

BAB V

PENUTUP

5.1. Kesimpulan

Penelitian ini berhasil menerapkan Genetic Algorithm untuk menyusun jadwal shift Customer Service Agent di CSC Solo, menggunakan data aktual agent, kebutuhan operasional, *Off Request*, dan *constraint* penjadwalan yang ditentukan. Algoritma mengevaluasi fitness berdasarkan penalti dari constraint tersebut dan menghasilkan solusi penjadwalan yang sesuai dengan kondisi operasional penelitian.

Pengujian parameter menunjukkan bahwa konfigurasi Population Size 100, Max Generations 400, dan Mutation Rate 0,01 memberikan performa terbaik. Peneliti mempertahankan Elite Rate 0,10, Tournament Size 3, dan Stagnation Limit 50 sepanjang pengujian, dan memilih konfigurasi berdasarkan nilai fitness sebagai indikator utama, dengan total penalty dan waktu komputasi sebagai pertimbangan pendukung. Konfigurasi terpilih menghasilkan fitness 0,0447 dan total penalty 21,39, setelah 152 generasi dan waktu komputasi 1 menit 24 detik. Nilai unfilled sebesar 0 berarti seluruh slot operasional pada skenario pengujian terisi. Penalty masih muncul pada SC4 (7,52), SC7 (10,36), dan SC8 (3,51), sementara SC1, SC2, SC3, SC5, dan SC6 tidak menghasilkan penalti. Pengujian black-box terhadap 59 kasus menghasilkan 59 kasus lolos tanpa kegagalan.

5.2. Saran

Berdasarkan hasil penelitian yang telah dilakukan, pengembangan selanjutnya dapat difokuskan pada peningkatan kualitas optimasi terhadap constraint yang masih menghasilkan penalti, khususnya SC4, SC7, dan SC8, sehingga pemerataan jadwal yang dihasilkan dapat ditingkatkan. Selain itu, pengujian Genetic Algorithm dapat diperluas dengan menguji parameter lain seperti Elite Rate, Tournament Size, dan Stagnation Limit untuk mengetahui pengaruhnya terhadap kualitas solusi dan waktu komputasi.

Penelitian selanjutnya juga dapat melakukan pengujian pada periode dan kondisi operasional yang lebih beragam untuk mengetahui konsistensi performa *Genetic Algorithm* terhadap variasi jumlah agent, kebutuhan operasional, dan *Off Request*. Pengembangan sistem juga dapat disesuaikan dengan kebutuhan operasional yang muncul setelah sistem digunakan pada lingkungan sebenarnya sehingga sistem dapat terus disempurnakan berdasarkan kebutuhan pengguna dan kondisi penjadwalan yang berkembang.