Analisis Kemacetan Lalu Lintas: Pendekatan Sistem Kompleks untuk Memahami Dinamika Pergerakan Kendaraan

Hamzah Firoos Fauzi, Nicholas Fabian Rustam, Muhamad Daffa Fawwaz F J, Rizanty Heningtyas
Fakultas Matematika dan Ilmu Pengetahuan Alam
Email: 10221005@mahasiswa.itb.ac.id, 10221054@mahasiswa.itb.ac.id, 10221040@mahasiswa.itb.ac.id, 10222097@mahasiswa.itb.ac.id

Abstrak

Sistem kompleks merujuk pada kumpulan elemen yang saling berinteraksi, menciptakan pola, dan perilaku yang sulit diprediksi. Dalam dinamika sistem ini, perubahan pada satu bagian dapat memiliki dampak kompleks pada keseluruhan. Dalam hal ini, analisis kemacetan lalu lintas dengan pendekatan sistem kompleks melibatkan pemahaman mendalam terhadap interaksi dinamis antara berbagai elemen dalam jaringan transportasi. Pada laporan ini juga akan dibahas pendekatan sistem kompleks untuk lalu lintas menggunakan analisis sederhana dan analisis lebih lanjut dengan simulasi program serta melihat perilakunya dari grafik keluaran.

Kata kunci: Kemacetan, lalu lintas, simulasi, sistem kompleks

1. Pendahuluan

Perilaku kolektif yang kompleks juga dapat muncul dari interaksi elemen-elemen sistem yang memang berperilaku dengan cara yang terarah. Dalam beberapa kasus, perilaku kolektif ini bahkan dapat terlihat berlawanan dengan tujuan yang mendorong elemen-elemen yang saling berinteraksi ini. Terjadinya kemacetan lalu lintas dalam arus kendaraan yang bergerak merupakan salah satu contohnya.

2. Formulasi Model

Pada model kemacetan lalu lintas, desain model dasar yang digunakan secara konseptual cukup sederhana. Jika sederet mobil yang berjumlah N bergerak dalam arah yang sama sepanjang laju satu jalur, maka seseorang yang mengemudikan mobil akan memperlambat kecepatan mobil jika mobil yang dikemudikan memiliki jarak terlalu dekat dengan mobil yang ada di depannya, kemudian seseorang akan mempercepat kecepatan mobil jika mobilnya dan mobil yang ada di depannya memiliki jarak yang cukup jauh. Posisi dan kecepatan mobil ke-k pada waktu t_n dilambangkan dengan x_k^n dan v_k^n (k=0, …, N-1) yang memenuhi aturan berikut.

I. Jarak δ antar mobil dapat dihitung dengan

$$\delta = x_{k+1}^n - x_k^n \quad (1)$$

II. Jika $\delta < 5$, mobil akan memperlambat dengan

$$v_k^{n+1} = v_k^n - 3 \quad (2)$$

III. Jika $\delta > 5$, mobil akan mempercepat dengan

$$v_k^{n+1} = v_k^n + 1 \quad (3)$$

IV. Kecepatan harus ada pada rentang [0, 10] dengan nilai 10 merupakan batas kecepatan maksimum dan nilai 0 merupakan batas kecepatan minimum (dalam hal ini kendaraan diamsusikan bisa diperlambat, tapi tidak sampai mundur).

V. Setiap kendaraan bergerak dengan kecepatan yang sama dengan iterasi yang diberikan sebagai berikut.

$$x_k^{n+1} = x_k^n + v_k^n \times \Delta t \quad (4)$$

VI. Tentunya di jalan tidak selalu ideal dan bisa saja ada hambatan eksternal seperti orang menyebrang jalan, beberapa pengendara yang mengerem ($k = r$):

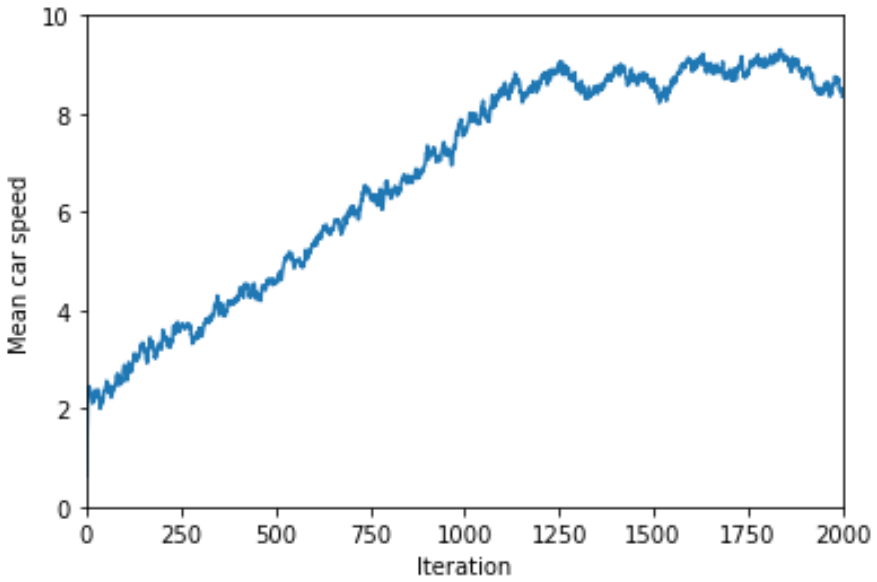
$$v_r^{n+1} = v_r - 3, \qquad r \in [0, N-1] \quad (5)$$

3. Simulasi Sederhana

Dalam hal ini, untuk model pengereman yang terjadi secara acak dan secara independen dari jarak mobil ke depan dengan *Phyton source* sebagai berikut.

```
# DISCRETE TRAFFIC MODEL ON A ONE-WAY STRAIGHT ROAD
import numpy as np
import matplotlib.pyplot as plt
#------
N=300
p_bozo=0.1
n_iter=2000
#------
v=np.zeros(N)
x=np.zeros(N)
mean_v=np.zeros(n_iter)
x[0]=1
for k in range(1, N):
    x[k]=x[k-1]+np.floor(np.random.uniform(3.,14.))

for iterate in range(0, n_iter) :
    for k in range (0, N-1):
        dx=x[k+1]-x[k]
        if dx < 5:
            v[k]=max(0, v[k]-3)
        if dx > 5:
            v[k]=min(10, v[k]+1)
        if x[N-1]-x[N-2] <= 10:
            v[N-1]=min(10,v[N-1]+1)

    for k in range(0,N):
        if np.random.random() <= p_bozo:
            v[k]=max(0, v[k]-3)

    for k in range(0, N-1):
        x[k]=min(x[k]+v[k], x[k+1]-1)
    x[N-1]+=v[N-1]
```

```
    mean_v[iterate]=(v.sum())/N
    print("iteration {0}, mean speed {1}.".format(iter,mean_v[iterate]))

plt.plot(range(0, n_iter), mean_v)
plt.axis([0, n_iter, 0., 10.])
plt.xlabel('Iteration')
plt.ylabel('Mean car speed')
plt.show()
```

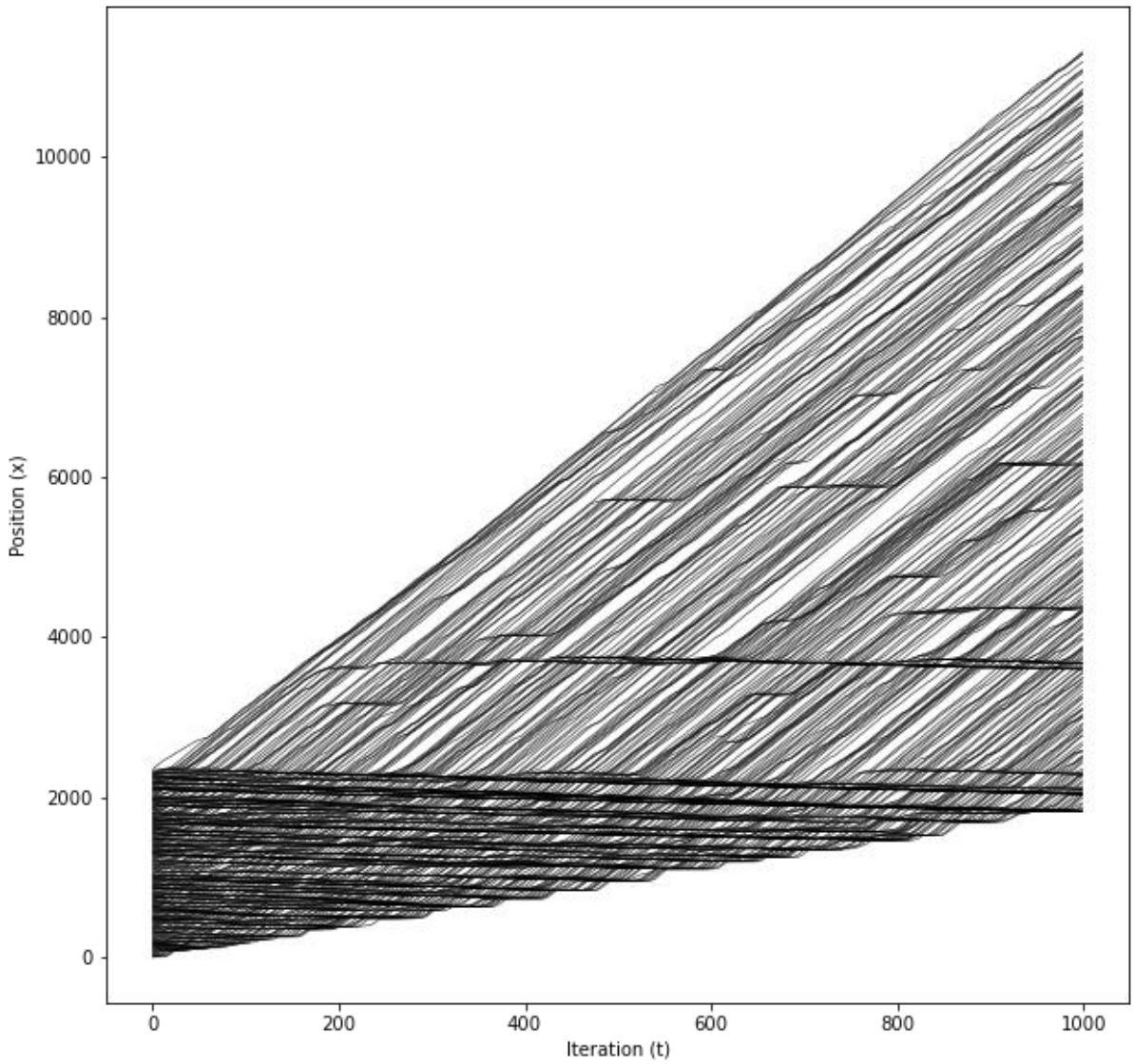


Gambar 1. Grafik rata-rata kecepatan terhadap waktu

Source Phyton di atas memenuhi aturan berikut.

I. Simulasi ini disusun di sekitar loop temporal luar yang mencakup tiga loop dalam berurutan di atas N mobil.
II. Posisi mobil diinisialisasi sebagai kenaikan positif bernilai acak dalam kisaran $3 \leq x_{k+1} - x_k \leq 17$, untuk jarak antar mobil rata-rata 10 unit. Prosedur ini memastikan bahwa $x_1 < x_2 < x_3 < \cdots < x_N$.
III. Perubahan kecepatan mobil dihitung untuk semua mobil di dua loop dalam pertama, dan baru setelah itu posisi mobil dihitung dan *array* x diperbarui, di loop dalam ketiga.
IV. Uji keamanan menggunakan fungsi Python min dan max memastikan bahwa kecepatan tidak boleh melebihi 10 atau turun di bawah 0.
V. Dipastikan juga tidak ada mobil yang bisa mendekat lebih dari 1 unit ke mobil di depan.
VI. Mobil nomor N yang berada di depan, tidak memiliki mobil di depannya; oleh karena itu, mobil ini menyesuaikan kecepatannya sesuai dengan jarak ke mobil berikutnya.

```
import numpy as np
import matplotlib.pyplot as plt
#------
N=300
p_bozo=0.1
n_iter=1000
#------
v=np.zeros(N)
x=np.zeros(N)
x[0]=1
for k in range(1, N):
  x[k]=x[k-1]+np.floor(np.random.uniform(3.,14.))

positions = np.zeros((n_iter, N))

# Update the positions of each car over time
for iterate in range(n_iter):
    for k in range(N-1):
        dx = x[k+1] - x[k]
        if dx < 5:
            v[k] = max(0, v[k] - 3)
        if dx > 5:
            v[k] = min(10, v[k] + 1)
        if x[N-1] - x[N-2] <= 10:
            v[N-1] = min(10, v[N-1] + 1)

    for k in range(N):
        if np.random.random() <= p_bozo:
            v[k] = max(0, v[k] - 3)

    for k in range(N-1):
        x[k] = min(x[k] + v[k], x[k+1] - 1)
    x[N-1] += v[N-1]

    positions[iterate, :] = x  # Store the positions of each car for this iteration

# Plot the position of each car over time
plt.figure(figsize=(10, 10))
for car in range(N):
    plt.plot(range(n_iter), positions[:, car], color='black', linewidth=0.5)
plt.xlabel('Iteration (t)')
plt.ylabel('Position (x)')
plt.show()
```

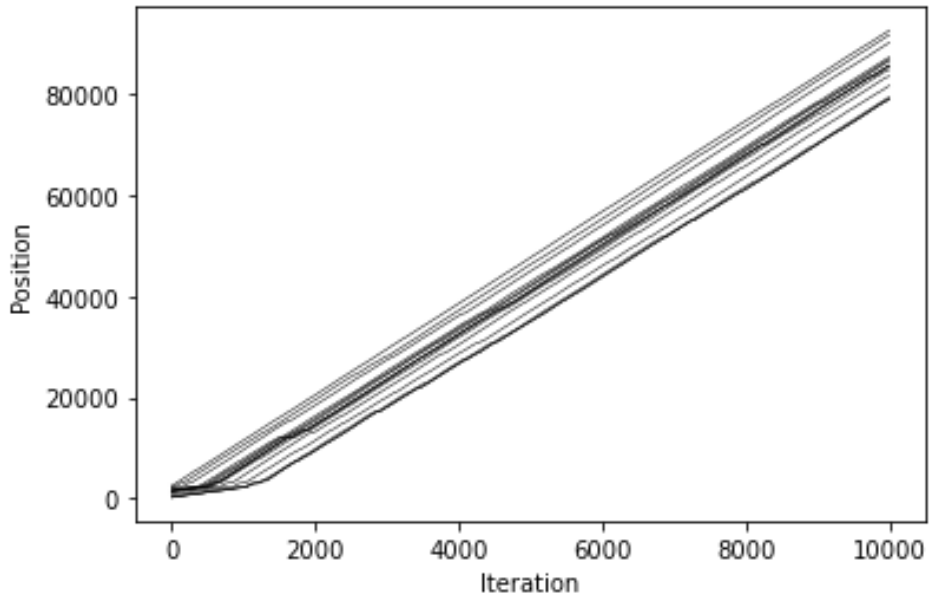


Gambar 2. Grafik posisi terhadap waktu untuk semua mobil

Pada Gambar 2 garfik paling bawah menggambarkan kendaraan ke $N = 1$ dan grafik paling atas menggambarkan kendaraan ke $N = 300$. Jika disimulasikan, kendaraan ke $N = 1$ terus bergerak (grafik tidak pernah mendatar). Sementara itu, kendaraan ke $N = 300$ grafiknya mendatar karena harus menunggu kendaraan di depannya bergerak begitu pula kendaraan di depannya lagi. Terdapat beberapa kendaraan acak yang menunjukkan respon posisi terhadap waktu yang mendatar, mengacu pada (5) terdapat beberapa hal (kendala eksternal) yang menyebabkan kendaraan tidak bergerak.

Pada rentang interasi yang cukup lama ($t \gg$) akan menunjukkan tren yang konstan bergerak (tidak mendatar atau berhenti). Kita dapat melihatkan dengan memperpanjang rentang iterasi dan mengambil sampel sebanyak 15 kendaraan.

```
import numpy as np
import matplotlib.pyplot as plt
#------
N=300
p_bozo=0.1
n_iter=10000
#------
v=np.zeros(N)
x=np.zeros(N)
x[0]=1
for k in range(1, N):
   x[k]=x[k-1]+np.floor(np.random.uniform(3.,14.))

positions = np.zeros((n_iter, N))

# Randomly select 15 cars
selected_cars = np.random.choice(N, 15, replace=False)

# Create an array to store the positions of each selected car over time
selected_positions = np.zeros((n_iter, 15))

# Update the positions of each car over time
for iterate in range(n_iter):
    for k in range(N-1):
        dx = x[k+1] - x[k]
        if dx < 5:
            v[k] = max(0, v[k] - 3)
        if dx > 5:
            v[k] = min(10, v[k] + 1)
        if x[N-1] - x[N-2] <= 10:
            v[N-1] = min(10, v[N-1] + 1)

    for k in range(N):
        if np.random.random() <= p_bozo:
            v[k] = max(0, v[k] - 3)

    for k in range(N-1):
        x[k] = min(x[k] + v[k], x[k+1] - 1)
    x[N-1] += v[N-1]

    positions[iterate, :] = x  # Store the positions of each car for this iteration
    selected_positions[iterate, :] = x[selected_cars]

# Plot the position of each car over time
for i, car in enumerate(selected_cars):
    plt.plot(range(n_iter), selected_positions[:, i], color='black', linewidth=0.5)
plt.xlabel('Iteration')
plt.ylabel('Position')
plt.show()
```



Gambar 3. Grafik posisi terhadap waktu untuk semua mobil

Dari Gambar 3 dibuktikan bahwasannya kendaraan terus bergerak tanpa mengalami perlambatan yang signifikan (kalaupun ada perlambatan terjadi dalam waktu yang sangat singkat).

4. Perilaku Model

Tinjauan kuantitatif diperlukan dalam memahami model ini berperilaku. Dua besaran global yang menjadi tajuk utama kali ini adalah kecepatan rata-rata dan jarak rata-rata antara mobil-mobil yang berurutan dalam barisan.

Kecepatan rata-rata untuk semua kendaraan dinyatakan sebagai:

$$\langle v \rangle = \frac{1}{N}\sum_{k=0}^{N-1} v_k^n \tag{6}$$

Dan jarak rata-rata antara mobil-mobil yang berurutan dalam barisan dinyatakan sebagai:

$$\langle \delta \rangle = \frac{1}{N-1}\sum_{k=0}^{N-2}(x_{k+1}^n - x_k^n) = \frac{x_{N-1}^n - x_0^n}{N-1} \tag{7}$$

Kepadatan rata-rata mobil adalah rasio kebalikan dari ekspresi:

$$\rho = \frac{N-1}{x_{N-1}^n - x_0^n} \tag{8}$$

Dengan mengetahui persamaan (6) dan (8), dapat dihitung *car flux* (ϕ):

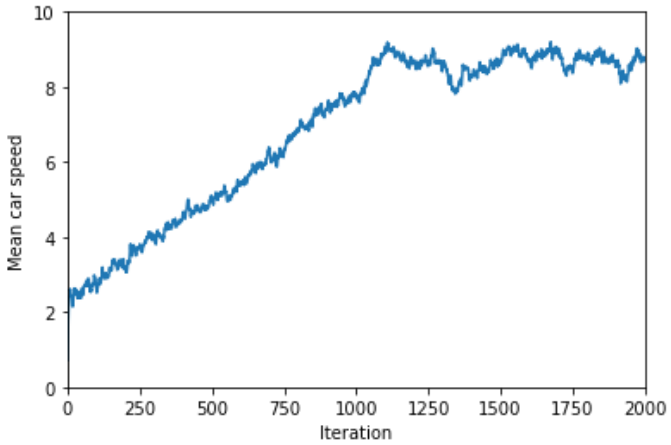
$$\phi = \rho \times \langle v \rangle \tag{9}$$

Car flux mengukur jumlah rata-rata mobil yang melewati x^* per satuan waktu.

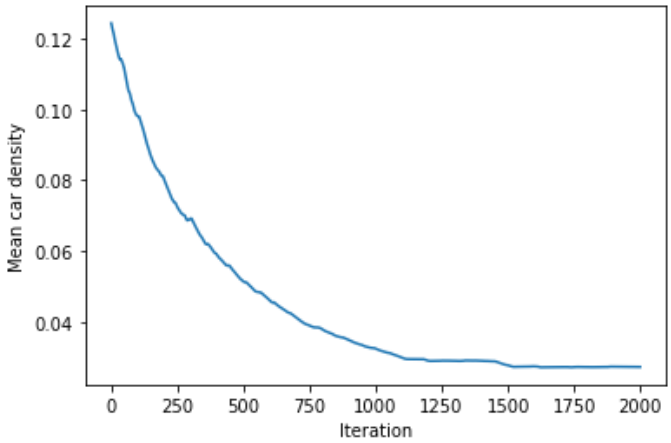
```
# DISCRETE TRAFFIC MODEL ON A ONE-WAY STRAIGHT ROAD
import numpy as np
import matplotlib.pyplot as plt
#------
N=300
p_bozo=0.1
n_iter=2000
#------
v=np.zeros(N)
x=np.zeros(N)
mean_v=np.zeros(n_iter)
```

```
mean_density = np.zeros(n_iter)
x[0]=1
for k in range(1, N):
    x[k]=x[k-1]+np.floor(np.random.uniform(3.,14.))

for iterate in range(0, n_iter) :
    for k in range (0, N-1):
        dx=x[k+1]-x[k]
        if dx < 5:
            v[k]=max(0, v[k]-3)
        if dx > 5:
            v[k]=min(10, v[k]+1)
        if x[N-1]-x[N-2] <= 10:
            v[N-1]=min(10,v[N-1]+1)

    for k in range(0,N):
        if np.random.random() <= p_bozo:
            v[k]=max(0, v[k]-3)

    for k in range(0, N-1):
        x[k]=min(x[k]+v[k], x[k+1]-1)
    x[N-1]+=v[N-1]

    mean_v[iterate]=(v.sum())/N
    #print("iteration {0}, mean speed {1}.".format(iter,mean_v[iterate]))
    mean_density[iterate] = (N - 1) / (x[N-1] - x[0])
    #print("iteration {0}, mean density {1}.".format(iter,mean_density[iterate]))
    # Update the positions and speeds of cars for the next iteration
    # Calculate car flux over time
    car_flux = mean_density * mean_v
    #car_flux = np.cross(mean_density, mean_v)
    #print("iteration {0}, car_flux {1}.".format(iter,car_flux[iterate]))

# Plot car flux over time
plt.plot(range(0, n_iter), car_flux)
plt.xlabel('Iteration')
plt.ylabel('Car Flux')
plt.show()

# Plot mean density of cars over time
plt.plot(range(0, n_iter), mean_density)
plt.xlabel('Iteration')
plt.ylabel('Mean car density')
plt.show()

# Plot mean speed of cars over time
plt.plot(range(0, n_iter), mean_v)
plt.axis([0, n_iter, 0., 10.])
plt.xlabel('Iteration')
```

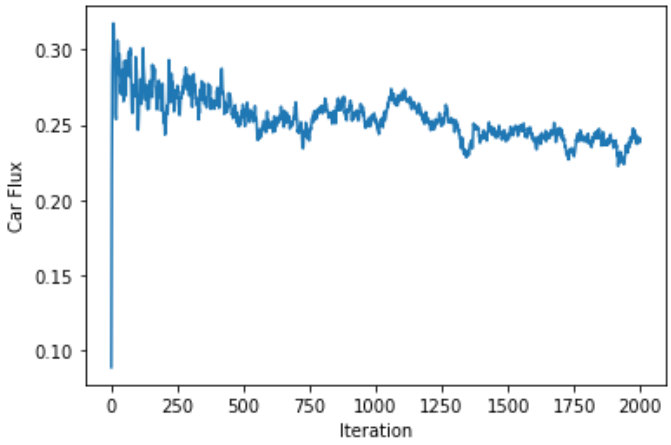
```
plt.ylabel('Mean car speed')
plt.show()
```



Gambar 4. Grafik kecepatan rata-rata kendaraan terhadap iterasi waktu



Gambar 5. Grafik kepadatan rata-rata kendaraan terhadap iterasi waktu



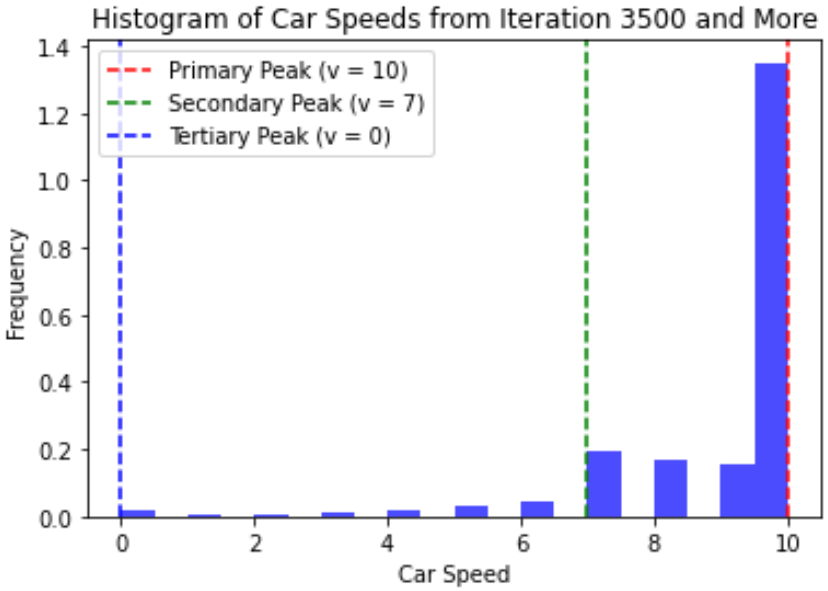
Gambar 6. Grafik *car flux* terhadap iterasi waktu

Gambar 4, Gambar 5, dan Gambar 6 menunjukkan variasi keadaan bervariasi hingga $t \simeq 1300\,s$. Namun, untuk kepadatan rata-rata dan fluks mobil hanya benar-benar stabil mulai sekitar $t \simeq 2000\,s$. Hal ini mengindikasikan bahwa untuk mencapai keadaan stasioner secara statistik masih membutuhkan waktu yang cukup lama setelah transisi fasa.

Sifat penting dari keadaan stasioner secara statistik ini adalah karakteristik globalnya, seperti kecepatan rata-rata dan kepadatan tidak bergantung pada kondisi awal simulasi. Kekacauan kemacetan lalu lintas yang sangat parah yang mencirikan apa yang disebut sebagai fase padat dalam simulasi ini tentu saja menunjukkan bahwa kondisi awal yang diterapkan di sini masih jauh dari optimal dalam hal melancarkan lalu lintas.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import gaussian_kde

#------
N = 300
p_bozo = 0.1
n_iter = 10000
#------
v = np.zeros(N)
x = np.zeros(N)
x[0] = 1
for k in range(1, N):
    x[k] = x[k-1] + np.floor(np.random.uniform(3., 14.))

positions = np.zeros((n_iter, N))
car_speeds = []

# Update the positions of each car over time
for iterate in range(n_iter):
    for k in range(N-1):
        dx = x[k+1] - x[k]
        if dx < 5:
            v[k] = max(0, v[k] - 3)
        if dx > 5:
            v[k] = min(10, v[k] + 1)
        if x[N-1] - x[N-2] <= 10:
            v[N-1] = min(10, v[N-1] + 1)

    for k in range(N):
        if np.random.random() <= p_bozo:
            v[k] = max(0, v[k] - 3)

    for k in range(N-1):
        x[k] = min(x[k] + v[k], x[k+1] - 1)
    x[N-1] += v[N-1]
```

```
    positions[iterate, :] = x  # Store the positions of each car for this iteration
    if iterate >= 3500:
        car_speeds.extend(v)  # Store the speeds of all cars at each temporal iteration

# Calculate the PDF of car speeds
car_speeds = np.array(car_speeds)
kde = gaussian_kde(car_speeds)
x_vals = np.linspace(0, 15, 1000)
y_vals = kde(x_vals)

# Plot histogram of car speeds
plt.hist(car_speeds, bins=20, density=True, alpha=0.7, color='b')
plt.axvline(x=10, color='r', linestyle='--', label='Primary Peak (v = 10)')
plt.axvline(x=7, color='g', linestyle='--', label='Secondary Peak (v = 7)')
plt.axvline(x=0, color='b', linestyle='--', label='Tertiary Peak (v = 0)')
plt.xlabel('Car Speed')
plt.ylabel('Frequency')
plt.title('Histogram of Car Speeds from Iteration 3500 and More')
plt.legend()
plt.show()
```



Gambar 7. Grafik fungsi rapat probabilitas terhadap kecepatan

Evolusi menuju kecepatan rata-rata mobil yang kuat (dan kepadatan) juga menunjukkan bahwa sebagian besar mobil pada akhirnya melakukan perjalanan hampir sepanjang waktu pada atau mendekati kecepatan tersebut. Dengan kata lain, distribusi kecepatan mobil adalah seperti Gaussian dan berpusat pada nilai rata-ratanya $\langle v \rangle$. Sementara itu, Gambar 7 tidak menunjukkan hal yang demikian. Gambar 7 menunjukkan fungsi rapat probabilitas yang dibangun dari semua mobil pada semua iterasi jauh di dalam fase yang stasioner secara statistik ($t > 3500$). Distribusi ini sama sekali tidak berbentuk Gaussian atau bahkan tidak simetris terhadap nilai rata-ratanya (segmen garis vertikal pada $\simeq 8{,}6$),

melainkan mencakup seluruh rentang yang dimungkinkan dengan puncaknya pada $v = 10$dan puncak sekunder pada $v = 7$ dan $v = 0$. Puncak $v = 7$ merupakan konsekuensi langsung dari aturan pengereman mengacu pada persamaan (2) yang bekerja pada puncak primer pada $v = 10$.

Gambar 7 lebih lanjut menggambarkan bahwa lebih dari 60% mobil menghabiskan waktunya untuk bergerak dengan kecepatan $v = 10$ dan sisanya kurang dari itu bisa saja terjebak dalam kemacetan lalu lintas.

5. **Kemacetan Lalu Lintas sebagai Longsoran**
   Penumpukan kemacetan lalu lintas mirip seperti longsor akibat pengereman yang beruntun. Kemacetan adalah objek semu karena mobil terus menerus menumpuk di bagian belakang kemacetan dan yang lainnya dipindahkan di depan.
6. **Lalu Lintas Mobil sebagai Sistem SOC**
   Lalu lintas mobil dapat dianalisis sebagai Sistem Self-Organizing Criticality (SOC) yang merupakan konsep dalam teori sistem kompleks. Dalam konteks SOC, sistem dikarakterisasi oleh perilaku emergen yang terjadi karena adanya keterkaitan antar elemen-elemen individu dalam sistem tersebut.

   Skala invariansi adalah ciri khas sistem kritis. Model lalu lintas menunjukkan sifat-sifat utama sistem kritis, di mana gangguan kecil dapat memicu efek global seperti terjadinya kemacetan yang memengaruhi semua kendaraan.

   Tujuan utama rekayasa lalu lintas untuk mencapai kelancaran arus lalu lintas dan memastikan pengendara mencapai tujuan mereka dengan cepat. Kemacetan diidentifikasi sebagai hambatan utama yang harus diatasi. *Trade-off* antara menjaga jarak aman antar mobil untuk menghindari kemacetan dengan risiko rendah fluks atau memadatkan mobil untuk fluks tinggi dengan risiko tinggi kemacetan.

Kondisi stasioner dalam simulasi lalu lintas bahkan dengan adanya kemacetan sebenarnya dapat memaksimalkan fluks mobil. Distribusi kemacetan yang bervariasi dianggap sebagai strategi sistem untuk mengurangi dampak global dari pengereman acak yang tidak terduga. Selain itu, kemacetan lalu lintas sulit diantisipasi dengan peraturan lalu lintas sederhana yang menunjukkan kompleksitas sistem ini.

Referensi

[1] Charbonneau, Paul. (2017). *Natural Complexity: A Modeling Handbook*. Princeton: Princeton University Press