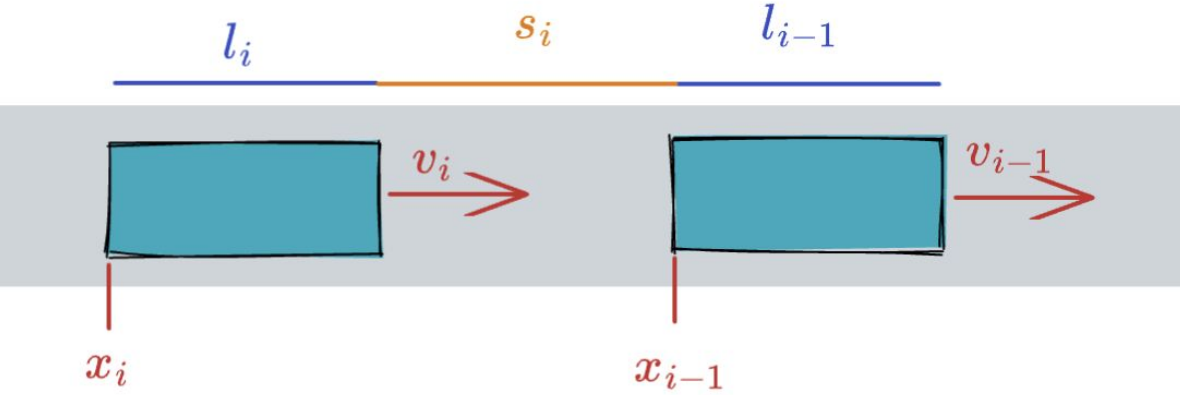
Simulación de tráfico

Pedro López. 60711
Valentino Riera Torraca. 60212

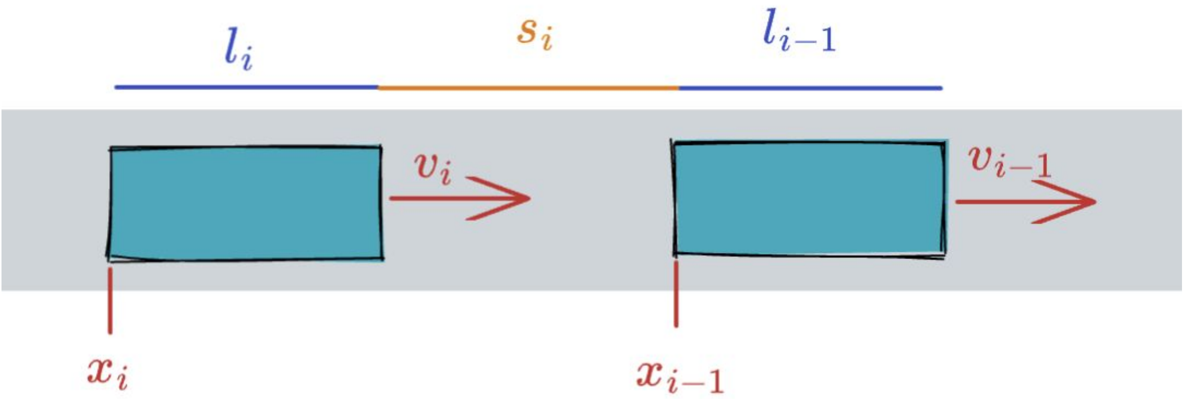
Modelo

Intelligent driver model (IDM)



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

Intelligent driver model (IDM)



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

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

Intelligent driver model (IDM)

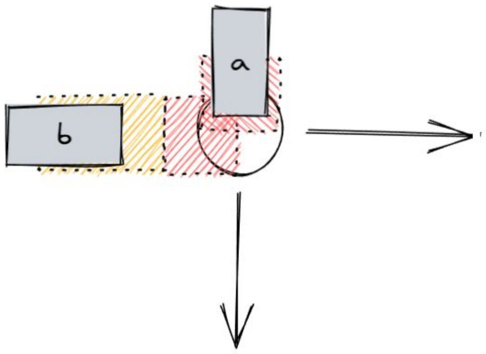
$$v_i(t + \Delta t) = v_i(t) + a_i(t + \Delta t) * \Delta t$$

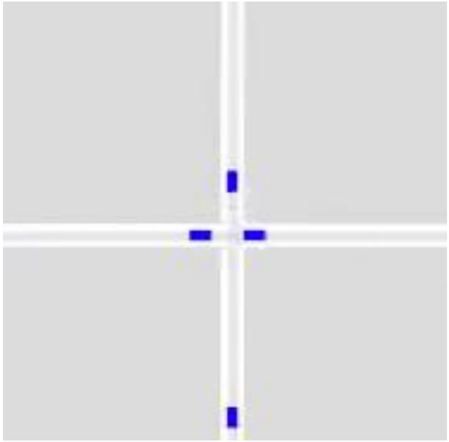
$$x_i(t + \Delta t) = x_i(t) + v_i(t) * \Delta t$$

- En caso de obtener una velocidad negativa, esta será tomada como v_i = 0 m/s

Intersecciones

- Lógica vive dentro de un nodo
- El de la derecha tiene prioridad a menos que doble
- Si los dos doblan, tiene prioridad el de la derecha nuevamente
- Zona amarilla de menos velocidad
- Zona roja de frenado
- El nodo no tiene tamaño (son superposiciones de calles)





Busqueda de rutas

Se utilizó el algoritmo A* con:

- Función de costo:
$$g(x) = g(x-1) + d + \omega * \gamma$$
    - ω : Costo de doblar
    - γ: {1,0} si se dobló o no se dobló, respectivamente.
    - d: Longitud de la ruta
    - g(0)= 0 , g(x-1): costo anterior

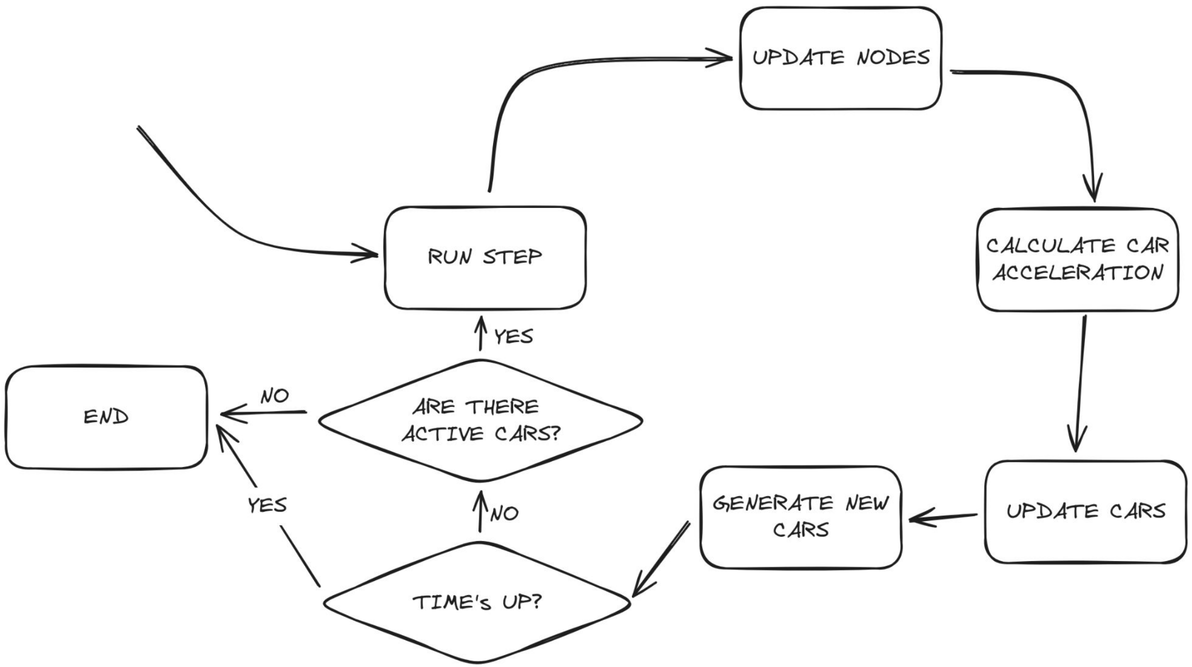
- Heurística:
$$Distancia = |x_{\text{objetivo}} - x_{\text{actual}}| + |y_{\text{objetivo}} - y_{\text{actual}}|$$

Implementación

Clases


Config
timeStep double
minimumDesiredDistance double
totalBlocksHeight int
accelerationExponent double
spawnTime long
maximumAcceleration double
maximumDeceleration double
yellowZoneLength double
turnWeight double
cars int
carLength double
yellowZoneSpeedMul double
totalBlocksWidth int
redZoneLength double
maximumDesiredSpeed double
reactionTime double
simulationTime long
toString() String
initConfig() void
Car
len() double
setVel(double) void
route() List<Road>
setAcc(double) void
hashCode() int
calculateValues(double, Config) void
isStopped() boolean
acc() double
setPos(double) void
update(double) void
isActive() boolean
wantsToTurn() boolean
setupCar(List<Road>, double, double, double) void
toString() String
vel() double
pos() double
road() Road
id() int
deactivate() void
Road
addCar(Car) void
end() Node
removeCar(Car) void
peekHeadCar() Car
length() double
getCarAhead(Car) Car
yellowZoneLength() double
start() Node
toString() String
redZoneLength() double
peekLastCar() Car
carCount() int
equals(Object) boolean
resetId() void
id() int
direction() Axis
OutputGenerator
PathFinder
HashedTreeSet
AStar
generatePath(Node[][], Road, Road) List<Road>
Node
id() int
equals(Object) boolean
getInboundRoads() List<Road>
initIntersectionPriority() void
isStopped(Road) boolean
addRoad(Road) void
getOutboundRoads() List<Road>
update() void
toString() String
x() double
hashCode() int
y() double
Axis
NORTH
WEST
SOUTH
EAST
valueOf(String) Axis
values() Axis[]
Simulation
initIntersectionPriorities() void
runStep() void
generateGrid() ArrayList<Road>
runSimulation(String) void
serializeStaticData() JSONObject
Heuristic
PathfindingNode
EuclidianDistance
ManhattanDistance
CarPool
activeCars() Set<Car>
getFreeCar() Car
getDeactivatedCars() Set<Car>
removeCar(Car) void
Main
main(String[]) void

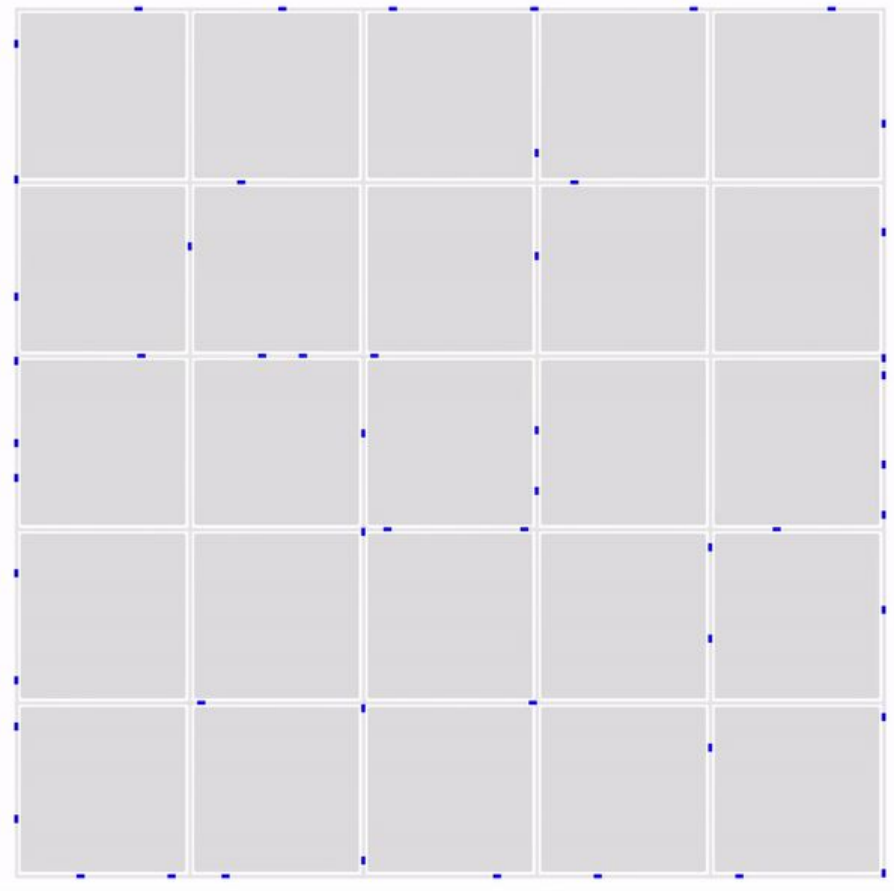
Flujo de simulación



Simulaciones

Mundo de simulación

- Area de N x N manzanas iguales cuadradas
- Calles unidireccionales de un solo carril
- Direcciones de las calles intercaladas
- Intersecciones SIN semáforos
- Los vehículos se generan en los bordes
- Los vehículos desaparecen al llegar a su destino
- Calles son aristas y las intersecciones son nodos



Parámetros fijos

- Cantidad de manzanas (Largo y ancho): 19 X 19
- Cantidad de vehículos: 8000
- Longitud del vehículo (m): 5m
- Tiempo de reacción (s): 1.5s
- Aceleración máxima: 0.73 m/s^2
- Desaceleración cómoda: -1.67 m/s^2
- Exponente de aceleración: 4
- Distancia deseada entre vehículos: 2m
- Tiempo máximo de simulación (s): 3600s
- Tiempo esperado para generar todos los vehículos (s): 1200s
- Time step Δt (s): 0.2s
- Longitud zona roja: 15m
- Longitud zona amarilla: 30m
- Multiplicador de velocidad en zona amarilla ($0 < a < 1$): 0.75

Parámetros variables

- Velocidad máxima deseada (m/s): v_0
- Costo de giro: ω
- Cantidad de vehículos: N

Observables

- Proporción de retraso:

$$\bar{\tau} = \frac{\sum_{i=0}^{N} \tau(i)}{N}$$

$$\tau(i) = \frac{t_{recorrido}(i)}{t_{min}(i)}$$

$$t_{min}(i) = \frac{d_i}{v_0}$$

Observables

- Promedio de velocidad promedio:

$$\bar{v} = \frac{\sum_{i=0}^{N} \bar{v_i}}{N}$$

$$\bar{v_i} = \sum_t \left(v_i[t] * \frac{\Delta t}{t_i} \right)$$

- Proporción de $\overline{v}$ con velocidad deseada

$$\bar{V} = \frac{\bar{v}}{v_0}$$

Observables

- Tiempo frenado promedio:

$$\bar{s} = \frac{\sum_{i=0}^{N} \bar{s_i}}{N}$$

$$\bar{s_i} = \sum_t \left(s_i[t] * \frac{\Delta t}{t_i} \right)$$

$$s_i[t] = \begin{cases} 1 & \text{si } v_i[t] < \epsilon \quad o \quad i \text{ está en una zona roja prendida} \\ 0 & \text{en otro caso} \end{cases}$$

*Se tomó ε = 0.01 (m/s)

Observables

- Promedio de momento de generación del último vehículo:

$$\bar{\theta}(b) = \frac{\sum_{i=0}^{S} Max(b,i)}{S}$$

- Promedio de Porcentaje de vehículos generados:

$$\%\bar{C} = \sum_{i} \left(\frac{n_i}{N}\right) * 100$$

Set de simulaciones 1: Variación de N

- $N=\{2000,3000,4000,6000,8000,10000,12000\}$
- $\omega = 0$ constante
- $v_0 = 11{,}11$ m/s
- 6 repeticiones por valor

Set de simulaciones 2: Variación V_0

- v_0={ {5,55},{11,11},{16,67},{22,22}} (m/s)
- $\omega = 0$
- N = 8000
- 8 repeticiones por valor

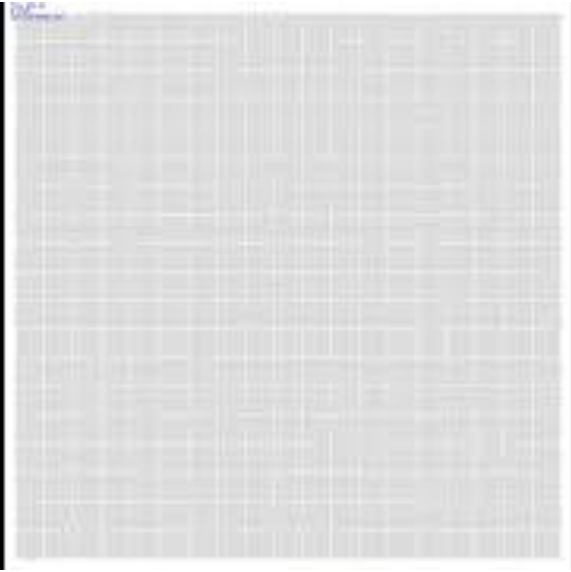
Set de simulaciones 3: Variación de ω

- ω ={ 0, 25 , 50 ,75}
- v_0= 11,11 m/s
- N = 8000
- 8 repeticiones por valor

Resultados

Primer grupo de simulaciones

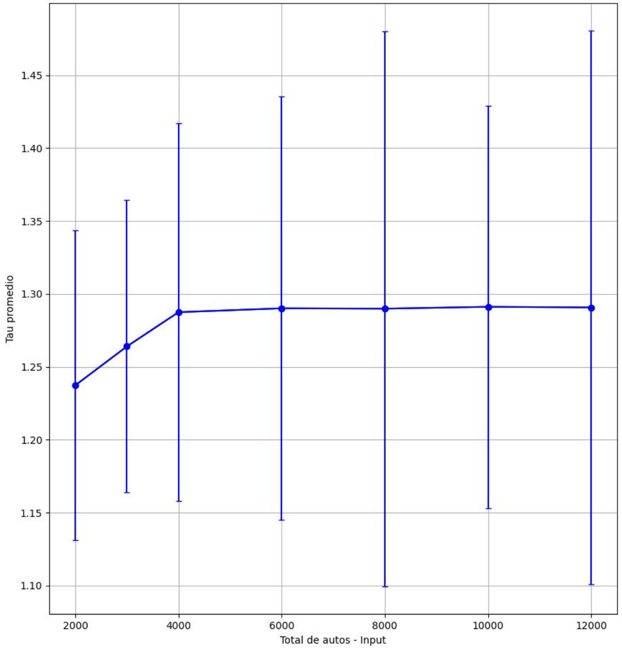
N = 2000



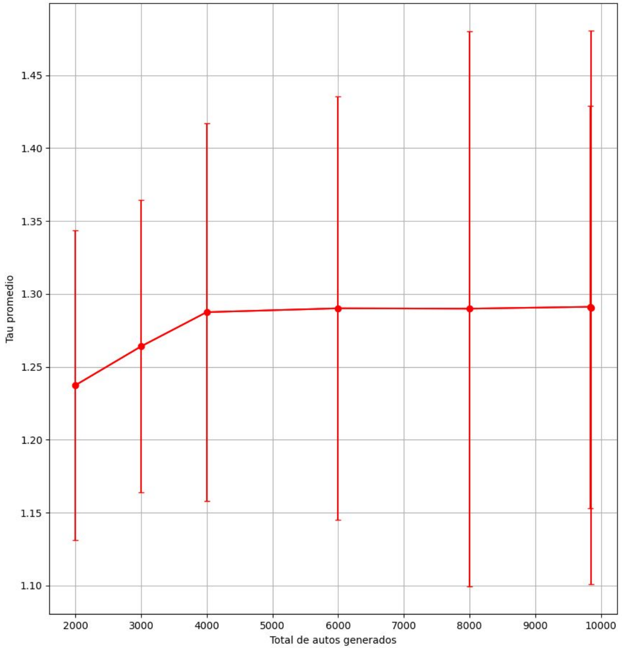
N = 12000

τ promedio

τ promedio vs #Autos objetivo

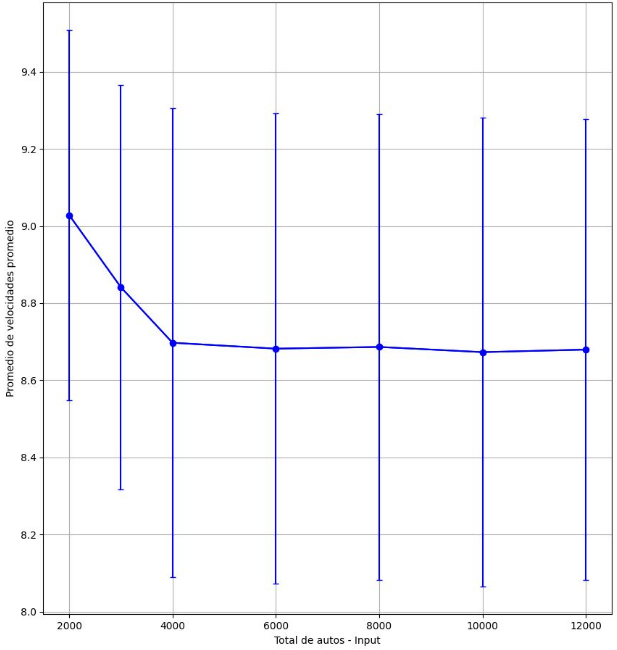


τ promedio vs #Autos generados



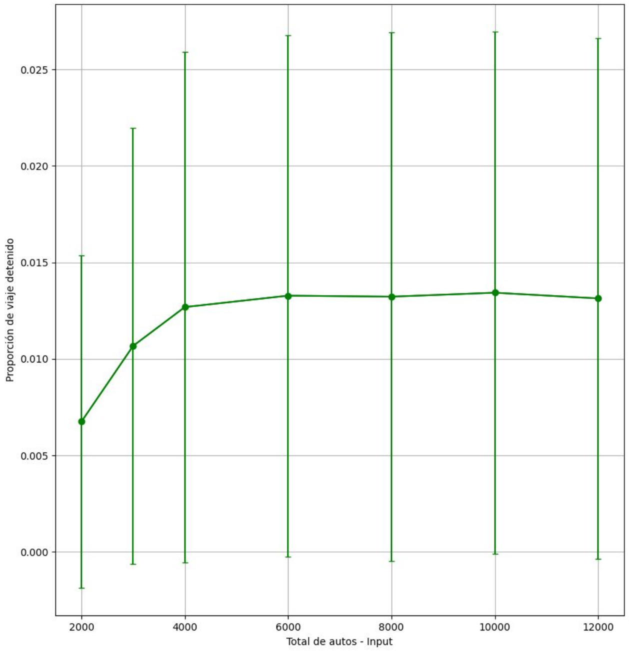
Promedio velocidades

$\overline{v}$ vs #Autos objetivo



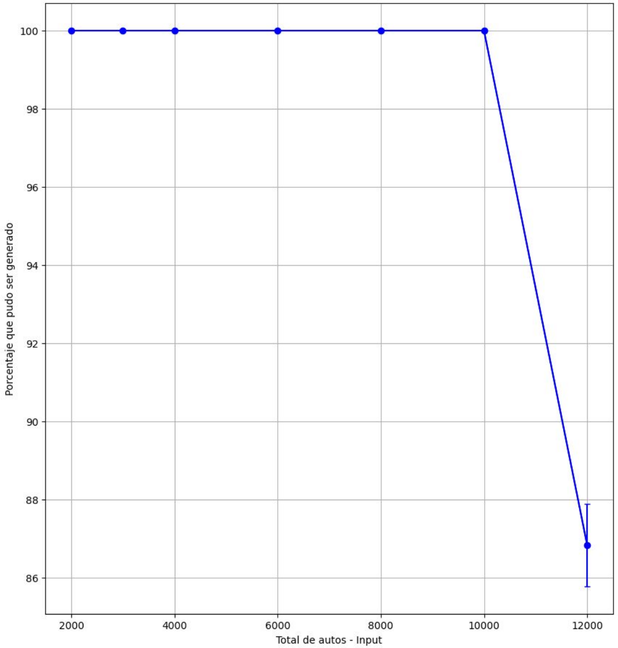
Tiempo frenado

Proporción de tiempo frenado vs #Autos objetivo

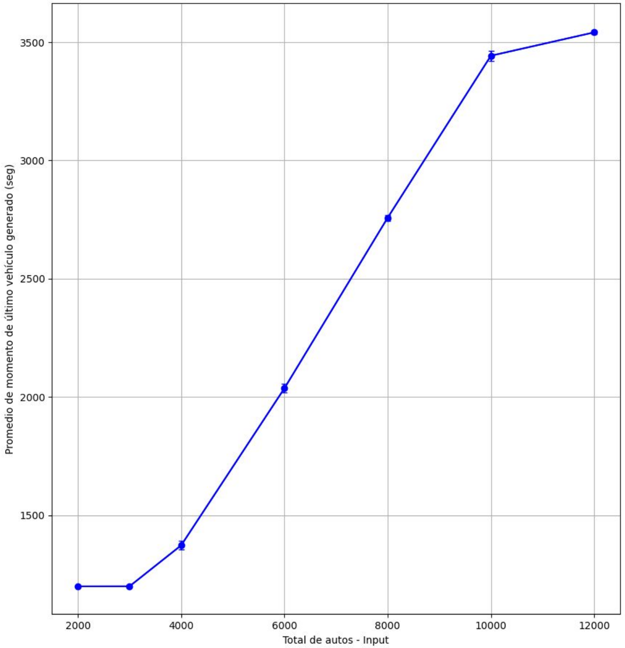


Resultados de la generación

Porcentaje de autos generados vs #Autos objetivo

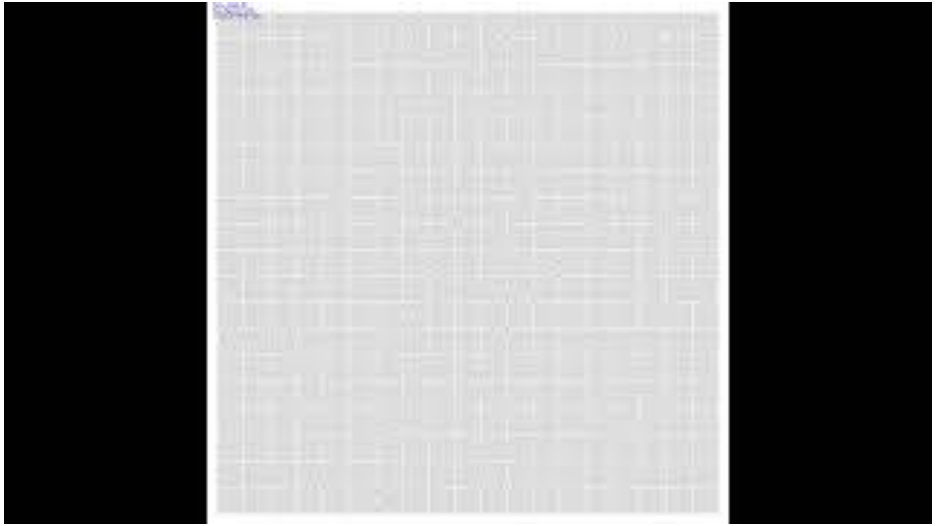


Momento de generación de último auto vs #Autos objetivo

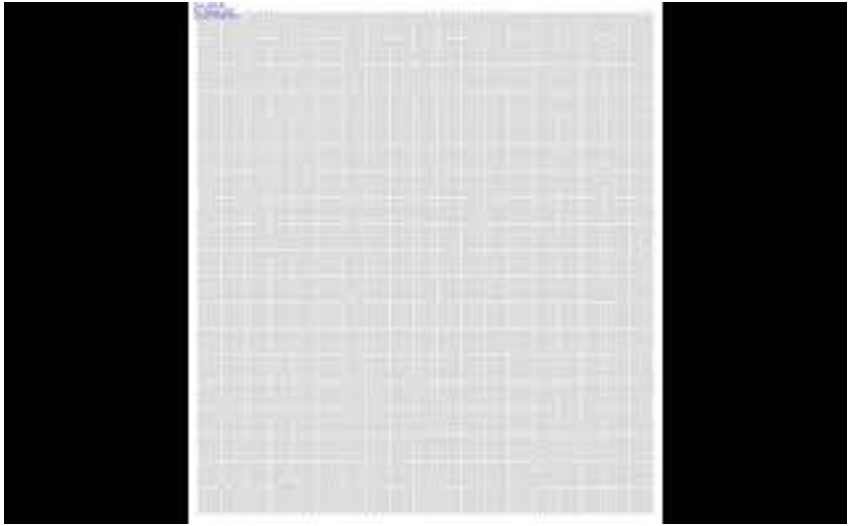


Segundo grupo de simulaciones

$v_0 = 5,55$ m/s

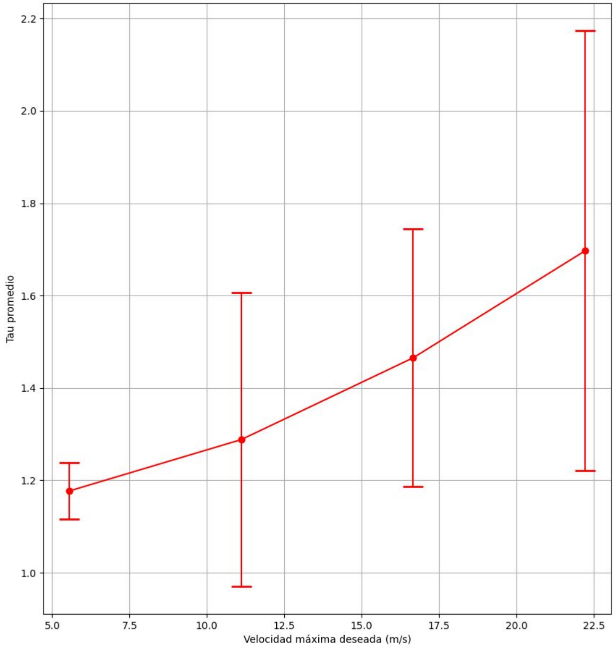


$v_0 = 22.22$ m/s

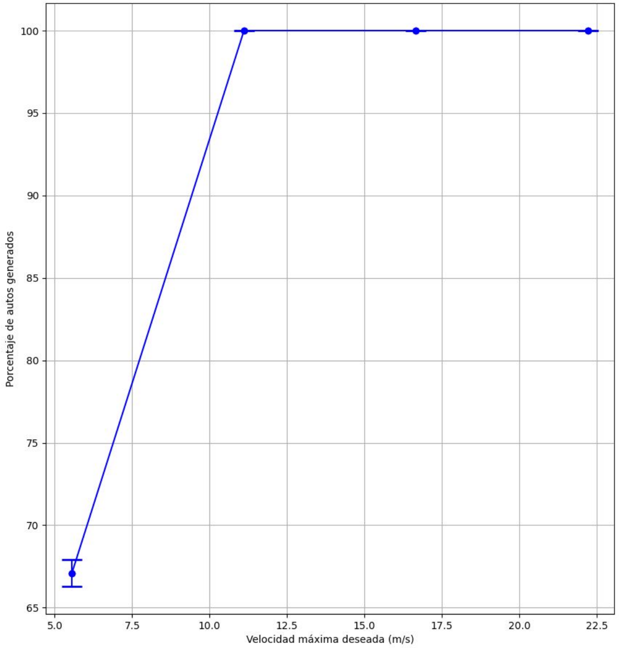


τ promedio

τ promedio vs v_0

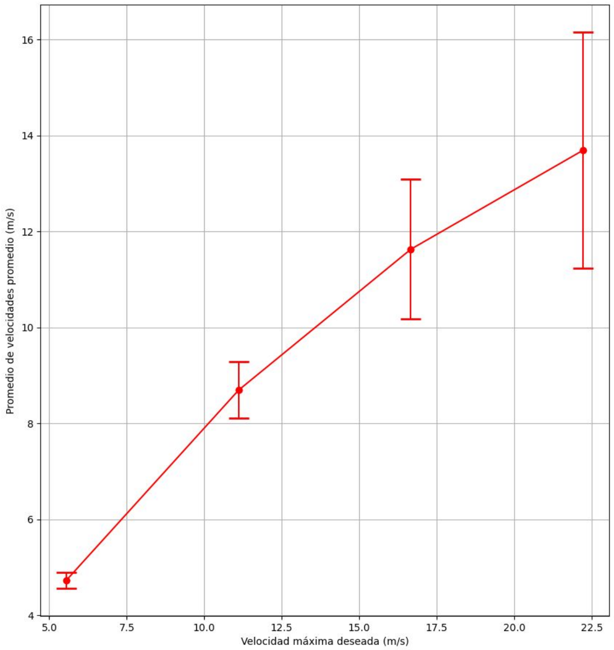


Porcentaje de vehículos generados vs v_0

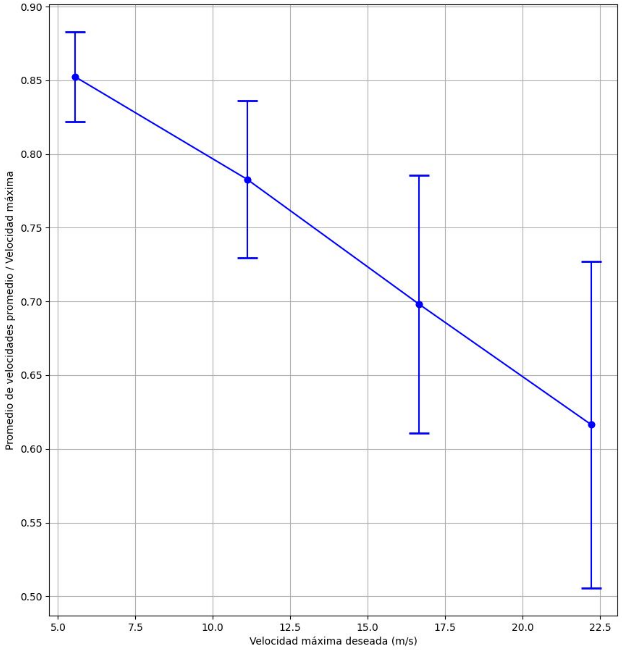


Promedio velocidades

$\overline{v}$ vs v_0

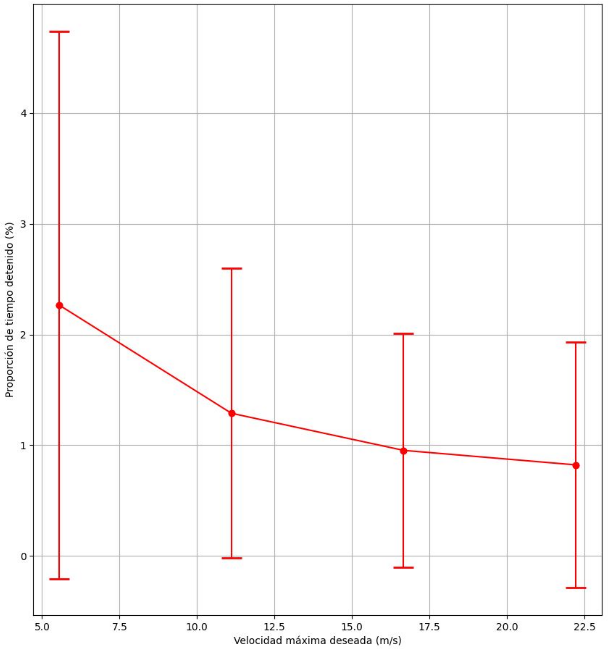


$\overline{V}$ vs v_0



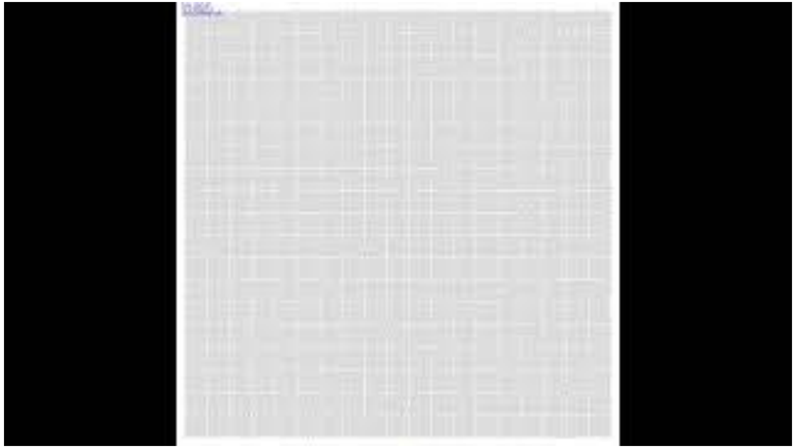
Tiempo frenado

Porcentaje de tiempo frenado vs v_0

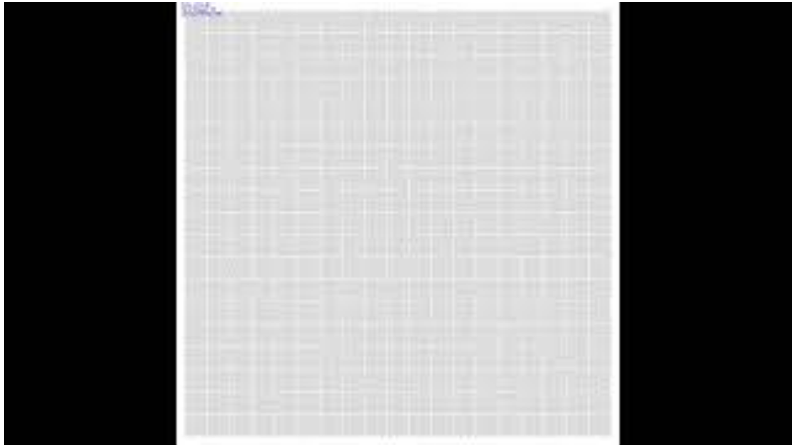


Tercer grupo de simulaciones

ω = 0

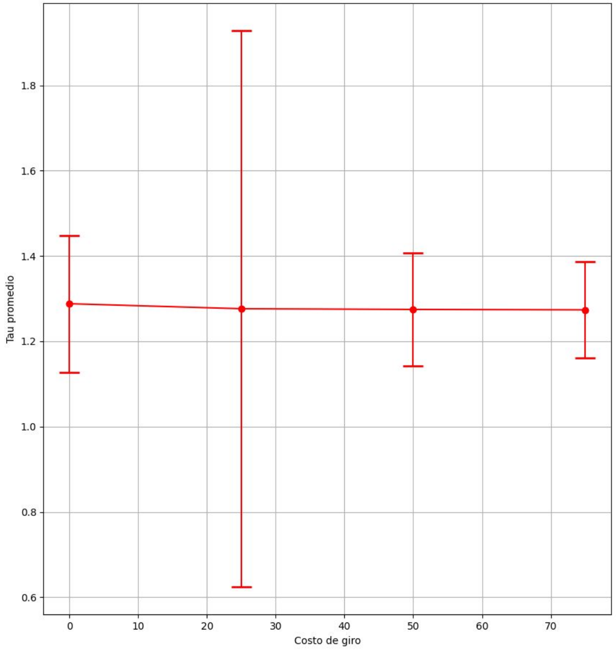


ω = 75

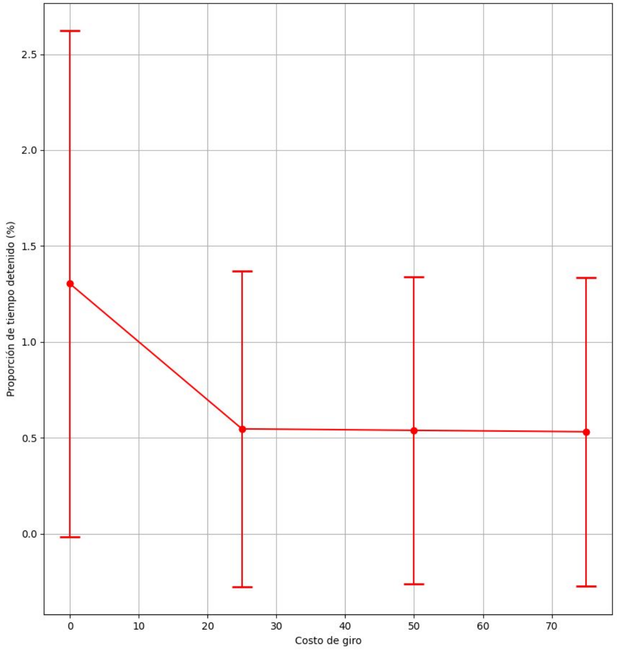


τ promedio y tiempo frenado

τ promedio vs ω

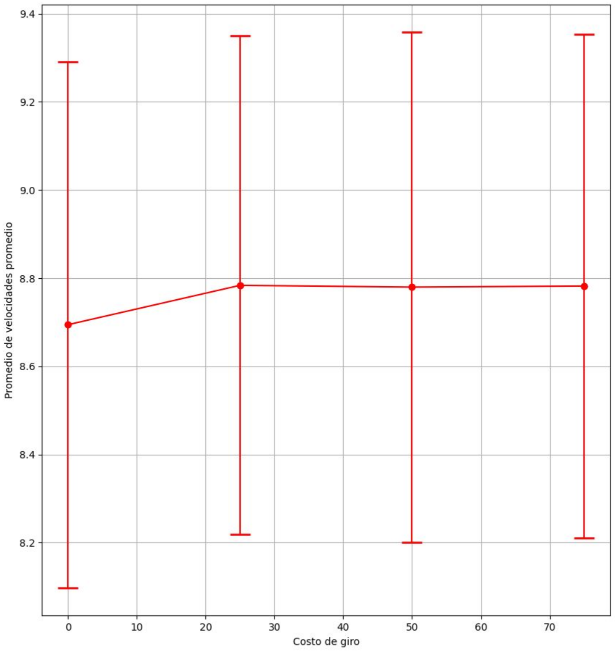


Porcentaje de tiempo frenado vs ω

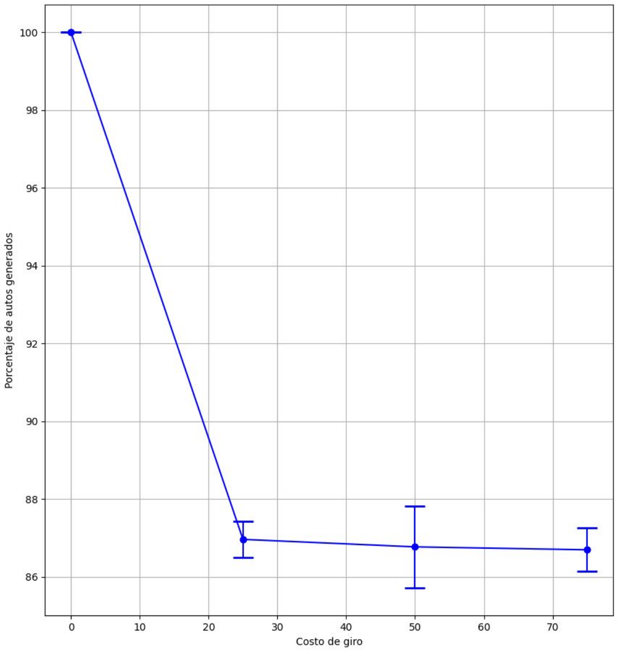


Promedio velocidades y autos generados

$\bar{v}$ vs ω



Porcentaje de autos generados vs ω



Conclusiones

Conclusiones

- Con los tiempos hay una capacidad de generar hasta 10000 vehículos aproximadamente.
- Generar muchos vehículos impacta negativamente al tiempo de ruta con respecto al tiempo óptimo.
- La disminución de $\overline{V}$ al aumentar el v_0 es coherente debido a que la aceleración máxima es constante.
- Se puede correlacionar la disminución en $\overline{V}$ con un aumento en el $\overline{\tau}$.
- Más costo al giro lleva a % stops, sabiendo que al girar hay una posibilidad de perder prioridad esto es lógico que suceda.
- El costo de giro impacta directamente en el % generado.

Muchas gracias!