Proyecto Integrador
Compilación + Simulación + Inteligencia Artificial
Garage-Sale

Frank Adrian Pérez Morales
C31

Camilo Rodríguez Veláquez
C32

Sergio Pantoja
C31

Índice general

1. Resumen del Problema

Las ventas de garaje son una nueva modalidad de tiendas que se están haciendo muy popular en Cuba, debido a los altos precios de los artículos importados. Un ejemplo de estas es la ubicada en 17 entre 2 y 4 la cual tiene como principal objetivo vender a la población artículos nuevos o de segunda mano con la mejor calidad posible para la satisfacción de todos sus clientes.

Esta tienda cuenta con varios empleados que realizan diferentes tipos de servicios como: el portero el cual es el encargado de la seguridad del local y de guardar los bolsos o mochilas que los clientes pueden tener, el vendedor que atiende a los clientes para orientarlos mejor acerca de los productos en venta y el cajero, encargado de los pagos a realizar por los clientes y de la contabilidad general.

El local es bastante amplio por lo tanto existen diversas formas de distribuir los productos en venta. Por ejemplo: los artículos en exhibición, los cuales son los que están a la vista de cualquier transeúnte que pase por los alrededores del local y se puedan interesar por alguno de estos o por ver q más ofrecen dentro, las prendas que se exhiben en los colgadores los cuales dan una mejor vista a los clientes de que forma son estos artículos y como les pudiera quedar disminuyendo el tiempo de estancia de estos en el establecimiento, y los productos de menor visibilidad los cuales se deben doblar para que ocupen menos espacio, el vendedor tiene que enseñarle a los clientes cual es la forma de estos. También se cuenta con un vestidor en donde los clientes se pueden hacer pruebas de los productos.

Debido a las medidas higiénicas que se deben tener solo pueden entrar al local una capacidad limitada de clientes, lo que hace que se creen colas fuera del local, lo cual genera inconformidad en los clientes debido a que esta produce una demora significativa que en algunos casos les obliga a abandonar su inteción de compra.

No se tiene a disposición ninguna información que permita calcular la cantidad de clientes que llegan a cierta hora del día, tampoco que cantidad de personal debe atender en cada momento del día para mantener la menor cantidad de personas en la cola. Como riesgo principal ante esta situación se puede identificar que si el tiempo de espera en la cola por parte de los clientes es excesivo, estos podrían optar para en una próxima oportunidad no ir a la tienda, es decir, se estaría perdiendo al cliente debido a una mala estrategia de atención.

El precio de cada artículo varía según su calidad. Se sabe que en el local deben existir tanto artículos colgados como artículos doblados debido a que la cantidad de mercancía es demasiado grande y se requiere que toda esta se le exhiba al cliente.

Se desea simular el proceso de las ventas, variando las formas de organizar los productos y también a los empleados para minimizar el tiempo de espera del cliente en las colas y para maximizar la ganancia total de la tienda.

2. Compilación

La compilación se divide en tres procesos:

- **Análisis sintáctico**: Se convierte el código fuente en una representación de un árbol para el posterior análisis. Cuenta de dos partes el análisis léxico, el cual es la primera fase del compilador y se apoya en expresiones regulares para machear el tipo de token devueltos por el tokenizador.

  Este proceso desglosa el código en una serie de tokens y se desechan todos los saltos de línea y los espacios en blancos a través de la asociación de una secuencia de caracteres con un tipo token con el uso de expresiones regulares que se generan mediante el parser según **Early's Parser Algorithm**.

  Los tokens resultantes almacenan el texto ,el tipo, la columna donde empieza, donde comienzan y terminan en el string contenido y la línea en donde está para luego si se encuentran errores en un token en específico poder saber en dónde se encuentra este. Luego de concluido el análisis se envía al parser, una lista de tokens, el cual lo analizará.

  La segunda parte del análisis sitáctico es la relacionada con el parser. En esta fase también utilizamos el **Early's Parser Algorithm** al cual le pasaremos la gramática y la lista de tokens obtenida anteriormente para generar el **AST** del programa.

- **Análisis semántico**: Para poder hacer el chequeo semántico se utiliza el patrón visitors para recorrer todos los nodos del ast.
  El chequeo semántico se divide en tres fases: TypeCollector: se agrega los tipos builtin del DSL, así como los atributos y los métodos de cadad uno.

TypeBuilder: se buscan todas las declaraciones de variables y funciones, y se guardan en el contexto principal su tipo correspondiente y en caso de ser una función la cantidad de argumentos y los tipos de estos.

La última fase del chequeo semántico es el TypeChecker en donde se comprueba que las variables y las funciones solo pueden ser utilizadas cuando estén declaradas en su contexto y también que los tipos declarados de estas correspondan con los tipos de su expresión asignada, para esto, siempre que una variable o función sean declaradas chequeamos que no se encuentre declarada en el contexto actual o en algunos de sus contextos padres, es en caso de la asignadas o en el caso de la función invocadas chequeamos que estas se encuentren declaradas y q el tipo de su declaración corresponda con el tipo que se le esta asignando utilizando el patrón visitors para chequear los nodos del ast, donde cada nodo va a saber cual es su tipo cuando sepa cual es el tipo de sus nodos hijos hasta q se llegue al nodo hoja que tendrá el tipo literal.

Si se encuentra algún error se añadirá a la lista de errores y por lo cual no se ejecutara el programa.

- **Transpilación**: Luego de que se realiza el análisis semántico se transpila el código resultante al lenguaje Python el cual fue el utilizado en el proyecto.

3. Simulación

La simulación se basa en un sistema de colas donde existen 2 tipos de empleados en serie y cada tipo de empleado tiene m y n cantidades de empleados en paralelo respectivamente. Los primeros empleados por los que deben pasar los clientes son los servidores y luego por los cajeros. Cada servidor y cajero tiene una cola en la cual estarán los clientes que el empleado deberá de atender.

Se simulará en un período de dos horas (7200 segundos) que varían según lo eventos que pueden pasar en la simulación. La cantidad máxima de clientes que pueden entrar a la tienda es 500. Cada cliente llega a la tienda en tiempos diferentes dada una **Distribución Beta** dado el tiempo total de la simulación.

Agentes del Sistema:

- Cliente: Es el agente más importante del sistema. Interactúa con los demás agentes que son los empleados de la tienda. Cada cliente tendrá un comportamiento acerca de cómo seleccionar al empleado que lo va a atender , también tiene un nivel de tolerancia, que según la cantidad de personas en la cola del empleado seleccionado para que lo atienda, el cliente decidirá si irse o no.

- Servidor : Es el agente por el cual primero pasará cada cliente, es el encargado de atender al cliente para que seleccione el producto a comprar. Su tiempo de servicio es el más elevado entre los empleados del local.

- Cajero: Es el agente encargado de la recepción de los pagos y el cual marca el final de la estancia en la tienda para el cliente, es decir, cuando el cliente realiza el pago al cajero se automáticamente este cliente saldrá de la tienda.

Cada empleado tiene un tiempo de servicio el cual está dado por un **Distribución de Poisson**, por simplicidad y para asegurarnos que obtengamos números discretos no negativos.

Los servidores , los cuales son los de más tiempo de servicio poseen tienen un tiempo aproximado de 160 segundos mientras que los cajeros tienen 100 segundos para operar.

Cada cliente tiene una cantidad de tolerancia la cual es la cantidad de personas máximas que deben de estar en la cola a la cual se va a unir para no irse de la tienda. Esto puede pasar tanto como en las colas de los servidores, como en las colas de los cajeros.

Para que la tienda pueda funcionar el jefe debe pagarle una cantidad diaria de dinero a cada empleado. Por cada cliente que salga de la tienda completamente atendido (haya pasado por un servidor y por un cajero) se obtiene una cantidad de dinero. EL objetivo de la simulación es: según las variaciones entre la cantidad de servidores y cajeros, en cual de estas se obtiene mayor ganancia , la cual es el dinero generado por la cantidad de personas que han sido atendidas menos la cantidad fija a pagar en total por los trabajadores.

Cada simulación también va a tener eventos aleatorios los cuales actualizan varios aspectos de la simulación de la tienda para saber si la variación idónea resultante se verá afectada o no con estas situaciones extremas .

Además de la ganancia total de la tienda también se evalúan otras métricas las cuales tienen relación con el resultado final de la simulación, como la cantidad total de personas que entraron a la tienda, la cantidad total de personas que se fueron de la tienda debido a que las colas no tenían la cantidad deseada según su nivel de tolerancia, entre otras .

4. Inteligencia Artificial

Se utiliza un Algoritmo Genético para poder tener una población de simulaciones y entre todas estas poder seleccionar a los mejores individuos según sus mejores ganancias.

Se crea un población de Individuos Aleatorios los cuales tienen una cantidad aleatoria de clientes y empleados para tener una base de individuos. Estos se irán actualizando para poder llegar a los de mejores ganancias.

Estos Individuos dada la iteración en que esté el algoritmo se convertián en Individuos Padres. Se seleccionará el 60 % de los mejores Individuos Padres lo cual está dado por los valores de la ganancia resultante de estos. Estos Individuos Padres van a mezclarse según una probabilidad para poder crear una población de igual tamaño que la inicial, pero de Individuos Hijos, los cuales son una mezcla de un par de Individuos Padres Resultantes.

Cuando se crean los Individuos Hijos también existe la posibilidad que estos muten, aunque es muy baja, al mutar no va a tomar algunos "genes"de los Individuos Padres sino que estos genes van a generarse de forma aleatoria .

Estos Individuos Hijos se convertirán nuevamente en Individuos Padres de los cuales se produción más hijos, este proceso se repetiá según la cantidad de iteraciones que el usuario desee, a mayor cantidad de iteraciones mayor va a ser el tiempo de demora del proceso pero el resultado va a ser mejor , debido a que va a abarcar una amplia gama de posibles individuos.

5. Lenguaje

EL lenguaje es un lenguaje de programación de tipado estático el cual contiene varios objetos predeterminados para facilitar nuestra programación. :

Esta contiene varios tipos de variables:

- `int`
- `float`
- `string`
- `void`
- `list`
- `Agent`
- `Customer`
- `Worker`
- `Server`
- `Cashier`
- `GarageSaleModel`
- `MakeSim`

- Para crear funciones primero se declara el tipo , luego el nombre de la función, luego paréntesis que deben contener argumentos (los cuales son opcionales) seguidos por llaves., donde adentro se define el cuerpo de la función que de no ser tipo void la función tiene que tener un valor de retorno.

- Para crear variables primero se declara el tipo, el nombre , luego se declara con := y la expresión a la cual va a ser igualada.

- Todo statement del lenguaje debe tener con punto y coma excepto cuando declaras el cuerpo de una función, un if o un else (en donde se utilizan llaves).

- Para la asignación se procede de igual manera pero sin declarar un tipo, debido a que sino sería una declaración lo cual da un error de contexto,

- A una variable le puedes asignar cualquier tipo de expresión no importa si es una asignación o una declaración siempre y cuando tenga el tipo a asignar, ya sea una función , una llamada a un tipo definido o una llamada a una lista.

- Se declara igual que una variable a diferencia de que estas tendrán un pareja de corchetes. Los elementos dentro de estas pueden ser de cualquier tipo pero todos tienen que ser del mismo tipo, separados por coma.

- Se puede indexar tanto como en listas declaradas o en funciones o métodos que retornen listas. El índice a la hora de indexar siempre debe de ser un elemento de tipo `int` o una función o método que retorne un elemento del mismo tipo `int`.

- Existen expresiones condicionales `if - else` que se declaran; if(condición) y de ser esta evaluada verdadera, ejecutará el código contenido dentro del cuerpo de este que va a estar contenido entre llaves. De evaluar falso se ejecutará el cuerpo del else (el cual es opcional) que igual va a estar contenido entre llaves.

- El statement while es parecido al anterior lo que este ejecutará el código contenido hasta que la condición sea falsa. Su código va a estar separado entre llaves.

- Se puede acceder a los métodos y atributos de los tipos declarados pero solo para chequearlos, no para modificarlos.