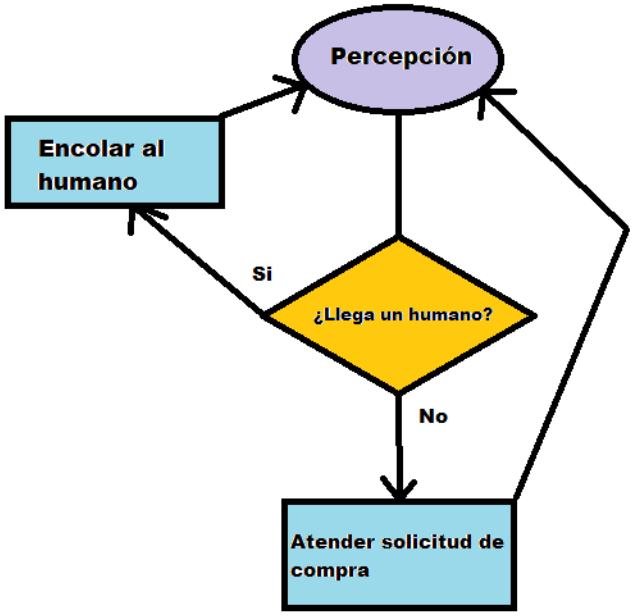

FIGURA 8. Algoritmo de comportamiento de los agentes de destino.

2.3.5. Sistema multiagente, comunicación e interacción:

El sistema que se modela en este proyecto contituye evidentemente un sistema multiagente [14], ya que el mismo está constituído por un conjunto de agentes (agentes humanos y de destino), cada uno con determinadas capacidades (moverse, negociar, realizar compras, responder a solicitudes de compras) y recursos computacionales (necesidades, dinero, ofertas) para la solución de una clase de problemas o tareas (compra y venta de productos). El objetivo de este sistema es la solución de problemas inherentemente distribuídos, donde los agentes que intervienen en el mismo deben interactuar entre ellos (negociación entre agentes humanos y de destino) de forma tal que esto conduzca a la solución de dichos problemas.

El principal proceso de interacción entre los agentes humanos y de destino es la negociación. En este, los agentes humanos revisan las ofertas que posee el destino, y realizan una política específica para seleccionar aquellas ofertas que mejor satisfacen su estado actual en base a algún criterio específico. Posteriormente, los agentes de destino reciben dicha solicitud de compra y actualizan, en caso de que esta sea válida, el estado interno de sus ofertas. Finalmente, una vez terminado este proceso, el agente humano trata de determinar el próximo destino al cual dirigirse.

2.4. Ejecución de la Simulación:

La ejecución de la simulación de un entorno es controlada internamente por el campo *schedule* en la clase *Environment*. Dicho campo es una cola de prioridad que almacena las acciones que deben ocurrir durante la simulación. Estas acciones son tuplas de la forma *(tiempo, humano, acción, destino)*, donde *tiempo* representa el momento en que debe ejecutarse la acción, *humano* es el agente humano que interviene en la acción, *acción* es el suceso que debe ejecutarse y *destino* es el agente de destino que interactuará con el agente humano.

Para la simulación se tienen implementadas dos acciones principales, las cuales determinan los cambios de estado en el sistema que se está tratando. Estas son: $arrival$ (encargada de simular el proceso en que un agente humano llega a un agente de destino y se decide si incluirlo en la cola o decidir el próximo destino al cual viajar) y $negotiation$

(simula la situación en que el cliente debe ser atendido por el agente de destino y comienza la fase de negociación).

En el momento inicial de la simulación cada uno de los agentes humanos está en su hogar, por lo que la primera acción de todos los agentes sería determinar el primer destino al cual dirigirse y luego, el propio sistema se encarga de colocar en la cola de prioridad general dicha acción como un $arrival$, donde el tiempo en que se ejecutaría sería el tiempo en que demora el agente humano en moverse desde su posición actual hasta dicho destino teniendo en cuenta su velocidad y la distancia a recorrer. En un primer momento, todas las acciones son desplazamientos de los agentes humanos hacia centros de destino. La cola de prioridad interna toma el primero de dichos arribos y lo ejecuta. Cada acción de arribo puede generar una acción de compra ($negotiation$) en caso de que el agente decida hacer la cola en dicho centro. En caso contrario se generaría una acción de arribo a otro destino de ser posible. Luego, las acciones de compra son las encargadas de simular toda la lógica de adquisición de productos por parte del humano y una vez terminado este suceso, se procede a generar una acción de arribo de dicho cliente a otro posible destino.

```
def run(self, time_step=10):
    while self.schedule.qsize() > 0 and self.total_time_elapsed < self.simulation_duration:
        self.execute(time_step)
    for human_agent in self.human_agents:
        if not (human_agent in self.dsat_list.keys()):
            self.dsat_list[human_agent] = human_agent.dissatisfaction(
                self.total_time_elapsed)
```

FIGURA 9. Algoritmo de ejecución del entorno.

En la figura 9 se muestra la implementación en *Python* de la ejecución del entorno. Básicamente, dicho método consiste en ejecutar las acciones en la cola de prioridad de acciones mientras no se cumplan los criterios de parada establecidos.

Una vez descrita la forma de ejecución de la clase *Environment*, solo quedaría analizar la ejecución de la clase *Experiment*. Esta última es la encargada de obtener una serie de conclusiones interesantes de los entornos realizando siempre un mínimo de 30 simulaciones. Para inicializar un experimento solo es necesario pasarle a su constructor un entorno ya definido, el cual será la base del mismo. Como se explicó en la sección

2.1.2, esta clase es utilizada para modelar los experimentos basados en diversos criterios de optimización. Para cada uno de los criterios mencionados se ejecuta el algoritmo *Hill Climbing* ya mencionado en 2.2.3. En total dicha clase cuenta con cuatro métodos de optimización, uno por cada factor definido en 2.1.2.

function HILL-CLIMBING(*problem*) **returns** a state that is a local maximum

current ← MAKE-NODE(*problem*.INITIAL-STATE)
loop do
neighbor ← a highest-valued successor of *current*
if neighbor.VALUE ≤ current.VALUE **then return** *current*.STATE
current ← *neighbor*

FIGURA 10. Pseudocódigo al algoritmo *Hill Climbing*.

2.5. Resultados de la Simulación:

Para la ejecución de la simulación primeramente crearemos una instancia de la clase *Environment* con los siguientes parámetros:

```
number_human_agents = 3
number_destination_agents = 2
number_needs = 6
simulation_duration = 100000
gini_coef = 0.2
mean_income = 400
human_needs_density = [0.1, 0.1, 0.1, 2, 2, 2]
offers_average_price = [50, 50, 50, 50, 50, 50]
store_offers_density = [1, 1, 1, 1, 1, 1]
stores_total_budget = 1800
store_distribution = [0.5, 0.5]
```

FIGURA 11. Parámetros para la ejecución del entorno.

El parámetro *gini_coef* es sencillamente el coeficiente de Gini [12] de la ciudad o región analizada. El coeficiente de Gini es una medida de la desigualdad de una población, indicada como la diferencia entre el área de la curva de la igualdad total y el área de la curva de la distribución real de los ingresos. Debe ser un valor real entre 0 y 1, sin incluir el 1. En el caso de *mean_income*, esta es la media de los ingresos de la región estudiada. Se usa junto al coeficiente de Gini para modelar los ingresos del área analizada.
El valor *offers_average_price* es la media de los precios del producto que suple cierta necesidad en la región estudiada. Para cada necesidad, en su índice específico, se tiene el

valor promedio del costo de satisfacer una unidad de esa necesidad. Este dato es almacenado en una lista de valores reales de tamaño igual a la cantidad de necesidades.
Los parámetros *human_needs_density* y *store_offers_density* son medidores de la proporción de una necesidad/producto en una persona/tienda. El primero se refiere a la proporción entre la media de la generación de una necesidad en específico en la población estudiada y la media global. Por citar ejemplos, en una población de Cuba se consume mucho más arroz en promedio que a nivel del hemisferio occidental o a nivel mundial o en una población de EE.UU. se consume mucho más papel sanitario en promedio que a nivel mundial. Por otro lado, *store_offers_density* es una proporción entre los propios productos de un agente destino, significando que el valor esperado de la proporción entre las existencias de un par de productos es igual a la proporción entre los valores de las densidades asociadas. Ambas variables son listas de valores reales de tamaño igual a la cantidad de necesidades.
Finalmente, *stores_total_budget* y *store_distribution* son los parámetros que caracterizan el presupuesto de todos los agentes destino del área estudiada. El primero indica el presupuesto global con el que se compran los productos que se encuentran disponibles a comprar por los agentes humanos, y el segundo indica la distribución de este presupuesto entre todos los agentes destino. El parámetro *stores_total_budget* es un valor real y *store_distribution* es una lista de valores reales de tamaño igual a la cantidad de agentes destino.
Como parte de la simulación de instancias de la clase *Environment*, se realizaron 10 pruebas, donde en cada una se ejecutaron unas 30 simulaciones sobre entornos distintos. En las figuras 12 y 13 se muestra el comportamiento de la media y la varianza de la insatisfacción de los agentes humanos.

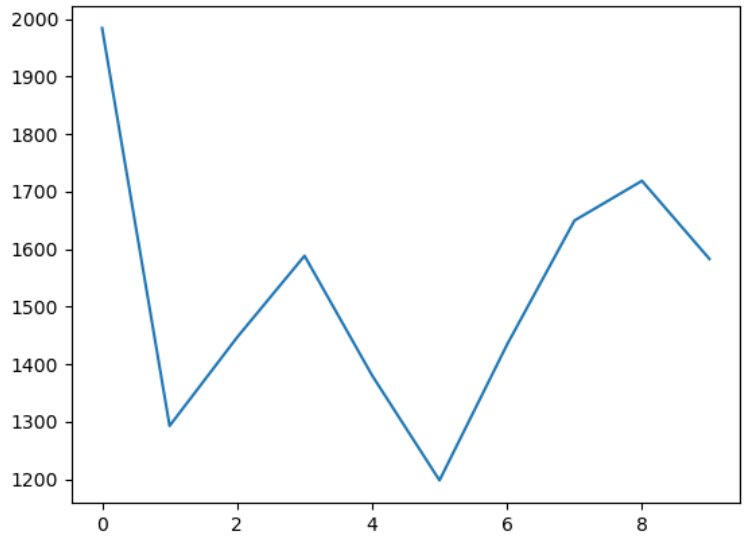

FIGURA 12. Comportamiento de la media de la insatisfacción de los agentes humanos por cada iteración.

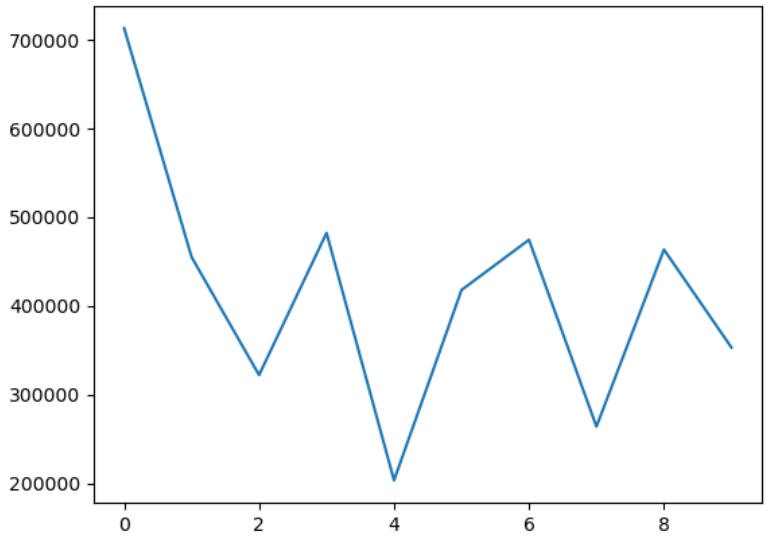

FIGURA 13. Comportamiento de la media de la insatisfacción de los agentes humanos por cada iteración.

2.6. Ejecución de los experimentos:

Luego de comprobar la ejecución de varias instancias de la clase *Environment*, procederemos a inicializar un objeto de tipo *Experiment* con el cual podamos realizar las optimizaciones propuestas en este trabajo. Para cada factor de optimización previamente establecido 2.1.2, el algoritmo *Hill Climbing* optimiza sobre un parámetro específico, donde se realizan 40 iteraciones y en cada una se ejecuta 40 veces el entorno:

1. STORE_OFFERS_DENSITY: En este caso, el valor inicial de *store_offers_density* fue $[1, 1, 1, 1, 1, 1]$ con evaluación $1330{,}30$. La media y la varianza obtenidas para la insatisfacción fueron $1330{,}30$ y $317205{,}72$. Al finalizar la optimización, el parámetro estudiado terminó con el valor $[1{,}33, 1{,}60, 2{,}03, 0{,}17, 0{,}08, 0{,}76]$ con una evaluación de $1238{,}58$ de media de insatisfacción.
2. OFFERS_PRICE_FACTOR: El parámetro *store_offers_density* comienza con valor 1 y un factor de penalización de $2{,}71$. Al finalizar la optimización, *store_offers_density* valía $4{,}73$ mientras que la penalización era de $1{,}23$. Es importante recordar que este factor hace referencia al valor que se multiplica al precio base de los productos. Como se observa en los resultados, al aumentar el costo de los productos aumenta la insatisfacción, pero la función de penalidad implementada disminuye su valor, de esta forma se explican los resultados obtenidos.
3. TOTAL_BUDGET_FACTOR: Es conocido que este factor hace referencia al multiplicador del presupuesto total de los agentes destino. Este comenzó con valor 1 y un factor de penalización de $2{,}71$. Al finalizar la optimización, el parámetro tenía un valor de $0{,}086$ mientras que la penalización era de $1{,}09$. Como se observa en los resultados, al disminuir la oferta de productos aumenta la insatisfacción, pero la función de penalidad implementada disminuye su valor, de esta forma se explican los resultados obtenidos.
4. STORE_DISTRIBUTION: La media y la varianza obtenidas para la insatisfacción fueron $1390{,}97$ y $254075{,}39$. El estado inicial fue establecido con el valor $[0{,}5, 0{,}5]$ para el parámetro *store_distribution*, con una insatisfacción de $1390{,}97$, mientras

que el estado final estableció que el valor óptimo de *store_distribution* es $[0{,}5, 0{,}5]$, con una insatisfacción de $1262{,}98$.

3. Conclusiones

Con la realización de este proyecto se logró diseñar un modelo capaz de simular un entorno donde un conjunto de personas se traslada hacia destinos posibles con el fin de satisfacer sus necesidades. Además, gracias a los resultados del análisis estadístico sobre la insatisfacción de los agentes humanos, se pudo probar la calidad de la implementación de los entornos. Finalmente, con la ejecución de los experimentos, se demostró la posibilidad de optimizar factores como la distribución de ofertas en las tiendas y el gasto del presupuesto inicial con una complejidad computacional asumible.

4. Recomendaciones

Se recomienda continuar con el estudio de los procesos relacionados con el movimiento de personas en ciudades con el objetivo de satisfacer necesidades. Resulta de interés ampliar el estudio de este campo a situaciones en las que intervengan la propagación de enfermedades, sistemas de transporte, reabastecimiento de los centros durante el día, simulación de entornos basados en días consecutivos, diseños de sistemas expertos que sirvan de oráculos para la toma de decisiones de los agentes e incluso la aplicación de estos resultados a modelos de escala regional, lo cual traería consigo significativos resultados para empresas, instituciones y gobiernos encargados de diversos centros orientados a la atención de clientes.

Referencias

[1] Michael Jaros, Monika Di Angelo, Peter Ferschin. *Modeling and Simulation of Pedestrian Behaviour As Planning Support for Building Design*. Journal of Economic Interaction and Coordination

[2] Paul M. Torrens. *Agent models of customer journeys on retail high streets*. The Conference on Pedestrian and Evacuation Dynamics 2014 (PED2014).

[3] Aloys W.J. Borgers, I.M.E. Smeets, A.D.A.M. Kemperman, and H.J.P. Timmermans. *Simulation of Micro Pedestrian Behaviour in Shopping Streets*. Urban planning Group, Eindhoven University of Technology, The Netherlands

[4] Jan Dijkstra, Harry Timmermans, Joran Jessurun. *Modeling planned and unplanned store visits within a framework for pedestrian movement simulation*. The Conference on Pedestrian and Evacuation Dynamics 2014 (PED2014).

[5] Tilman A. Schenk, Günter Löffler, Jürgen Rauh. *Agent-based simulation of consumer behavior in grocery shopping on a regional level*. Journal of Business Research 60 (2007) 894 – 903

[6] Python. URL: https://www.python.org/. Consultado en 1 de enero de 2023.

[7] Gladys Bonilla Enríquez, Diana Sánchez Partida, Santiago Omar Caballer Morales. *Algoritmo genético para el problema logístico de asignación de la mochila (Knapsack Problem)*. Universidad Popular Autónoma del Estado de Puebla A.C., Puebla, Puebla, México

[8] Wikipedia. URL: https://es.wikipedia.org/wiki/Algoritmo_hill_climbing. Consultado en 1 de enero de 2023.

[9] Gunter Dueck, Tobias Scheuer. *Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing*. Journal of Computational Physics, Volume 90, Issue 1, September 1990, Pages 161-175

[10] Medium. URL: https://medium.com/@PuchkovOleg/name-generator-tutorial-1c9a3dfb2ab0. Consultado en 1 de enero de 2023.

[11] Gallup. URL: https://news.gallup.com/poll/166211/worldwide-median-household-income-000.aspx. Consultado en 1 de enero de 2023.

[12] Thitithep Sitthiyot, Kanyarat Holasut. *A simple method for estimating the Lorenz curve*. Humanities and Social Sciences Communications volume 8, Article number: 268 (2021)

[13] Amir Sohrab Sahaleh, Michel Bierlaire, Bilal Farooq, Antonin Danalet, Flurin Silvan Hänseler . *Scenario Analysis of Pedestrian Flow in Public Spaces*. School of Architecture, Civil and Environmental Engineering - ENAC, Mayo 2012

[14] Luciano García, Luis Martí, Luis Orosa. *Temas de Simulación*. Facultad de Matemática y Computación. Universidad de la Habana. Página 87.