Optimasi Penjadwalan Imam Masjid Menggunakan Algoritma Genetika dengan Antarmuka Streamlit

Disusun Oleh:

1. **Fatihah Hanin Mahmudah**
2. **Syifa Fatimah**

STMIK AL MUSLIM

PROGRAM STUDI ILMU KOMPUTER

2025

1. Latar Belakang

Penjadwalan imam masjid merupakan tantangan yang cukup kompleks, terutama jika terdapat banyak imam dengan ketersediaan dan preferensi yang berbeda-beda. Jika penjadwalan dilakukan secara manual, maka akan sangat rawan terjadi ketidakseimbangan tugas, bentrok jadwal, hingga kekosongan imam pada waktu tertentu. Untuk mengatasi masalah tersebut, diperlukan solusi cerdas berbasis algoritma. Algoritma Genetika (Genetic Algorithm) merupakan salah satu metode AI yang efektif untuk menyelesaikan permasalahan penjadwalan dengan pendekatan evolusioner.

2. Tujuan

Membangun aplikasi penjadwalan otomatis untuk imam masjid menggunakan algoritma genetika dengan antarmuka Streamlit yang interaktif. Aplikasi ini bertujuan untuk meminimalkan konflik jadwal, membagi tugas imam secara adil, serta memberikan solusi yang efisien dan dapat dimodifikasi dengan mudah.

3. Studi Kasus

Terdapat 7 hari dalam seminggu dan 3 waktu salat utama yang ingin dijadwalkan secara otomatis: Subuh, Maghrib, dan Isya. Tersedia 5 imam dengan preferensi atau ketersediaan tertentu. Tujuannya adalah menjadwalkan imam untuk setiap waktu salat selama 7 hari dengan pembagian yang seimbang dan meminimalkan bentrok.

4. Dataset

Berikut adalah contoh dataset imam dan jadwal:

Imam	Hari Tersedia	Waktu Salat
Ust. Ahmad	Senin, Rabu, Jumat	Subuh, Maghrib
Ust. Budi	Selasa, Kamis	Isya
Ust. Cholid	Setiap Hari	Subuh, Isya
Ust. Dani	Senin - Jumat	Maghrib, Isya
Ust. Eko	Sabtu, Minggu	Subuh

5. Metodologi Algoritma Genetika

a. Data

Dataset berisi daftar imam beserta ketersediaan hari dan batas maksimal tugas per minggu. Contoh format data disimpan dalam file CSV dengan kolom: Nama, Hari, Kapasitas.

b. Preprocessing

Data CSV dibaca menggunakan Pandas dan diubah menjadi struktur data list of dictionaries agar mudah diolah. Setiap imam memiliki atribut nama, daftar hari tersedia, dan kapasitas maksimal tugas.

c. Arsitektur Algoritma Genetika

Tahapan GA meliputi:

1) Inisialisasi populasi: Membuat jadwal acak.
2) Evaluasi fitness: Menghitung skor berdasarkan ketersediaan dan batas tugas.
3) Seleksi: Memilih individu terbaik menggunakan metode seleksi turnamen.
4) Crossover: Menggabungkan dua jadwal untuk membuat keturunan baru.
5) Mutasi: Mengubah sebagian jadwal secara acak untuk menjaga keragaman populasi.
6) Iterasi: Mengulang proses hingga generasi maksimum tercapai.

6. Implementasi

Aplikasi dibangun menggunakan bahasa Python dan framework Streamlit. Pengguna dapat mengunggah file CSV berisi data imam dan parameter algoritma seperti ukuran populasi, jumlah generasi, tingkat crossover, dan mutasi dapat diatu rmelalui antarmuka web.

Berikut implementasi kode dengan komentar:

File app.py

app.py scheduler.py

app.py > ...

```python
import streamlit as st  # Streamlit untuk membuat tampilan web
import pandas as pd  # Untuk memproses data CSV
import matplotlib.pyplot as plt  # Untuk menampilkan grafik
from scheduler import load_imams, run_ga  # Import fungsi dari backend scheduler.py

# Judul halaman utama
st.title("🕌 Penjadwalan Imam Masjid Otomatis (Algoritma Genetika)")

# Sidebar untuk upload data CSV imam
st.sidebar.markdown("### Upload Data Imam (CSV)")
uploaded_file = st.sidebar.file_uploader("Upload file CSV", type="csv")

# Jika file diupload, simpan dan proses
if uploaded_file:
    with open("imams.csv", "wb") as f:
        f.write(uploaded_file.read())
    st.success("File berhasil diupload!")

    # Muat data imam dari file CSV
    imam_data = load_imams("imams.csv")

    # Tampilkan pengaturan parameter algoritma genetika
    st.markdown("### Parameter Algoritma Genetika")
    pop_size = st.slider("Ukuran Populasi", 10, 100, 30)  # Jumlah jadwal dalam populasi
    n_gen = st.slider("Jumlah Generasi", 10, 200, 50)  # Berapa kali iterasi
    cross_rate = st.slider("Tingkat Crossover", 0.1, 1.0, 0.8)  # Kemungkinan kawin silang
    mut_rate = st.slider("Tingkat Mutasi", 0.01, 1.0, 0.05)  # Kemungkinan perubahan random

    # Tombol untuk menjalankan optimasi
    if st.button("Jalankan Optimasi"):
        # Jalankan algoritma genetika
        best, score, history, hari_list, shalat_list = run_ga(
```

```python
    # Proses evolusi selama beberapa generasi
    for _ in range(n_gen):
        new_pop = []
        selected = selection(population, imams, hari_list, shalat_list)
        while len(new_pop) < pop_size:
            p1, p2 = random.sample(selected, 2)
            c1, c2 = crossover(p1, p2, crossover_rate)
            new_pop.append(mutate(c1, mutation_rate, imams))
            if len(new_pop) < pop_size:
                new_pop.append(mutate(c2, mutation_rate, imams))
        population = new_pop
        best = max(population, key=lambda x: evaluate(x, imams, hari_list, shalat_list))
        fitness_history.append(evaluate(best, imams, hari_list, shalat_list))

    # Kembalikan jadwal terbaik dan riwayat fitness-nya
    best = max(population, key=lambda x: evaluate(x, imams, hari_list, shalat_list))
    return best, evaluate(best, imams, hari_list, shalat_list), fitness_history, hari_list, shalat_list
```

File scheduler.py

app.py scheduler.py

scheduler.py > run_ga

```python
import pandas as pd
import random

# Fungsi untuk membaca data imam dari CSV
def load_imams(filepath="imams.csv"):
    df = pd.read_csv(filepath)
    imam_data = []
    for _, row in df.iterrows():
        imam_data.append({
            "nama": row["Nama"],
            "hari": row["Hari"].split(","),
            "kapasitas": int(row["Kapasitas"])
        })
    return imam_data

# Buat jadwal acak (satu individu/kromosom)
def generate_individual(total_slots, imams):
    return [random.choice(imams)["nama"] for _ in range(total_slots)]

# Hitung nilai fitness dari satu jadwal
def evaluate(individual, imams, hari_list, shalat_list):
    tugas_per_imam = {imam["nama"]: 0 for imam in imams}
    score = 0
    for i, imam_name in enumerate(individual):
        hari = hari_list[i // len(shalat_list)]
        imam = next((x for x in imams if x["nama"] == imam_name), None)
        if imam and hari in imam["hari"] and tugas_per_imam[imam_name] < imam["kapasitas"]:
            score += 1  # Dapat poin jika imam tersedia dan belum melebihi batas
        tugas_per_imam[imam_name] += 1
    return score
```

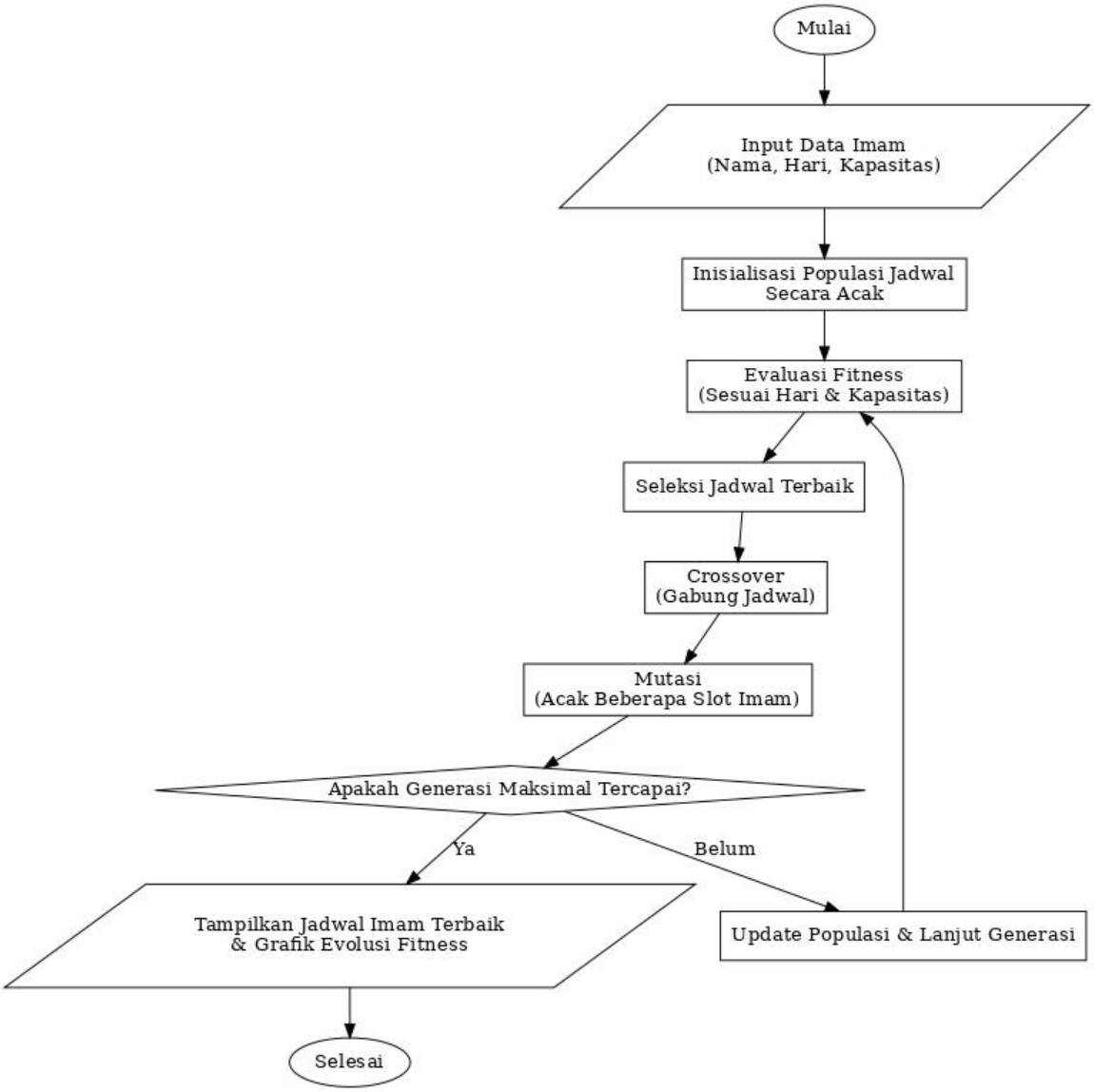
app.py scheduler.py

scheduler.py > run_ga

```python
# Pilih 2 individu terbaik dari populasi
def selection(population, imams, hari_list, shalat_list):
    return sorted(population, key=lambda x: evaluate(x, imams, hari_list, shalat_list), reverse=True)[:2]

# Gabungkan 2 individu menjadi 2 anak baru
def crossover(p1, p2, rate):
    if random.random() > rate:
        return p1[:], p2[:]
    point = random.randint(1, len(p1)-2)
    c1 = p1[:point] + [x for x in p2 if x not in p1[:point]]
    c2 = p2[:point] + [x for x in p1 if x not in p2[:point]]
    return c1, c2

# Mutasi: tukar imam secara acak dengan peluang tertentu
def mutate(ind, rate, imams):
    for i in range(len(ind)):
        if random.random() < rate:
            ind[i] = random.choice(imams)["nama"]
    return ind

# Fungsi utama algoritma genetika
def run_ga(pop_size, n_gen, crossover_rate, mutation_rate, imams):
    hari_list = ["Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Minggu"]
    shalat_list = ["Subuh", "Dzuhur", "Ashar", "Maghrib", "Isya"]
    total_slots = len(hari_list) * len(shalat_list)

    # Inisialisasi populasi awal (kumpulan jadwal)
    population = [generate_individual(total_slots, imams) for _ in range(pop_size)]
    fitness_history = []
```

```python
    # Proses evolusi selama beberapa generasi
    for _ in range(n_gen):
        new_pop = []
        selected = selection(population, imams, hari_list, shalat_list)
        while len(new_pop) < pop_size:
            p1, p2 = random.sample(selected, 2)
            c1, c2 = crossover(p1, p2, crossover_rate)
            new_pop.append(mutate(c1, mutation_rate, imams))
            if len(new_pop) < pop_size:
                new_pop.append(mutate(c2, mutation_rate, imams))
        population = new_pop
        best = max(population, key=lambda x: evaluate(x, imams, hari_list, shalat_list))
        fitness_history.append(evaluate(best, imams, hari_list, shalat_list))

    # Kembalikan jadwal terbaik dan riwayat fitness-nya
    best = max(population, key=lambda x: evaluate(x, imams, hari_list, shalat_list))
    return best, evaluate(best, imams, hari_list, shalat_list), fitness_history, hari_list, shalat_list
```

Flowchart



7. Hasil & Visualisasi

Aplikasi menampilkan jadwal terbaik untuk imam selama 1 minggu dengan visualisasi fitness terbaik dari generasi ke generasi. Jadwal ditampilkan dalam bentuk tabel dan fitness ditampilkan dalam grafik evolusi.

8. Kesimpulan

Penerapan Algoritma Genetika pada penjadwalan imam masjid terbukti efektif dalam mengoptimalkan pembagian tugas dan mengurangi konflik jadwal. Aplikasi ini dapat digunakan sebagai solusi cerdas dan fleksibel yang mudah dioperasikan oleh pengurus masjid.

9. Referensi

- Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley.
- Streamlit Documentation. (2025). Streamlit Docs. https://docs.streamlit.io/
- ChatGPT. (2025). Optimasi Penjadwalan Imam Masjid Menggunakan Algoritma Genetika.