Integrative Perspectives of Social and Science Journal (IPSSJ)
ipssj.com ippsj2024@gmail.com E-ISSN: 3064-4011

Analisis Waktu Eksekusi Serial dan Paralel pada Simulasi Kendaraan Otonom Berbasis Multi-Agent Systems Menggunakan Metode Regresi Linear

Muhammad Thoriq Alamsyah[1*], Tata Sutabri[2]
[1,2]Program Studi Teknik Informatika, Institut Teknologi Indonesia
*Corresponding email: thoriqalamsyah24@gmail.com

Abstrak: Simulasi kendaraan otonom berbasis Multi-Agent Systems (MAS) membutuhkan waktu komputasi yang meningkat seiring bertambahnya jumlah agen, sementara pemrosesan paralel tidak selalu lebih cepat akibat overhead pembuatan proses. Penelitian ini bertujuan memodelkan hubungan jumlah kendaraan dengan waktu eksekusi serial dan paralel serta memperkirakan titik impas keduanya. Simulasi berbasis Python dengan agen reaktif diuji pada 10 hingga 10.000 kendaraan (lima ulangan) menggunakan empat proses pekerja, lalu dianalisis dengan regresi linear sederhana. Hasil menunjukkan R^2 sebesar 0,9998 (serial) dan 0,9996 (paralel). Titik impas model sekitar 1.123 kendaraan, sesuai pengamatan bahwa paralel mulai lebih cepat di antara 1.000 dan 2.000 kendaraan, dengan speedup tertinggi 1,38 kali. Kebaruannya, titik impas dan batas speedup dapat diperkirakan secara analitik dengan selisih model terhadap data maksimal 0,07 detik. Paralel layak dipakai pada skala besar, sedangkan serial lebih efisien pada skala kecil.
Kata kunci: simulasi kendaraan otonom, Multi-Agent Systems, pemrosesan paralel, waktu eksekusi, regresi linear

Abstract: *Autonomous vehicle simulation based on Multi-Agent Systems (MAS) requires computation time that grows with the number of agents, while parallel processing is not always faster because of process-creation overhead. This study aims to model the relationship between the number of vehicles and serial and parallel execution time, and estimate their break-even point. A Python simulation with reactive agents was tested on 10 to 10,000 vehicles (five repetitions) using four worker processes and analyzed with simple linear regression. The coefficients of determination (R^2) were 0.9998 (serial) and 0.9996 (parallel). The model's break-even point was about 1,123 vehicles, consistent with parallel gains observed between 1,000 and 2,000 vehicles, with a maximum speedup of 1.38 times. The novelty is that the break-even point and speedup limit can be estimated analytically, with a maximum model-to-data difference of 0.07 seconds. Parallel processing suits large scales, whereas serial processing is more efficient at small scales.*
Keywords: *autonomous vehicle simulation, Multi-Agent Systems, parallel processing, execution time, linear regression*

PENDAHULUAN

Kendaraan otonom banyak dikaji melalui pemodelan dan simulasi untuk menilai dampaknya (Sadid & Antoniou, 2023). Pendekatan Multi-Agent Systems (MAS) memodelkan setiap kendaraan sebagai agen yang mengambil keputusan sendiri dan berinteraksi dengan agen lain, dan telah dikaji untuk koordinasi kendaraan otonom (Cabri et al., 2023). Domain lalu lintas sesuai dengan pendekatan ini karena sistem transportasi tersebar secara geografis, berada pada lingkungan yang dinamis, dan dapat memuat ribuan entitas otonom (Bazzan & Klügl, 2014; Chen & Cheng, 2010). Semakin banyak agen, semakin besar beban komputasi simulasi. Pemrosesan paralel dapat memangkas waktu eksekusi, tetapi tidak selalu menguntungkan karena adanya biaya pembuatan dan pengelolaan proses (Ekawijana & Noviansyah, 2026).

Di bidang simulasi lalu lintas, pemodelan berbasis agen yang diparalelkan dengan Python multiprocessing melaporkan percepatan hingga sekitar empat kali (Tomás et al., 2026). Studi pada pemrosesan audio juga mengindikasikan bahwa komputasi paralel menurunkan waktu eksekusi dibandingkan proses serial, meskipun peningkatannya tidak selalu linear akibat overhead (Chyan & Sumarta, 2026). Pada simulasi lalu lintas berbasis agen berskala besar, algoritma paralel terdistribusi dilaporkan mencapai percepatan 43 kali dibandingkan satu core (Laudan et al., 2025). Regresi linear sendiri telah dipakai untuk memprediksi waktu eksekusi tugas di lingkungan cloud dengan R^2 sebesar 0,971 (Bahaswan & Pramono, 2024).

Namun, berdasarkan penelusuran penulis, belum ditemukan penelitian yang memodelkan hubungan jumlah kendaraan dengan waktu eksekusi serial dan paralel pada simulasi MAS menggunakan regresi linear. Berbeda dari penelitian yang melaporkan kinerja paralel secara deskriptif, penelitian ini memodelkan waktu eksekusi serial dan paralel dengan regresi linear sehingga titik impas dan batas speedup dapat diperkirakan secara analitik pada simulasi MAS kendaraan otonom.

Penelitian ini dilakukan untuk (1) memodelkan hubungan jumlah kendaraan dengan waktu eksekusi pada pemrosesan serial dan paralel, (2) menilai kecocokan model melalui nilai R^2 dan selisih terhadap data, (3) memperkirakan jumlah kendaraan saat pemrosesan paralel mulai lebih cepat daripada serial, dan (4) menghitung efisiensi paralel serta bagian program yang dapat diparalelkan.

LANDASAN TEORI

Multi-Agent Systems

MAS adalah sistem yang terdiri atas banyak agen otonom yang berinteraksi untuk mencapai tujuan masing-masing atau bersama (Wooldridge, 2009). Agen reaktif memilih aksi berdasarkan kondisi lingkungan saat itu tanpa model simbolik yang kompleks (Wooldridge, 2009). Teknik agen dan MAS telah diterapkan pada banyak aspek sistem lalu lintas, antara lain pemodelan dan simulasi, pengaturan rute dan manajemen kemacetan, serta kontrol lalu lintas cerdas (Chen & Cheng, 2010).

Perkembangan MAS dapat dikelompokkan menjadi era sebelum dan sesudah Large Language Model (LLM), dirangkum pada Tabel 1. Pada era pra-LLM, akar MAS berada pada riset *Distributed Artificial Intelligence* (DAI), dengan karya awal *contract net protocol* untuk pembagian tugas melalui negosiasi (Smith, 1980). Arus reaktif menolak representasi simbolik (Brooks, 1991), dan arus deliberatif Belief-Desire-Intention (BDI) (Rao & Georgeff, 1995). Agen kemudian dipadukan dengan pembelajaran (multi-agent reinforcement learning, MARL) (Buşoniu et al., 2008), termasuk untuk kendaraan otonom (Dinneweth et al., 2022) serta pengelolaan arus dan sinyal lalu lintas (Mushtaq et al., 2023). Pada era pasca-LLM, agen berkomunikasi dan bernalar dengan bahasa alami, misalnya generative agents (Park et al., 2023). Survei sistem multi-agen berbasis LLM dapat dilihat pada (Guo et al., 2024; Li et al., 2024), sedangkan untuk kendaraan otonom pada (Wu et al., 2025). Pengelompokan periode ini adalah pengelompokan penulis, bukan batas baku. Simulasi pada penelitian ini berada pada kelompok pra-LLM, yaitu agen reaktif berbasis aturan.

Tabel 1. Perkembangan Multi-Agent Systems sebelum dan setelah era LLM

Era	Periode	Karakteristik utama	Rujukan
Pra-LLM	1980-an	DAI; negosiasi dan pembagian tugas antarsimpul	Smith (1980)
Pra-LLM	1990-an	Agen reaktif tanpa representasi simbolik dan agen deliberatif BDI	Brooks (1991); Rao & Georgeff (1995)
Pra-LLM	2008 hingga kini	Agen yang belajar (MARL), termasuk untuk kendaraan otonom dan arus lalu lintas	Buşoniu et al. (2008); Dinneweth et al. (2022); Mushtaq et al. (2023)
Pasca-LLM	2023 hingga kini	Agen berbasis LLM: komunikasi bahasa alami, memori dan refleksi, simulasi dunia	Park et al. (2023); Guo et al. (2024); Li et al. (2024); Wu et al. (2025)

Pemrosesan serial dan paralel

Dari sisi komputasi, pada pemrosesan serial tugas dijalankan berurutan oleh satu proses, sedangkan pada pemrosesan paralel tugas dibagi ke beberapa proses yang berjalan bersamaan, misalnya dengan modul multiprocessing pada Python (Aziz et al., 2021). Modul ini menjalankan pekerjaan pada proses-proses terpisah, dan dokumentasinya menganjurkan agar

keadaan bersama (shared state) sebisa mungkin dihindari (Python Software Foundation, n.d.). Percepatan (speedup, S) dan efisiensi (E) dihitung dengan Persamaan (1) dan (2), dengan T_{serial} dan T_{paralel} masing-masing waktu eksekusi serial dan paralel serta p jumlah proses pekerja (Ekawijana & Noviansyah, 2026).

$$S = \frac{T_{\text{serial}}}{T_{\text{paralel}}} \tag{1}$$

$$E = \frac{S}{p} \tag{2}$$

Hukum Amdahl membatasi speedup yang dapat dicapai. Jika f adalah bagian program yang dapat diparalelkan, speedup ideal dinyatakan oleh Persamaan (3) (Hill & Marty, 2008). Dengan membalik persamaan tersebut dan menyubstitusikan S dari Persamaan (1), diperoleh estimasi f pada Persamaan (4). Bagian program yang tetap berjalan berurutan membatasi peningkatan kinerja paralel. Berapa pun jumlah proses, speedup tidak dapat melampaui $1/(1-f)$.

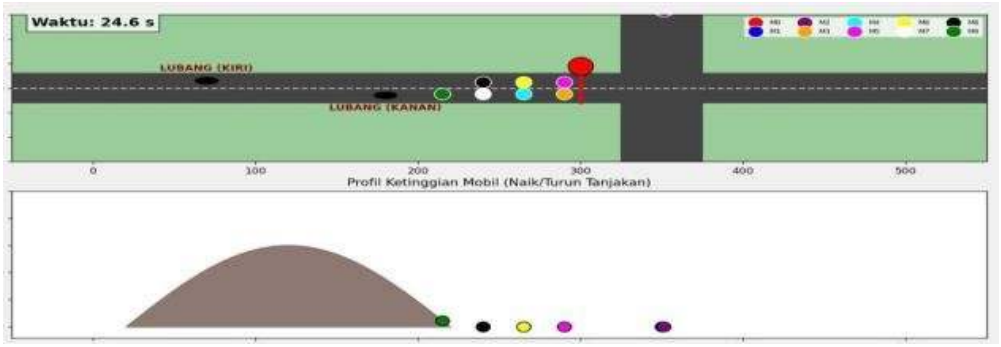
$$S(p) = \frac{1}{(1-f) + \frac{f}{p}} \tag{3}$$

$$f = \frac{1 - \frac{1}{S}}{1 - \frac{1}{p}} \tag{4}$$

METODE

Rancangan simulasi

Simulasi diimplementasikan dengan Python memakai pendekatan waktu diskrit (discrete time simulation) dan data sintetis. Simulasi memandang sistem lalu lintas sebagai sekumpulan elemen yang saling berhubungan untuk mencapai tujuan (Sutabri, 2012). Setiap kendaraan adalah agen otonom reaktif dengan atribut identitas, posisi (x, y), tipe manuver (lurus, kanan, atau kiri), dan kecepatan dasar. Keputusan agen bergantung pada kondisi lingkungan, bukan pada agen lain secara langsung. Pada setiap langkah waktu, agen menentukan status lampu lalu lintas (merah pada 12 detik pertama dari siklus 30 detik). Bila lampu merah, agen berhenti dan mengantre di belakang garis henti. Agen juga berpindah lajur untuk menghindari dua lubang jalan dan bermanuver belok setelah melewati titik tengah persimpangan. Selain itu, ketinggian jalan (z) dihitung pada segmen berbukit dengan fungsi trigonometri. Gambar 1 menunjukkan visualisasi simulasi.



Gambar 1. Visualisasi simulasi pergerakan kendaraan otonom pada jalan dengan tanjakan, lubang, dan persimpangan berlampu lalu lintas

Durasi simulasi pada pengujian waktu eksekusi adalah 1 detik waktu simulasi dengan langkah 0,1 detik (10 langkah), sehingga seluruh langkah berada pada fase lampu merah. Gambar 1 merupakan tangkapan layar simulasi terpisah dengan waktu simulasi yang lebih panjang. Agar beban komputasi dapat diukur, setiap agen pada tiap langkah juga menjalankan 500 iterasi perhitungan matematika buatan (akar kuadrat dan sinus). Beban ini bukan bagian dari aturan pergerakan dan nilainya sama untuk semua pengujian, sehingga hasil waktu eksekusi berlaku untuk tingkat beban tersebut.

Pengujian waktu eksekusi

Skenario yang sama dijalankan dengan dua pendekatan eksekusi menggunakan fungsi logika agen yang sama. Pada pemrosesan serial, seluruh agen diproses berurutan dalam satu alur program pada satu core CPU dan dijadikan acuan dasar. Pada pemrosesan paralel, digunakan multiprocessing.Pool dengan maksimum 4 proses pekerja. Daftar kendaraan dibagi rata menjadi 4 bagian yang diproses terpisah, lalu hasilnya digabung. Antrean di lampu lalu lintas tidak memakai struktur data global. Posisi antrean ditentukan secara lokal berdasarkan indeks kendaraan, sehingga tiap proses bekerja tanpa berbagi keadaan dan tidak terjadi race condition. Rancangan ini sejalan dengan anjuran dokumentasi (Python Software Foundation, n.d.).

Pengujian dijalankan pada komputer dengan sistem operasi Windows 11 Home Single Language, prosesor Intel Core i3-1005G1 (2 core fisik dan 4 thread), dan RAM 12 GB. Variabel bebas (x) adalah jumlah kendaraan, yaitu 10, 100, 500, 1.000, 2.000, 5.000, dan 10.000. Variabel terikat (y) adalah waktu eksekusi dalam detik untuk pemrosesan serial dan paralel, diukur dengan time.perf_counter(). Waktu paralel mencakup pembuatan Pool, pembagian data, perhitungan, dan penggabungan hasil. Setiap konfigurasi dijalankan 5 kali dan dicatat nilai rata-rata serta simpangan bakunya (total 70 pengujian). Satu pengujian awal untuk pemanasan tidak dicatat. Pengujian serial dan paralel dijalankan berurutan, bukan bersamaan, dengan jeda 1 detik antarpengujian. Data kendaraan dibuat baru pada setiap pengujian dan waktu pembuatannya tidak dihitung.

Analisis regresi linear

Analisis memakai regresi linear sederhana. Pada simulasi waktu diskrit ini, setiap agen menjalankan jumlah operasi yang sama pada setiap langkah dan diproses secara independen. Atas dasar tersebut, waktu serial diharapkan sebanding dengan jumlah agen: biaya per agen dikali jumlah agen, ditambah biaya tetap yang kecil. Pada pemrosesan paralel, pekerjaan dibagi ke beberapa proses sehingga biaya per agen turun. Namun, muncul biaya tetap yang tidak bergantung pada jumlah agen (pembuatan proses dan pengiriman data), serta biaya pengiriman hasil yang ikut bertambah. Bentuk garis lurus $y = a + bx$ sesuai dengan gambaran ini: a mewakili biaya tetap dan b mewakili tambahan waktu setiap penambahan satu kendaraan. Bentuk ini juga memungkinkan titik impas dan batas speedup dihitung secara analitik. Kelinearan hubungan tidak diasumsikan begitu saja, tetapi diuji dengan R^2 dan selisih model terhadap data pada bagian Hasil dan Pembahasan.

Untuk tiap metode, hubungan jumlah kendaraan (x) dan waktu eksekusi (y) dinyatakan oleh Persamaan (5), dengan a sebagai intersep (detik) dan b sebagai kemiringan (detik per kendaraan).

$$y = a + bx \tag{5}$$

Parameter dicari dengan metode kuadrat terkecil, yaitu garis yang meminimalkan jumlah kuadrat selisih antara data dan garis. Dari $k = 7$ titik (rata-rata 5 ulangan tiap jumlah kendaraan), dihitung dua besaran pada Persamaan (6). S_{xx} adalah jumlah kuadrat selisih x terhadap rata-ratanya, dan S_{xy} adalah jumlah hasil kali selisih x dan y terhadap rata-ratanya. Kemiringan dan intersep diperoleh dari Persamaan (7) dan (8), dengan tanda garis di atas menyatakan rata-rata.

$$S_{\mathrm{xx}} = \sum_{i=1}^{k} (x_i - \overline{x})^2, \quad S_{\mathrm{xy}} = \sum_{i=1}^{k} (x_i - \overline{x})(y_i - \overline{y}) \tag{6}$$

$$b = \frac{S_{\mathrm{xy}}}{S_{\mathrm{xx}}} \tag{7}$$

$$a = \overline{y} - b\overline{x} \tag{8}$$

Sebagai gambaran, Tabel 2 memuat komponen perhitungan untuk data penelitian ini.

Tabel 2. Komponen perhitungan regresi linear (*k* = 7 titik)

Besaran	Serial	Paralel
Rata-rata *x* (kendaraan)	2.658,57	2.658,57
Rata-rata *y* (detik)	2,0088	1,6469
S_{xx} (kendaraan^2)	80.784.085,71	80.784.085,71
S_{xy} (kendaraan·detik)	59.766,54	40.728,92
Kemiringan $b = S_{xy} / S_{xx}$ (detik per kendaraan)	0,000740	0,000504
Intersep *a* (detik)	0,0419	0,3065

Kecocokan diukur dengan koefisien determinasi R^2 pada Persamaan (9), dengan $\hat{y}_i$ nilai prediksi garis. Semakin mendekati 1, semakin besar proporsi ragam waktu yang dijelaskan model (Bahaswan & Pramono, 2024). Karena R^2 dapat tinggi walaupun galat di skala kecil relatif besar, model juga dievaluasi dengan selisih mutlak antara prediksi dan data pada tiap jumlah kendaraan.

$$R^2 = 1 - \frac{\sum_{i=1}^{k} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{k} (y_i - \overline{y})^2} \tag{9}$$

Dari kedua garis diturunkan dua besaran. Titik impas, yaitu jumlah kendaraan saat waktu serial dan paralel sama, diperoleh dari Persamaan (10). Persamaan ini hanya bermakna bila kemiringan serial lebih besar daripada paralel; indeks ser dan par menyatakan serial dan paralel. Bila model linear tetap berlaku pada skala sangat besar, speedup mendekati rasio kemiringan pada Persamaan (11) karena intersep menjadi tidak berarti dibanding suku *bx*.

$$x^* = \frac{a_{\mathrm{par}} - a_{\mathrm{ser}}}{b_{\mathrm{ser}} - b_{\mathrm{par}}} \tag{10}$$

$$S_\infty = \frac{b_{\mathrm{ser}}}{b_{\mathrm{par}}} \tag{11}$$

Metode ini mengandaikan hubungan yang linear pada rentang yang diuji, galat yang saling bebas, dan ragam galat yang cukup seragam. Asumsi terakhir tidak sepenuhnya terpenuhi karena simpangan baku membesar pada jumlah kendaraan yang lebih besar. Regresi dihitung dari 7 rata-rata, bukan dari 70 pengukuran mentah, sehingga ketidakpastian tiap ulangan tidak ikut dimodelkan, dan hasilnya tidak boleh diekstrapolasi di luar 10.000 kendaraan tanpa pengujian.

Analisis dilakukan dalam empat tahap: pertama, mengumpulkan waktu eksekusi; kedua, menghitung speedup, efisiensi, dan bagian paralel; ketiga, menyusun persamaan regresi linear untuk serial dan paralel; serta keempat, menghitung R^2, titik impas, dan batas speedup.

HASIL DAN PEMBAHASAN

Waktu eksekusi, speedup, dan efisiensi

Tabel 3 memuat rata-rata dan simpangan baku waktu eksekusi dari lima ulangan, beserta speedup (kali), efisiensi (%), dan bagian paralel *f* (%) pada $p = 4$. Pada 10 sampai 1.000 kendaraan, pemrosesan serial lebih cepat (speedup 0,06 sampai 0,98 kali). Pemrosesan

paralel mulai unggul pada 2.000 kendaraan (1,15 kali) dan mencapai 1,38 kali pada 5.000 dan 10.000 kendaraan. Pada 10.000 kendaraan, waktu eksekusi terpangkas sekitar 27,3%. Simpangan baku paralel kecil, paling besar 0,0979 detik. Simpangan baku serial pada 1.000 kendaraan (0,1406 detik) relatif besar karena ulangan pertama (1,0171 detik) lebih lambat daripada empat ulangan lainnya (0,6949 sampai 0,9228 detik). Data ini tetap dipakai apa adanya. Nilai *f* hanya dihitung bila speedup lebih dari 1, karena Persamaan (4) memberi nilai negatif bila paralel lebih lambat daripada serial.

Tabel 3. Waktu eksekusi (rata-rata ± simpangan baku, 5 ulangan), speedup, efisiensi, dan bagian paralel ($p = 4$)

Jumlah kendaraan	Serial (detik)	Paralel (detik)	Speedup (kali)	Efisiensi (%)	f (%)
10	0,0197 ± 0,0057	0,3107 ± 0,0445	0,06	1,6	-
100	0,1274 ± 0,0388	0,3804 ± 0,0123	0,33	8,4	-
500	0,3882 ± 0,0393	0,5635 ± 0,0158	0,69	17,2	-
1.000	0,8234 ± 0,1406	0,8408 ± 0,0979	0,98	24,5	-
2.000	1,4872 ± 0,0574	1,2930 ± 0,0268	1,15	28,8	17,4
5.000	3,8026 ± 0,1698	2,7539 ± 0,0461	1,38	34,5	36,8
10.000	7,4133 ± 0,1989	5,3858 ± 0,0911	1,38	34,4	36,5

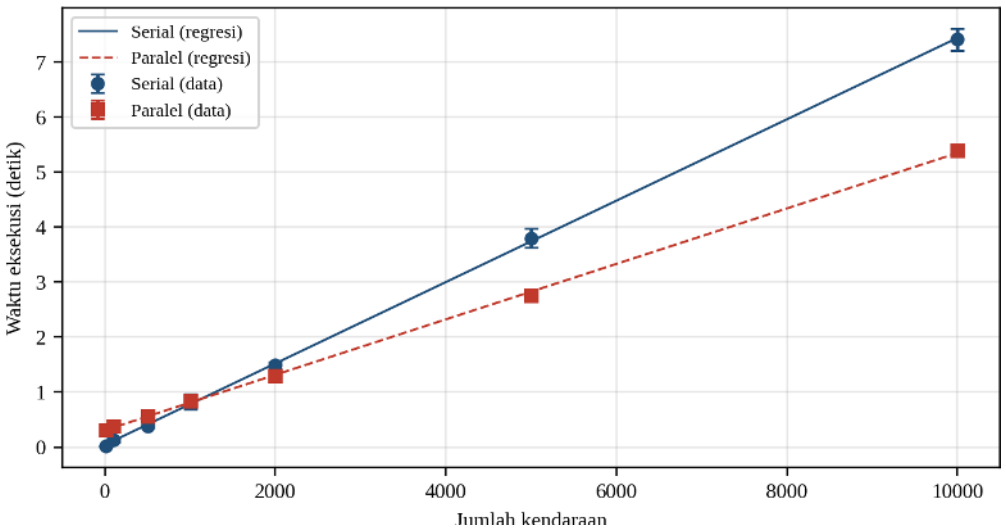
Catatan: tanda "-" berarti *f* tidak terdefinisi karena speedup kurang dari 1.

Model regresi linear

Tabel 4 menunjukkan persamaan regresi hasil perhitungan pada bagian Metode. Kemiringan serial (0,000740 detik per kendaraan) sekitar 1,47 kali kemiringan paralel (0,000504 detik per kendaraan), artinya setiap tambahan 1.000 kendaraan menambah waktu sekitar 0,74 detik pada serial dan 0,50 detik pada paralel. Intersep paralel (0,3065 detik) jauh lebih besar daripada serial (0,0419 detik), yang mencerminkan biaya tetap pembuatan proses dan pengiriman data. Nilai R^2 keduanya di atas 0,999, sehingga hubungan jumlah kendaraan dan waktu eksekusi pada rentang yang diuji sangat mendekati linear (Gambar 2). Dengan demikian, pada rentang tersebut waktu eksekusi dapat ditaksir dari jumlah kendaraan dengan galat yang kecil (Tabel 5).

Tabel 4. Hasil regresi linear (y dalam detik, x dalam jumlah kendaraan)

Metode	Persamaan regresi	R^2
Serial	$y = 0{,}0419 + 0{,}000740x$	0,9998
Paralel	$y = 0{,}3065 + 0{,}000504x$	0,9996



Gambar 2. Waktu eksekusi terhadap jumlah kendaraan beserta garis regresi (batang galat: simpangan baku)

Tabel 5 membandingkan prediksi model dengan data. Selisih terbesar adalah 0,0734 detik (paralel, 5.000 kendaraan); pada 1.000 kendaraan selisihnya 0,0416 detik untuk serial dan 0,0301 detik untuk paralel. Secara relatif, kesalahan terbesar muncul pada 10 kendaraan, ketika model serial menaksir 0,0493 detik padahal hasil pengujian 0,0197 detik, meskipun selisih mutlaknya hanya 0,0296 detik.

Tabel 5. Perbandingan data pengujian dan prediksi model (selisih = model dikurangi data, dalam detik)

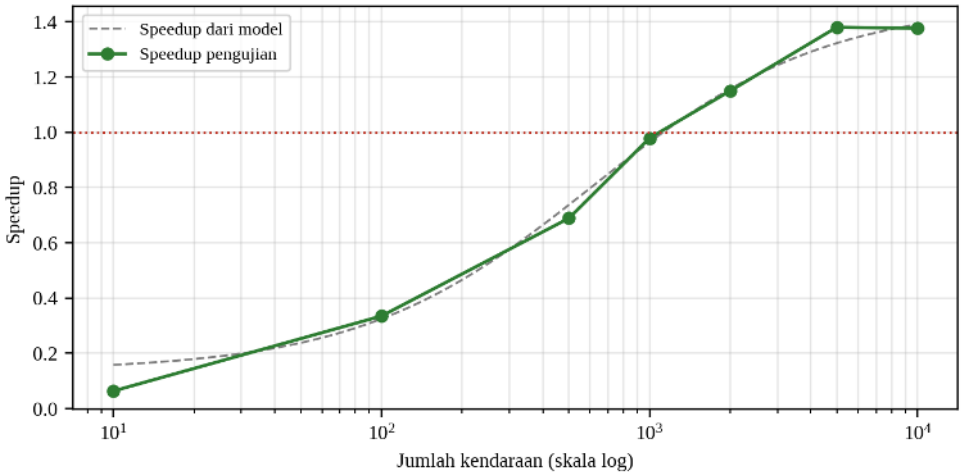
Jumlah	Serial (detik)			Paralel (detik)		
kendaraan	Data	Model	Selisih	Data	Model	Selisih
10	0,0197	0,0493	+0,0296	0,3107	0,3115	+0,0008
100	0,1274	0,1159	-0,0115	0,3804	0,3569	-0,0235
500	0,3882	0,4119	+0,0237	0,5635	0,5586	-0,0049
1.000	0,8234	0,7818	-0,0416	0,8408	0,8107	-0,0301
2.000	1,4872	1,5216	+0,0344	1,2930	1,3148	+0,0218
5.000	3,8026	3,7411	-0,0615	2,7539	2,8273	+0,0734
10.000	7,4133	7,4402	+0,0269	5,3858	5,3482	-0,0376

Titik impas, batas speedup, dan implikasinya

Dari Persamaan (10), titik impas model berada di sekitar 1.123 kendaraan. Nilai ini sesuai dengan pengamatan. Pada 1.000 kendaraan, serial masih sedikit lebih cepat (0,8234 detik berbanding 0,8408 detik; speedup 0,98 kali). Pada 2.000 kendaraan, paralel sudah lebih cepat (1,15 kali). Dengan demikian, titik impas sebenarnya berada di antara 1.000 dan 2.000 kendaraan. Namun, di sekitar titik impas selisih kedua metode (sekitar 0,017 detik pada 1.000 kendaraan) lebih kecil daripada simpangan baku pengukuran (0,1406 detik untuk serial dan 0,0979 detik untuk paralel). Atas dasar tersebut, angka 1.123 sebaiknya dipandang sebagai perkiraan, bukan batas yang pasti. Meskipun demikian, perkiraan ini tetap dapat menjadi informasi pendukung bagi pengembang simulasi dalam memilih metode eksekusi yang sesuai dengan skala simulasi (Sutabri, 2005).

Pada jumlah kendaraan kecil, paralel lebih lambat karena biaya pembuatan proses dan pengiriman data antarproses lebih besar daripada waktu komputasinya sendiri. Pada 10 kendaraan, paralel membutuhkan 0,3107 detik, sekitar 16 kali waktu serial (0,0197 detik). Efisiensi paralel memang cenderung turun ketika overhead mulai dominan (Chyan & Sumarta, 2026; Ekawijana & Noviansyah, 2026). Pengukuran waktu eksekusi sebagai ukuran efisiensi pendekatan komputasi juga dipakai pada pengujian algoritma rekursif untuk aplikasi berbasis kecerdasan buatan (Pratama et al., 2025), dan pada perbandingan algoritma rekursif dan iteratif dengan Python (Febriansyah et al., 2025). Keuntungan paralel baru terlihat ketika beban komputasi cukup besar.

Efisiensi paralel pada Tabel 3 meningkat dari 1,6% pada 10 kendaraan menjadi 34,5% pada 5.000 kendaraan ketika beban kerja yang dapat dibagi ke empat proses bertambah, lalu relatif tetap (34,4%) pada 10.000 kendaraan. Pada 10.000 kendaraan, f sekitar 36,5%. Artinya, sekitar 36,5% program dapat berjalan paralel, sedangkan sekitar 63,5% tetap berurutan atau tertahan biaya pengelolaan proses. Sebagai pembanding, untuk $p = 2$ (jumlah core fisik), efisiensi pada 10.000 kendaraan sekitar 68,8% dan f sekitar 54,7%.



Gambar 3. Speedup hasil pengujian dan speedup yang diturunkan dari model regresi

Dari Persamaan (11), speedup model mendekati 1,47 kali ketika jumlah kendaraan makin besar (Gambar 3). Speedup hasil pengujian (1,38 kali pada 5.000 dan 10.000 kendaraan) sejalan dengan batas ini, tetapi jauh di bawah speedup ideal 4 kali untuk 4 proses. Atas dasar tersebut, keuntungan paralel pada jumlah kendaraan di atas 10.000 tidak dapat diasumsikan makin besar. Prosesor yang dipakai memiliki 2 core fisik dan 4 thread, sehingga empat proses pekerja tidak berjalan pada empat core fisik. Kondisi ini kemungkinan menjadi pembatas utama, di samping biaya mengirim hasil simulasi antarproses (sekitar 100.000 baris hasil pada 10.000 kendaraan) dan pembuatan Pool pada setiap pemanggilan. Dugaan ini belum diuji secara langsung, misalnya dengan memvariasikan jumlah proses pekerja.

Hasil regresi memiliki dua implikasi praktis. Pertama, kemiringan b menunjukkan biaya tambahan per kendaraan, sedangkan intersep a menunjukkan biaya tetap. Pemrosesan paralel menurunkan biaya per kendaraan dari 0,000740 menjadi 0,000504 detik, tetapi menambah biaya tetap sekitar 0,26 detik. Karena titik impas pada Persamaan (10) berasal dari perbandingan kedua biaya ini, nilainya bergantung pada implementasi dan beban komputasi yang dipakai. Perubahan keduanya dapat menggeser titik impas, sehingga angka 1.123 tidak boleh dipakai pada konfigurasi lain tanpa pengujian ulang. Kedua, model sebaiknya dipakai hanya untuk menaksir waktu eksekusi pada rentang 10 sampai 10.000 kendaraan. Nilai R^2 yang tinggi tidak menjamin ketepatan pada skala kecil, karena kesalahan relatifnya di sana besar (Tabel 5). Dari sisi penerapan, pemrosesan paralel pada penelitian ini dibatasi oleh tiga hal: biaya tetap yang membuatnya merugikan di bawah titik impas, batas speedup sekitar 1,47 kali, dan jumlah core fisik yang lebih sedikit daripada jumlah proses pekerja.

Selain waktu eksekusi, pengamatan penulis terhadap keluaran simulasi menunjukkan bahwa kedua metode berperilaku konsisten secara konsep. Kendaraan mematuhi antrean di lampu lalu lintas, menghindari lubang, dan mengikuti jalur yang ditentukan. Jumlah baris data keluaran juga sama pada setiap skala, misalnya 100.000 baris pada 10.000 kendaraan. Pada studi yang dilakukan, keluaran simulasi tidak dibandingkan secara otomatis selama pengujian waktu.

SIMPULAN

Waktu eksekusi simulasi kendaraan otonom berbasis MAS meningkat hampir linear terhadap jumlah kendaraan pada pemrosesan serial maupun paralel (R^2 = 0,9998 dan 0,9996). Laju pertumbuhan waktu serial sekitar 1,47 kali laju paralel. Pemrosesan paralel mulai lebih cepat di antara 1.000 dan 2.000 kendaraan, sesuai dengan titik impas model sekitar 1.123 kendaraan, dan mencapai speedup 1,38 kali pada 5.000 sampai 10.000 kendaraan dengan

efisiensi sekitar 34% pada $p = 4$. Pada skala kecil, pemrosesan serial lebih efisien. Model linear dari tujuh titik uji cukup akurat dengan selisih terbesar 0,07 detik, tetapi galat relatifnya paling besar pada jumlah kendaraan kecil. Titik impas perlu dibaca sebagai perkiraan karena selisih kedua metode di sekitarnya lebih kecil daripada simpangan baku. Batas speedup model (1,47 kali) dan hasil pengujian (1,38 kali) jauh di bawah speedup ideal 4 kali. Dugaan penyebabnya, yaitu jumlah core fisik, biaya pengiriman hasil, dan pembuatan Pool, belum diuji langsung. Hasil ini hanya berlaku untuk simulasi MAS berbasis aturan (pra-LLM), perangkat, dan beban komputasi buatan yang digunakan. Penelitian lanjutan disarankan memvariasikan jumlah proses pekerja dan beban komputasi, menguji pada perangkat lain, serta membandingkan dengan regresi polinomial atau model piecewise.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Program Studi Teknik Informatika, Institut Teknologi Indonesia, atas dukungan akademik dan fasilitas yang diberikan sehingga penelitian ini dapat diselesaikan.

DAFTAR PUSTAKA

Aziz, Z. A., Abdulqader, D. N., Sallow, A. B., & Omer, H. K. (2021). Python parallel processing and multiprocessing: A review. *Academic Journal of Nawroz University*, *10*(3), 345–354. https://doi.org/10.25007/ajnu.v10n3a1145

Bahaswan, J., & Pramono, P. (2024). Pemodelan prediktif waktu eksekusi tugas di lingkungan cloud menggunakan metrik pemanfaatan sumber daya. In *Prosiding Seminar Nasional Teknologi Informasi dan Bisnis (SENATIB)* (pp. 31–36). https://ojs.udb.ac.id/index.php/Senatib/article/view/4334

Bazzan, A. L. C., & Klügl, F. (2014). A review on agent-based technology for traffic and transportation. *The Knowledge Engineering Review*, *29*(3), 375–403. https://doi.org/10.1017/S0269888913000118

Brooks, R. A. (1991). Intelligence without representation. *Artificial Intelligence*, *47*(1–3), 139–159. https://doi.org/10.1016/0004-3702(91)90053-M

Buşoniu, L., Babuška, R., & De Schutter, B. (2008). A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *38*(2), 156–172. https://doi.org/10.1109/TSMCC.2007.913919

Cabri, G., Leonardi, L., & Rotonda, E. (2023). A survey on the use of the multi-agent paradigm in coordination of connected and autonomous vehicles. In *Intelligent Distributed Computing XV* (pp. 118–124). Springer. https://doi.org/10.1007/978-3-031-29104-3_13

Chen, B., & Cheng, H. H. (2010). A review of the applications of agent technology in traffic and transportation systems. *IEEE Transactions on Intelligent Transportation Systems*, *11*(2), 485–497. https://doi.org/10.1109/TITS.2010.2048313

Chyan, P., & Sumarta, S. C. (2026). Implementasi dan analisis kinerja komputasi paralel menggunakan Ray pada lingkungan multi-core untuk pemrosesan audio. *JURIKOM (Jurnal Riset Komputer)*, *13*(3), 899–908. https://doi.org/10.30865/jurikom.v13i3.9649

Dinneweth, J., Boubezoul, A., Mandiau, R., & Espié, S. (2022). Multi-agent reinforcement learning for autonomous vehicles: A survey. *Autonomous Intelligent Systems*, *2*(1), 27. https://doi.org/10.1007/s43684-022-00045-z

Ekawijana, A., & Noviansyah, B. (2026). Pengaruh jumlah proses terhadap kinerja multiple processing pada Linux. *VOCATECH: Vocational Education and Technology Journal*, *8*(1), 70–78. https://doi.org/10.38038/vocatech.v8i1.288

Febriansyah, M. F., Gunawan, Rhamadani, M., & Sutabri, T. (2025). Perbandingan pemanfaatan algoritma rekursif dan iteratif dalam penyelesaian struktur data pohon. *Jurnal Manajemen Informatika & Teknologi (MIFORTEKH)*, *5*(1), 46–56. https://doi.org/10.51903/2mxmmg85

Guo, T., Chen, X., Wang, Y., Chang, R., Pei, S., Chawla, N. V., Wiest, O., & Zhang, X. (2024). Large language model based multi-agents: A survey of progress and challenges. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI-24)* (pp. 8048–8057). International Joint Conferences on Artificial Intelligence Organization. https://doi.org/10.24963/ijcai.2024/890

Hill, M. D., & Marty, M. R. (2008). Amdahl's law in the multicore era. *Computer*, *41*(7), 33–38. https://doi.org/10.1109/MC.2008.209

Laudan, J., Heinrich, P., & Nagel, K. (2025). High-performance mobility simulation: Implementation of a parallel distributed message-passing algorithm for MATSim. *Information*, *16*(2), 116. https://doi.org/10.3390/info16020116

Li, X., Wang, S., Zeng, S., Wu, Y., & Yang, Y. (2024). A survey on LLM-based multi-agent systems: Workflow, infrastructure, and challenges. *Vicinagearth*, *1*, 9. https://doi.org/10.1007/s44336-024-00009-2

Mushtaq, A., Haq, I. U., Sarwar, M. A., Khan, A., Khalil, W., & Mughal, M. A. (2023). Multi-agent reinforcement learning for traffic flow management of autonomous vehicles. *Sensors*, *23*(5), 2373. https://doi.org/10.3390/s23052373

Park, J. S., O'Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., & Bernstein, M. S. (2023). Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23)* (Article 2, pp. 1–22). Association for Computing Machinery. https://doi.org/10.1145/3586183.3606763

Pratama, M. B., Setiawan, R., & Sutabri, T. (2025). Integrasi algoritma rekursif pada pemrosesan data multilevel untuk aplikasi berbasis AI. *Jurnal Manajemen Informatika & Teknologi (MIFORTEKH)*, *5*(1), 57–66.

Python Software Foundation. (n.d.). *multiprocessing — Process-based parallelism*. Python documentation. https://docs.python.org/3/library/multiprocessing.html

Rao, A. S., & Georgeff, M. P. (1995). BDI agents: From theory to practice. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)* (pp. 312–319). AAAI Press.

Sadid, H., & Antoniou, C. (2023). Modelling and simulation of (connected) autonomous vehicles longitudinal driving behavior: A state-of-the-art. *IET Intelligent Transport Systems*, *17*(6), 1051–1071. https://doi.org/10.1049/itr2.12337

Smith, R. G. (1980). The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, *C–29*(12), 1104–1113. https://doi.org/10.1109/TC.1980.1675516

Sutabri, T. (2005). *Sistem Informasi Manajemen*. Andi.

Sutabri, T. (2012). *Konsep Sistem Informasi*. Andi.

Tomás, V. R., Castillo, M., Monzón-Catalán, I., & García, L. A. (2026). Parallel agent-based modeling for improving traffic flow simulation. *Transportation Research Interdisciplinary Perspectives*, *37*, 101930. https://doi.org/10.1016/j.trip.2026.101930

Wooldridge, M. (2009). *An Introduction to Multiagent Systems* (2nd ed.). John Wiley & Sons.

Wu, Y., Li, D., Chen, Y., Jiang, R., Zou, H. P., Huang, W.-C., Li, Y., Fang, L., Wang, Z., & Yu, P. S. (2025). *Multi-agent autonomous driving systems with large language models: A survey of recent advances* (arXiv:2502.16804). arXiv. https://doi.org/10.48550/arXiv.2502.16804