

# Penerapan Algoritma A\* dalam Estimasi Parameter Model Lotka-Volterra : Pendekatan Heuristik untuk Pemodelan Interaksi Populasi

Muh. Rusmin Nurwadin - 13523068<sup>1</sup>

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia

[rusmn17@gmail.com](mailto:rusmn17@gmail.com), [13523068@std.stei.itb.ac.id](mailto:13523068@std.stei.itb.ac.id)

**Abstrak**— Estimasi parameter yang akurat dalam model Lotka-Volterra merupakan tantangan krusial dalam pemodelan dinamika interaksi predator-mangsa. Metode optimasi konvensional seringkali mengalami kesulitan dalam menavigasi ruang parameter yang kompleks dan mudah terjebak pada solusi suboptimal. Penelitian ini mengusulkan pendekatan novel dengan mengadaptasi algoritma A\* untuk menyelesaikan masalah estimasi parameter model Lotka-Volterra. Adaptasi dilakukan dengan memodelkan ruang parameter sebagai graf pencarian terstruktur, dimana setiap node merepresentasikan kombinasi spesifik empat parameter kunci model ( $\alpha$ ,  $\beta$ ,  $\delta$ ,  $\gamma$ ). Algoritma menggunakan fungsi heuristik admissible yang mengombinasikan estimasi error minimum berdasarkan variabilitas data dengan penalti jarak parameter dari region optimal. Evaluasi komprehensif dilakukan melalui serangkaian eksperimen menggunakan data sintesis atau fiktif, mencakup analisis robustness terhadap noise, pengaruh granularity pencarian, ukuran ruang parameter, dan konsistensi hasil. Hasil menunjukkan bahwa A\* mampu menghasilkan estimasi parameter dengan akurasi yang memuaskan, meski performa sangat dipengaruhi oleh tingkat noise data dan pengaturan parameter algoritma. Terdapat *trade-off* signifikan antara akurasi estimasi dan efisiensi komputasi yang perlu dipertimbangkan dalam implementasi praktis. Penelitian ini memberikan kontribusi dalam pengembangan metode optimasi heuristik untuk pemodelan dinamika populasi dan membuka peluang eksplorasi lebih lanjut dalam penerapan algoritma pencarian untuk masalah estimasi parameter model matematika.

**Kata Kunci**— Lotka-Volterra, algoritma A\*, estimasi parameter, optimasi heuristik, dinamika populasi

## I. PENDAHULUAN

Interaksi antara predator dan mangsa merupakan fenomena yang umum dijumpai di alam. Untuk memahami bagaimana kedua populasi ini saling mempengaruhi, para peneliti menggunakan berbagai pendekatan matematis. Model Lotka-Volterra menjadi salah satu pilihan yang populer karena mampu menjelaskan pola perubahan populasi dengan menggunakan sistem persamaan diferensial [1]. Walaupun tampak sederhana, model ini cukup efektif dalam menggambarkan dinamika dasar hubungan predator-mangsa.

Masalah utama dalam penerapan model Lotka-Volterra terletak pada penentuan parameter yang sesuai. Model ini bergantung pada empat parameter kunci, yakni laju pertumbuhan populasi mangsa, tingkat predasi, efisiensi konversi energi, dan laju kematian predator. Masing-masing parameter memiliki pengaruh besar terhadap perilaku model,

sehingga estimasi yang tepat sangat diperlukan untuk mendapatkan hasil yang bermakna.

Pendekatan optimasi yang biasa digunakan, seperti metode berbasis gradien atau teknik evolusioner, seringkali mengalami kesulitan ketika berhadapan dengan karakteristik model yang kompleks. Ruang pencarian parameter dapat memiliki banyak lembah dan puncak lokal, sehingga algoritma mudah terjebak pada solusi suboptimal. Hal ini menjadi kendala serius dalam mencari parameter yang benar-benar memberikan hasil terbaik.

A\* telah lama dikenal sebagai algoritma yang handal untuk menyelesaikan masalah pencarian jalur [2]. Kekuatan utama algoritma ini terletak pada penggunaan informasi heuristik yang membantu mengarahkan proses pencarian secara lebih efisien. Sifat ini membuat A\* menarik untuk dicoba dalam konteks yang berbeda, termasuk untuk masalah optimasi parameter.

Adaptasi A\* untuk estimasi parameter model Lotka-Volterra dapat dilakukan dengan memandang setiap kombinasi parameter sebagai node dalam graf. Tujuan pencarian adalah menemukan node yang menghasilkan error terkecil antara output model dengan data yang diamati. Fungsi heuristik dapat didesain berdasarkan ukuran kesalahan ini, memberikan petunjuk kepada algoritma tentang arah pencarian yang paling berpotensi.

Studi ini fokus pada implementasi A\* untuk menyelesaikan masalah estimasi parameter dalam model Lotka-Volterra. Perbandingan dilakukan dengan beberapa metode optimasi lain untuk melihat seberapa baik performa algoritma yang diusulkan. Evaluasi meliputi tingkat akurasi, waktu komputasi, dan konsistensi hasil yang diperoleh. Data yang digunakan bersifat sintesis untuk memudahkan kontrol dan validasi eksperimen.

Diharapkan hasil dari studi ini dapat menunjukkan potensi algoritma A\* sebagai alternatif yang layak untuk optimasi parameter model dinamika populasi. Selain itu, pendekatan ini dapat memberikan inspirasi untuk penerapan algoritma pencarian dalam masalah optimasi lain di bidang pemodelan matematika.

## II. LANDASAN TEORI

### A. Dinamika Ekosistem

Dinamika ekosistem mengacu pada perubahan dan interaksi yang terjadi dalam suatu ekosistem akibat pengaruh faktor biotik, yaitu makhluk hidup, dan abiotik, yaitu lingkungan fisik. Setiap ekosistem terdiri dari berbagai komponen seperti produsen, konsumen, dan pengurai yang



antara model dan data observasi. Dengan memilih heuristik yang sesuai, algoritma A\* dapat mempercepat pencarian solusi dengan lebih efisien dibandingkan metode optimasi tradisional.

Namun, meskipun kompleksitas waktu A\* dalam kasus terburuk adalah  $O(b^d)$ , dengan  $b$  sebagai faktor percabangan dan  $d$  sebagai kedalaman solusi optimal, penerapan heuristik yang tepat akan mengurangi waktu pencarian dan meningkatkan performa algoritma secara praktis. Meskipun demikian, kompleksitas ruang tetap  $O(b^d)$  karena A\* perlu menyimpan semua node yang telah dieksplorasi. Oleh karena itu, penerapan algoritma A\* dalam estimasi parameter model Lotka-Volterra sangat bergantung pada pemilihan heuristik yang efektif, agar solusi global yang optimal dapat ditemukan dengan efisien, baik dari segi waktu maupun ruang

### E. Fungsi Heuristik

Heuristik berasal dari kata Yunani "heuriskein" yang berarti mencari atau menemukan. Dalam konteks pemrograman, heuristik merujuk pada pendekatan yang digunakan dalam merancang algoritma atau metode pemecahan masalah yang didasarkan pada eksperimen atau aturan praktis lainnya. Pendekatan ini bertujuan untuk menemukan solusi yang efisien dan efektif meskipun tidak selalu menghasilkan solusi yang sempurna atau optimal. Heuristik sangat berguna untuk mengatasi masalah yang terlalu kompleks untuk diselesaikan dengan algoritma deterministik konvensional, atau ketika mencari solusi yang tepat tidak memungkinkan karena keterbatasan waktu [8].

Dalam penerapan algoritma A\* untuk estimasi parameter model Lotka-Volterra, heuristik memegang peran penting dalam mengarahkan pencarian solusi optimal. Fungsi heuristik haruslah admissible, yang berarti untuk setiap simpul  $n$ , nilai  $h(n)$  (estimasi biaya menuju tujuan) tidak boleh melebihi biaya sebenarnya  $h^*(n)$  yang diperlukan untuk mencapai tujuan dari simpul tersebut. Dengan kata lain, heuristik yang admissible tidak pernah melebihi biaya yang dibutuhkan untuk mencapai tujuan, yang berarti bersifat optimistik [7].

Untuk masalah estimasi parameter Lotka-Volterra, fungsi heuristik dapat dirancang berdasarkan karakteristik *error fitting* dan jarak parameter dalam ruang pencarian. Salah satu pendekatan adalah menggunakan estimasi error minimum berdasarkan variabilitas data observasi, dikombinasikan dengan penalti jarak Euclidean dari parameter yang diperkirakan optimal. Teorema yang penting di sini adalah jika suatu heuristik  $h(n)$  admissible, maka algoritma A\* yang menggunakan tree search akan menghasilkan solusi yang optimal. Hal ini menjamin bahwa kombinasi parameter yang ditemukan benar-benar memberikan *fitting* terbaik terhadap data observasi.

## III. METODOLOGI

Penelitian ini diawali dengan studi literatur mengenai dinamika ekosistem, model Lotka-Volterra, serta algoritma A\* dan fungsi heuristik. Studi ini bertujuan untuk memahami teori dasar yang mendukung optimasi parameter dalam model Lotka-Volterra, termasuk bagaimana ruang pencarian parameter dapat dipetakan dan dijelajahi menggunakan pendekatan heuristik. Selain itu, dilakukan analisis literatur mengenai adaptasi algoritma pencarian untuk masalah optimasi guna memperkuat kerangka evaluasi yang akan digunakan.

Adaptasi algoritma A\* untuk estimasi parameter model Lotka-Volterra dilakukan dengan memodelkan ruang parameter sebagai graf pencarian terstruktur. Setiap node dalam graf merepresentasikan kombinasi spesifik dari empat parameter kunci model ( $\alpha, \beta, \delta, \gamma$ ), dimana  $\alpha$  adalah laju pertumbuhan mangsa tanpa kehadiran predator,  $\beta$  adalah tingkat kematian mangsa karena interaksi dengan predator,  $\delta$  adalah tingkat pertumbuhan predator karena konsumsi mangsa, dan  $\gamma$  adalah laju kematian predator tanpa kehadiran mangsa. Edges dalam graf menghubungkan node-node yang berbeda dalam nilai parameter tertentu dengan increment yang telah ditetapkan, memungkinkan eksplorasi sistematis kombinasi parameter yang berdekatan. Algoritma A\* kemudian mencari jalur optimal dalam graf ini menggunakan fungsi evaluasi  $f(n) = g(n) + h(n)$ , dimana  $g(n)$  merepresentasikan biaya kumulatif dari node awal menuju node  $n$ , dan  $h(n)$  adalah fungsi heuristik yang memperkirakan biaya untuk mencapai kombinasi parameter optimal.

Desain fungsi heuristik admissible menjadi kunci keberhasilan pendekatan ini. Fungsi  $h(n)$  dirancang berdasarkan karakteristik sistem Lotka-Volterra dan sifat data observasi, dengan mempertimbangkan bahwa kombinasi parameter yang baik akan menghasilkan prediksi populasi yang mendekati data observasi. Heuristik ini mengombinasikan estimasi error minimum yang mungkin dicapai berdasarkan variabilitas data dengan penalti jarak parameter dari region yang diperkirakan optimal. Hal ini memastikan bahwa algoritma tidak akan meremehkan biaya sebenarnya untuk mencapai solusi optimal, sehingga tetap mempertahankan sifat admissible yang diperlukan untuk menjamin optimalitas solusi A\*. Fungsi objektif yang diminimalkan adalah Mean Squared Error (MSE) sebagaimana didefinisikan dalam persamaan yang telah dijelaskan sebelumnya, yang dihitung melalui simulasi sistem persamaan diferensial Lotka-Volterra untuk memprediksi dinamika populasi predator dan mangsa.

Data yang digunakan dalam penelitian ini adalah data sintetis atau fiktif yang mencerminkan dinamika interaksi populasi predator-mangsa dalam ekosistem. Data ini digenerasi dengan mensimulasikan sistem persamaan diferensial Lotka-Volterra menggunakan parameter yang telah ditentukan, kemudian menambahkan noise untuk merepresentasikan variabilitas alami dan ketidakpastian pengamatan yang umum dijumpai dalam data ekologi. Pendekatan data sintetis dipilih karena memungkinkan validasi akurasi algoritma secara objektif, dimana parameter sebenarnya sudah diketahui sehingga kualitas estimasi dapat diukur secara kuantitatif. Data dirancang untuk merepresentasikan berbagai skenario dinamika populasi, termasuk kondisi awal yang berbeda dan periode observasi yang bervariasi, untuk menguji robustness algoritma dalam berbagai kondisi pemodelan interaksi populasi.

Evaluasi performa algoritma A\* dilakukan melalui serangkaian eksperimen terkontrol dengan memvariasikan beberapa aspek kunci. Pertama, tingkat noise dalam data observasi divariasikan untuk menguji robustness algoritma terhadap ketidakpastian data, yang mencerminkan kondisi nyata dimana pengamatan populasi selalu mengandung gangguan dan ketidakpastian. Kedua, ukuran ruang pencarian parameter

divariasikan dari ruang yang sempit hingga luas untuk menganalisis skalabilitas dan efisiensi algoritma dalam mengeksplorasi space parameter yang kompleks. Ketiga, granularity pencarian parameter diatur pada tingkat yang berbeda untuk menganalisis *trade-off* antara akurasi estimasi dengan efisiensi komputasi. Setiap variasi eksperimen dijalankan secara berulang untuk memperoleh statistik yang konsisten dan reliable mengenai performa algoritma.

Untuk memvalidasi keefektifan pendekatan heuristik yang diusulkan, hasil algoritma A\* dibandingkan dengan metode optimasi yang telah mapan dalam estimasi parameter model dinamika populasi. Perbandingan dilakukan dengan metode berbasis gradien dan teknik pencarian acak untuk menganalisis keunggulan dan keterbatasan relatif masing-masing pendekatan. Metrik evaluasi meliputi akurasi estimasi parameter yang diukur melalui error relatif terhadap parameter sebenarnya, efisiensi komputasi yang dinilai dari waktu eksekusi dan jumlah evaluasi fungsi objektif, serta konsistensi hasil dalam mereproduksi dinamika interaksi predator-mangsa yang diamati. Analisis komprehensif ini bertujuan untuk mengidentifikasi kondisi optimal penggunaan algoritma A\* dalam konteks pemodelan interaksi populasi dan memberikan rekomendasi praktis untuk implementasi metode ini dalam studi ekologi yang lebih luas.

#### IV. IMPLEMENTASI

##### A. Implementasi Persamaan Lotka-Volterra

```

1 class LVModel:
2
3     def __init__(self, alpha: float, beta: float, delta: float, gamma: float):
4         self.alpha = alpha # laju pertumbuhan mangsa
5         self.beta = beta # tingkat predasi
6         self.delta = delta # efisiensi konversi
7         self.gamma = gamma # laju kematian predator
8
9     def equations(self, state: List[float], t: float) -> List[float]:
10        x, y = state
11        dxdt = self.alpha * x - self.beta * x * y
12        dydt = self.delta * x * y - self.gamma * y
13        return [dxdt, dydt]
14
15    def simulate(self, init_state: List[float], time_pts: np.ndarray) -> np.ndarray:
16        solution = odeint(self.equations, init_state, time_pts)
17        return solution
18

```

Gambar 2. Implementasi Persamaan Lotka-Volterra  
Sumber : Dokumen Pribadi

Fungsi ini mengimplementasikan sistem persamaan diferensial Lotka-Volterra sesuai persamaan (1) dan (2) yang telah didefinisikan di Bab 2 bagian B. Implementasi ini menjadi dasar simulasi dinamika populasi predator-mangsa untuk memprediksi interaksi populasi berdasarkan parameter yang diberikan.

##### B. Fungsi Heuristik

```

1 def heuristic_func(self, node: ParamNode) -> float:
2     data_var = np.var(self.obs_data)
3     min_error = np.sqrt(data_var) * 0.1
4
5     ref_alpha = (self.param_bounds['alpha'][0] + self.param_bounds['alpha'][1]) / 2
6     ref_beta = (self.param_bounds['beta'][0] + self.param_bounds['beta'][1]) / 2
7     ref_delta = (self.param_bounds['delta'][0] + self.param_bounds['delta'][1]) / 2
8     ref_gamma = (self.param_bounds['gamma'][0] + self.param_bounds['gamma'][1]) / 2
9
10    dist_penalty = (abs(node.alpha - ref_alpha) + abs(node.beta - ref_beta) +
11                  abs(node.delta - ref_delta) + abs(node.gamma - ref_gamma)) * 0.01
12
13    return min_error + dist_penalty

```

Gambar 3. Implementasi Fungsi Heuristik  
Sumber : Dokumen Pribadi

Fungsi heuristik admissible ini dirancang berdasarkan metodologi pada Bab 3, mengombinasikan estimasi error minimum dari variabilitas data observasi dengan penalti jarak parameter dari region optimal. Desain ini memastikan sifat admissible tetap terjaga sehingga algoritma A\* dapat menjamin optimalitas solusi yang ditemukan.

##### C. Representasi Graf

```

1 def get_neighbors(self, node: ParamNode) -> List[ParamNode]:
2     neighbors = []
3     steps = [-self.step_size, self.step_size]
4
5     for a_step in steps:
6         for b_step in steps:
7             for d_step in steps:
8                 for g_step in steps:
9                     new_alpha = node.alpha + a_step
10                    new_beta = node.beta + b_step
11                    new_delta = node.delta + d_step
12                    new_gamma = node.gamma + g_step
13
14                    if (self.param_bounds['alpha'][0] <= new_alpha <= self.param_bounds['alpha'][1] and
15                        self.param_bounds['beta'][0] <= new_beta <= self.param_bounds['beta'][1] and
16                        self.param_bounds['delta'][0] <= new_delta <= self.param_bounds['delta'][1] and
17                        self.param_bounds['gamma'][0] <= new_gamma <= self.param_bounds['gamma'][1]):
18
19                        neighbor = ParamNode(new_alpha, new_beta, new_delta, new_gamma)
20                        neighbor.parent = node
21                        neighbors.append(neighbor)
22
23    return neighbors

```

Gambar 4. Representasi Graf  
Sumber : Dokumen Pribadi

Fungsi ini mengimplementasikan konsep edges dalam graf pencarian yang menghubungkan node-node dengan increment parameter yang telah ditetapkan sesuai metodologi. Eksplorasi sistematis ini memungkinkan algoritma menjelajahi ruang parameter secara terstruktur untuk menemukan kombinasi optimal.

### D. Implementasi A\*

```

1 def optimize(self, max_iter: int = 500, tolerance: float = 1e-6) -> Tuple[ParamNode, float, dict]:
2     start_alpha = (self.param_bounds['alpha'][0] + self.param_bounds['alpha'][1]) / 2
3     start_beta = (self.param_bounds['beta'][0] + self.param_bounds['beta'][1]) / 2
4     start_delta = (self.param_bounds['delta'][0] + self.param_bounds['delta'][1]) / 2
5     start_gamma = (self.param_bounds['gamma'][0] + self.param_bounds['gamma'][1]) / 2
6
7     start_node = ParamNode(start_alpha, start_beta, start_delta, start_gamma)
8     start_node.h_cost = self.heuristic_func(start_node)
9     start_node.f_cost = start_node.g_cost + start_node.h_cost
10
11     open_list = [start_node]
12     closed_set = set()
13     iterations = 0
14     error_history = []
15     start_time = time.time()
16
17     print("Memulai optimasi A* untuk estimasi parameter...")
18
19     while open_list and iterations < max_iter:
20         current = heapq.heappop(open_list)
21         current_key = (round(current.alpha, 3), round(current.beta, 3),
22                     round(current.delta, 3), round(current.gamma, 3))
23
24         if current_key in closed_set:
25             continue
26
27         closed_set.add(current_key)
28
29         current_error = self.calc_mse(current.get_params())
30         error_history.append(current_error)
31
32         if current_error < self.best_error:
33             self.best_error = current_error
34             self.best_params = current
35             print(f"Iter {iterations}: MSE = {current_error:.6f}")
36
37         if current_error < tolerance:
38             print(f"Konvergen pada iterasi {iterations}")
39             break
40
41         neighbors = self.get_neighbors(current)
42
43         for neighbor in neighbors:
44             neighbor_key = (round(neighbor.alpha, 3), round(neighbor.beta, 3),
45                             round(neighbor.delta, 3), round(neighbor.gamma, 3))
46
47             if neighbor_key in closed_set:
48                 continue
49
50             neighbor.g_cost = current.g_cost + self.step_size
51             neighbor.h_cost = self.heuristic_func(neighbor)
52             neighbor.f_cost = neighbor.g_cost + neighbor.h_cost
53
54             heapq.heappush(open_list, neighbor)
55
56         iterations += 1
57
58         if iterations % 50 == 0:
59             print(f"Iter {iterations}, Best MSE: {self.best_error:.6f}")
60
61     end_time = time.time()
62
63     results = {
64         'iterations': iterations,
65         'comp_time': end_time - start_time,
66         'error_hist': error_history,
67         'final_error': self.best_error,
68         'nodes_explored': len(closed_set)
69     }
70
71     return self.best_params, self.best_error, results

```

Gambar 5. Implementasi A\*  
Sumber : Dokumen Pribadi

Implementasi lengkap algoritma A\* dengan priority queue untuk open list dan closed set sesuai teori pada Bab 2 bagian D. Fungsi ini menggunakan evaluasi  $f(n) = g(n) + h(n)$  untuk mengarahkan pencarian menuju solusi optimal dengan efisiensi komputasi yang baik.

### E. MSE

```

1 def calc_mse(self, params: Tuple[float, float, float, float]) -> float:
2     try:
3         alpha, beta, delta, gamma = params
4         model = LVModel(alpha, beta, delta, gamma)
5         predicted = model.simulate(self.init_state, self.time_pts)
6         mse = np.mean((self.obs_data - predicted) ** 2)
7         return mse
8     except:
9         return 1e6
10
11 def calc_mse(self, params: Tuple[float, float, float, float]) -> float:
12     alpha, beta, delta, gamma = params
13     model = LVModel(alpha, beta, delta, gamma)
14     predicted = model.simulate(self.init_state, self.time_pts)
15     mse = np.mean((self.obs_data - predicted) ** 2)
16     return mse
17
18 except:
19     return 1e6

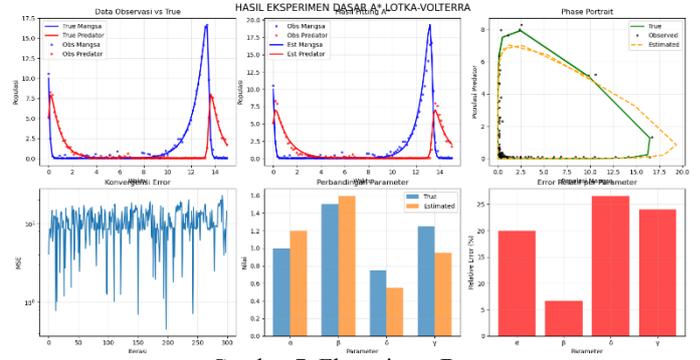
```

Gambar 6. Perhitungan MSE  
Sumber : Dokumen Pribadi

Fungsi objektif yang menghitung Mean Squared Error sesuai formula yang didefinisikan di Bab 2 bagian C sebagai kriteria estimasi parameter. Implementasi ini menjadi jembatan antara prediksi model dengan data observasi untuk mengukur kualitas fitting parameter.

## V. HASIL DAN PEMBAHASAN

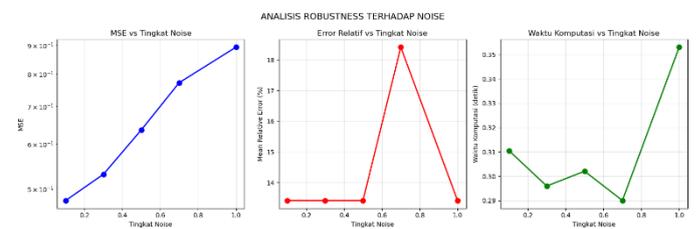
### A. Eksperimen Dasar



Gambar 7. Eksperimen Dasar  
Sumber : Dokumen Pribadi

Eksperimen ini memperlihatkan bagaimana estimasi parameter menggunakan A\* berhasil menghasilkan solusi yang mendekati nilai parameter sebenarnya untuk model Lotka-Volterra. Grafik yang menunjukkan konvergensi error menunjukkan bahwa meskipun ada fluktuasi pada iterasi awal, model akhirnya mencapai konvergensi dengan MSE yang cukup rendah. Hasil fitting parameter menunjukkan bahwa estimasi parameter yang dihasilkan oleh A\* cukup mendekati nilai sebenarnya dengan kesalahan relatif yang wajar, tetapi masih ada ruang untuk perbaikan dalam akurasi estimasi, terutama untuk parameter dengan tingkat sensitivitas yang lebih tinggi.

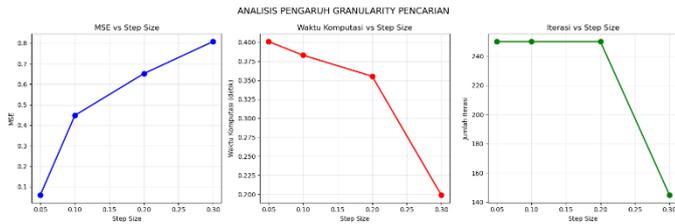
### B. Test Robustness Terhadap Noise



Gambar 8. Test Robustness Terhadap Noise  
Sumber : Dokumen Pribadi

Hasil eksperimen ini menunjukkan bahwa MSE meningkat secara signifikan ketika tingkat noise dalam data meningkat. Ini menunjukkan bahwa model Lotka-Volterra yang dioptimasi dengan A\* cenderung lebih sensitif terhadap noise yang lebih tinggi. Pada tingkat noise yang lebih rendah, model dapat lebih akurat dalam estimasi parameter, sedangkan pada noise tinggi, model kesulitan menghasilkan estimasi yang optimal, bahkan menyebabkan peningkatan error yang tajam. Kecepatan komputasi relatif stabil, namun lebih tinggi pada noise yang lebih besar, karena lebih banyak iterasi dan evaluasi yang diperlukan.

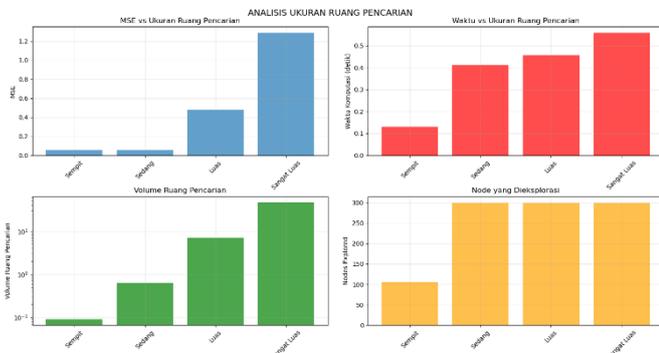
### C. Test Pengaruh Granularity Pencarian



Gambar 9. Test Pengaruh Granularity Pencarian  
Sumber : Dokumen Pribadi

Dari grafik yang dihasilkan, terlihat bahwa MSE meningkat seiring dengan meningkatnya ukuran *step size*. Hal ini menunjukkan bahwa pencarian yang lebih kasar (dengan langkah lebih besar) tidak mampu mengoptimalkan model dengan akurat, sedangkan pencarian yang lebih halus memberikan hasil yang lebih baik meskipun membutuhkan lebih banyak iterasi dan waktu komputasi. Oleh karena itu, dalam pengaturan optimasi, pemilihan ukuran granularity yang tepat sangat penting untuk mencapai keseimbangan antara akurasi dan efisiensi komputasi.

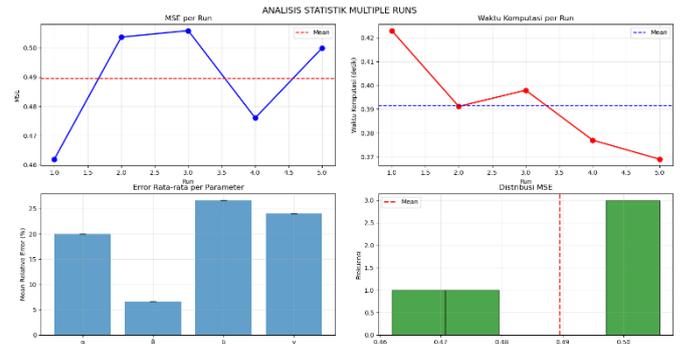
### D. Test Ukuran Ruang Pencarian



Gambar 10. Test Ukuran Ruang Pencarian  
Sumber : Dokumen Pribadi

Pengaruh ukuran ruang pencarian terhadap MSE dan waktu komputasi menunjukkan bahwa semakin besar ruang pencarian yang digunakan, semakin tinggi MSE yang dihasilkan. Namun, hal ini diimbangi dengan peningkatan waktu komputasi yang signifikan. Oleh karena itu, ukuran ruang pencarian yang sangat besar mengorbankan kecepatan pencarian dan memperburuk akurasi model. Sebaliknya, ruang pencarian yang lebih kecil menghasilkan performa yang lebih cepat tetapi dengan MSE yang lebih rendah. Ini memberikan wawasan penting dalam pengaturan ukuran ruang pencarian yang optimal.

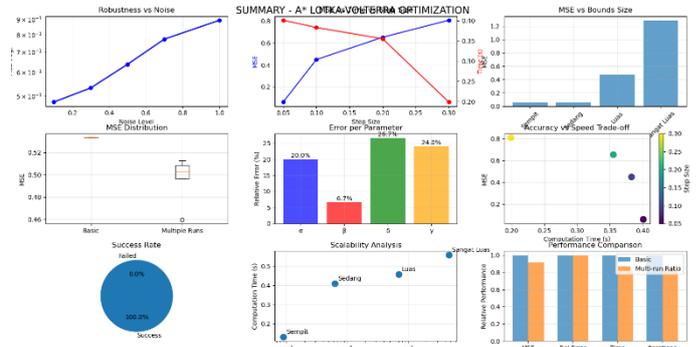
### E. Multiple Runs Untuk Analisis Statistik



Gambar 11. Multiple Runs Untuk Analisis Statistik  
Sumber : Dokumen Pribadi

Melalui eksperimen ini, kita bisa melihat fluktuasi pada MSE dan waktu komputasi di berbagai run, menunjukkan adanya variabilitas dalam hasil optimasi. Meskipun rata-rata kesalahan relatif (mean relative error) menunjukkan kinerja model yang cukup baik, masih ada ketidakteraturan antar run yang menunjukkan bahwa solusi yang lebih stabil bisa diperoleh dengan peningkatan metode optimasi atau pemilihan parameter lebih lanjut. Distribusi MSE yang lebih tersebar pada run yang berbeda mengindikasikan adanya potensi peningkatan dalam algoritma pencarian untuk meningkatkan stabilitas hasil.

### F. Analisis Kesimpulan



Gambar 12. Analisis Kesimpulan  
Sumber : Dokumen Pribadi

Dalam ringkasan hasil optimasi, terlihat bahwa A\* memberikan *trade-off* antara kecepatan dan akurasi. Semakin kecil *step size* dan ruang pencarian, semakin rendah MSE yang dihasilkan, tetapi dengan waktu komputasi yang lebih lama. Di sisi lain, penggunaan bounds yang lebih besar menghasilkan lebih banyak node yang dieksplorasi, yang meningkatkan waktu komputasi dan meningkatkan MSE. Ini menekankan pentingnya memilih ukuran ruang pencarian yang tepat untuk mencapai hasil optimasi terbaik dengan kompromi antara akurasi dan kecepatan.

## VI. KESIMPULAN

Penelitian ini berhasil menunjukkan bahwa algoritma A\* dapat diadaptasi secara efektif untuk menyelesaikan masalah estimasi parameter model Lotka-Volterra. Adaptasi dilakukan dengan memodelkan ruang parameter sebagai graf pencarian, dimana setiap node merepresentasikan kombinasi spesifik

parameter  $\alpha$ ,  $\beta$ ,  $\delta$ , dan  $\gamma$ , serta menggunakan fungsi heuristik admissible yang mengombinasikan estimasi error minimum dengan penalti jarak parameter.

Hasil eksperimen menunjukkan bahwa pendekatan A\* mampu menghasilkan estimasi parameter yang cukup akurat dengan error relatif yang wajar. Algoritma menunjukkan kemampuan konvergensi yang baik dalam eksperimen dasar, meski terdapat fluktuasi pada iterasi awal. Analisis robustness mengungkapkan bahwa performa algoritma sangat dipengaruhi oleh tingkat noise dalam data observasi, dimana peningkatan noise secara signifikan meningkatkan MSE dan memperburuk kualitas estimasi.

Studi pengaruh granularity pencarian mendemonstrasikan adanya *trade-off* penting antara akurasi dan efisiensi komputasi. *Step size* yang lebih kecil menghasilkan estimasi yang lebih akurat namun membutuhkan waktu komputasi yang lebih lama. Sebaliknya, penggunaan ruang pencarian yang terlalu luas cenderung menurunkan akurasi dan meningkatkan kompleksitas komputasi tanpa memberikan manfaat yang sebanding.

Analisis multiple runs mengkonfirmasi adanya variabilitas dalam hasil optimasi, mengindikasikan perlunya peningkatan stabilitas algoritma. Meskipun demikian, algoritma A\* menunjukkan potensi yang menjanjikan sebagai alternatif metode optimasi untuk estimasi parameter model dinamika populasi, khususnya dalam konteks dimana interpretabilitas proses pencarian dan kontrol terhadap eksplorasi ruang parameter menjadi prioritas.

## VII. LAMPIRAN

Github :

<https://github.com/Rusmn/Makalah-Strategi-Algoritma>

## VIII. SARAN DAN UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih atas berkah yang senantiasa diberikan oleh Allah SWT, karena-Nya makalah ini dapat terselesaikan tanpa ada hambatan yang berarti. Selanjutnya, penulis juga mengucapkan terima kasih kepada kedua orang tua penulis yang selalu mendukung dan mendoakan selama proses pengerjaan makalah ini. Penulis juga menyampaikan terima kasih kepada Dr. Ir. Rinaldi Munir, M.T., selaku dosen pengampu mata kuliah IF2211 Strategi Algoritma, yang telah memberikan arahan dan materi-materi yang mendasari penelitian ini. Terakhir, penghargaan juga diberikan kepada rekan-rekan penulis atas masukan dan diskusi yang sangat membantu selama pengerjaan makalah ini.

Untuk pengembangan penelitian lebih lanjut, penulis menyarankan beberapa hal berikut.

1. Penggunaan dataset nyata dari ekosistem tertentu untuk menguji validitas algoritma A\* secara lebih komprehensif dalam estimasi parameter model Lotka-Volterra.
2. Analisis lebih mendalam terhadap perbandingan dengan algoritma optimasi lain, seperti algoritma genetika, simulated annealing, atau metode berbasis gradien.
3. Pengembangan fungsi heuristik adaptif yang dapat menyesuaikan diri berdasarkan karakteristik data untuk meningkatkan akurasi dan efisiensi algoritma.

Diharapkan dengan saran-saran ini, penelitian selanjutnya dapat memberikan analisis yang lebih komprehensif serta

implementasi yang lebih efektif untuk optimasi parameter dalam pemodelan dinamika populasi.

## REFERENCES

- [1] S. I. Salwa, L. A. Shakira and D. Savitri, "DINAMIKA MODEL MANGSA-PEMANGSA LOTKA VOLTERRA DENGAN ADANYA KERJA SAMA BERBURU PADA PEMANGSA," *Jurnal Riset dan Aplikasi Matematika (JRAM)*, vol. 7, no. 2, pp. 195-205, 2023.
- [2] Trivusi, "Trivusi," 20 January 2023. [Online]. Available: <https://www.trivusi.web.id/2023/01/algoritma-a-star.html>. [Accessed 24 June 2025].
- [3] S. Academy, "Sampoerna Academy," 2 May 2022. [Online]. Available: <https://www.sampoernaacademy.sch.id/id/news/dinamika-populasi>. [Accessed 24 June 2025].
- [4] S. Pratiwi, Y. N. Nasution and M. N. Huda, "Analisis Model Matematika Predator-Prey Perikanan Pada Ekosistem Perairan Tercemar," *Basis Jurnal Ilmiah Matematika*, vol. 1, no. 1, pp. 51-60, 2022.
- [5] R. Monica, L. Deswita and R. Pane, "KESTABILAN POPULASI MODEL LOTKA-VOLTERRA TIGA SPESIES DENGAN TITIK KESETIMBANGAN," *JOM FMIPA*, vol. 1, no. 2, pp. 133-141, 2014.
- [6] Liputan6, "Liputan6.com," Liputan6, 28 October 2024. [Online]. Available: [https://www.liputan6.com/feeds/read/5755534/estimasi-adalah-proses-perkiraan-pengertian-jenis-dan-penerapannya?utm\\_source=chatgpt.com&page=2](https://www.liputan6.com/feeds/read/5755534/estimasi-adalah-proses-perkiraan-pengertian-jenis-dan-penerapannya?utm_source=chatgpt.com&page=2). [Accessed 24 June 2025].
- [7] R. Munir, 2025. [Online]. Available: [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/22-Route-Planning-\(2025\)-Bagian2.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/22-Route-Planning-(2025)-Bagian2.pdf). [Accessed 24 June 2025].
- [8] DevX.com, "DevX.com," 11 April 2025. [Online]. Available: <https://www.devx.com/terms/heuristic-programming/#:~:text=Heuristic%20Programming%201%20Definition%20Heuristic%20programming%20is%20an,Terms%20...%208%20Sources%20for%20More%20Information%20>. [Accessed 24 June 2025].

## PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 24 Juni 2025



Muh. Rusmin Nurwadin, 13523068