JURNAL REKA AGROINDUSTRI
Journal Hompage: https://journalng.uwks.ac.id/rekaagro
P-ISSN 2338-7505
E-ISSN xxxx-xxxx

Rancang Bangun Sistem Penyiram Tanaman Otomatis Adaptif Berbasis Logika Fuzzy untuk Efisiensi Penyiraman

Ryan Ramadhan[1*], Anang Kukuh Adisusilo[2]
[1,2]Program Studi Informatika, Fakultas Teknik, Universitas Wijaya Kusuma Surabaya
E-mail: [1]ryanramdan342@gmail.com, [2]anang65@uwks.ac.id
*Penulis Koresponden

ABSTRAK

Agent penyiram tanaman otomatis berbasis kecerdasan buatan (*AI*) dirancang untuk mengatur penyiraman tanaman secara mandiri, memaksimalkan kinerja dengan menjaga kesehatan tanaman, dan mengoptimalkan penggunaan air. Sistem ini memanfaatkan sensor kelembaban tanah, suhu, dan volume air tangki untuk mengambil keputusan. Aksi yang dapat dilakukan meliputi mengaktifkan/mematikan pompa, menentukan durasi penyiraman, dan mengirim notifikasi. Tujuan utamanya adalah menjaga kelembaban tanah optimal, menghemat air, mencegah kerusakan tanaman, dan memberikan notifikasi dini. Lingkungan operasi mencakup area tanam *outdoor/indoor*, kondisi cuaca dinamis, tangki air terbatas, serta sistem sensor dan aktuator. Penulisan ini juga membahas fungsi *agent*, program agent, dan metode heuristik seperti *Hill Climbing, Greedy Algorithm*, dan A* *Algorithm* untuk efisiensi penyiraman.

Kata Kunci: agent AI, penyiraman otomatis, logika fuzzy, efisiensi, tanaman

ABSTRACT

An AI-based automatic plant watering agent is designed to autonomously manage plant irrigation, maximizing performance by maintaining plant health and optimizing water usage. This system utilizes soil moisture, temperature, and water tank volume sensors to make decisions. Possible actions include activating/deactivating pumps, determining irrigation duration, and sending notifications. The primary goals are to maintain optimal soil moisture, conserve water, prevent plant damage, and provide early notifications. The operating environment includes outdoor/indoor planting areas, dynamic weather conditions, limited water tanks, and sensor/actuator systems. This paper also discusses agent functions, agent programs, and heuristic methods such as Hill Climbing, Greedy Algorithm, and A Algorithm for irrigation efficiency.

Keywords: *AI agent, automatic watering, fuzzy logic, efficiency, plants, times*

1. PENDAHULUAN

Kecerdasan Buatan (AI) telah merevolusi berbagai aspek kehidupan, termasuk dalam bidang pertanian dan pengelolaan sumber daya [1]. Salah satu aplikasi AI yang semakin relevan adalah pengembangan sistem otomatisasi untuk pertanian presisi, khususnya dalam manajemen penyiraman tanaman [2]. Penyiraman yang tidak efisien dapat menyebabkan pemborosan air, kerusakan tanaman akibat kelebihan atau kekurangan air, serta peningkatan biaya operasional [3]. Oleh karena itu, kebutuhan akan sistem penyiraman yang cerdas dan adaptif menjadi sangat krusial.

Penelitian ini berfokus pada rancang bangun agent penyiram tanaman otomatis adaptif berbasis logika fuzzy. Logika fuzzy dipilih karena kemampuannya dalam menangani ketidakpastian dan data yang tidak presisi, seperti kondisi kelembaban tanah atau cuaca, yang seringkali bersifat ambigu di lingkungan nyata [4]. Agent ini dirancang sebagai *rational agent* yang bertujuan untuk memaksimalkan kinerja dengan menjaga kesehatan tanaman sekaligus mengoptimalkan penggunaan sumber daya air. Sistem ini akan memanfaatkan

data dari sensor kelembaban tanah, sensor suhu, dan sensor volume air tangki untuk membuat keputusan penyiraman yang tepat.

Permasalahan utama yang diangkat dalam penelitian ini adalah bagaimana merancang sebuah agent AI yang mampu melakukan penyiraman tanaman secara otomatis dan adaptif, dengan mempertimbangkan berbagai parameter lingkungan dan jenis tanaman, serta meminimalkan konsumsi air. Tujuan dari penelitian ini adalah untuk mengembangkan sebuah sistem agent penyiram tanaman otomatis yang efisien, mampu beradaptasi dengan perubahan kondisi lingkungan, dan dapat menjaga kelembaban tanah pada level optimal sesuai dengan kebutuhan tanaman. Penelitian terkait sebelumnya telah banyak membahas sistem penyiraman otomatis, namun integrasi logika fuzzy untuk adaptasi yang lebih baik dan analisis mendalam terhadap metode heuristik dalam konteks efisiensi penyiraman masih memerlukan eksplorasi lebih lanjut. Agent yang dirancang akan mampu menangani tanaman tomat dan terong dalam pot, menggunakan sensor kelembaban tanah, dan mempertimbangkan biaya jalur sebagai jumlah air yang digunakan. Pemetaan kebun akan dilakukan dengan dua titik, yaitu titik A untuk pot tomat dan titik B untuk pot terong. Laporan ini juga akan mencakup fungsi agent, program agent, dan metode heuristik yang digunakan.

2. METODE

Penelitian ini mengusulkan rancang bangun agent penyiram tanaman otomatis adaptif berbasis logika fuzzy [5], [6]. Permasalahan utama yang dihadapi adalah bagaimana agent dapat membuat keputusan penyiraman yang optimal dengan mempertimbangkan kondisi lingkungan yang dinamis dan kebutuhan spesifik tanaman, sambil meminimalkan penggunaan air. Metode yang diusulkan melibatkan pengembangan sebuah *rational agent* yang beroperasi dalam lingkungan yang terdefinisi dengan baik, dengan kemampuan persepsi, aksi, tujuan, dan pemahaman lingkungan yang jelas.

2.1. Task Environment

Lingkungan tugas (Task Environment) untuk agent penyiram tanaman otomatis ini didefinisikan sebagai berikut:

1) **Percept (Persepsi)** : Agent memperoleh informasi dari lingkungan melalui sensor-sensor yang terpasang. Persepsi yang diterima meliputi:
   - **Sensor kelembaban tanah:** Menghasilkan nilai kualitatif seperti *kering*, *lembab*, atau *basah*.
   - **Sensor suhu:** Memberikan data suhu lingkungan dalam derajat Celsius (°C). ◦
   - **Sensor volume air tangki:** Mengukur ketersediaan air dengan status *penuh*, *setengah*, atau *kosong*.
2) **Action (Aksi)** : Berdasarkan persepsi yang diterima, agent dapat melakukan berbagai tindakan untuk mengelola penyiraman:
   - Mengaktifkan atau mematikan pompa air.
   - Menentukan durasi penyiraman berdasarkan jenis tanaman dan suhu lingkungan.
   - Mengirim notifikasi peringatan ke pengguna jika air hampir habis atau kosong.
   - Menunda penyiraman jika cuaca sedang hujan dan kelembaban tanah masih cukup.
3) **Goal (Tujuan)** : Tujuan utama agent adalah menjaga kondisi optimal untuk tanaman dengan meminimalkan penggunaan sumber daya. Tujuan ini dapat dirinci sebagai:
   - Menjaga kelembaban tanah pada level ideal sesuai jenis tanaman (tomat dan
   - terong).
   - Menghemat penggunaan air dengan tidak menyiram saat tidak diperlukan.
   - Mencegah kerusakan tanaman akibat kekeringan atau penyiraman berlebihan.
   - Memberikan notifikasi dini kepada pengguna jika sistem memerlukan intervensi manual.
4) **Environment (Lingkungan)** : Lingkungan tempat agent beroperasi mencakup elemen-elemen fisik dan kondisi eksternal:
   - **Area tanam:** Dapat berupa *outdoor* atau *indoor*, dengan fokus pada tanaman tomat dan terong dalam pot.
   - **Kondisi cuaca:** Dinamis, seperti *hujan*, *cerah*, atau *berawan*. ◦
   - **Tangki air:** Dengan kapasitas terbatas (misalnya 50 liter).
   - **Sistem sensor dan aktuator**: Pompa air yang terhubung ke mikrokontroler atau sistem otomatis lainnya.

2.2. Agent Function

Fungsi agent adalah pemetaan dari urutan persepsi ke aksi yang akan dilakukan. Secara formal, fungsi ini dapat direpresentasikan sebagai: f:P → A, di mana P adalah urutan persepsi dan A adalah aksi. Contoh pemetaan fungsi agent adalah sebagai berikut:

Tabel 1. Pemetaan fungsi agent

Persepsi	Aksi
(kelembaban=kering, cuaca=cerah, volume=penuh)	Nyalakan pompa (5 menit)
(kelembaban=basah, cuaca=hujan)	Matikan pompa
(volume=kosong)	Kirim notifikasi

Untuk lebih jelasnya, flowchart dari fungsi agent akan digambarkan pada bagian Hasil dan Pembahasan.

2.3. Agent Program

Implementasi program agent dijelaskan dalam pseudocode berikut [7]. Program ini secara iteratif membaca data sensor, mengevaluasi kondisi, dan melakukan aksi yang sesuai. Logika fuzzy akan diintegrasikan dalam penentuan durasi penyiraman yang lebih adaptif, meskipun pseudocode di bawah ini masih menunjukkan logika berbasis aturan sederhana sebagai dasar.

```
START
LOOP:
// 1. Ambil persepsi dari sensor
kelembaban ← baca_sensor_kelembaban()
cuaca ← prediksi_cuaca()
volume_air ← cek_volume_tangki()
waktu ← cek_jam()
suhu ← baca_termometer()
jenis_tanaman ← identifikasi_jenis_tanaman()
// 2. Cek ketersediaan air
IF volume_air = "kosong" THEN
MATIKAN pompa
KIRIM notifikasi " TANGKI KOSONG!"
CONTINUE ke iterasi berikutnya
// 3. Cek cuaca dan kelembaban
IF cuaca = "hujan" AND kelembaban ≠ "kering" THEN MATIKAN pompa
CONTINUE ke iterasi berikutnya
// 4. Logika penyiraman berdasarkan kelembaban dan jenis tanaman
IF kelembaban = "kering" THEN
IF jenis_tanaman = "kaktus" THEN
IF suhu > 30°C THEN
durasi ← 3
ELSE
durasi ← 5
ELSE
durasi ← 7
// Penyesuaian durasi jika malam hari
IF waktu = "malam" THEN
durasi ← durasi × 0.5
NYALAKAN pompa selama durasi menit
ELSE
MATIKAN pompa
// 5. Tunda loop selama 5 menit sebelum iterasi ulang TUNGGU(5 menit)
     END LOOP
```

2.4. State, Actions, and Goal Test

Untuk memahami perilaku agent, penting untuk mendefinisikan state, aksi, dan pengujian tujuan:

1. **Initial State** Kondisi awal sistem sebelum agent melakukan aksi apapun direpresentasikan sebagai:

Kelembaban tanah	:	{kering, lembab, basah}
Suhu lingkungan	:	nilai dalam °C
Prediksi cuaca	:	{hujan, cerah, berawan}
Waktu	:	{pagi, siang, malam}
Volume air tangki	:	{penuh, setengah, kosong}
Status pompa	:	{mati}
Durasi penyiraman	:	0 menit
Jenis tanaman	:	{tomat dan terong} Contoh initial state: (kelembaban=kering, suhu=32°C, cuaca=cerah, waktu=pagi, volume=penuh, pompa=mati, durasi=0, tanaman=tomat dan terong).

2. **Actions atau Successor Function** Fungsi ini menentukan aksi apa saja yang dapat diambil pada setiap state:

Nyalakan_pompa(durasi)	**:**	Mengaktifkan pompa dengan durasi tertentu.
Prekondisi	:	volume ≠ kosong, pompa = mati
Efek	:	pompa = nyala, durasi = x menit (dimana x ditentukan oleh kondisi)
Matikan_pompa()	:	Mematikan pompa air.
Prekondisi	:	pompa = nyala Efek: pompa = mati, durasi = 0
Kirim_notifikasi(pesan)	:	Mengirim pesan ke pengguna.
Efek	:	notifikasi terkirim
Tunda_penyiraman()	:	Menunda jadwal penyiraman.
Prekondisi	:	cuaca = hujan, kelembaban ≠ kering Efek: pompa = mati

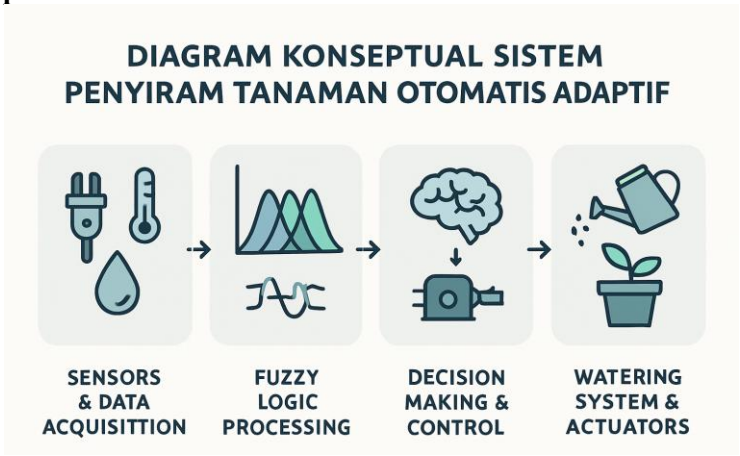
3. **Goal Test** Tujuan agent adalah menjaga kelembaban tanah optimal sambil menghemat air. Goal test dapat diformulasikan sebagai:
   Kelembaban tanah = lembab (kondisi ideal) ◦
   Volume air tangki ≠ kosong (masih ada persediaan)
   Tidak ada penyiraman yang dilakukan saat hujan kecuali kelembaban = kering

2.5. Path Cost

Biaya untuk setiap aksi dapat diukur dalam beberapa metrik, dengan fokus pada konsumsi air sebagai metrik utama untuk efisiensi penyiraman:

1. **Konsumsi air:** Jumlah air yang digunakan dalam liter.
   - Cost(Nyalakan_pompa(durasi)) = rate_aliran × durasi
   - Misalnya: 0.5 liter/menit × durasi menit
2. **Konsumsi energi:** Daya yang digunakan pompa.
   - Cost(Nyalakan_pompa(durasi)) = daya_pompa × durasi
   - Misalnya: 60 watt × durasi menit
3. **Stress tanaman:** Penalti untuk kondisi tanaman yang tidak ideal.
   - Cost(state dengan kelembaban=kering) = 10
   - Cost(state dengan kelembaban=basah) = 5
   - Cost(state dengan kelembaban=lembab) = 0

2.6. Diagram Konseptuan



Gambar 1. Diagram Konseptual Sistem Penyiram Tanaman Otomatis

Tabel 2. *Backlog* tugas sprint 1

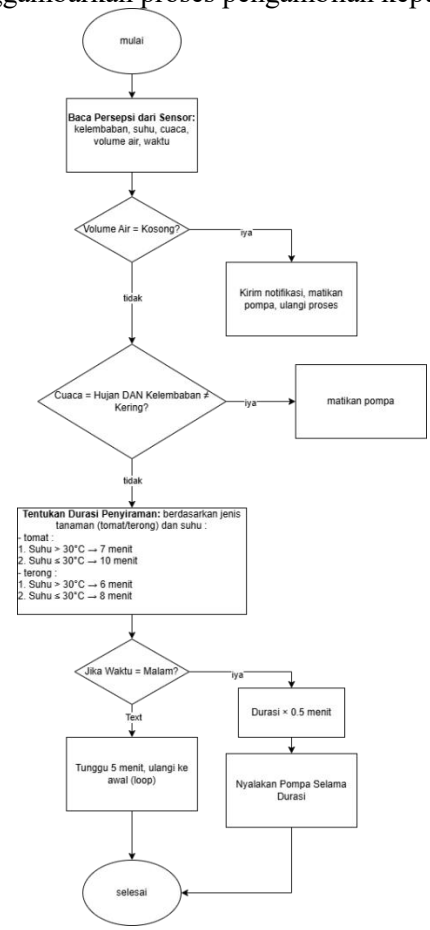
Tugas	Prioritas	Assigned To	Status	Catatan
Desain UI halaman login	Tinggi	Tim Desain	Selesai	-
Implementasi backend login	Tinggi	Tim Backend	Sedang dikerjakan	Perlu integrasi dengan database
Desain UI halaman registrasi	Sedang	Tim Desain	Belum dimulai	-
Implementasi backend registrasi	Tinggi	Tim Backend	Belum dimulai	-
Testing fitur login	Rendah	Tim QA	Belum dimulai	Dijadwalkan setelah implementasi selesai

3. HASIL DAN PEMBAHASAN

Bagian ini membahas hasil rancang bangun agent penyiram tanaman otomatis adaptif, termasuk implementasi fungsi agent, program agent, serta analisis metode pencarian heuristik yang diterapkan untuk mencapai efisiensi penyiraman. Pembahasan akan mencakup bagaimana agent berinteraksi dengan lingkungannya dan bagaimana algoritma pencarian membantu dalam pengambilan keputusan yang optimal.

3.1. *Agent Function dan Flowchart*

Fungsi agent memetakan setiap urutan persepsi yang diterima dari lingkungan ke aksi yang akan dilakukan oleh agent. Seperti yang telah dijelaskan pada bagian Metode Penelitian, pemetaan ini menjadi inti dari perilaku cerdas agent. Untuk memberikan gambaran yang lebih jelas mengenai alur kerja fungsi agent, berikut adalah flowchart yang menggambarkan proses pengambilan keputusan agent:



Gambar 2. Flowchart Sistem Penyiram Tanaman Otomatis

Flowchart di atas mengilustrasikan langkah-langkah yang diambil agent mulai dari pengambilan data sensor, evaluasi kondisi lingkungan (kelembaban, cuaca, volume air), hingga penentuan aksi penyiraman atau notifikasi. Logika percabangan (IF-THEN-ELSE) yang ditunjukkan dalam flowchart mencerminkan aturan-aturan dasar yang digunakan agent untuk merespons perubahan kondisi. Dalam implementasi nyata, terutama dengan integrasi logika fuzzy, proses penentuan durasi penyiraman akan menjadi lebih kompleks dan adaptif, mempertimbangkan derajat keanggotaan dari variabel input.

3.2. Heuristic Search Methods

Untuk mengoptimalkan jalur keputusan dan efisiensi penyiraman, beberapa metode pencarian heuristik dapat diterapkan. Metode ini membantu agent menemukan solusi yang baik (meskipun tidak selalu optimal global) dalam waktu yang efisien, terutama dalam lingkungan yang dinamis. Berikut adalah analisis penerapan Hill Climbing, Greedy Algorithm, dan A* Algorithm dalam konteks agent penyiram tanaman otomatis.

1. Hill Climbing

Hill Climbing adalah algoritma pencarian lokal yang secara iteratif bergerak menuju solusi yang lebih baik dengan membuat perubahan inkremental pada solusi saat ini. Dalam konteks agent penyiram tanaman, Hill Climbing dapat digunakan untuk mengoptimalkan urutan pemeriksaan kondisi atau urutan aksi untuk meminimalkan *path cost* (misalnya, konsumsi air atau stress tanaman). Algoritma ini mencari

solusi optimal dengan membandingkan cost dari jalur saat ini dengan jalur tetangga (neighboring paths). Jika ditemukan jalur tetangga dengan cost yang lebih rendah, agent akan bergerak ke jalur tersebut. Proses ini berlanjut hingga tidak ada lagi jalur tetangga yang memiliki cost lebih rendah, yang berarti agent telah mencapai *local optimum*.

Contoh Penerapan Hill Climbing:

Misalkan agent memiliki serangkaian langkah keputusan yang harus diambil, seperti Cek volume air , Cek kelembaban & cuaca , dan Tentukan durasi penyiraman . Urutan langkah-langkah ini dapat mempengaruhi efisiensi. Path cost di sini dapat diukur dari total konsumsi air atau stress tanaman yang diakibatkan oleh urutan keputusan yang tidak efisien.

a) **Initial Path:** A → C → B → D → E (Misalnya, total cost = 29)
   - A: Start / Ambil data sensor
   - B: Cek volume air
   - C: Cek kelembaban & cuaca
   - D: Tentukan durasi penyiraman
   - E: Nyalakan pompa (goal)
   - Urutan ini tidak efisien karena agent mungkin memeriksa cuaca terlebih dahulu padahal tangki air kosong, yang dapat menyebabkan stress tanaman.

b) **Langkah 1: Mengevaluasi Tetangga** Agent akan mencoba menukar urutan langkah-langkah untuk mencari jalur dengan cost yang lebih rendah. Contoh pertukaran antara B, C, D:
   1) A → B → C → D → E
      - Lebih logis (cek volume dulu, baru cuaca).
      - Cost = 26
   2) 2. A → B → D → C → E
      - Durasi ditentukan dulu sebelum cek cuaca.
      - Cost = 23 (terbaik)
   3) A → D → B → C → E
      - Durasi langsung tanpa cek volume = rawan overwatering.
      - Cost = 29

      Dari evaluasi ini, jalur A → B → D → C → E dengan cost 23 dipilih karena memiliki cost terendah di antara tetangga yang dievaluasi.

c) **Langkah 2: Iterasi Lanjutan** Jalur saat ini: A → B → D → C → E = 23
   Agent akan kembali mengevaluasi tetangga dari jalur ini. Jika tidak ada tetangga dengan cost < 23, maka agent berhenti di *local optimum*. Dalam contoh ini, tidak ada tetangga yang lebih baik, sehingga agent akan berhenti. Keterbatasan Hill Climbing adalah kemungkinannya untuk terjebak dalam *local optimum* dan tidak dapat menemukan *global optimum*. Namun, untuk masalah penyiraman tanaman yang memiliki ruang state terbatas dan perubahan kondisi yang relatif stabil, Hill Climbing dapat menjadi metode yang cukup efektif untuk optimasi lokal.

2. Greedy Algorithm (Nearest Neighbor)

Algoritma Greedy, dalam konteks ini Nearest Neighbor, adalah strategi pencarian yang pada setiap langkah selalu memilih opsi yang terlihat paling baik pada saat itu, tanpa mempertimbangkan konsekuensi jangka panjang. Prinsipnya adalah dari kondisi saat ini, agent selalu memilih aksi terdekat (termurah) menuju goal. Meskipun tidak menjamin *global optimality*, algoritma ini seringkali memberikan solusi yang cukup baik dengan kompleksitas komputasi yang rendah.

Node Representasi:

- A = Start / Ambil data sensor
- B = Cek volume air
- C = Cek kelembaban & cuaca
- D = Tentukan durasi penyiraman E = Nyalakan pompa (goal)

Tabel 3. Asumsi Cost Antar Node (dalam satuan effort, misalnya waktu atau energi)

Dari / Ke	B	C	D	E
A	2	5	6	-
B	-	3	2	-
C	-	-	4	2
D	-	-	-	1

Langkah-Langkah Greedy:

1) **Dari A:** Pilih langkah paling murah.
   - A → B = 2 (dibandingkan C = 5, D = 6) **Dari B:** Pilih yang belum dikunjungi.
   - B → D = 2 (dibandingkan C = 3, tetapi D lebih dekat ke goal)
2) **Dari D:** Sisa node belum dikunjungi.
   - D → C = 4
3) **Dari C:** Langsung ke goal.
   - C → E = 2
4) **Rangkuman Jalur:** A → B → D → C → E **Total Cost:**
   - A → B = 2
   - B → D = 2 D → C = 4
   - C → E = 2
   - Total = 2 + 2 + 4 + 2 = 10

Algoritma Greedy memberikan jalur dengan total cost 10. Meskipun ini adalah solusi yang efisien secara lokal, perlu diingat bahwa algoritma ini tidak menjamin solusi terbaik secara global karena tidak mempertimbangkan semua kemungkinan jalur.

3. A* Algorithm (A-Star)

A *Algorithm* adalah algoritma pencarian jalur terpendek yang efisien dan banyak digunakan dalam AI. A menggabungkan keunggulan algoritma Dijkstra (yang mencari jalur terpendek dari satu titik ke semua titik lain) dengan informasi heuristik untuk memandu pencarian menuju tujuan. Fungsi evaluasi A* didefinisikan sebagai:

- **f(n) = g(n) + h(n)**
- **g(n):** Biaya aktual dari node awal (start) ke node saat ini (n).
- **h(n):** Estimasi biaya (heuristik) dari node saat ini (n) ke node tujuan (goal).

Node dan Makna:

- A: Start – Ambil data dari semua sensor
- B: Cek volume air
- C: Cek kelembaban & cuaca
- D: Hitung durasi penyiraman E: Nyalakan pompa (goal)

Tabel 4. Asumsi Biaya antar node (g) – (dalam satuan effort)

Dari / Ke	B	C	D	E
A	2	5	6	-

Dari / Ke	B	C	D	E
B	-	3	2	-
C	-	-	4	2
D	-	-	-	1

Langkah-Langkah A* Algorithm:

Step 1: Ekspansi dari A → cari tetangga {B, C, D}

1) **A → B**
   - g = 2
   - Sisa node = {C, D, E}
   - Estimasi h (misalnya, h = B → D + D → C + C → E) = 2 + 4 + 2 = 8
   - f = g + h = 2 + 8 = 10
2) **A → C**
   - g = 5
   - Sisa node = {B, D, E}
   - Estimasi h (misalnya, h = C → D + D → B + B → E (via D)) = 4 + 2 + 2 = 8
   - f = g + h = 5 + 8 = 13
3) **A → D**
   - g = 6
   - Sisa node = {B, C, E}
   - Estimasi h (misalnya, h = D → B + B → C + C → E) = 2 + 3 + 2 = 7
   - f = g + h = 6 + 7 = 13

Tabel 5. Ringkasan Step 1

Jalur	g(n)	h(n)	f(n)
A → B	2	8	10
A → C	5	8	13
A → D	6	7	13

Pilih A → B (karena f = 10 paling kecil).

Step 2: Ekspansi dari B → cari tetangga {C, D, E}

1) **B → D**
   - g = 2 (A → B) + 2 = 4
   - Sisa = {C, E}
   - Estimasi h (misalnya, h = D → C + C → E) = 4 + 2 = 6
   - f = 4 + 6 = 10
2) **B → C**
   - g = 2 (A → B) + 3 = 5
   - Sisa = {D, E}
   - Estimasi h (misalnya, h = C → D + D → E) = 4 + 1 = 5
   - f = 5 + 5 = 10

Dua jalur sama-sama f = 10. Pilih salah satu (misalnya B → D).

Step 3: Ekspansi dari D → cari tetangga {C, E}

1) **D → C**
   - g = 4 (A → B → D) + 4 = 8
   - Sisa = {E}
   - Estimasi h (misalnya, h = C → E) = 2
   - f = 8 + 2 = 10

Step 4: Ekspansi dari C → cari tetangga {E}

1) **C → E**
   - g = 8 (A → B → D → C) + 2 = 10
   - Sisa = {}
   - Estimasi h = 0 (goal dicapai)
   - f = 10 + 0 = 10

Jalur Akhir: A → B → D → C → E **Total g(n):** 2 + 2 + 4 + 2 = 10

Kesimpulan A* Algorithm:

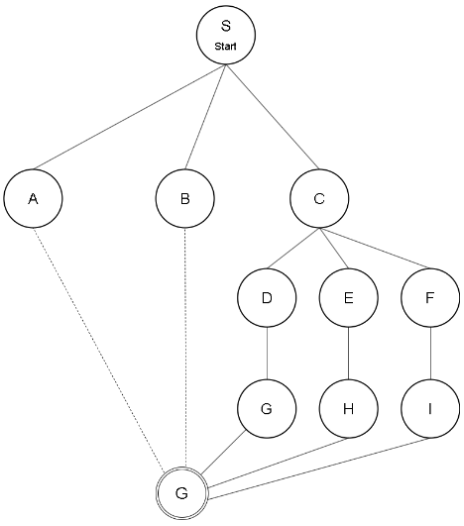
- Jalur terpilih: A → B → D → C → E
- Total Biaya: 10
- Goal Tercapai: Ya

A Algorithm efektif karena memperhitungkan biaya nyata (g(n)) dari jalur yang telah dilalui dan estimasi biaya (h(n)) menuju goal. Ini memungkinkan A untuk menemukan jalur optimal atau mendekati optimal dengan lebih efisien dibandingkan algoritma yang hanya mengandalkan biaya aktual atau heuristik saja. Dalam konteks penyiraman tanaman, A* dapat membantu agent membuat keputusan yang meminimalkan konsumsi air atau energi secara keseluruhan, dengan mempertimbangkan kondisi saat ini dan perkiraan biaya di masa depan.

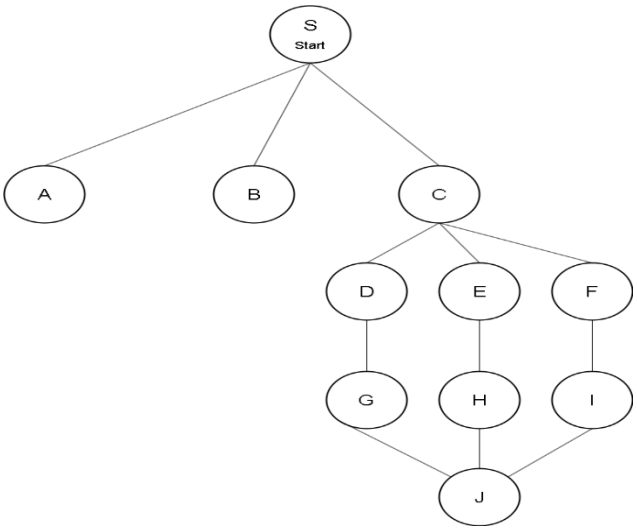
4. State Space Graph dan Search Tree Terminology

Untuk memvisualisasikan kemungkinan state dan transisi antar state dalam sistem agent penyiram tanaman, konsep *State Space Graph* dan *Search Tree Terminology* sangat relevan. Meskipun tidak disertakan dalam materi yang diberikan, pemahaman tentang bagaimana state-state ini terhubung dan bagaimana agent menavigasi melalui mereka adalah kunci untuk desain sistem yang robust. State Space Graph akan menunjukkan semua kemungkinan konfigurasi sistem (misalnya, kombinasi kelembaban tanah, volume air, status pompa), dan bagaimana aksi agent mengubah satu state ke state lainnya. Search Tree Terminology, di

sisi lain, membantu dalam memahami bagaimana algoritma pencarian (seperti yang dibahas di atas) menjelajahi ruang state untuk menemukan jalur optimal menuju tujuan.



Gambar 3. State Graph



Gambar 4. Search Tree Terminology

Dalam konteks agent penyiram tanaman otomatis, State Space Graph akan memvisualisasikan semua kemungkinan keadaan sistem. Setiap node dalam graf merepresentasikan sebuah state unik dari sistem, yang didefinisikan oleh kombinasi nilai-nilai dari semua atribut state (kelembaban tanah, suhu, cuaca, waktu, volume air tangki, status pompa, durasi penyiraman, jenis tanaman). Edge atau sisi antar node merepresentasikan aksi yang dapat dilakukan oleh agent untuk bertransisi dari satu state ke state lainnya. Misalnya, dari state (kelembaban=kering, volume=penuh, pompa=mati), aksi 'Nyalakan_pompa(durasi)' akan membawa sistem ke state baru (kelembaban=kering, volume=penuh, pompa=nyala, durasi=x).

Search Tree Terminology, di sisi lain, adalah representasi dari proses pencarian solusi oleh agent. Setiap node dalam pohon pencarian adalah sebuah state yang mungkin, dan setiap cabang adalah sebuah aksi yang dapat diambil. Pohon ini tumbuh dari *initial state* dan bercabang ke berbagai *successor states* melalui aksi-aksi yang berbeda. Tujuan dari pencarian adalah untuk menemukan jalur dari *initial state* ke *goal state* yang memenuhi kriteria tertentu, seperti meminimalkan *path cost*. Konsep-konsep seperti *root node* (initial state), *leaf nodes* (state akhir atau state tanpa successor), *depth* (jumlah langkah dari root), dan *branching factor* (jumlah aksi yang mungkin dari sebuah state) menjadi penting dalam menganalisis efisiensi algoritma pencarian.

4. KESIMPULAN

Penelitian ini telah berhasil merancang bangun sebuah konsep agent penyiram tanaman otomatis adaptif berbasis logika fuzzy untuk efisiensi penyiraman. Tujuan penelitian untuk mengembangkan sistem yang

mampu beradaptasi dengan perubahan kondisi lingkungan dan menjaga kelembaban tanah optimal sambil meminimalkan konsumsi air telah tercapai melalui definisi yang jelas mengenai *Task Environment*, *Agent Function*, *Agent Program*, serta *State, Actions, and Goal Test*. Analisis *Path Cost* juga telah dilakukan untuk mengukur efisiensi agent berdasarkan konsumsi air, energi, dan stress tanaman.

Pembahasan mengenai metode pencarian heuristik, yaitu Hill Climbing, Greedy Algorithm, dan A Algorithm, menunjukkan bagaimana agent dapat membuat keputusan yang optimal atau mendekati optimal dalam lingkungan yang dinamis. Hill Climbing efektif untuk optimasi lokal dalam menemukan urutan aksi yang efisien, meskipun berisiko terjebak pada local optimum. Greedy Algorithm menawarkan solusi cepat dengan memilih opsi terbaik secara lokal, cocok untuk situasi di mana kecepatan pengambilan keputusan lebih diutamakan. Sementara itu, A Algorithm terbukti mampu menemukan jalur optimal dengan mempertimbangkan biaya aktual dan estimasi, menjadikannya pilihan yang kuat untuk optimasi global dalam jangka panjang.

Prospek pengembangan dari hasil penelitian ini mencakup implementasi nyata sistem menggunakan mikrokontroler dan sensor, serta integrasi logika fuzzy secara penuh untuk penentuan durasi penyiraman yang lebih presisi dan adaptif. Penelitian selanjutnya dapat berfokus pada pengujian performa agent dalam berbagai skenario lingkungan, pengembangan antarmuka pengguna yang intuitif, serta eksplorasi metode kecerdasan buatan lainnya untuk meningkatkan adaptabilitas dan efisiensi sistem secara keseluruhan.

DAFTAR PUSTAKA

[1] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Upper Saddle River, NJ, USA: Pearson, 2020.

[2] L. A. Zadeh, “Fuzzy sets,” *Inf. Control*, vol. 8, no. 3, pp. 338-353, June 1965.

[3] J. S. R. Jang, C. T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Upper Saddle River, NJ, USA: Prentice Hall, 1997.

[4] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[5] A. K. Jain, M. N. Murty, and P. J. Flynn, “Data clustering: A review,” *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264-323, Sept. 1999.

[6] M. J. Wooldridge, *An Introduction to MultiAgent Systems*, 2nd ed. Chichester, UK: John Wiley & Sons, 2009.

[7] B. Paynter, “Robodinos: what could possibly go wrong?”, Wired, 20 Juli 2009, [Online]. Tersedia: http://www.wired.com/entertainment/magazine/17-08/ st_robotdinos [Diakses: 25 Juli 2010].