

Implementasi Algoritma *Greedy* untuk Optimasi Penjadwalan Tugas pada Sistem Otomatisasi *AI Agent* untuk Manajemen *Lead* Pelanggan

Muhammad Dzaky Atha Fadhillah - 18223124

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung, Jalan Ganesha 10 Bandung

E-mail: dzakyatha8@gmail.com , 18223124@std.stei.itb.ac.id

Abstrak— Perkembangan teknologi otomatisasi hingga saat ini memungkinkan unit bisnis/perusahaan melakukan berbagai aktivitas operasional secara otomatis, salah satunya adalah manajemen *lead* pelanggan. Dalam praktiknya, sistem otomatisasi dapat memanfaatkan *AI Agent* untuk mengeksekusi tugas yang memiliki karakteristik berbeda-beda mulai tingkat prioritas, urgensi, nilai bisnis, hingga biaya eksekusi. Dalam mengeksekusi setiap tugas, diperlukan sebuah pendekatan optimasi agar penjadwalan tugas pada sistem otomatisasi ini dapat berjalan secara efektif dan efisien. Penelitian ini menerapkan algoritma *Greedy* untuk optimasi penjadwalan tugas pada sistem tersebut. Setiap tugas diberikan skor berdasarkan tingkat prioritas, urgensi, nilai bisnis, dan biaya eksekusi. Kinerja algoritma dievaluasi menggunakan tiga skenario dengan jumlah tugas sebanyak 20, 50, dan 100 tugas. Algoritma *Greedy* dibandingkan dengan dua pendekatan penjadwalan lainnya yaitu *First In First Out* (FIFO) dan *Random Scheduling*. Ketiga pendekatan dievaluasi menggunakan metrik rata-rata waktu tunggu, rata-rata waktu respons, dan persentase penyelesaian tugas dengan prioritas tinggi. Hasil eksperimen menunjukkan bahwa algoritma *Greedy* secara konsisten unggul dalam menghasilkan waktu tunggu dan waktu respons yang lebih rendah dibandingkan FIFO dan *Random Scheduling*. Algoritma *Greedy* juga mampu menyelesaikan lebih banyak tugas berprioritas tinggi pada seluruh skenario pengujian. Hasil penelitian menunjukkan bahwa algoritma *Greedy* efektif digunakan untuk mengoptimasi penjadwalan tugas pada sistem otomatisasi *AI Agent* untuk manajemen *lead* pelanggan.

Kata kunci— Otomatisasi; Manajemen *Lead* Pelanggan; *AI Agent*; Penjadwalan Tugas; Optimasi; Algoritma *Greedy*

I. PENDAHULUAN

Manajemen *Lead* Pelanggan adalah proses mengidentifikasi, mengelompokkan, memantau, dan menindaklanjuti calon pelanggan suatu unit bisnis/perusahaan yang menghubungi pada berbagai kanal komunikasi seperti *WhatsApp*, *email*, media sosial, *website* resmi perusahaan, dan berbagai kanal lainnya. Seiring dengan meningkatnya jumlah interaksi pelanggan, unit bisnis/perusahaan sering kali menghadapi tantangan dalam menangani banyak *lead*/potensi/calon pelanggan dalam waktu yang bersamaan. Penanganan *lead* pelanggan yang tidak dikelola dengan baik berpotensi dapat mengurangi peluang konversi *lead* dengan potensi bisnis yang

tinggi menjadi pelanggan aktual. Maka dari itu, diperlukan adanya sebuah sistem dengan mekanisme yang dapat membantu unit bisnis/Perusahaan dalam proses pengelolaan *lead* secara lebih efektif dan efisien [1].

Perkembangan teknologi kecerdasan buatan (*Artificial Intelligence*) dan sistem otomatisasi telah mengubah cara unit bisnis/perusahaan dalam menjalankan proses bisnis. Aktivitas-aktivitas operasional yang sebelumnya banyak dilakukan secara manual kini dapat diterapkan otomatisasi untuk meningkatkan efisiensi waktu dan biaya serta mempercepat pengambilan Keputusan. Manajemen *lead* pelanggan menjadi salah satu area operasional yang mengalami transformasi akibat perkembangan otomatisasi ini. Bisnis yang memprioritaskan kecepatan dan ketepatan dalam mengelola calon pelanggan dapat lebih unggul karena kedua faktor tersebut penting dalam mempengaruhi tingkat konversi penjualan dan kepuasan pelanggan.

Sistem otomatisasi semakin berkembang hingga munculnya konsep *AI Agent*, sebuah sistem AI yang mampu mengamati lingkungan, mengambil keputusan, dan melakukan sebuah Tindakan secara independe untuk mencapai tujuan tertentu. Jika *chatbot* konvensional hanya mampu memberikan respons/jawaban di saat terdapat permintaan pengguna, maka *AI Agent* mampu menjalankan aktivitas yang lebih kompleks seperti mengelola percakapan dengan pelanggan, pemrosesan gambar/dokumen, pencarian informasi eksternal, hingga mengakses sistem perangkat lunak lainnya. Dalam konteks bisnis, *AI Agent* mulai digunakan untuk menangani proses/aktivitas operasional yang repetitif seperti melayani permintaan pelanggan, manajemen prospek penjualan, menangani tugas administrasi, dan berbagai aktivitas lainnya. Tetapi, implementasi *AI Agent* ini sebagian besar masih berfokus pada kemampuan mengeksekusi tugas sementara permasalahan penjadwalan dan prioritas tugas belum mendapatkan perhatian yang dibutuhkan.

Pada implementasinya, sistem otomatisasi berbasis *AI Agent* sering menerima banyak tugas secara bersamaan dari berbagai macam sumber. Dalam sistem manajemen *lead* pelanggan, *AI Agent* dapat menerima tugas untuk membalas pesan calon pelanggan, melakukan *follow-up*, memperbarui data pada sistem *Customer Relationship Management* (CRM), membuat laporan penjualan, membuat proposal bisnis, dan lain sebagainya.

Keterbatasan sumber daya komputasi dan waktu untuk merespons *lead* menyebabkan tidak semua tugas dapat dieksekusi secara bersamaan. Kondisi ini merupakan permasalahan penjadwalan tugas (*task scheduling*) untuk menentukan pelaksanaan tugas yang paling optimal berdasarkan karakteristik dan prioritas masing-masing tugas. Berbagai penelitian menunjukkan bahwa pada sistem *multi-agent*, penjadwalan dan alokasi tugas menjadi permasalahan yang sulit dipecahkan secara komputasional sehingga memerlukan pendekatan yang dapat memperoleh solusi optimal dalam waktu yang singkat [2].

Salah satu pendekatan yang akan dibahas pada makalah ini adalah pendekatan algoritmik yang banyak digunakan untuk menyelesaikan permasalahan penjadwalan yaitu algoritma *Greedy*. Algoritma *Greedy* bekerja dengan cara memilih solusi terbaik di setiap langkah untuk memperoleh solusi optimal secara keseluruhan. Algoritma ini unggul dalam implementasinya yang sederhana, kebutuhan sumber daya komputasi yang rendah, dan mampu menghasilkan Keputusan dengan cepat [3]. Karakteristik-karakteristik ini menjadikan algoritma *Greedy* sebagai algoritma yang cocok diterapkan pada sistem otomatisasi *AI Agent* yang membutuhkan respons cepat dan *real-time* dalam menangani berbagai tugas secara bersamaan. Berbagai penelitian menemukan bahwa pada sistem *multi-agent* dan alokasi tugas, pendekatan *Greedy* menghasilkan efisiensi tugas yang baik dengan komputasi yang relatif rendah [2].

Berdasarkan permasalahan yang telah dijelaskan, makalah ini mengusulkan implementasi algoritma *Greedy* untuk optimasi penjadwalan tugas pada sistem otomatisasi *AI Agent* untuk manajemen *lead* pelanggan. Algoritma *Greedy* akan digunakan untuk menentukan prioritas eksekusi tugas berdasarkan beberapa parameter seperti tingkat prioritas, tingkat urgensi, nilai bisnis yang dihasilkan, dan biaya eksekusi. Pendekatan ini diharapkan dapat menghasilkan sistem yang cepat dalam merespons *lead* pelanggan bernilai tinggi, meningkatkan efisiensi operasional, dan mendukung proses pengambilan keputusan secara otomatis oleh unit bisnis/perusahaan.

II. LANDASAN TEORI

A. Manajemen *Lead* Pelanggan

Manajemen *lead* pelanggan adalah proses mengelola calon (*lead*) pelanggan yang dilakukan sejak tahap identifikasi hingga mengkonversi menjadi pelanggan. *Lead* dapat berasal dari berbagai sumber media sosial, *website*, *email*, maupun kanal komunikasi seperti *WhatsApp* dan kanal lainnya [6]. Setiap *lead* memiliki karakteristik yang berbeda mulai dari tingkat urgensi, nilai bisnis, hingga peluang konversi. Semakin banyak jumlah *lead* yang diterima unit bisnis/perusahaan, semakin besar pula tantangan dalam memprioritaskan penanganannya. Keterlambatan merespons *lead* dalam menurunkan potensi dan peluang unit bisnis/perusahaan untuk mendapatkan *lead* bernilai tinggi bahkan dapat menurunkan penjualan.

B. Sistem Otomatisasi

Sistem otomatisasi adalah mesin yang mampu mengubah perilakunya sebagai respons terhadap peristiwa yang tidak

terduga selama pengoperasian tugas yang mana terdapat kemungkinan tidak adanya kontrol manusia secara langsung [7]. Agar berfungsi secara efektif, sistem ini harus mampu mengembangkan dan memodifikasi rencana eksekusi tugas secara dinamis, bereaksi dengan tepat terhadap sebuah peristiwa, dan berkoordinasi dengan manusia. Sistem ini biasanya mencapai perilaku kompleks ini melalui arsitektur kontrol berlapis yang menggabungkan *loop* kontrol reaktif yang cepat untuk stabilitas dan keamanan sistem segera dengan proses perencanaan deliberatif tingkat tinggi yang mengelola tujuan misi secara keseluruhan. Karena mengurangi biaya dan risiko operasional sekaligus memungkinkan kemampuan baru di seluruh domain, sistem otomatisasi dianggap sebagai teknologi rekayasa sistem lintas industri yang transformatif.

C. *AI Agent*

AI Agent adalah sistem komputer yang berada dalam suatu lingkungan yang bertindak secara otonom untuk memenuhi tujuan tertentu tanpa memerlukan instruksi eksplisit secara konstan [8]. Tidak seperti objek perangkat lunak standar yang secara pasif mengeksekusi perintah ketika dipanggil, *AI Agent* memiliki alur kendali dan memutuskan sendiri apakah akan melakukan tindakan yang diminta atau tidak. Agar dianggap cerdas, sebuah agen harus menunjukkan perilaku yang fleksibel dengan menyeimbangkan tiga kemampuan utama yaitu reaktivitas untuk secara dinamis menanggapi perubahan lingkungan yang tidak terduga, proaktivitas untuk mengambil inisiatif yang berorientasi pada tujuan, dan kemampuan sosial untuk berinteraksi, bekerja sama, serta bernegosiasi dengan *agent* otonom lainnya. Meskipun mereka dapat menggunakan teknik kecerdasan buatan, *AI Agent* pada dasarnya adalah sistem rekayasa perangkat lunak kompleks yang berfokus pada pengintegrasian kemampuan untuk pengambilan keputusan independen dan tidak perlu menyelesaikan semua masalah *AI* yang menyeluruh agar berfungsi secara efektif.

D. Penjadwalan Tugas

Penjadwalan tugas adalah persoalan dimana sekelompok agen independen harus memilih tugas dari kumpulan tugas dengan tujuan menemukan penugasan yang memaksimalkan utilitas global secara keseluruhan [3]. Utilitas global ini ditentukan oleh jumlah utilitas tugas individu yang didefinisikan sebagai fungsi submodular berdasarkan himpunan agen spesifik yang ditugaskan pada tugas tertentu. Menemukan penugasan optimal secara matematis kompleks dan umumnya dianggap sebagai masalah *NP-hard* sehingga persoalan ini sering diatasi menggunakan metode aproksimasi seperti algoritma *Greedy*. Penerapan metode seperti ini pada penjadwalan tugas memungkinkan setiap agen untuk bertindak sebagai pengambil keputusan independen dan secara asinkron memilih tugasnya dengan mengandalkan informasi dan jaringan komunikasi lokal.

E. Algoritma *Greedy*

Algoritma *Greedy* adalah algoritma yang bekerja dengan cara membuat serangkaian solusi lokal yang optimal dengan tujuan untuk mencapai solusi global terbaik berdasarkan sifat pilihan *greedy* (*greedy-choice property*) dan substruktur optimal (*optimal substructure*) [9]. Algoritma ini cocok untuk persoalan yang berkaitan dengan optimasi dan salah satunya adalah

penjadwalan tugas. Dalam konteks penjadwalan tugas, algoritma ini akan memaksimalkan jumlah tugas yang kompatibel dengan memilih tugas yang selesai paling awal. Jika diformulasikan secara matematika, *greedy* akan membangun penjadwalan tugas dengan memilih a_i sedemikian sehingga f_i minimal dari himpunan tugas yang kompatibel dengan pilihan sebelumnya. Proses ini diulang dengan mengurutkan aktivitas $S = \{a_1, a_2, \dots, a_n\}$ berdasarkan waktu selesai $\{f_1 \leq f_2 \leq \dots \leq f_n\}$ dan memilih a_i berikutnya yang memenuhi syarat $S_i \geq f_{last_selected}$.

F. First In First Out (FIFO)

Algoritma FIFO adalah metode penjadwalan non-preemptive yang memproses setiap entitas sesuai urutan kedatangannya [14]. Dalam konteks sistem operasi, algoritma ini dikenal juga sebagai *First Come First Served* (FCFS), di mana setiap proses dijalankan hingga selesai sebelum proses berikutnya dimulai. Penelitian oleh Parinduri & Hutagalung (2019) menunjukkan bahwa meskipun algoritma FIFO sederhana, efisiensinya dapat berkurang ketika variasi waktu proses sangat tinggi, karena proses yang panjang dapat menyebabkan keterlambatan signifikan bagi proses lain.

G. Random Scheduling

Dalam konteks minimisasi *makespan online* pada mesin identik, penjadwalan dalam model urutan acak (*Random*) mengasumsikan bahwa meskipun pihak lawan dapat memilih contoh input pekerjaan tertentu, pekerjaan tersebut sebenarnya tiba untuk dijadwalkan sebagai permutasi acak [15]. Kerangka kerja ini memungkinkan algoritma penjadwalan untuk melampaui skenario adversarial terburuk, di mana suatu urutan mungkin diurutkan secara jahat dengan semua pekerjaan besar tiba di akhir; sebaliknya, dalam urutan acak, ada kemungkinan besar bahwa pekerjaan besar akan tiba lebih awal, mengungkapkan informasi penting tentang *makespan* optimal. Untuk memanfaatkan hal ini, para peneliti telah mengembangkan algoritma online deterministik baru yang mencapai rasio kompetitif 1,8478, yang secara signifikan mengurangi daya saing terburuk terbaik yang mungkin sebesar 1,88. Algoritma ini mencapai hal tersebut dengan menganalisis urutan pekerjaan "stabil" dan secara proaktif mempertahankan profil beban tertentu di seluruh mesin—menjadwalkan pekerjaan yang masuk pada mesin yang paling sedikit bebannya, mesin yang sangat banyak bebannya, atau mesin menengah—untuk mencegah jadwal datar yang dapat dirusak oleh pekerjaan besar yang tiba terlambat. Terlepas dari peningkatan kinerja ini, batas bawah matematis membuktikan bahwa tidak ada algoritma online deterministik dalam model urutan acak yang dapat mencapai rasio kompetitif yang lebih kecil dari 4/3, atau lebih kecil dari 3/2 dengan probabilitas tinggi.

III. ALGORITMA METODOLOGI PENELITIAN

A. Gambaran Umum

Penelitian ini menggunakan pendekatan simulasi untuk mengevaluasi penerapan algoritma *greedy* pada sistem otomatisasi AI *agent* untuk manajemen *lead* pelanggan. Simulasi dibangun dalam sebuah program sederhana untuk

mensimulasikan operasional unit bisnis/perusahaan saat menerima berbagai tugas manajemen *lead* dari berbagai media komunikasi seperti *WhatsApp*, *email*, *website*, dan media lainnya. AI *Agent* berperan sebagai eksekutor dari tugas-tugas tersebut yang akan ditentukan urutan eksekusi tugasnya melalui mekanisme penjadwalan berbasis *greedy*. Penelitian ini membandingkan performa algoritma *greedy* dengan dua pendekatan penjadwalan lainnya yaitu *First In First Out* (FIFO) dan *Random Scheduling*.

B. Representasi Tugas

Setiap tugas direpresentasikan sebagai sebuah tuple yang memiliki 4 atribut/karakteristik yaitu prioritas tugas (P), urgensi tugas (U), nilai bisnis (B), dan biaya eksekusi (E). Setiap atribut tugas diberikan rentang nilai dari 1 hingga 10. Setiap kriteria diberikan bobot perhitungan berdasarkan pertimbangan domain bisnis manajemen *lead* pelanggan. Prioritas tugas diberikan bobot sebesar 0,4 karena secara langsung memengaruhi kualitas layanan pelanggan dan peluang konversi bisnis. Urgensi tugas diberikan bobot 0,3 untuk memastikan respons yang cepat terhadap *lead*. Nilai bisnis diberikan bobot 0,2 untuk merepresentasikan potensi keuntungan yang dihasilkan dari suatu tugas. Terakhir, biaya eksekusi diberikan bobot negatif sebesar 0,1 untuk meminimalisir kecenderungan sistem memilih tugas yang membutuhkan sumber daya komputasi tinggi. Untuk mempermudah interpretasi dan perhitungan skor, total bobot dinormalisasi menjadi 1.

C. Implementasi Algoritma *Greedy*

Algoritma *greedy* diimplementasikan dengan fokus menentukan urutan tugas yang akan dieksekusi AI *agent* berdasarkan akumulasi skor dari keempat atribut tugas.

Algorithm 1 Algoritma *Greedy* Penjadwalan Tugas

Input: List of tasks T

```

1: while T is not empty do
2:    $score = P*0.4 + U*0.3 + B*0.2 + E*-0.1$ 
3:   choose task with maximum score
4:   execute selected task
5:   remove task from T
6: end while
7: return Execution order

```

D. Implementasi Simulasi

Komponen simulasi meliputi:

- generator data tugas,
- FIFO, *Random*, *Greedy scheduler*,
- simulator eksekusi,
- modul pencatatan metrik

Data tugas disimpan dalam format CSV agar mudah diproses dan dikembangkan lebih lanjut.

E. Metrik Evaluasi

Evaluasi performa dilakukan berdasarkan tiga metrik kinerja:

- rata-rata waktu yang diperlukan sebelum suatu tugas mulai dieksekusi,
- rata-rata waktu sejak tugas masuk ke antrian sistem hingga selesai dieksekusi,
- persentase tugas prioritas tinggi yang berhasil diselesaikan. Jika diformulakan secara matematis akan seperti ini:

$$CompletionRate = \frac{CompletedHighPriorityTasks}{TotalHighPriorityTasks} \times 100\% \quad (1)$$

IV. EKSPERIMEN DAN HASIL

A. Lingkungan Eksperimen

Eksperimen menggunakan simulator sistem otomatisasi AI *Agent* yang diimplementasikan menggunakan bahasa pemrograman *Python*. Bahasa ini dipilih karena memiliki dukungan pustaka yang memadai untuk simulasi dan analisis data.

B. Dataset

Dataset yang digunakan pada penelitian ini adalah berbentuk daftar tugas simulasi yang merepresentasikan kegiatan operasional bisnis pada konteks manajemen *lead* pelanggan. Setiap tugas memiliki atribut beserta skornya masing-masing. Berikut adalah dataset yang digunakan untuk eksperimen pada penelitian ini:

TABLE I. DATASET EKSPERIMEN

Tugas	P	U	B	E
Membalas pesan <i>lead</i> (calon) pelanggan	10	9	3	2
Mengirim pesan tindak lanjut (<i>follow-up</i>)	9	9	3	2
Memperbarui data CRM	7	5	5	5
Membuat proposal bisnis	5	3	8	7
Membuat laporan analitik	5	4	8	7

Pemilihan skor atribut dari tiap tugas tidak didasarkan oleh apapun dan hanya sekadar asumsi dari penulis saja. Eksperimen akan berfokus pada bagaimana algoritma *greedy* memanfaatkan nilai-nilai ini untuk menghasilkan jadwal eksekusi tugas yang optimal. Agar eksperimen tidak deterministik, pada pembuatan dataset, setiap skor ditambahkan nilai variasi acak sebesar 1 atau -1.

C. Skenario Eksperimen

Eksperimen dilakukan dengan membandingkan algoritma *greedy* dengan metode penjadwalan FIFO dan *Random Scheduling*. Eksperimen dilakukan melalui 3 skenario: 20 tugas, 50 tugas, dan 100 tugas. Setiap skenario dijalankan sebanyak 30 kali untuk mengurangi pengaruh variasi acak pada sistem.

D. Analisis Rata-Rata Waktu Tunggu

TABLE II. HASIL EKSPERIMEN RATA-RATA WAKTU TUNGGU

Jumlah Tugas	FIFO	Random	Greedy
20	37.86 ms	37.26 ms	25.83 ms
50	96.13 ms	96.68 ms	69.83 ms
100	206.29 ms	210.34 ms	152.25 ms

Berdasarkan hasil eksperimen, algoritma *Greedy* menghasilkan rata-rata waktu tunggu yang lebih kecil dibandingkan penjadwalan FIFO dan *Random* pada ketiga skenario eksperimen. Pada skenario 20 tugas, *Greedy* menurunkan rata-rata waktu tunggu kurang lebih sebesar 30% daripada FIFO dan *Random*. Sedangkan pada skenario 50 tugas, *Greedy* menurunkan kurang lebih sebesar 27% dan pada skenario 100 tugas, *Greedy* menurunkan kurang lebih sebesar 26%. Hasil tersebut menunjukkan bahwa algoritma *Greedy* unggul dalam pemilihan tugas yang dianggap paling penting terlebih dahulu berdasarkan akumulasi skor dari keempat atribut tugas. Sebaliknya, FIFO hanya memproses tugas berdasarkan urutan pada daftar tugas dan *Random* tidak mempertimbangkan karakteristik tugas sama sekali. Akibatnya, FIFO dan *Random* sering kali menunda tugas-tugas penting dengan tugas lain yang sebenarnya memiliki kontribusi lebih rendah.

E. Analisis Rata-Rata Waktu Respons

TABLE III. HASIL EKSPERIMEN RATA-RATA WAKTU RESPON

Jumlah Tugas	FIFO	Random	Greedy
20	41.86 ms	41.26 ms	29.83 ms
50	100.07 ms	99.62 ms	73.77 ms
100	210.53 ms	214.58 ms	156.49 ms

Berdasarkan hasil eksperimen, didapatkan bahwa algoritma *Greedy* menghasilkan waktu respons paling rendah secara konsisten di ketiga skenario eksperimen. Pada skenario 20 tugas, rata-rata waktu respons *Greedy* hanya sebesar 29.83 ms dan rata-rata waktu respons untuk FIFO dan *Random* adalah sebesar 41 ms. Pada skenario 50 tugas, rata-rata waktu respons *Greedy* hanya sebesar 73.77 ms dan rata-rata waktu respons untuk FIFO dan *Random* adalah sebesar 99-100 ms. Pada skenario 100 tugas, rata-rata waktu respons *Greedy* hanya sebesar 156.49 ms dan rata-rata waktu respons untuk FIFO dan *Random* adalah sebesar 210 ms. Penurunan rata-rata waktu respons oleh *Greedy* disebabkan karena algoritma ini selalu memilih tugas-tugas dengan akumulasi skor tertinggi untuk dieksekusi terlebih dahulu sehingga tugas-tugas bernilai tinggi terselesaikan dengan lebih cepat.

F. Analisis Persentase Tugas Prioritas Tinggi terselesaikan

TABLE IV. HASIL EKSPERIMEN PERSENTASE TUGAS PRIORITAS TINGGI TERSELESAIKAN

Jumlah Tugas	FIFO	Random	Greedy
20	7.33 %	6.33 %	20.0 %
50	2.47 %	2.47 %	6.45 %
100	1.64 %	1.31 %	3.28 %

Berdasarkan hasil eksperimen, terlihat bahwa algoritma *Greedy* selalu menghasilkan skor persentase tertinggi di ketiga skenario. Pada skenario 20 tugas, *Greedy* dapat menyelesaikan tugas prioritas tinggi hampir sebesar tiga kali lebih banyak dibandingkan FIFO. Pada skenario 50 tugas, *Greedy* menghasilkan juga menghasilkan nilai hampir tiga kali lebih besar daripada FIFO dan *Random*. Pada skenario 100 tugas, *Greedy* masih unggul dengan nilai sebesar 3.28 % sedangkan FIFO hanya sebesar 1.64 % dan *Random* sebesar 1.31 %. Hasil ini menunjukkan bahwa mekanisme penilaian tugas berdasarkan prioritas yang digunakan algoritma *Greedy* berhasil mendorong sistem untuk lebih fokus pada tugas yang lebih tinggi tingkat kepentingannya. Jika dikaitkan dengan konteks manajemen *lead* pelanggan, perolehan hasil ini sangat penting karena *lead* yang memiliki urgensi dan potensi nilai bisnis lebih tinggi dapat menerima respons yang lebih cepat.

G. Analisis Pengaruh Jumlah Tugas

Berdasarkan eksperimen yang telah dilakukan, peningkatan jumlah tugas mulai dari 20 hingga 100 menyebabkan penurunan performa pada seluruh pendekatan penjadwalan yang diujikan. Hal ini diakibatkan oleh semakin banyak tugas yang harus diproses, semakin panjang pula antrian sistem sehingga waktu tunggu dan waktu respons meningkat. Akan tetapi, algoritma *Greedy* tetap menunjukkan keunggulannya dalam mempertahankan performa dibandingkan FIFO dan *Random*. Dapat dikatakan bahwa strategi pemilihan tugas berdasarkan nilai prioritas memberikan keuntungan yang besar di saat sebuah sistem menghadapi beban kerja yang tinggi. Dengan demikian, semakin kompleks sebuah kondisi antrian tugas, semakin terlihat pula keunggulan dari penggunaan algoritma *Greedy*.

V. KESIMPULAN DAN SARAN

A. Kesimpulan

Secara keseluruhan, hasil eksperimen menunjukkan bahwa algoritma *Greedy* terbukti menjadi metode penjadwalan tugas yang efektif dan optimal dalam sistem otomatisasi *AI Agent* untuk manajemen *lead* pelanggan. Algoritma ini unggul pada kemampuannya memanfaatkan informasi dan karakteristik sebuah tugas seperti prioritas, urgensi, nilai bisnis, dan biaya eksekusi untuk menentukan urutan eksekusi tugas. Berbeda dengan pendekatan FIFO yang hanya mempertimbangkan urutan kedatangan tugas dan *Random* yang hanya memilih tugas secara acak, *Greedy* dapat menghasilkan keputusan yang lebih efektif dan sesuai dengan kebutuhan bisnis. Hasil penelitian ini

menunjukkan bahwa penerapan algoritma *Greedy* mampu menurunkan rata-rata waktu tunggu dan waktu respons serta meningkatkan jumlah tugas prioritas tinggi yang berhasil diselesaikan. Oleh karena itu, algoritma *Greedy* dapat menjadi pilihan solusi yang efektif dalam peningkatan efisiensi operasional bisnis pada sistem otomatisasi *AI Agent* untuk pengelolaan *lead* pelanggan.

B. Saran

Penelitian ini memiliki beberapa keterbatasan sehingga terdapat beberapa saran pengembangan yang dapat dipertimbangkan:

- Penelitian ini menggunakan bobot prioritas yang ditentukan berdasarkan pertimbangan domain bisnis manajemen *lead* pelanggan. Pada penelitian selanjutnya, penentuan bobot dapat dilakukan menggunakan metode yang lebih sistematis seperti *Analytic Hierarchy Process* (AHP) [10] atau pendekatan optimasi lainnya agar hasil penjadwalan dapat lebih objektif dan sesuai dengan kebutuhan unit bisnis/perusahaan.
- Penelitian ini menggunakan lingkungan simulasi dengan jenis tugas yang terbatas. Penelitian selanjutnya dapat menggunakan data yang lebih bervariasi dan mendekati kondisi operasional yang sebenarnya seperti integrasi dengan sistem *Customer Relationship Management* (CRM), platform belanja *online*, atau *workflow* otomatisasi yang digunakan oleh unit bisnis/perusahaan. Dengan demikian, performa algoritma dapat dievaluasi pada lingkungan dan kondisi yang lebih dinamis dan kompleks.
- Algoritma *Greedy* yang digunakan pada penelitian ini hanya mempertimbangkan kondisi saat ini (*local optimum*) dalam penentuan urutan penyelesaian tugas. Pada penelitian selanjutnya, dapat dilakukan perbandingan performa *Greedy* dengan algoritma lainnya seperti *Dynamic Programming* [11], *Genetic Algorithm* [12], *Reinforcement Learning* [13], atau algoritma lainnya untuk mengetahui metode mana yang paling efektif dalam menangani penjadwalan tugas pada sistem otomatisasi *AI Agent*.
- Eksperimen yang dilakukan pada penelitian ini dilakukan pada sistem otomatisasi yang diasumsikan hanya memiliki satu *AI Agent*. Penelitian selanjutnya dapat dikembangkan ke dalam sistem *multi-agent*, dimana terdapat beberapa *AI Agent* yang bekerja secara bersamaan dalam menangani berbagai tugas operasional. Pendekatan ini diharapkan dapat meningkatkan skalabilitas sistem dan memberikan gambaran yang lebih realistis mengenai implementasi *AI Agent* pada sebuah unit bisnis/perusahaan.

LAMPIRAN

Repositori *GitHub*:

<https://github.com/dzakyatha/lead-management-scheduler>

PENGAKUAN

Ucapan terima kasih penulis berikan kepada Tuhan Yang Maha Esa, Institut Teknologi Bandung, Program Studi Teknik Informatika, dan dosen mata kuliah IF2211 Strategi Algoritma kelas 01, Bapak Rinaldi Munir. Diharapkan makalah ini dapat bermanfaat bagi pembacanya dan dapat dikembangkan lebih lanjut.

REFERENSI

- [1] Laaksonen, H. (2024). *Improving customer retention with marketing and sales integrated lead management: case Finnish B2B consulting company*.
- [2] Zhu, J., Shi, J., Yang, Z., & Li, B. (2019). *A real-time decentralized algorithm for task scheduling in multi-agent system with continuous damage*. *Applied Soft Computing*, 83, 105628. <https://doi.org/10.1016/j.asoc.2019.105628>
- [3] Qu, G., Brown, D., & Li, N. (2019). *Distributed greedy algorithm for multi-agent task assignment problem with submodular utility functions*. *Automatica*, 105, 206–215. <https://doi.org/10.1016/j.automatica.2019.03.007>
- [4] Shin, H., Li, T., Lee, H., & Tsourdos, A. (2022). *Sample greedy based task allocation for multiple robot systems*. *Swarm Intelligence*, 16(3), 233–260. <https://doi.org/10.1007/s11721-022-00213-0>
- [5] Amador, S., Okamoto, S., & Zivan, R. (2014). *Dynamic Multi-Agent Task Allocation with Spatial and Temporal Constraints*. *Proceedings of the AAAI Conference on Artificial Intelligence*, 28(1). <https://doi.org/10.1609/aaai.v28i1.8889>
- [6] Järvinen, J., & Taiminen, H. (2015). *Harnessing marketing automation for B2B content marketing*. *Industrial Marketing Management*, 54, 164–175. <https://doi.org/10.1016/j.indmarman.2015.07.002>
- [7] Watson, D. P., & Scheidt, D. H. (2005). *Autonomous systems*. *Johns Hopkins APL technical digest*, 26(4), 368–376.
- [8] Wooldridge, M. (2002). *An introduction to multiagent systems*. John Wiley & Sons, Ltd. https://uranos.ch/research/references/Wooldridge_2001/TLTK.pdf
- [9] Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to Algorithms* (4th ed.). The MIT Press.
- [10] Ergu, D., Kou, G., Peng, Y., Shi, Y., & Shi, Y. (2013). *The analytic hierarchy process: task scheduling and resource allocation in cloud computing environment*. *The Journal of Supercomputing*, 64(3), 835–848.
- [11] Psaraftis, H. N. (1980). *A dynamic programming approach for sequencing groups of identical jobs*. *Operations Research*, 28(6), 1347–1359.
- [12] Jang, S. H., Kim, T. Y., Kim, J. K., & Lee, J. S. (2012). *The study of genetic algorithm-based task scheduling for cloud computing*. *International Journal of Control and Automation*, 5(4), 157–162.
- [13] Shyalika, C., Silva, T., & Karunananda, A. (2020). *Reinforcement learning in dynamic task scheduling: A review*. *SN Computer Science*, 1(6), 306.
- [14] Haikal, A. A., Indra, Z., Arfansyah, A., Abror, A. K. A., & Bastio, A. (2025). Implementasi Algoritma First In First Out (FIFO) Berbasis Web untuk Perhitungan dan Simulasi Berbagai Kasus Antrean. *Jurnal Publikasi Ilmu Komputer Dan Multimedia*, 5(1), 53–61. <https://doi.org/10.55606/jupikom.v5i1.5615>
- [15] Susanne Albers and Maximilian Janke. *Scheduling in the Random-Order Model*. In *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*. Leibniz International Proceedings in Informatics (LIPIcs), Volume 168, pp. 68:1–68:18, Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2020)

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.

Jakarta, 19 Juni 2026



M. Dzaky A. F.
18223124