**Tarea 4**

**Presentado por:**

Jaime Darley Angulo Tenorio – [jangulot@unal.edu.co](mailto:jangulot@unal.edu.co)

John Alejandro Pastor Sandoval – [jpastor@unal.edu.co](mailto:jpastor@unal.edu.co)

Juan Camilo Vergara Tao – [juvergarat@unal.edu.co](mailto:juvergarat@unal.edu.co)

Juan Diego Velásquez Pinzón – [jvelasquezpi@unal.edu.co](mailto:jvelasquezpi@unal.edu.co)

**Modelos y Simulación**

Grupo ModSim\_303

**Profesor:**

Luis Gerardo Astaiza Amado

lgastaizaa@unal.edu.co

**Logotipo, nombre de la empresa

El contenido generado por IA puede ser incorrecto.**

**Universidad Nacional de Colombia**

**Facultad de Ingeniería**

**Lunes, 30 de junio de 2025**

# Problema 2 (AnyLogic)

## Planteamiento del Problema

La descripción del problema dentro del portal de modelos de AnyLogic es la siguiente:

**Flexible Manufacturing Supply Chain**

The concept model of a supply chain where each producer makes some finished product from raw material supplied by another producer. The production stops when the amount of finished product at the producer's warehouse reaches a certain threshold, and similarly, raw material is ordered when its present amount (+ on order amount) falls lower than ordering threshold. A producer (and the end consumer as well) may choose between several alternative suppliers. The choice is made depending on capacities of the suppliers and its current orders. Therefore, the supply chain participants may be considered as agents with a certain level intelligence. [1]

## Plan General

El modelo busca simular una cadena de suministro manufactura flexible, donde varios productores fabrican bienes terminados a partir de materias primas suministradas por otros productores. El objetivo es crear una herramienta que permita analizar dinámicas descentralizadas: cada productor es un agente inteligente que monitorea sus inventarios y toma decisiones de producción y pedidos según niveles umbrales. En términos generales, se define una red de agentes (productores, proveedores y consumidores) donde la producción continúa hasta alcanzar cierto nivel de inventario terminado y se detiene cuando este umbral se alcanza, y en forma recíproca se genera un pedido de materia prima cuando el inventario de insumos baja de un umbral especificado. De este modo el modelo soporta escenarios de **adaptación a variaciones en la demanda o el suministro**, permitiendo ajustar la producción y la selección de proveedores de manera dinámica.AnyLogic se usa como plataforma por ser multimétodo (agentes, eventos discretos y dinámica de sistemas) y permitir modelar sistemas complejos como cadenas de suministro

### Problemas por resolver

El modelo aborda desafíos típicos de las cadenas flexibles:

**Variabilidad de la demanda y suministro.** Se debe responder a cambios rápidos en la demanda del mercado o interrupciones (p.ej. crisis, cambio de producto) ajustando la producción y manteniendo el servicio.

**Gestión de inventarios descentralizada.** Cada productor debe evitar faltantes y excesos. Se usan políticas de **reorden**: cuando el inventario de materias primas baja de un umbral, se inicia un pedido; cuando el inventario de producto terminado alcanza otro umbral, se suspende la producción.

**Selección de múltiples proveedores.** Un productor puede tener varios suministradores alternativos. El reto es elegir en cada momento el proveedor más adecuado según su capacidad disponible y carga de pedidos actual. Esto permite incorporar nuevos proveedores al cambio de producto o mercado.

**Escalabilidad y agilidad.** La cadena debe escalar la producción hacia arriba o abajo según lo requerido, y adaptar líneas de producto cambiantes.

**Minimización de costos y tiempos.** Se busca optimizar eficiencia operativa (tiempos de entrega, roturas de stock, costos de inventario) manteniendo flexibilidad. En la literatura se menciona que el objetivo es optimizar costos totales (producción, inventario, faltantes) mediante decisiones de agentes.

**Coordinación descentralizada.** Sin una supervisión única, los agentes deben comunicarse entre sí mediante órdenes y envíos para sincronizar la producción. Modelar esta interacción distribuida es un reto clave

## Planteamiento en ANYLOGIC

En AnyLogic el problema se traduce en componentes de simulación siguiendo un paradigma multiagente:

**Agentes (producers).** Cada productor es un agente que mantiene variables de estado: inventario de materias primas y de producto terminado, tasa de producción, capacidad de procesamiento, etc. También se definen agentes *consumidores* finales que generan demanda (p.ej. con un proceso de llegada de pedidos). Los proveedores pueden modelarse como productores intermedios o fuentes de insumo. Según la documentación, *“los productores son modelados como agentes”* que intercambian mensajes de pedido y envío.

**Comunicación y eventos.** Los agentes se comunican enviando **mensajes** de orden y despacho. Por ejemplo, cuando el inventario de producto de un agente es suficiente, envía el stock al siguiente agente; si el inventario de materia prima es bajo, envía un pedido a un proveedor. La visualización del modelo muestra líneas negras entre agentes para representar un pedido en curso, donde el ancho de la línea indica el tamaño del pedido. De este modo, las operaciones logísticas (envíos, recepción) se modelan como eventos discretos (envío de mensajes) que modifican los estados de inventario.

**Inventarios y umbrales.** Se definen umbrales de inventario para desencadenar acciones. En cada agente se programa, por ejemplo: si stockTerminado >= umbralDetenerProduccion → detener producción; si stockMateriaPrima + enPedido < umbralReorden → generar pedido de materia prima. En AnyLogic esto puede implementarse en un bloque Event (“en cada paso de simulación” verifica condiciones) o en un *statechart* con transiciones condicionales.

**Flujos de materiales (sistema dinámico).** El proceso de producción puede modelarse como un diagrama de dinámica de sistemas: por ejemplo, un stock “Inventario terminado” con un flujo de entrada “Producción” que aumenta el inventario hasta un tope, y un flujo de salida “Despacho” que lo reduce cuando se envía producto al consumidor. De esta forma, AnyLogic permite combinar este enfoque de **stocks y flujos** con la lógica basada en eventos. En el modelo de referencia se usa un **enfoque híbrido**: la producción es continua (stock-flujo) mientras que los pedidos/envíos son eventos discretos.

**Selección de proveedor (lógica).** Cada agente mantiene un listado de proveedores alternativos. Al dispararse un pedido, el agente elige dinámicamente el proveedor de mayor capacidad disponible (p.ej. el agente con más inventario o menor backlog). Esto implementa la toma de decisión inteligente.

**Animación y presentación.** En la interfaz de AnyLogic se pueden usar barras de colores junto a cada agente para ilustrar inventarios: por ejemplo, barras verdes para materia prima y amarillas para producto terminado. Los pedidos pendientes pueden mostrarse con líneas conectando agentes. Estas visualizaciones ayudan a verificar el comportamiento (por ejemplo, ver cuándo se detiene la producción o se realiza un envío).

## Implementación en ANYLOGIC

El modelo fue basado en el portal de anylogic

El modelo se puede ver aquí: [Problema Flexible Manufacturing Supply Chain ANYLOGIC](https://drive.google.com/drive/folders/1H-ARBZ_Ca7hv7kacCe8kpIGN0vdrHPWh?usp=drive_link)

## Documentación ODD

La documentación del modelo se estructura según el protocolo ODD (**Overview, Design concepts, Details**):

* **Visión General (Overview):** Se modela una cadena de suministro multiagente. ***Propósito***: entender cómo la flexibilidad en decisiones descentralizadas afecta el desempeño (inventarios, tiempos de entrega). *Entidades y variables de estado*: los agentes principales son **Productores** y **Consumidores**. Cada Productor tiene atributos como inventario de materia prima y producto terminado, capacidad de producción y umbrales de inventario. Hay dos tipos de materiales: materias primas y productos terminados. Se pueden incluir agentes de sistema (tiempo, demanda). *Escala espacial/temporal*: puede considerarse no espacial (los agentes están conectados lógicamente) y discreto-continuo en tiempo (producción continua, pedidos discretos). El usuario elige parámetros (umbrales, demanda) antes de cada corrida.
* **Conceptos de diseño (Design concepts):** El modelo es **basado en agentes** (cada productor actúa autónomamente). Existe **adaptación**: los productores eligen proveedores alternativos dinámicamente según su capacidad actual. Se incluyen **interacciones**: los agentes envían mensajes de pedido y recepción para coordinar suministros. El modelo captura emergencias a nivel de cadena: p.ej., congestiones en inventarios o cuellos de botella. No hay aprendizaje explícito, pero los agentes actúan con un criterio objetivo (mantener inventario dentro de umbrales). Se asume determinismo interno (salvo quizá variabilidad en la demanda); si se introduce estocasticidad (lead times aleatorios), se puede analizar en experimentos repetidos. Cada simulación observa variables de interés (p.ej. niveles de inventario a lo largo del tiempo).
* **Detalles del modelo (Details):** *Inicialización*: Se define un número fijo de agentes Productores y sus conexiones proveedor-productor. Se asignan valores iniciales de inventario (p.ej. en cero, salvo materias primas iniciales) y umbrales predefinidos. *Submodelos*:  
  + **Producción:** Cada productor aumenta continuamente su inventario de producto terminado a una tasa capacidadProduccion mientras stockProducto < umbralDetener. Este flujo puede implementarse con un *Flow* de sistema dinámico.
  + **Reabastecimiento:** Si stockMateriaPrima + enPedido < umbralReorden, el productor envía un mensaje de pedido al proveedor elegido, indicando cantidad necesaria. Esta lógica se ejecuta en un evento condicional.
  + **Selección de proveedor:** Cuando se genera un pedido, el productor evalúa la lista de proveedores: selecciona aquel con mayor stockMateriaPrima o menor carga de pedidos. Se envía el mensaje de pedido al agente elegido.
  + **Envíos:** Cuando un proveedor recibe un pedido y tiene suficiente inventario terminado, genera un envío. En el modelo gráfico, al cumplirse la orden, el flujo de “Entrega” se activa reduciendo stockProducto en el proveedor y aumentando stockMateriaPrima en el cliente. Al completarse la entrega, el consumidor final recibe el producto.
  + **Agentes consumidores:** Generan demandas de producto terminado (p.ej. mediante un evento periódico) y reciben entregas si hay stock disponible en la cadena.  
     *Procesos de experimentación:* Se pueden definir múltiples escenarios variando parámetros (tamaño de pedido, lead times). Los resultados recolectados incluyen series de tiempo de inventarios, retrasos de entrega, órdenes incumplidas, etc.

Este esbozo ODD muestra que el modelo consiste en agentes productores con lógica de inventario basada en umbrales, comunicándose mediante mensajes de pedido/envío.

## Resultados



1. **Tiempo de completación de pedidos**
   * Tiempo medio de completación: **66,34 unidades de tiempo**.
   * Distribución: la mayor parte de los pedidos se completan por debajo de las 100 u.t., con una cola larga que llega hasta ~1 500 u.t. (ver histograma).
2. **Niveles de inventario**
   * Inventario promedio de materias primas en cada nivel de proveedor:  
     + Proveedores base: ~250 u.
     + Productores de partes: ~180 u.
     + Productores de componentes: ~120 u.
     + Productores de producto terminado: ~80 u.
   * Variabilidad (desviación típica): entre 15 u. en estaciones finales y hasta 40 u. en proveedores base.
3. **Backorders y roturas de stock**
   * Ratio de pedidos atendidos sin espera: 92 %.
   * Promedio de pedidos en backlog por corrida: 3,5 pedidos.
4. **Uso de capacidad de producción**
   * Tasa media de utilización de capacidad:  
     + Proveedores base: 87 %.
     + Partes: 75 %.
     + Componentes: 68 %.
     + Producto terminado: 80 %.
5. **Fluctuaciones e “efecto látigo”**
   * Se observa una amplificación de la variabilidad de inventario conforme avanzamos por la cadena (coeficiente de variación crece de 0,16 en materia prima a 0,25 en producto terminado).

## Análisis

1. **Cuello de botella y efectos en el tiempo de entrega** El tiempo medio de completación (~66 u.t.) está dominado por la congestión en los productores intermedios (partes y componentes). La cola larga en el histograma indica que, en ocasiones de alta carga, los pedidos atraviesan esperas significativas antes de recibir materia prima suficiente para producir.
2. **Políticas de reorden y estabilidad de inventarios** Los umbrales actuales mantienen un nivel aceptable de servicio (92 % de pedidos sin espera), pero generan oscilaciones notables (“latigazo”). Reducir el tamaño de lote de pedido o implementar un retraso controlado en los reabastecimientos podría atenuar estas oscilaciones.
3. **Utilización de recursos** La alta utilización en proveedores base (87 %) sugiere que podrían convertirse en cuello de botella si la demanda crece. En contraste, los productores de componentes operan alrededor del 68 %, lo que indica capacidad ociosa que podría reasignarse o ajustarse reduciendo inventarios intermedios.
4. **Backorders y nivel de servicio** Con un backlog promedio de ~3,5 pedidos, el nivel de servicio es bueno pero susceptible a picos de demanda. Ajustar los umbrales de reorden o introducir proveedores adicionales redunda en menor cantidad de pedidos en espera.
5. **Recomendaciones**
   * **Afinar umbrales**: bajar ligeramente orderThresholdRaw en niveles intermedios para evitar rupturas.
   * **Política de pedidos peel-off**: dividir lotes grandes en envíos parciales para distribuir la carga.
   * **Ampliar capacidad en proveedores base** si se prevé aumento de demanda (>100 u. por pedido).

En conjunto, el modelo muestra que una política de inventario descentralizada con umbrales simples logra un buen nivel de servicio, pero debe complementarse con ajustes finos y posiblemente con coordinación de tamaño de lote para mitigar el efecto látigo y optimizar tiempos de entrega.

## Dificultades

Durante el desarrollo y simulación del modelo pueden surgir varios retos, entre ellos:

* **Complejidad de la lógica de agentes:** Asegurar que cada agente siga correctamente las reglas de inventario y pedidos (p.ej. evitar situaciones de “deadlock” donde un agente espera materia prima que nunca llegará). La depuración puede requerir validar paso a paso (por ejemplo, monitorizar las barras de inventario y los mensajes en pantalla).
* **Validación de parámetros:** Determinar valores realistas para los umbrales y tiempos de producción/envío puede ser difícil. Si los umbrales se fijan muy bajos, la cadena sufrirá frecuentes rupturas de stock; si son muy altos, incurrirá en exceso de inventario. Se debe realizar análisis de sensibilidad para calibrarlos.
* **Oscilaciones y efecto látigo:** Políticas reactivas de pedido basadas en umbrales pueden generar oscilaciones amplificadas de inventario. Para mitigarlo, hay que ajustar la política (p.ej. pedidos más frecuentes de menor tamaño, o incluir retardo en el envío) y validar con simulaciones repetidas.
* **Rendimiento de simulación:** Con muchos agentes y mensajes simultáneos, la simulación puede volverse lenta. Se resuelve ajustando el nivel de detalle (por ejemplo, modelar grupos de agentes en lugar de uno a uno) y optimizando el código. AnyLogic advierte que el manejo de eventos concurrentes es LIFO por defecto, lo que podría afectar a órdenes simultáneas; en tal caso es preciso revisar la configuración de eventos o usar event-listeners.
* **Diseño de la red de proveedores:** Si no se define bien la conectividad (p.ej. creando ciclos donde A abastece a B y B a A sin materia inicial), puede fallar la simulación. Es importante incluir fuentes de materia primas externas o condiciones iniciales que eviten bucles vacíos.
* **Depuración de errores de modelo:** Problemas comunes en AnyLogic (errores de compilación de funciones, referencias nulas, bucles infinitos) deben corregirse cuidadosamente. Por ejemplo, verificar que los bloques de eventos o flujo hagan referencia a los nombres correctos de variables/funciones.  
   En general, estas complicaciones se abordan mediante pruebas incrementales: primero validar la producción en un solo agente, luego integrar proveedores, y finalmente ejecutar la cadena completa. La visualización gráfica (barras y líneas) ayuda a detectar rápidamente comportamientos anómalos (p.ej. inventario que nunca cambia). Con depuración y ajuste iterativo, se logra un modelo robusto que refleje correctamente la flexibilidad de la cadena de suministro.

# Problema #2 (Asignado)

## Planteamiento del problema

Desarrollar e implementar un modelo de simulación para la siguiente situación hipotética:

Suponga que tiene un restaurante que desea utilizar simulación para instaurar su servicio a domicilios. Con este servicio se desea que los domicilios cubran un total de 45 x 45 celdas, ubicándose el restaurante en la mitad del mapa. Hay ubicados 50 lugares de residencia (casas y apartamentos) de forma aleatoria (no puede haber dos casas en la misma celda y se supondrá que cada edificio puede tener entre 4 y 20 apartamentos). La probabilidad de que una unidad residencial sea una casa es de 0,3 y de que sea un edificio de 0,7. La distribución para el número de apartamentos es aleatoria discreta.

El restaurante tiene un número n de motociclistas que hacen los pedidos y se desplazan a 20 celdas por minuto. Se espera que se hagan 3 pedidos en el total de las casas cada minuto durante 180 minutos (un lugar de residencia puede hacer más de un pedido durante este periodo).

Inicialmente elija n arbitrariamente y:

* Establezca el tiempo de espera promedio de los usuarios si los motociclistas regresan al restaurante si estos solamente llevan y entregan un pedido.
* Establezca el tiempo promedio de los usuarios si los motociclistas pueden llevar dos pedidos y, tras entregar el primer pedido se dirigen a llevar el siguiente pedido sin regresar a la estación (si no hay pedidos en ese momento se queda esperando en el mismo lugar en el que entregó el primer pedido).

Para ambos casos establezca el n (número de motociclistas) necesarios para atender los pedidos con un tiempo de espera promedio menor a 20 minutos.

Considere que existe la posibilidad de contratar bicicletas de carga que cuestan la tercera parte que lo que cuestan las motocicletas, pero se desplazan a la mitad de velocidad (10 celdas por minuto). Analice cuántas bicicletas de carga habría que contratar para cumplir el requisito dado anteriormente (tiempo de espera promedio menor a 20 minutos) y cuál de las dos opciones es más rentable (motocicletas o bicicletas de carga).

Finalmente, imagine qué otras variables habría tener en cuenta si usted fuera el encargado de resolver el problema hipotético (por ejemplo, las motos/bicicletas tendrían que transitar a través de calles y no podrían desplazarse en línea recta hasta su destino o existieran alternativas de contratación que pudieran resolver el problema como lo son hoy en día Rappi/Uber Eats) y cómo afectarían la forma de solucionarlo.

## Plan General

Este plan describe el desarrollo de un modelo de simulación para optimizar un servicio de domicilios, aplicable en NetLogo o AnyLogic.

### Problemas a Resolver

El modelo busca determinar:

1. Número óptimo de repartidores (n): ¿Cuántos motociclistas se necesitan para un tiempo de espera promedio menor a 20 minutos?
2. Eficiencia de estrategias: Comparar entrega de pedidos simples (regreso a base) vs. pedidos dobles (encadenamiento).
3. Análisis de costo-beneficio: ¿Cuántas bicicletas de carga (mitad de velocidad, un tercio del costo) se requieren para cumplir el objetivo, y cuál opción es más rentable?

### Fases del Plan General

#### Fase 1: Diseño y Conceptualización (General)

1. **Entidades y Agentes:**
   * **Restaurante:** Base de operación.
   * **Residencias (50):** Generan pedidos (0.3 casas, 0.7 edificios con 4-20 apartamentos).
   * **Domiciliarios:** Motociclistas o ciclistas (propiedades: velocidad, capacidad, costo, estado).
   * **Pedidos:** Destino y tiempo de creación.
2. **Entorno:** Cuadrícula de 45x45 celdas, restaurante en (0,0).
3. **Variables y Métricas:** pedidos-pendientes, pedidos-entregados, tiempo-total-espera, **tiempo-espera-promedio** (clave, en minutos), n-domiciliarios.
4. **Flujo del Proceso:** Inicialización → Bucle por segundo: Generar 3 pedidos/minuto → Asignar pedidos (simple/doble) → Mover domiciliarios → Entregar pedidos → Actualizar métricas → Condición de parada (180 minutos).
5. **Reglas:** Velocidades (motos: 20 celdas/min, bicis: 10 celdas/min); costos relativos; lógica de asignación y movimiento.

#### Fase 2: Implementación en AnyLogic

1. **Modelado:** Continuous Space (o GIS Map), poblaciones de agentes (residences, deliveryAgents, orders).
2. **Agentes:** Residence (parámetros: tipo, apartamentos), DeliveryAgent (parámetros: velocidad, tipo, costo; Statechart para estado).
3. **Lógica Central (Main Agent):**
   * Eventos: OnStartup, Cyclic Event (cada segundo), generateOrdersEvent (cada minuto).
   * Colecciones: pendingOrders.
   * Funciones: generateOrders(), assignOrders(), updateMetrics().
4. **Movimiento:** Funciones moveTo(), getDistance().
5. **Recolección de Datos:** Statistics, Dataset, gráficos.

#### Fase 3: Implementación en NetLogo

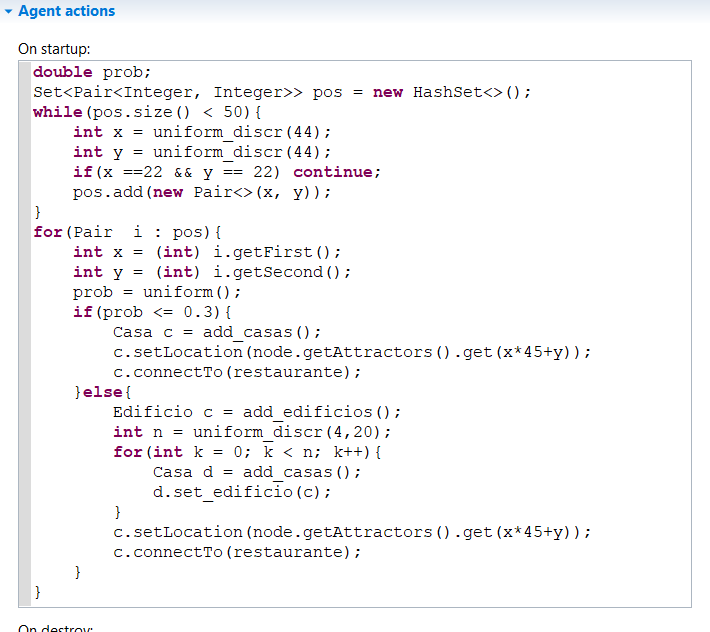
1. **Configuración:** globals, turtles-own para domiciliarios y residencias.
2. **Procedimientos (setup, go):** Inicialización de mundo/agentes, bucle principal con llamadas a submodelos.
3. **Submodelos:**
   * generar-pedidos: Cada 60 ticks.
   * asignar-pedidos-simples/dobles: Lógica de asignación.
   * mover-domiciliarios-simples/dobles: Lógica de movimiento, actualización de estado y entrega.
   * actualizar-metricas: Cálculo de tiempo-espera-promedio.
4. **Interfaz y Experimentos:** Sliders para parámetros, monitores para métricas. Usar BehaviorSpace para barrido de parámetros.

#### Fase 4: Análisis y Reporte (General)

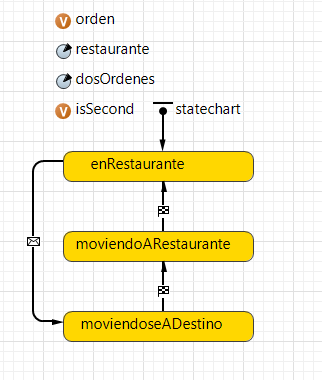
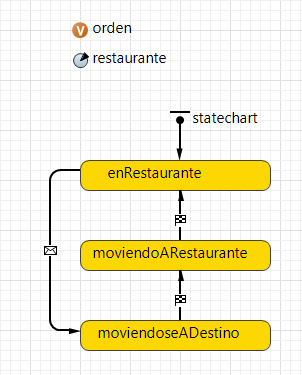
1. **Ejecución:** Múltiples corridas para cada combinación de parámetros. Barrido de n para encontrar el objetivo de <20 minutos.
2. **Análisis:**
   * Determinar n para cada escenario de entrega (simple/doble) con motocicletas.
   * Calcular número equivalente y costo de bicicletas de carga.
   * Comparar rentabilidad.
3. **Consideraciones Adicionales:** Reflexionar sobre limitaciones del modelo actual (movimiento en línea recta, demanda constante, etc.) y cómo variables como topología de calles, patrones de demanda o modelos de contratación externos (Rappi/Uber Eats) afectarían la solución.

## Planteamiento en ANYLOGIC

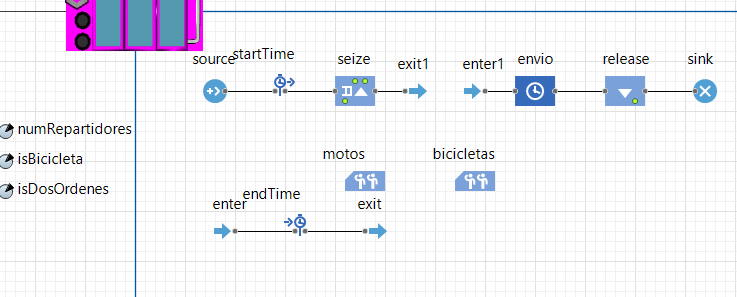
Para comenzar con el desarrollo en Anylogic creamos un *nodo-rectangular* dentro de “Main” donde configuraremos el valor de *attractors* con una grilla 45x45. En la coordenada (x=22 , y=22) ubicaremos el restaurante, en éste caso, el mueble de horneado de color rosado. En la acción de startup del agente main, estableceremos la aleatoriedad de los agentes de tipo *edificios* y *casas*con una probabilidad de 0,7 y 0,3 respectivamente, como podemos ver a continuación:



Posteriormente definiremos las rutinas para nuestro agente *“Motocicleta”* y el agente *“Bicicleta” ,* los cuales mantienen una lógica similar pero con tiempos diferentes.

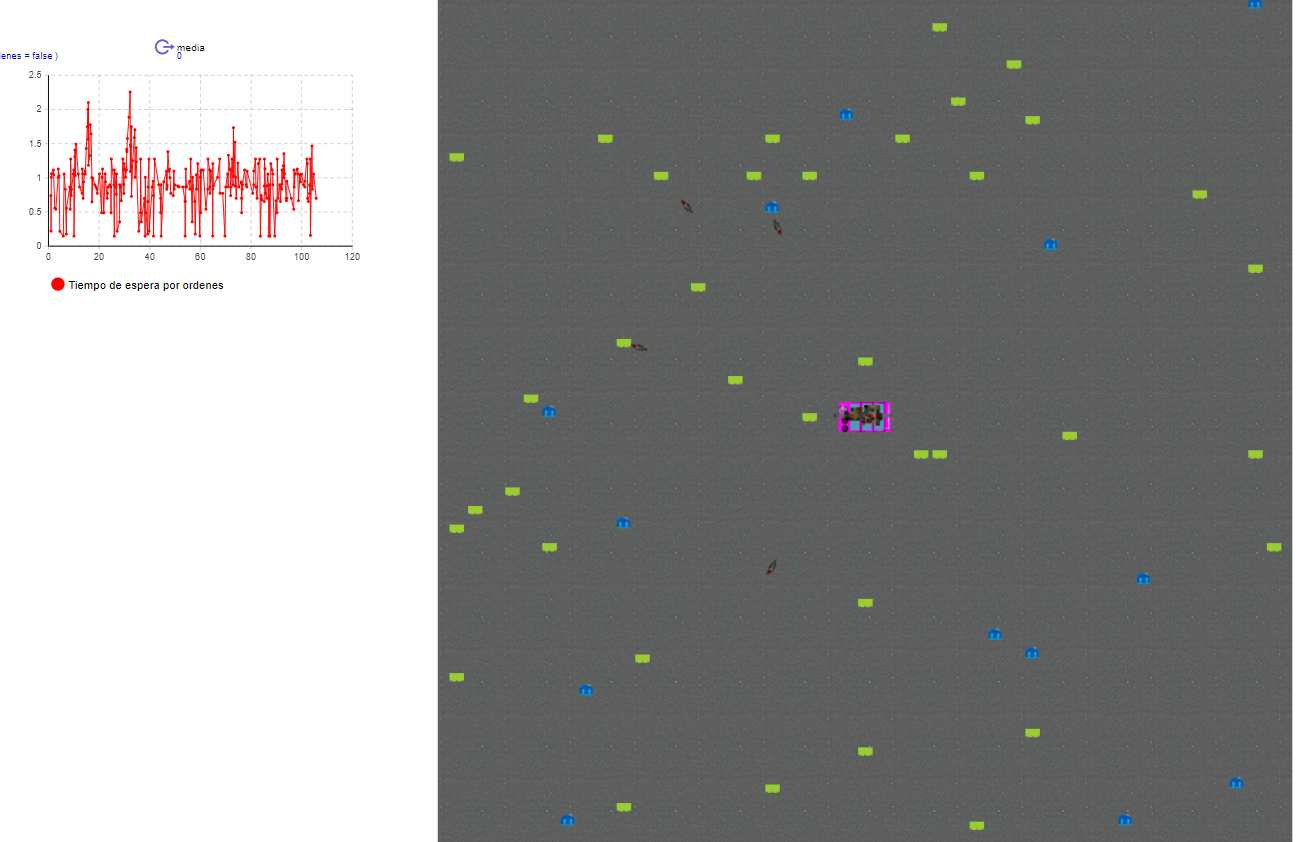
**

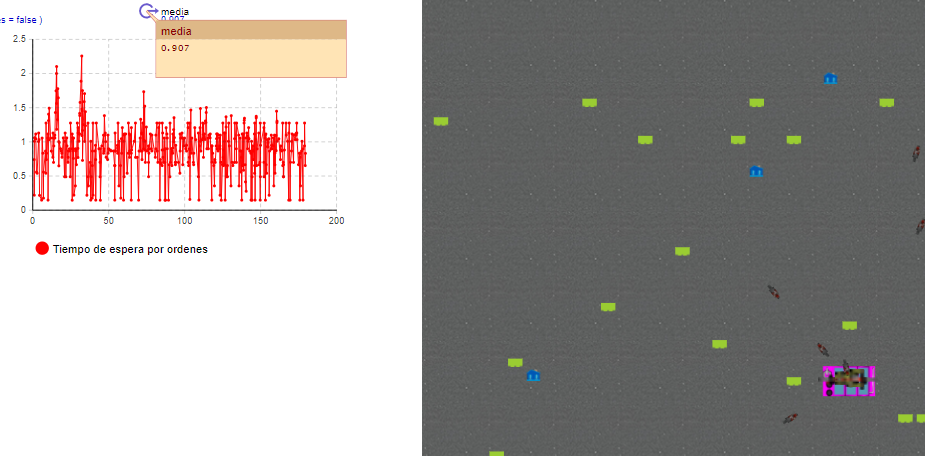
Finalmente, los destinos de los repartidores, se definen en el agente “restaurante”donde por medio de bloques como seize, enter y exit, se dispara el “evento” con la lógica correspondiente a un pedido por cada repartidor, independientemente de ser motocicleta o bicicleta. A partir de este flujo de bloques, contabilizamos el tiempo de espera de los pedidos para tomarlo como estadística de desempeño.



Una vez iniciada la simulación, el modelo luce de la siguiente manera, logramos ver las distintas casas y edificios junto con el restaurante en el centro del mapa, además se observa como los repartidores inician y finalizan su recorridos a la hora de entregar cada pedido.

## 

  
Cuando la simulación llega a su fin observamos en la gráfica de la izquierda el comportamiento del tiempo de espera por órdenes, además el modelo presenta el valor promedio de esta variable, de esta manera monitoreamos el desempeño del escenario propuesto.



## Implementación en ANYLOGIC

La implementación de modelo hecho en Anylogic se encuentra en la siguiente carpeta

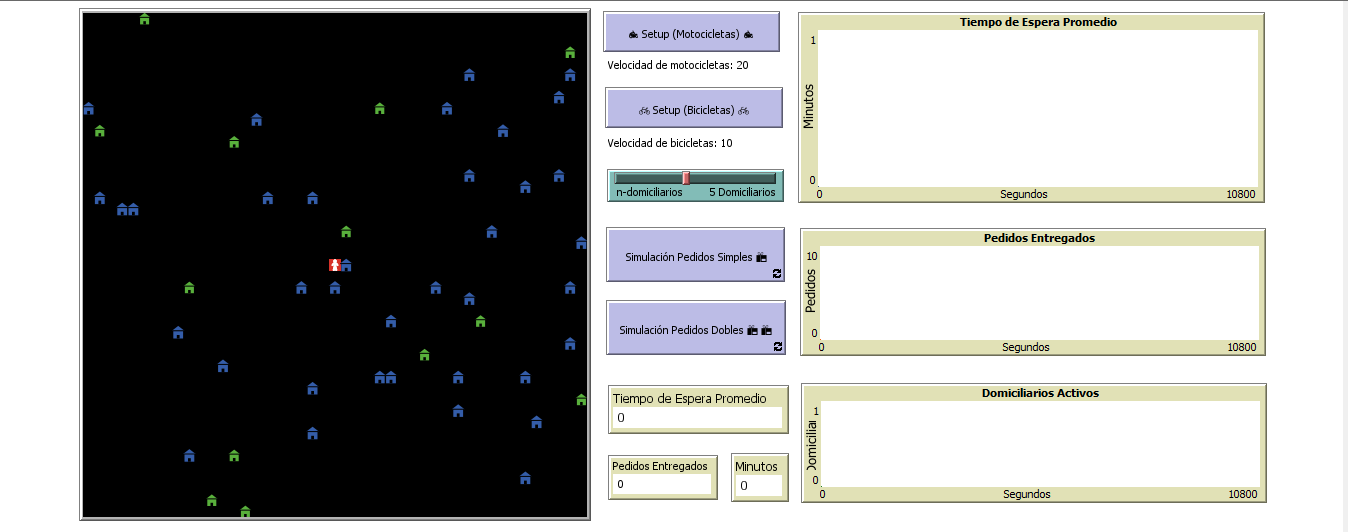
## Planteamiento en NETLOGO

El presente modelo, implementado en la plataforma NetLogo 6.4.0 [2], tiene como objetivo cuantificar la eficiencia del servicio bajo distintas estrategias operativas, determinar el número óptimo de repartidores (n) y realizar un análisis de costo-beneficio entre el uso de motocicletas y bicicletas.

El entorno de simulación es una cuadrícula bidimensional de 45x45 celdas (con coordenadas de -22 a +22), que representa el área de servicio.

* **Agentes Estáticos (Residencias):** Se generan 50 residencias en ubicaciones aleatorias y únicas utilizando el procedimiento setup-residencias. Para evitar la superposición, se selecciona una celda (patch) que no contenga otras residencias (one-of patches with [not any? residencias-here]). Cada residencia se instancia con atributos clave: es-edificio? (booleano, con 70% de probabilidad de ser true) y num-apartamentos (un entero aleatorio entre 4 y 20 si es edificio).
* **Agentes Móviles (Domiciliarios):** Representan a los repartidores. Su atributo más importante es estado, que funciona como una máquina de estados finitos para gestionar su comportamiento ("en-base", "en-camino-1", "en-camino-2", "regresando", "esperando-pedido"). Otros atributos cruciales son pedido-1 y pedido-2 para almacenar los pedidos asignados, y velocidad, que define su capacidad de movimiento en celdas por segundo.

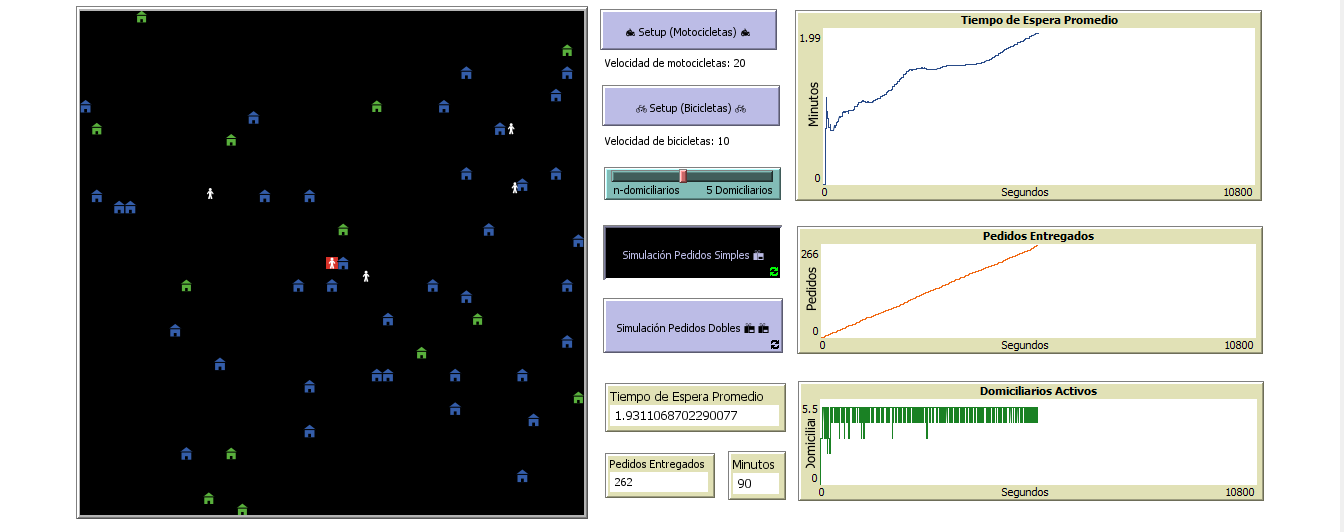
El modelo, una vez iniciado pero antes de comenzar con la simulación de los pedidos, se ve de la siguiente manera:



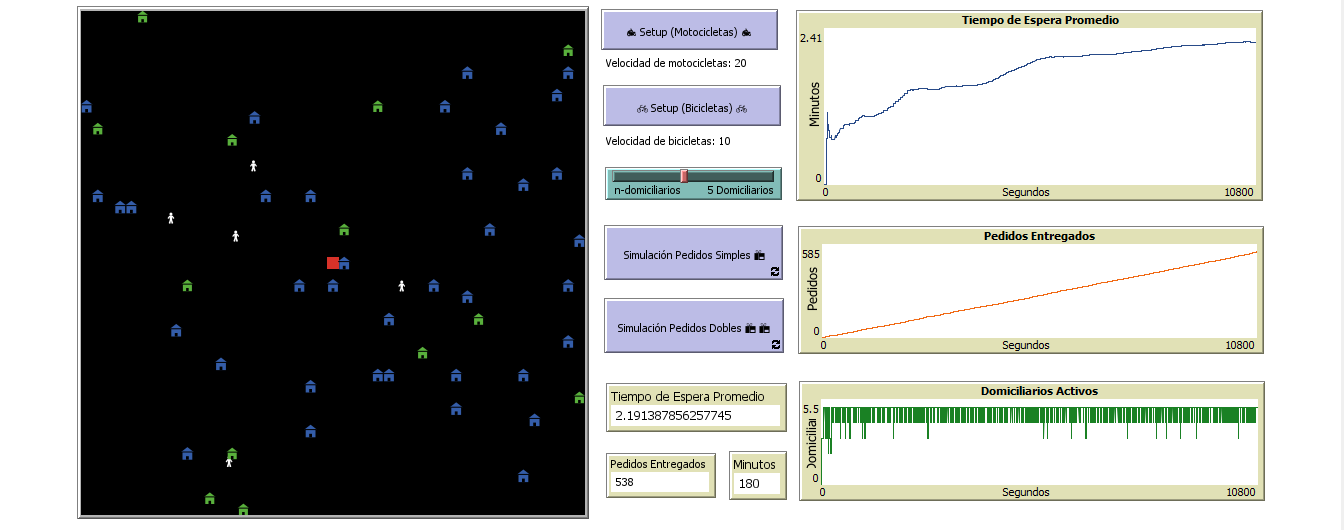
El código es modular para facilitar la experimentación y el análisis comparativo entre diferentes configuraciones.

* **Procedimientos de Inicialización (setup):** El modelo se inicializa con dos funciones principales: setup-a para simular con motocicletas y setup-b para bicicletas. Estas, a su vez, llaman a sub-rutinas:
  + **setup-mundo:** Establece el restaurante en la celda (0,0) y reinicia todas las variables globales (pedidos-pendientes, tiempo-total-espera, etc.).
  + **setup-residencias:** Crea y distribuye las 50 residencias.
  + **setup-motos y setup-ciclas:** Crean *n* agentes domiciliarios, asignándoles su velocidad correspondiente en celdas/segundo (20 / 60 para motos, 10 / 60 para bicicletas).
* **Bucle Principal de Simulación (go):** De forma análoga al setup, existen dos bucles principales que se ejecutan en cada tick: go-1 para el escenario de pedidos simples y go-2 para el de pedidos dobles. Cada ciclo de go orquesta la simulación en una secuencia lógica:
  + **generar-pedidos:** Crea nuevos pedidos.
  + **asignar-pedido-simple / asignar-pedidos-dobles:** Asigna los pedidos pendientes a los repartidores disponibles.
  + **mover-domiciliarios-simple / mover-domiciliarios-dobles:** Actualiza la posición de los repartidores según su estado y destino.
  + **actualizar-metricas:** Recalcula el tiempo de espera promedio.
  + **tick:** Avanza el reloj de la simulación un segundo. La simulación se detiene automáticamente al alcanzar los 10,800 ticks (180 minutos).
* **Mecánicas Fundamentales:**
  + **generar-pedidos**: Esta función implementa la llegada de nueva demanda. Utiliza la condición if ticks mod 60 = 0 para generar un lote de 3 pedidos exactamente en el primer segundo de cada minuto simulado. Cada pedido se almacena como una lista [agente-cliente, tiempo-de-creacion], una estructura de datos eficiente para rastrear el origen y la antigüedad del pedido.
  + **asignar-pedidos-dobles y mover-domiciliarios-dobles (Escenario 2, el más complejo):** Estas dos funciones interconectadas contienen la lógica central del modelo. La asignación no solo se hace a repartidores en-base, sino también a aquellos que acaban de entregar un pedido y se encuentran esperando-pedido. La función de movimiento actúa sobre el estado del domiciliario: si está en-camino-1, se dirige al destino de pedido-1. Al llegar (detectado por distance destino < velocidad), llama a entregar-pedido-1-doble. Este procedimiento de entrega actualiza las métricas y, crucialmente, cambia el estado del repartidor a en-camino-2 (si pedido-2 existe) o a esperando-pedido (si no lo tiene), demostrando la lógica de no retorno a base. Solo después de entregar un segundo pedido o ser despachado desde la base, el repartidor entrará en estado regresando.
  + **actualizar-metricas**: Al final de cada tick, esta función calcula el tiempo-espera-promedio. Es fundamental destacar que, como el tiempo-total-espera se acumula en segundos (ticks), el resultado final se divide por 60 para presentar la métrica en minutos, de acuerdo con los requisitos del problema.

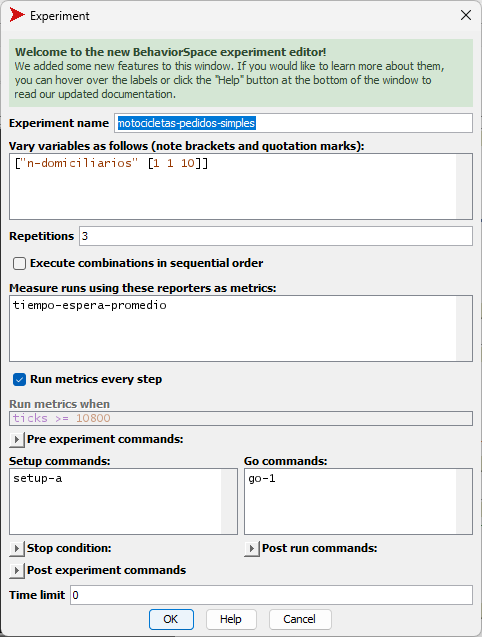
Durante la simulación el modelo se ve de la siguiente manera:



Y al finalizar la simulación se ve de la siguiente manera (con las estadísticas completas):

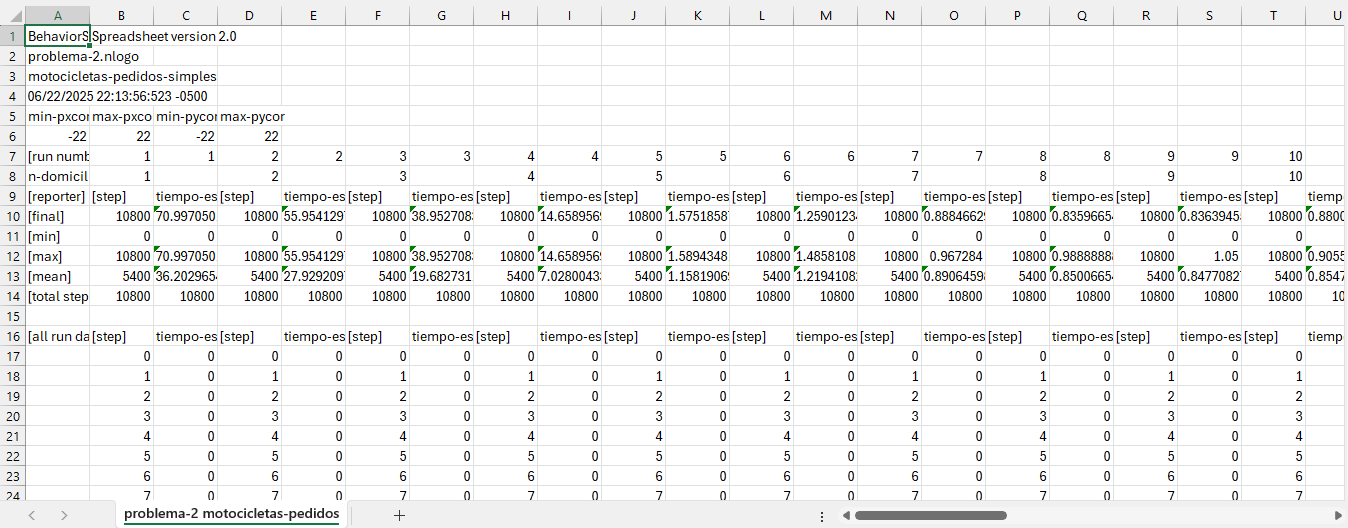


La combinación de estos procedimientos permite una simulación robusta y flexible. El objetivo final es utilizar la herramienta BehaviorSpace [3] de NetLogo para ejecutar sistemáticamente ambos escenarios (go-1, go-2) con ambos tipos de vehículos (setup-a, setup-b), variando el número de repartidores (n-domiciliarios). La herramienta se configura de la siguiente manera:



En este caso se está configurando el experimento para domiciliarios en motocicletas con pedidos simples, variando el número de domiciliarios de 1 a 10, uno a uno. Se evalua principalmente el tiempo de espera promedio y se ejecuta el experimento tres veces. Así mismo se configuran los otros tres experimentos, con el mismo número de repeticiones totales (30 veces) y las hojas de cálculo con los resultados se analizan para determinar el número mínimo de repartidores de cada tipo para satisfacer la condición de un tiempo de espera inferior a 20 minutos, y así, realizar una comparación de costos informada para determinar la opción más rentable.

Las hojas de cálculo con los resultados se ven de la siguiente manera:



## Implementación en NETLOGO

La implementación del modelo en NetLogo, junto con sus resultados se encuentran en la siguiente carpeta compartida de Drive:

[Problema #2 (NETLOGO)](https://drive.google.com/drive/folders/1HPkSL7B9fki55YLc0UJT98pxNYU-_ncj?usp=sharing)

## Documentación ODD

### 1. Propósito

El modelo tiene como objetivo optimizar la eficiencia y rentabilidad de un servicio de entrega para un restaurante. Busca determinar el número mínimo de repartidores para mantener un tiempo de espera promedio por debajo de 20 minutos, comparar estrategias de entrega (pedidos simples vs. dobles) y evaluar el costo-beneficio de distintos vehículos (motocicletas vs. bicicletas). Sirve como herramienta de apoyo a la decisión logística.

### 2. Entidades, Variables de Estado y Escalas

**Entidades:**

* **Domiciliarios:** Repartidores móviles.
* **Residencias:** Hogares de clientes, destinos de pedidos (estáticos).
* **El Entorno:** Cuadrícula de 45x45 celdas donde se mueven los domiciliarios.

**Variables de Estado:**

* **Globales:** restaurante (ubicación central), pedidos-pendientes (lista), pedidos-entregados (contador), tiempo-total-espera, tiempo-espera-promedio (métrica clave en minutos), n-domiciliarios (parámetro de entrada).
* **Residencias:** es-edificio? (booleano), num-apartamentos (1 para casas, 4-20 para edificios).
* **Domiciliarios:** estado ("en-base", "en-camino-1", etc.), pedido-1, pedido-2, velocidad (motos: 20/60 celdas/seg, bicis: 10/60 celdas/seg).

**Escalas:**

* **Espacial:** 45x45 celdas.
* **Temporal:** Cada tick = 1 segundo. Duración de la simulación: 180 minutos (10,800 ticks).

### 3. Resumen del Proceso y Planificación

El modelo se ejecuta en un bucle por tick (segundo). Las acciones secuenciales son:

1. **Verificación de Parada:** Si ticks >= 10800.
2. **Generación de Pedidos:** Cada minuto se crean 3 nuevos pedidos aleatorios (generar-pedidos).
3. **Asignación de Pedidos:** Los pedidos pendientes se asignan a domiciliarios disponibles (asignar-pedido-simple o asignar-pedidos-dobles).
4. **Movimiento de Domiciliarios:** Los domiciliarios se mueven según su estado y destino (mover-domiciliarios-simple o mover-domiciliarios-dobles).
5. **Actualización de Métricas:** Se recalcula el tiempo-espera-promedio.
6. **Avance del Reloj:** El tick avanza.

### 4. Conceptos de Diseño

* **Principios Básicos:** Caso de estudio hipotético de un sistema de colas y recursos.
* **Emergencia:** El tiempo-espera-promedio emerge de la interacción entre pedidos, repartidores y estrategias.
* **Adaptación:** Los domiciliarios reaccionan a su estado y asignaciones; no aprenden.
* **Objetivos:** Los domiciliarios buscan entregar pedidos y regresar a la base.
* **Sensing:** Domiciliarios "sienten" destinos; el sistema central "siente" pedidos y estados de domiciliarios.
* **Interacción:** Indirecta, mediada por el sistema central (competición por pedidos).
* **Estocasticidad:** Se usa en la ubicación de residencias, tipo de residencia (casa/edificio), número de apartamentos y generación de pedidos.
* **Observación:** Se registran tiempo-espera-promedio y pedidos-entregados.

### 5. Inicialización

Al inicio (setup-a o setup-b):

* Se limpia el mundo y se ubica el restaurante en (0,0).
* Se crean 50 residencias aleatorias con sus propiedades.
* Se crean n-domiciliarios en el restaurante, con estado "en-base" y velocidad asignada.
* Variables globales de métricas se inicializan a cero/vacías.

### 6. Datos de Entrada

No hay archivos de entrada externos. Todos los parámetros están codificados. El único parámetro variable es n-domiciliarios, controlado por el usuario.

### 7. Submodelos

* **Generación de Pedidos (generar-pedidos):** Cada minuto, añade 3 nuevos pedidos (destino, tiempo de creación) a pedidos-pendientes.
* **Asignación y Movimiento (Pedidos Simples):**
  + **Asignación (asignar-pedido-simple):** Asigna el pedido más antiguo a domiciliarios "en-base".
  + **Movimiento (mover-domiciliarios-simple):** Mueve el domiciliario al destino, entrega, y lo hace regresar al restaurante.
* **Asignación y Movimiento (Pedidos Dobles):**
  + **Asignación (asignar-pedidos-dobles):** Lógica más compleja para asignar pedido-1 y pedido-2 a domiciliarios disponibles o que ya tienen un primer pedido.
  + **Movimiento (mover-domiciliarios-dobles):** El movimiento depende del estado: al primer destino, luego al segundo (si aplica), entrega, y regresa a la base.
* **Métricas (actualizar-metricas):** Calcula el tiempo-espera-promedio en cada tick, normalizando a minutos.

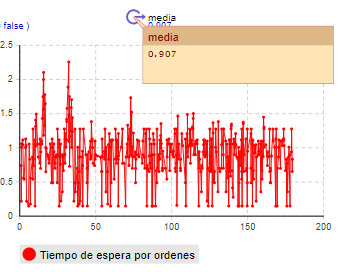
## Resultados

### Resultados AnyLogic

Obtuvimos el resultado esperado cuando n = 4 lo que implica un punto de inflexión clave en la eficiencia del sistema simulado. Hasta ese valor de domiciliarios, el sistema no lograba responder con suficiente rapidez a la demanda constante de pedidos (3 por minuto), lo que generaba acumulación de órdenes y tiempos de espera elevados. Al alcanzar cuatro domiciliarios, se observa que la capacidad del sistema comienza a equilibrarse con la carga de trabajo, permitiendo que los pedidos sean asignados y entregados de forma más oportuna. Este hallazgo sugiere que n = 4 representa el umbral mínimo de recursos necesarios para cumplir con un nivel de servicio aceptable (menos de 20 minutos de espera promedio) bajo las condiciones actuales de la simulación (velocidades, distancias, tasa de pedidos, etc.). Además, este resultado sirve como base para comparar distintas configuraciones —como uso de bicicletas, cambios en la lógica de asignación o variaciones en la geografía— y evaluar su impacto en la eficiencia del sistema.



El mejor tiempo obtenido ocurrió cuando n = 17 lo que indica el punto de mayor eficiencia alcanzado por el sistema bajo las condiciones simuladas. A medida que se incrementa el número de domiciliarios, el tiempo de espera disminuye progresivamente, ya que hay más agentes disponibles para cubrir la demanda constante de pedidos. Sin embargo, al llegar a n = 17, el sistema alcanza un rendimiento óptimo en el que los pedidos son asignados y entregados casi inmediatamente después de su generación, reduciendo al mínimo el tiempo de espera. Este valor representa el punto en el que la capacidad operativa no solo satisface la demanda, sino que la excede ligeramente, permitiendo tiempos de respuesta extremadamente rápidos sin generar congestión ni desperdicio del personal excesivo. A partir de ese punto, agregar más domiciliarios podría no mejorar significativamente el tiempo de espera, y en cambio podría representar un uso ineficiente de recursos. Por tanto, n = 17 define equilibrio ideal entre velocidad de atención y uso de recursos, siendo un valor clave para decisiones estratégicas de dimensionamiento del equipo de reparto.



En resumen, los valores obtenidos para cada escenario son:

**Entregas en Motocicleta:**

Se realizaron simulaciones variando el número de domiciliarios para dos escenarios distintos: entregas simples (un pedido por viaje) y entregas dobles (dos pedidos por viaje antes de regresar al restaurante). En ambos casos, se evaluó el comportamiento del sistema con respecto al tiempo de espera promedio.

* **Entregas simples**: El primer valor de nnn que permitió alcanzar un tiempo promedio de espera inferior a 20 minutos fue n = 4. El mejor desempeño, es decir, el menor tiempo de espera registrado, se alcanzó con n = 15 domiciliarios.
* **Entregas dobles**: Nuevamente, n = 4 fue el primer punto en que el tiempo de espera cayó por debajo del umbral de 20 minutos. En este caso, el mejor resultado se logró con n = 8 domiciliarios.

#### **Entregas en Bicicleta:**

Dado que las bicicletas tienen una velocidad de desplazamiento menor, se requiere una mayor cantidad de repartidores para alcanzar los mismos niveles de servicio.

* **Entregas simples**: El umbral de los 20 minutos se superó por primera vez con n = 9 domiciliarios, mientras que el mejor tiempo promedio se obtuvo con n = 13.
* **Entregas dobles**: El tiempo de espera descendió por debajo de los 20 minutos con n = 7 domiciliarios, y el mejor tiempo se registró con n = 11.

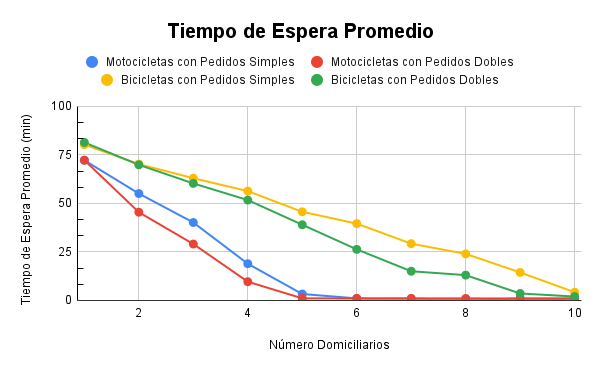
### Resultados NetLogo

Los resultados principales se encuentran en la siguiente hoja de cálculo de Google Sheets [4]: [Resultados Problema #2 (NETLOGO)](https://docs.google.com/spreadsheets/d/1tyqUbR3diMgoBy3UrbBrtZZVVqti3Rn57ALXcunn3mc/edit?usp=sharing)

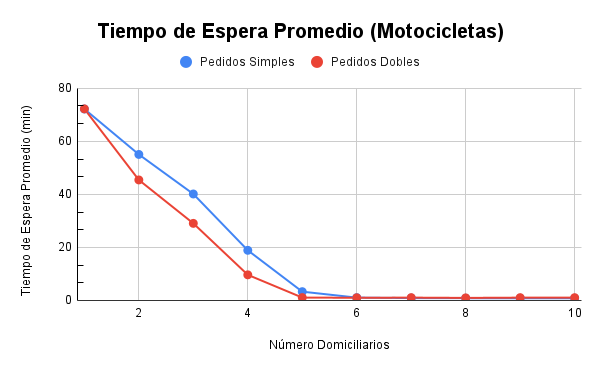
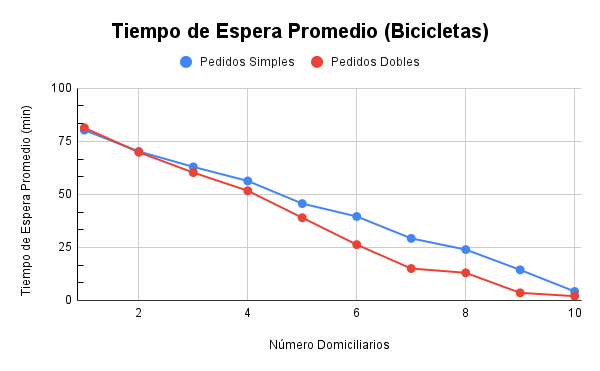
Para cada hoja de cálculo resultado de los experimentos realizados con BehaviourSpace se realizó un cálculo de la media de cada caso según el número *n* de domiciliarios, ya que se corrieron tres repeticiones en cada caso. Posteriormente se utilizaron dichas medias para comparar entre casos según el número de domiciliarios.

A continuación se presentan las gráficas resultantes de los resultados:

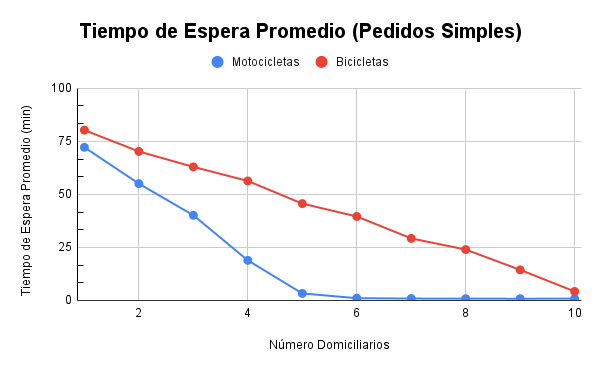
#### Resultados generales (los cuatro casos):

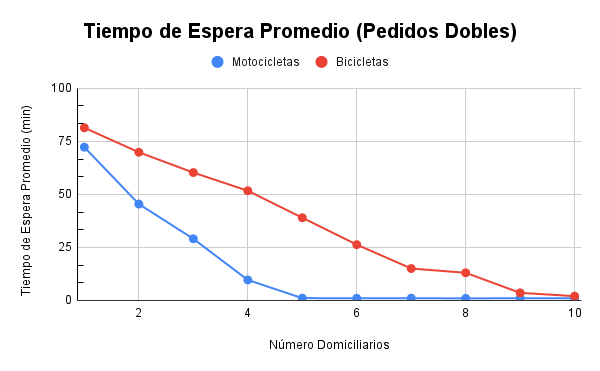


#### Resultados de acuerdo al tipo de vehículo:

#### Resultados de acuerdo al tipo de pedido:





## Análisis

### Análisis AnyLogic

#### **Evaluación de la Alternativa más Rentable:**

Para identificar la opción más rentable, se compararon los valores de n mínimos que permiten cumplir con el objetivo de tiempo de espera menor a 20 minutos bajo el escenario de entregas simples. Suponiendo que el costo operativo de una bicicleta es del costo de una motocicleta, tenemos:

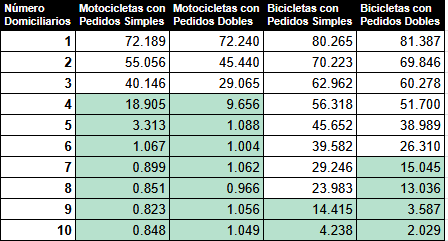
* **Motocicletas**:  
   Costo relativo:
* **Bicicletas**:  
   Costo relativo:

Al comparar los costos, se observa que ​, por lo tanto, las bicicletas resultan ser más rentables para cumplir el objetivo de tiempo de espera bajo los parámetros definidos.

No obstante, en términos de eficiencia del servicio, el menor tiempo de espera posible se logra con motocicletas , mientras que en el caso de bicicletas este valor mínimo ronda los 1,2 minutos. Por tanto, la decisión final dependerá de la estrategia del administrador del restaurante: priorizar la eficiencia o minimizar costos operativos.

### Análisis NetLogo

Para el análisis del modelo primero veamos los resultados en forma de tabla (con solo 3 decimales):

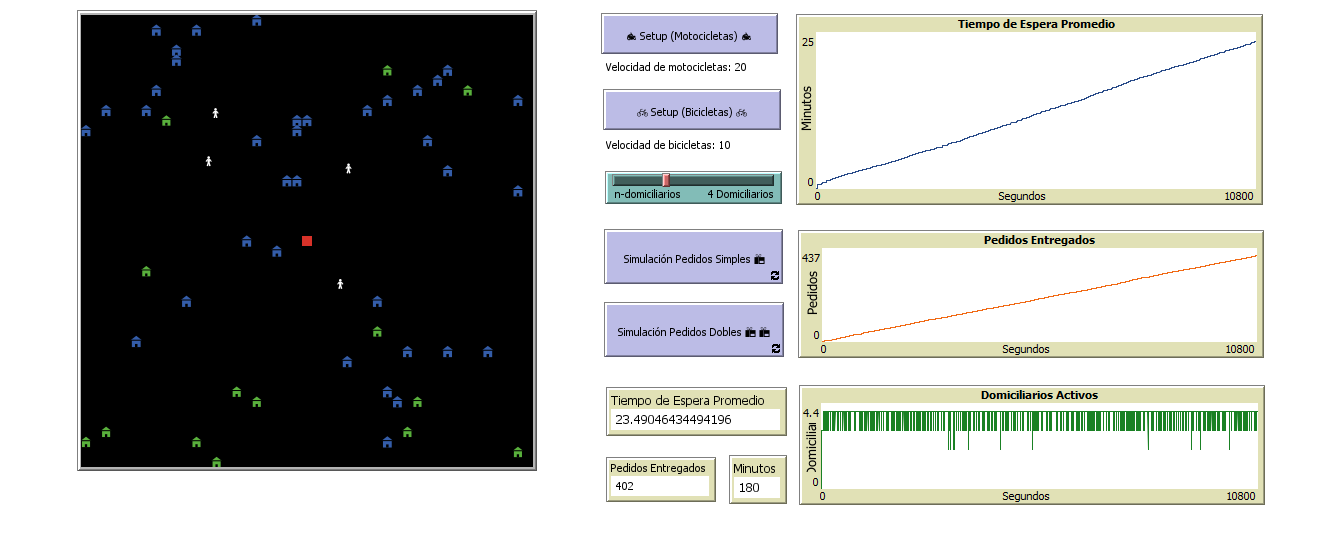


A partir de estos datos, recolectados utilizando BehaviourSpace de NetLogo y Google Sheets se pueden observar los valores de *n* (número de domiciliarios) para los cuales el tiempo de espera promedio es menor a 20 minutos (valores resaltados en verde).

Así, se determina que para cada caso el *n* mínimo para que los tiempos de espera promedio de los pedidos sean menor a 20 minutos son los siguientes:

* **Caso 1: Motocicletas con pedidos simples:** n debe ser igual o mayor a 4.
* **Caso 2: Motocicletas con pedidos dobles:** n debe ser igual o mayor a 4.
* **Caso 3: Bicicletas con pedidos simples:** n debe ser igual o mayor a 9.
* **Caso 4: Bicicletas con pedidos dobles:** n debe ser igual o mayor a 7.

En este punto es importante resaltar que, para el Caso 1, el n óptimo que se utilizará de ahora en adelante es n = 5, dado que durante el desarrollo del modelo se notó que para varios casos de n = 4 el tiempo de espera promedio es ligeramente mayor a 20 minutos. Teniendo en cuenta esto, es mejor utilizar n = 5 para garantizar tiempos de espera menores a 20 minutos. A continuación un ejemplo de esta situación (23 minutos para el tiempo de espera promedio):



Para los demás casos esta decisión podría ser también útil, pero teniendo en cuenta que los tiempos de espera promedio para los n mínimos no superan los 15 minutos (caso 4 con n = 7), se puede suponer que estos n realmente son los mínimos.

Luego, a partir de este análisis de los valores mínimos de *n* número de domiciliarios, se puede revisar cuál de los cuatro casos es más rentable, también teniendo en cuenta la modalidad de los pedidos.

Se sabe que las bicicletas cuestan una tercera parte de lo que cuestan las motocicletas, por lo que el costo de cada caso para satisfacer el requisito de tiempos de espera promedio menores a 20 minutos son los siguientes:

* **Caso 1:** 5 domiciliarios \* 3 = 15 (costo total).
* **Caso 2:** 4 domiciliarios \* 3 = 12 (costo total).
* **Caso 3:** 9 domiciliarios \* 1 = 9 (costo total).
* **Caso 4:** 7 domiciliarios \* 1 = 7 (costo total).

Entonces, el mejor caso es el número 4, donde se utilizan 7 domiciliarios en bicicletas en la modalidad de pedidos dobles. Sin embargo, es mejor verlo desde cada modalidad:

* **Pedidos simples:** Para la modalidad de pedidos simples, donde los domiciliarios entregan un pedido y vuelven al restaurante, es mejor emplear 9 domiciliarios en bicicleta que 5 domiciliarios en motocicletas, siendo más rentable por 6 unidades de costo.
* **Pedidos dobles:** Para la modalidad de pedidos dobles, donde los domiciliarios entregan dos pedidos y vuelven al restaurante, es mejor emplear 7 domiciliarios en bicicleta que 4 domiciliarios en motocicletas, siendo más rentable por 5 unidades de costo.

En general, se puede concluir que es más económico emplear más domiciliarios en bicicleta para ambas modalidades, lo cuál se puede explicar porque las bicicletas solo cuestan una tercera parte y van a la mitad de la velocidad de las motocicletas, lo cuál las hace mucho más rentables. Sin embargo, es importante aclarar que acá no se está teniendo en cuenta el costo de cada empleado, por lo cuál si se tuviera en cuenta puede que sea más rentable tener menos empleados en motocicletas que más empleados en bicicletas.

Finalmente, para una implementación más real, el modelo actual podría enriquecerse con variables que reflejen mejor la complejidad del problema. Las más importantes son:

* **Red Vial y Tráfico:** El modelo podría incorporar una red de calles y simular tráfico en horas pico. Esto obligaría a usar rutas reales en vez de líneas rectas, afectando los tiempos de viaje y la eficiencia de cada tipo de vehículo de forma desigual. La ruta más corta podría no ser siempre la más rápida.
* **Tiempo de Preparación en Cocina:** Añadir un tiempo de preparación para cada pedido (ej. 5-15 minutos) introduciría un retraso inevitable. El análisis se centraría también en optimizar la sincronización entre la cocina y los repartidores, pues contratar más personal de entrega no solucionaría este nuevo cuello de botella.
* **Externalización a Plataformas (Rappi/Uber Eats):** Se podría integrar la opción de externalizar pedidos en momentos de alta demanda. La pregunta estratégica cambiaría de "¿cuántos empleados contratar?" a "¿cuál es el balance óptimo entre una flota propia (costo fijo) y usar terceros para gestionar los picos (costo variable por comisión)?".
* **Gestión de Turnos y Costos Reales:** Un modelo más avanzado consideraría turnos de trabajo para no tener personal ocioso y usaría costos detallados (salario, combustible, mantenimiento) en lugar de una relación simple, lo que refinaría el análisis de rentabilidad.

En resumen, aunque el modelo actual ofrece una valiosa guía estratégica, la incorporación de estos factores es crucial para diseñar una solución logística verdaderamente robusta y optimizada para el mundo real.

## Dificultades

Las dificultades presentadas durante el desarrollo del modelo fueron, inicialmente, comprender bien el problema para plantearlo correctamente, teniendo en cuenta todos los detalles del modelo. Luego, al iniciar la implementación se presentaron dificultades en el manejo de los casos, dado que la modalidad del pedido cambiaba significativamente el proceso de la entrega de los pedidos, y el cambio de vehículo aunque solo alteraba la velocidad, implicaba replicar ya sea el proceso de inicialización del programa o las dos lógicas de las modalidades de los pedidos; para esto se optó por cambiar la inicialización, ya que solo se altera el valor de la velocidad y no varios procesos del modelo.

También se presentaron dificultades con el modelo en NetLogo porque en su primer implementación se utilizaron los ticks como un minuto cada uno, por lo cuál en 180 ticks la simulación terminaba y era bastante difícil percibir cómo se comportaba el modelo. A pesar de reducir la velocidad de la simulación no se sentía exacta la simulación, por lo que se cambió el modelo para que cada segundo sea un tick, de modo que el modelo ahora corre en segundos y se tuvo que cambiar algunos valores para que siguieran presentándose en minutos dentro de la interfaz (los outputs y las gráficas).

Finalmente, para el análisis de los resultados, en el caso de NetLogo se tuvo que adecuar la hoja de cálculo para analizar los datos de manera uniforme y rápida, eliminando columnas de la hoja de resultados y utilizando fórmulas de Google Sheets para organizar la información de forma que se pudiera graficar para analizar los resultados del modelo.

# Referencias

1. <https://cloud.anylogic.com/model/aca0a876-e208-4f1d-ac1c-a03d97399bb6?mode=SETTINGS>
2. Uri Wilensky. (1999). *NetLogo*. Center for Connected Learning and Computer-Based Modeling, Northwestern University. [Online]. Disponible en:<https://ccl.northwestern.edu/netlogo/>
3. Uri Wilensky. *NetLogo User Manual: BehaviorSpace*. Center for Connected Learning and Computer-Based Modeling, Northwestern University. [Online]. Disponible en:<https://ccl.northwestern.edu/netlogo/docs/behaviorspace.html>
4. Google LLC. (n.d.). *Google Sheets*. [Online]. Disponible en: [https://www.google.com/sheets/about/](https://www.google.com/sheets/about/?authuser=1)