

BAB 4

IMPLEMENTASI DAN PENGUJIAN SISTEM

4.1 Implementasi Sistem

4.1.1 Implementasi Metode Algoritma Genetika

Pada proses implementasi sistem, terdapat batasan dalam jumlah iterasi yang digunakan, yaitu maksimal 1000 iterasi. Batasan ini ditentukan berdasarkan spesifikasi perangkat yang telah dijelaskan pada Subbab 3.7 Spesifikasi Pengembangan Sistem. Saat iterasi melebihi 1000, perangkat yang digunakan mengalami error selama proses generate. Hal ini disebabkan oleh keterbatasan penyimpanan dan durasi waktu yang dibutuhkan. Oleh karena itu, peneliti membatasi jumlah iterasi hingga 1000 untuk memastikan proses pencarian jadwal optimal dapat berjalan dengan lancar tanpa mengganggu kinerja sistem.

Proses perhitungan menggunakan metode Algoritma Genetika terdapat beberapa tahap dalam menghitung data dalam sistem.

Tabel 4.1 Source Code Pembentukan Populasi Awal

```
// Proses Generate Populasi Awal
for ($i = 1; $i <= $jumlahIndividu; $i++) {
    $individu = []; // Menyimpan data individu per
generasi

    foreach ($penjadwalan as $jadwal) { // Loop over each
'penjadwalan'
        // Randomly select ruangan and waktu
        $gen2 = $ruangan->random()-
>only(['kode_ruangan', 'kapasitas']);
        $gen3 = $waktu->random();

        $kodeRuangan = $gen2['kode_ruangan'] ?? null;
        $kapasitasRuangan = $gen2['kapasitas'] ?? null;

        if (!$kodeRuangan || !$kapasitasRuangan ||
!$gen3) {
            continue; // Lewati jika data tidak valid
        }

        $isBentrokKapasitas = $jadwal->jumlah_peserta >
$kapasitasRuangan;

        $bentrokRuangan = [];
        foreach ($individu as $existing) {
            if (
```

```

$kodeRuangan && $existing['gen2']['ruangan'] ==
$gen3->kode_waktu $existing['gen3']['kode_waktu'] ==
) {
    $bentrokRuangan[] =
    $existing['gen1']['kode_pengguna'];
}

$individu[] = [
    'gen1' => [
        'kode_pengguna' => $jadwal->
        kode_pengguna,
        'lembaga' => $jadwal->lembaga->
        nama_lembaga ?? 'N/A',
        'kegiatan' => $jadwal->kegiatan->
        nama_kegiatan ?? 'N/A',
        'jumlah_peserta' => [
            'value' => $jadwal->jumlah_peserta,
            'is_bentrok' => $isBentrokKapasitas,
        ],
    ],
    'gen2' => [
        'ruangan' => $kodeRuangan,
        'kapasitas' => [
            'value' => $kapasitasRuangan,
            'is_bentrok' => $isBentrokKapasitas,
        ],
    ],
    'gen3' => [
        'kode_waktu' => $gen3->kode_waktu,
        'hari' => $gen3->kode_hari,
        'sesi' => $gen3->kode_sesi,
        'is_bentrok' => !empty($bentrokRuangan),
        // Pastikan kunci ini selalu ada
        'bentrok_ruangan' => $bentrokRuangan,
    ],
    'is_bentrok' => $isBentrokKapasitas ||
    !empty($bentrokRuangan),
];

// Add the individual to the population
$generasiData[] = $individu;
}

```

Proses generate populasi awal pada Tabel 4.1 membentuk kumpulan individu, di mana setiap individu merepresentasikan kombinasi jadwal yang terdiri dari informasi pengguna (kode pengguna, lembaga, kegiatan, dan jumlah peserta), ruangan (kode dan kapasitas), serta waktu (hari dan sesi) yang dipilih secara acak. Setiap individu diuji untuk mendeteksi potensi

konflik, seperti jumlah peserta yang melebihi kapasitas ruangan (**konflik kapasitas**) atau penggunaan ruangan yang sama pada waktu yang sama (**konflik ruangan**). Populasi awal yang dihasilkan ini menjadi dasar untuk tahapan optimasi lebih lanjut dalam algoritma genetika.

Tabel 4.2 Source Code Mencari Nilai Fitness

```

    foreach ($individu as $innerKey => $innerKromosom) {
        if (
            $key !== $innerKey && // Jangan
membandingkan dengan dirinya sendiri
            $kromosom['gen2']['ruangan'] ===
$innerKromosom['gen2']['ruangan'] && // Ruangan sama
            $kromosom['gen3']['hari'] ===
$innerKromosom['gen3']['hari'] && // Hari sama
            $kromosom['gen3']['sesi'] ===
$innerKromosom['gen3']['sesi'] // Sesi sama
        ) {
            $totalKR++;
            break; // Hanya hitung satu kali per
konflik
        }
    }

    // Hitung nilai fitness individu dengan rumus: 1
/ (1 + KR + KK)
    $fitnessValue = 1 / (1 + $totalKR + $totalKK);
    $fitnessResults[] = [
        'individu' => $individuIndex + 1,
        'fitness' => $fitnessValue,
        'kr' => $totalKR,
        'kk' => $totalKK
    ];
    $totalFitness += $fitnessValue;

```

Pada Tabel 4.2, kode ini menghitung nilai **fitness** setiap individu berdasarkan konflik ruangan (**KR**) dan konflik kapasitas (**KK**). Konflik ruangan dihitung dengan memeriksa jadwal yang menggunakan ruangan, hari, dan sesi yang sama dalam satu individu. Nilai fitness dihitung menggunakan rumus: $\text{fitness} = 1 / (1 + \text{jumlah konflik ruangan} + \text{konflik kapasitas})$, di mana semakin sedikit konflik, semakin tinggi nilai fitness. Hasil perhitungan mencakup indeks individu, nilai fitness, jumlah konflik ruangan, dan kapasitas, yang menjadi dasar untuk proses seleksi algoritma genetika.

Tabel 4.3 Source Code Proses Seleksi

```
// Langkah 1: Hitung Probabilitas
$probabilities = [];
foreach ($fitnessResults as $fitness) {
    $probabilities[] = $fitness['fitness'] /
$totalFitness;
}

// Langkah 2: Hitung Kumulatif
$cumulativeProbabilities = [];
$currentCumulative = 0;
foreach ($probabilities as $probability) {
    $currentCumulative += $probability;
    $cumulativeProbabilities[] = $currentCumulative;
}

// Langkah 3: Bangkitkan Bilangan Acak
$randomNumbers = [];
$numOfIndividuals = count($fitnessResults);
for ($i = 0; $i < $numOfIndividuals; $i++) {
    $randomNumbers[] = mt_rand(0, 100) / 100; //
Generate random number between 0 and 1
}

// Langkah 4: Menggantikan Individu Lama berdasarkan
Random Number
$newGeneration = [];
foreach ($randomNumbers as $random) {
    foreach ($cumulativeProbabilities as $index =>
$cumulative) {
        if ($random <= $cumulative) {
            $newGeneration[] =
$generasiData[$index];
            break;
        }
    }
}
```

Pada Tabel 4.3, kode ini menjelaskan proses **seleksi individu** dalam algoritma genetika menggunakan metode **roulette wheel selection**. Proses dimulai dengan menghitung **probabilitas seleksi** untuk setiap individu berdasarkan nilai fitness relatif terhadap total fitness populasi. Kemudian, dihitung **probabilitas kumulatif** untuk membentuk rentang seleksi individu. Selanjutnya, dihasilkan bilangan acak antara 0 dan 1 yang digunakan untuk memilih individu dari populasi berdasarkan probabilitas kumulatif tersebut.

Hasil akhirnya adalah generasi baru yang terdiri dari individu-individu terpilih untuk melanjutkan ke tahap optimasi berikutnya.

Tabel 4.4 Source Code Proses *Crossover*

```
// Step 1: Pilih individu berdasarkan crossover rate
for ($i = 0; $i < $numOfIndividuals; $i++) {
    $random = mt_rand(0, 100) / 100; // Generate
bilangan acak antara 0 dan 1
    $output['randomNumbers'][] = $random;

    if ($random <= $crossoverRate) {
        $output['selectedIndividuals'][] = $i; //
Simpan indeks individu yang terpilih
    } else {
        $output['originalIndividuals'][] =
$newGeneration[$i]; // Simpan individu yang tidak terpilih
    }
}

$selected = $output['selectedIndividuals'];
$numOfSelected = count($selected);

// Step 2: Bentuk pasangan dan lakukan crossover jika
lebih dari 1 individu terpilih
if ($numOfSelected > 1) {
    for ($i = 0; $i < $numOfSelected; $i++) {
        $parent1 = $selected[$i];
        $parent2 = $selected[(($i + 1) %
$numOfSelected)]; // Pasangan cyclic

        $crossoverPoint = mt_rand(1,
count($newGeneration[$parent1]) - 1);

        $offspring1 = array_merge(
            array_slice($newGeneration[$parent1],
0, $crossoverPoint),
            array_slice($newGeneration[$parent2],
$crossoverPoint)
        );

        $output['pairs'][] = [
            'parent1' => $parent1 + 1,
            'parent2' => $parent2 + 1,
            'crossoverPoint' => $crossoverPoint,
            'offspring' => [$offspring1],
        ];
        $output['offspring'][] = $offspring1;
    }
}
```

Pada Tabel 4.4, kode ini menjelaskan proses crossover dalam algoritma genetika, yang bertujuan menghasilkan individu baru (offspring) dengan

menggabungkan gen dari dua individu induk. Proses dimulai dengan memilih individu berdasarkan crossover rate, di mana bilangan acak digunakan untuk menentukan apakah individu terpilih atau tidak. Induk yang terpilih kemudian dipasangkan secara siklis, dan proses crossover dilakukan dengan menentukan crossover point secara acak. Gen sebelum crossover point diambil dari induk pertama, sedangkan sisanya diambil dari induk kedua. Hasilnya adalah individu baru (offspring) yang akan digunakan dalam generasi berikutnya untuk meningkatkan keberagaman populasi.

Tabel 4.5 Source Code Mutasi

```
// Identifikasi kromosom bentrok
foreach ($newGeneration as $individuIndex =>
$individu) {
    foreach ($individu as $kromosomIndex =>
$kromosom) {
        if ($kromosom['is_bentrok'] ?? false) {
            $bentrokKromosom[] = [
                'individu' => $individuIndex + 1,
                'kromosom' => $kromosomIndex + 1,
                'detail' => $kromosom,
            ];
        }
    }
}

$totalBentrok = count($bentrokKromosom);

// Pilih gen untuk dimutasi
$totalGenes = count($newGeneration) *
count($newGeneration[0]);
$numGenesToMutate = floor($totalGenes *
$mutationRate);
$randomGenes = [];

while (count($randomGenes) < $numGenesToMutate) {
    $randomIndex = mt_rand(1, $totalGenes);
    if (!in_array($randomIndex, $randomGenes)) {
        $randomGenes[] = $randomIndex;
    }
}
```

Pada Tabel 4.5, kode ini menjelaskan proses **mutasi** dalam algoritma genetika, yang bertujuan memperbaiki individu dengan konflik atau meningkatkan keberagaman populasi. Proses diawali dengan

mengidentifikasi **kromosom bentrok** dalam setiap individu, yaitu kromosom yang menyebabkan konflik seperti ruangan atau waktu yang tumpang tindih. Selanjutnya, ditentukan jumlah gen yang akan dimutasi berdasarkan **mutation rate**, lalu dipilih secara acak gen-gen tersebut menggunakan bilangan acak. Proses mutasi ini mengganti gen yang terpilih dengan data baru, seperti ruangan atau waktu yang berbeda, untuk mengurangi konflik dalam populasi.

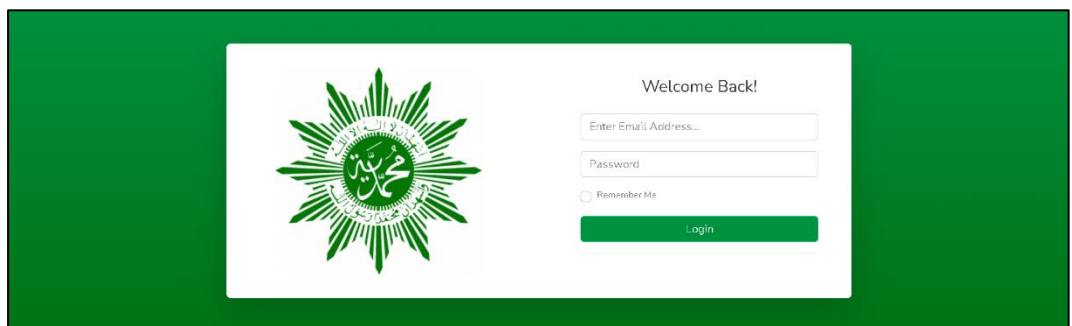
Setelah ditemukan individu dengan nilai fitness sebesar 1, individu tersebut akan dianggap sebagai solusi optimal dan digunakan sebagai jadwal akhir yang kemudian disimpan ke dalam database.

4.2 Pengujian Sistem

Pengembangan sistem jadwal penggunaan ruangan di Gedung Dakwah Muhammadiyah cabang Dukun menggunakan software teks editor “Visual Studio Code” dan software web server “XAMPP”. Pengembangan sistem berbasis website dengan bahasa pemrograman PHP Framework Laravel.

4.2.1 Antarmuka Halaman Login

Antarmuka halaman login adalah tampilan awal Ketika sistem dijalankan. Pada halaman ini user harus memasukkan email dan password agar bisa masuk tampilan selanjutnya. Berikut tampilan antarmuka halaman login ditunjukkan pada gambar 4.1.



Gambar 4.1 Halaman Login

4.2.2 Halaman Dashboard

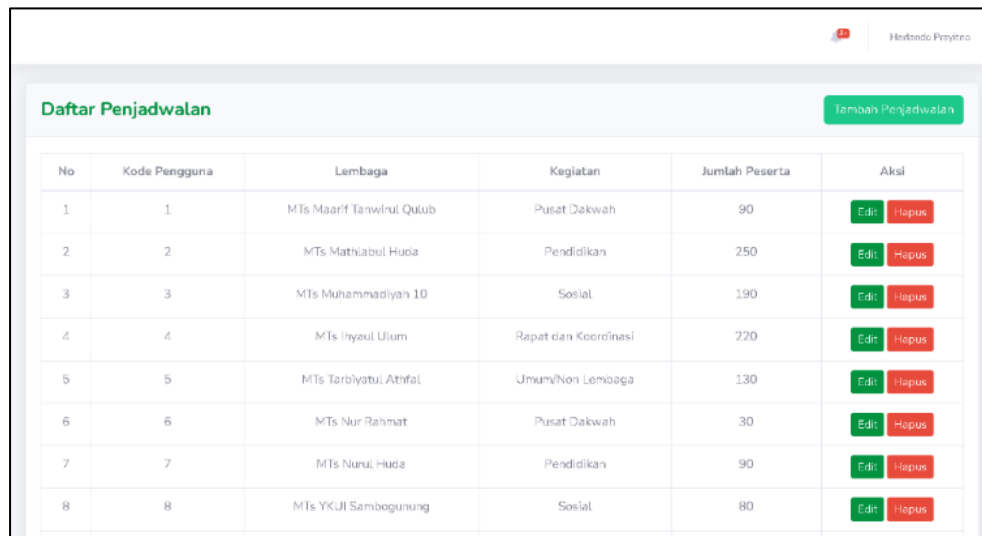
Antarmuka halaman awal atau dashboard adalah tampilan awal ketika aplikasi di jalankan. Pada halaman ini berisi informasi nama sistem dan ucapan selamat datang. Berikut tampilan antarmuka halaman dashboard ditunjukkan pada gambar 4.2:



Gambar 4.2 Halaman Dashboard

4.2.3 Halaman Data Pengguna

Antarmuka halaman data pengguna adalah tampilan pengguna yang akan dilibatkan dalam penjadwalan, halaman ini menjadi GEN 1 dalam proses algoritma genetika. Pada halama ini admin bisa menambah pengguna, edit dan delete data pengguna. Berikut tampilan antarmuka halaman data pengguna ditunjukkan pada gambar 4.3.

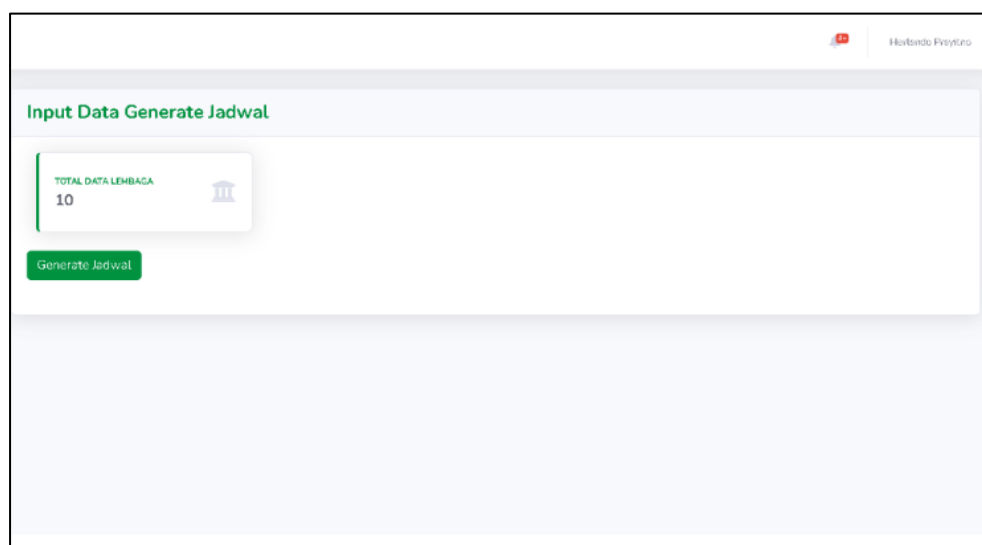


No	Kode Pengguna	Lembaga	Kegiatan	Jumlah Peserta	Aksi
1	1	MTs Maarif Tanwirul Qulub	Pusat Dakwah	90	<button>Edit</button> <button>Hapus</button>
2	2	MTs Mathiabul Huda	Pendidikan	250	<button>Edit</button> <button>Hapus</button>
3	3	MTs Muhammadiyah 10	Sosial	190	<button>Edit</button> <button>Hapus</button>
4	4	MTs Ihyaul Ulum	Rapat dan Koordinasi	220	<button>Edit</button> <button>Hapus</button>
5	5	MTs Tarbiyatul Atrifal	Umum/Non Lembaga	130	<button>Edit</button> <button>Hapus</button>
6	6	MTs Nur Rahmat	Pusat Dakwah	30	<button>Edit</button> <button>Hapus</button>
7	7	MTs Nurul Huda	Pendidikan	90	<button>Edit</button> <button>Hapus</button>
8	8	MTs YKUI Sambogunung	Sosial	80	<button>Edit</button> <button>Hapus</button>

Gambar 4.3 Halaman Pengguna

4.2.4 Halaman *Generate Jadwal Algoritma Genetika*

Antarmuka halaman *generate* jadwal Algoritma Genetika dapat dilihat pada gambar 4.4, tombol *generate* tersebut yang digunakan untuk memproses data dengan parameter yang telah dilakukan secara acak. Bagian pemrosesan jadwal dapat dilihat pada gambar 4.5.



Input Data Generate Jadwal

TOTAL DATA LEMBAGA

10

Generate Jadwal

Gambar 4.4 Halaman *Generate Jadwal*

Gambar 4.4 menunjukkan antarmuka Halaman *Generate Jadwal* untuk memulai proses optimasi jadwal menggunakan algoritma genetika. Pengguna

dapat memasukkan jumlah individu yang akan diproses, kemudian menekan tombol "Generate Jadwal" untuk memulai pengolahan data dengan parameter seperti crossover rate dan mutation rate yang dihasilkan secara acak. Hasil jadwal yang dioptimalkan akan ditampilkan pada halaman output seperti yang terlihat pada Gambar 4.5.

4.2.5 Halaman Jadwal

Antarmuka halaman jadwal adalah halaman yang menampilkan hasil jadwal yang telah dihasilkan melalui proses *generate* jadwal. Jadwal penggunaan ruangan dapat dilihat pada Gambar 4.5.

No	Nama Lembaga	Kegiatan	Jumlah Peserta	Ruangan	Kapasitas Ruangan	Hari	Sesi
1	MTs YKUI Sambogunung	Sosial	80	Lantai 1 - Ruang 3	150	SENIN	1
2	MTs Muhammadiyah 1	Umum/Non Lembaga	220	Lantai 1 - Ruang 4	250	SENIN	2
3	MTs Maarif Tanwirul Qulub	Pusat Dakwah	90	Lantai 1 - Ruang 4	250	SELASA	2
4	MTs YKUI Maskumambang	Rapat dan Koordinasi	40	Lantai 1 - Ruang 4	250	SELASA	1
5	MTs Muhammadiyah 10	Sosial	190	Lantai 1 - Ruang 5	400	RABU	1
6	MTs Nur Rahmat	Pusat Dakwah	30	Lantai 1 - Ruang 4	250	RABU	1
7	MTs Mathlabul Huda	Pendidikan	250	Lantai 1 - Ruang 5	400	KAMIS	2
8	MTs Ihyaul Ulum	Rapat dan Koordinasi	220	Lantai 1 - Ruang 4	250	JUMAT	1
9	MTs Tarbiyatul Athfal	Umum/Non Lembaga	130	Lantai 1 - Ruang 4	250	JUMAT	2
10	MTs Nurul Huda	Pendidikan	90	Lantai 1 - Ruang 5	400	JUMAT	2

Gambar 4.5 Halaman Jadwal

Pada gambar 4.5 menampilkan hasil jadwal penggunaan ruangan yang dihasilkan melalui proses generate jadwal. Tabel tersebut menyajikan informasi jadwal secara terstruktur, meliputi nomor urut, nama kegiatan, lembaga penyelenggara, jumlah peserta, ruangan, kapasitas ruangan, hari, dan sesi waktu. Jadwal ini merupakan hasil optimasi menggunakan algoritma genetika.

4.3 Hasil Analisis Sistem

Adapun pengujian sistem dilakukan sebagai berikut:

1. Data yang digunakan:

- a. Pengguna yang terdiri atas:
 - Lembaga
 - Kegiatan
 - Jumlah Peserta
 - b. Ruangan terdiri atas ruangan dan kapasitas ruangan
 - c. Waktu terdiri atas hari dan sesi
2. Melakukan pengujian terhadap
 - 5 Data Kromosom
 - 10 Data Kromosom
 - 15 Data Kromosom

Dengan melakukan 5 kali generate tiap pengujian

Tabel 4.6 Skenario Pengujian

Jumlah Data Kromosom	Pengujian	Jumlah Generasi	Crossover Rate (%)	Mutasi Rate (%)	Waktu
5	1	15	95	25	0,0592
	2	1	89	69	0,0186
	3	3	39	76	0,0453
	4	1	84	100	0,0166
	5	74	72	6	0,0573
10	1	9	71	23	0,0726
	2	3	34	57	0,0698
	3	4	23	30	0,0338
	4	22	84	19	0,1626
	5	14	2	18	0,0599
15	1	52	96	51	1,8500
	2	234	83	8	0,8621
	3	22	90	10	0,1010
	4	55	89	4	0,1197
	5	18	72	6	0,0914

3. Hasil tiap pengujian ditampilkan seperti pada tabel 4.7.

Tabel 4.7 Hasil Pengujian

Lembaga	Kegiatan	Jumlah Peserta	Ruangan	Kapasitas	Hari	Sesi
MTs Maarif Tanwirul Qulub	Pusat Dakwah	90	Lantai 1 – Ruangan 1	150	Senin	2
MTs Mathlabul Huda	Pendidikan	250	Lantai 1 – Ruangan 4	250	Selasa	2

MTs Muhammadiyah 10	Sosial	190	Lantai 1 – Ruangan 5	400	Selasa	1
MTs Ihyaul Ulum	Rapat dan Koordinasi	220	Lantai 1 – Ruangan 5	400	Senin	1
MTs Tarbiyatul Athfal	Umum/Non Lembaga	130	Lantai 1 – Ruangan 5	400	Kamis	2

Pada tabel 4.7 menampilkan hasil pengujian jadwal optimal untuk data yang digunakan dalam pengujian pertama dengan 5 data, 15 generasi, crossover rate 95%, mutation rate 25%, dan waktu eksekusi 0,0592 detik. Setiap jadwal mencakup informasi lembaga, kegiatan, jumlah peserta, ruangan yang digunakan, kapasitas ruangan, hari, dan sesi. Contoh hasil menunjukkan jadwal yang berhasil mengalokasikan ruangan dan waktu tanpa konflik.

- Lembaga: Mengindikasikan lembaga yang melakukan kegiatan.
- Kegiatan: Jenis kegiatan yang dijadwalkan.
- Jumlah Peserta: Total peserta yang hadir dalam kegiatan.
- Ruangan dan Kapasitas: Alokasi ruangan yang sesuai dengan jumlah peserta.
- Hari dan Sesi: Waktu pelaksanaan kegiatan.

Hasil pengujian menunjukkan masih adanya ketidak relevan antara jumlah peserta dan kapasitas ruangan, seperti pada kegiatan MTs Muhammadiyah 10 dengan 190 peserta di ruangan berkapasitas 400, menyisakan kelebihan 210 tempat. Hal serupa terjadi pada MTs Ihyaul Ulum (220 peserta) dan MTs Tarbiyatul Athfal (130 peserta) yang menggunakan ruangan berkapasitas 400, sehingga efisiensi pemanfaatan ruangan belum optimal. Hal tersebut disebabkan karena penelitian ini difokuskan pada optimasi penjadwalan untuk menghindari konflik jadwal, sehingga penyesuaian jumlah peserta dengan kapasitas ruangan tidak menjadi prioritas utama. Akibatnya, algoritma genetika lebih mengutamakan alokasi jadwal yang efisien tanpa mempertimbangkan secara mendalam kesesuaian

kapasitas ruangan dengan jumlah peserta. Namun, algoritma genetika tetap berhasil menghasilkan jadwal tanpa konflik, dengan rincian pengujian disajikan pada Lampiran 6 hingga Lampiran 20.