

Laporan Tugas Besar 1
IF3170 Inteligensi Artifisial
Pencarian Solusi Penjadwalan Kelas Mingguan
dengan Local Search
Semester I Tahun 2025/2026



Disusun oleh :

Shannon Aurelius Anastasya Lie (13523019)
Jessica Allen (13523059)
Benedict Presley (13523067)

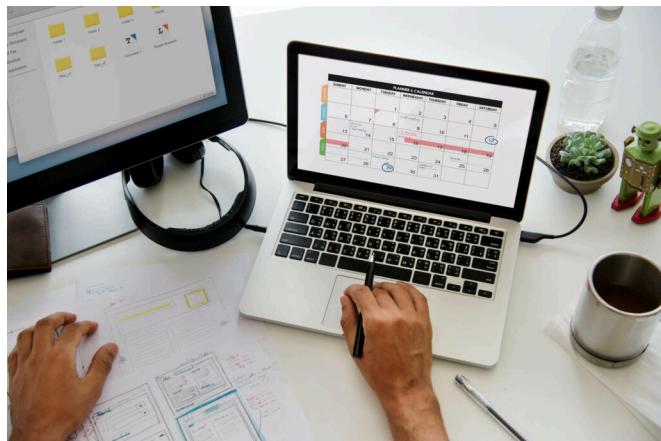
Program Studi Teknik Informatika
Sekolah Teknik Elektro Dan Informatika
Institut Teknologi Bandung
2025

DAFTAR ISI

BAB I DESKRIPSI TUGAS.....	3
BAB II PEMBAHASAN.....	4
2.1. Pemilihan Objective Function.....	4
2.2.1. Konflik Waktu pada Mahasiswa.....	4
2.2.2. Konflik Waktu pada Dosen.....	4
2.2.3. Konflik Ruangan dan Waktu.....	5
2.2.4. Nilai Akhir (Return).....	5
2.2. Implementasi Algoritma Local Search.....	7
2.2.1. Hill-Climbing Algorithm.....	7
2.2.2. Simulated Annealing Algorithm.....	27
2.2.3. Genetic Algorithm.....	33
2.3. Implementasi Lainnya.....	36
2.3.1. models.py.....	36
2.3.2. utils.py.....	39
2.3.3. runners.py.....	40
2.3.4. scheduler.py.....	42
2.3.5. test_generator.py.....	43
2.3.6. main.py.....	44
2.4. Hasil Eksperimen dan Analisis.....	46
2.4.1. Hasil Eksperimen.....	46
1. Test Case Input dari Spesifikasi (input.json).....	46
2. Test Case Semi Large (semi_large_test.json).....	71
3. Test Case Large (large_test.json).....	125
2.4.2. Analisis.....	180
BAB III KESIMPULAN DAN SARAN.....	185
3.1. Kesimpulan.....	185
3.2. Saran.....	185
REFERENSI.....	186
LAMPIRAN.....	187

BAB I

DESKRIPSI TUGAS



Gambar 1 Ilustrasi Penjadwalan

Sumber: <https://easyroster.net/blog/work-schedule/>

Penyusunan jadwal kelas di suatu institusi pendidikan merupakan salah satu proses operasional paling fundamental dan kompleks. Sebuah jadwal yang efektif tidak hanya memaksimalkan pemanfaatan sumber daya seperti ruangan dan waktu, tetapi juga secara langsung memengaruhi kelancaran proses belajar-mengajar bagi ribuan mahasiswa. Namun, dengan banyaknya variabel seperti jumlah mata kuliah, kapasitas ruangan yang terbatas, dan jadwal individual setiap mahasiswa yang tidak boleh tumpang tindih, pencarian solusi penjadwalan yang optimal secara manual menjadi tugas yang hampir mustahil dan sangat tidak efisien.

Untuk mengatasi kompleksitas ini, Intelelegensi Buatan (AI) menawarkan pendekatan optimasi yang kuat melalui algoritma *local search*. Teknik ini sangat cocok untuk masalah penjadwalan karena kemampuannya menjelajahi ruang solusi yang sangat besar untuk menemukan jadwal berkualitas tinggi dengan cara yang sistematis. Algoritma seperti *Hill-Climbing*, *Simulated Annealing*, dan *Genetic Algorithm* dapat secara iteratif memperbaiki sebuah jadwal awal yang acak, dengan cerdas membuat perubahan kecil untuk terus mengurangi konflik hingga tercapai keadaan yang paling optimal atau mendekati optimal.

Dalam Tugas Besar I mata kuliah Intelelegensi Buatan, Anda diminta untuk merancang dan mengimplementasikan sebuah sistem penjadwalan kelas mingguan otomatis. Sistem ini akan memanfaatkan kekuatan tiga algoritma *local search* yang berbeda untuk memecahkan masalah penjadwalan yang diberikan. Dengan mengimplementasikan varian *Hill-Climbing*, *Simulated Annealing*, dan *Genetic Algorithm*, Anda akan melakukan serangkaian eksperimen untuk mengevaluasi dan menganalisis performa, efisiensi, serta konsistensi dari setiap metode dalam menemukan solusi penjadwalan terbaik.

BAB II

PEMBAHASAN

2.1. Pemilihan *Objective Function*

2.2.1. Konflik Waktu pada Mahasiswa

Pada setiap mahasiswa, sistem memeriksa slot waktu yang telah terisi. Jika seorang mahasiswa memiliki dua kelas di jam yang sama (misalnya Selasa 9:00 - 11:00), maka setiap konflik waktu tersebut menambah 1 poin penalti. Logika ini memastikan bahwa seorang mahasiswa tidak dijadwalkan menghadiri dua kelas bersamaan.

```
for student in students:
    filled: set[int] = set()

    for course_id in student.course_list:
        current_course_assignments: List[Assignment] = schedule.course_assignments[course_id]
        for assignment in current_course_assignments:
            hour_idx: int = assignment.time_slot.hour_index()
            if hour_idx in filled:
                penalty += 1
            else:
                filled.add(hour_idx)
```

Gambar 2 *Screenshot* Implementasi *Objective Function* Konflik Waktu Pada Mahasiswa

2.2.2. Konflik Waktu pada Dosen

Bagian ini bekerja dengan cara yang sama seperti pada mahasiswa, tetapi diterapkan pada dosen pengajar. Setiap dosen hanya boleh mengajar satu kelas dalam sebuah slot waktu. Jika dosen dijadwalkan untuk dua kelas bersamaan, sistem menambahkan penalti sebesar 1 poin per konflik.

```
for lecturer in lecturers:
    filled: set[int] = set()

    for course_id in lecturer.course_list:
        current_course_assignments: List[Assignment] = schedule.course_assignments[course_id]
        for assignment in current_course_assignments:
            hour_idx: int = assignment.time_slot.hour_index()
            if hour_idx in filled:
                penalty += 1
            else:
                filled.add(hour_idx)
```

Gambar 3 *Screenshot* Implementasi *Objective Function* Konflik Waktu pada Dosen

2.2.3. Konflik Ruangan dan Waktu

Bagian ini mendeteksi tabrakan dua atau lebih mata kuliah yang menggunakan ruangan dan waktu yang sama. Untuk setiap konflik, sistem menghitung total penalti berdasarkan jumlah dan bobot prioritas mahasiswa yang berdampak. Contohnya, jika dua kelas bertabrakan di ruangan 7606 pada pukul 13:00 - 14:00, dan peserta terdiri atas dua mahasiswa prioritas 1 (bobot 1.75), satu mahasiswa prioritas 2 (bobot 1.5), satu mahasiswa prioritas 3 (bobot 1.25), maka penalti total untuk konflik tersebut adalah $(1.75 \times 2) + (1.5 \times 1) + (1.25 \times 1) = 6.25$.

```
room_time_courses: Dict[Tuple[str, int], List[str]] = {}

for assignment in schedule.assignments:
    room_id: str = assignment.room.room_id
    hour_index: int = assignment.time_slot.hour_index()
    course_id: str = assignment.course.course_id
    if (room_id, hour_index) not in room_time_courses:
        room_time_courses[(room_id, hour_index)] = []
    room_time_courses[(room_id, hour_index)].append(course_id)

for courses_list in room_time_courses.values():
    if len(courses_list) == 1:
        continue

    for course_id in courses_list:
        students_list: List[Student] = students_in_course[course_id]
        for student in students_list:
            penalty += priority_weight(student.priority_map[course_id])
```

Gambar 4 Screenshot Implementasi *Objective Function* Konflik Ruangan dan Waktu

2.2.4. Nilai Akhir (Return)

Fungsi mengembalikan nilai **-penalty** karena optimasi dilakukan untuk memaksimalkan nilai fitness, bukan meminimalkan penalti. Dengan begitu, semakin kecil total penalti, semakin tinggi nilai *objective function* yang dihasilkan.

```
return -penalty
```

Gambar 4 Screenshot return *Objective Function*

Berikut merupakan implementasi fungsi objective secara keseluruhan.

```
def objective(schedule: Schedule, students: List[Student], lecturers: List[Lecturer]) -> float:
    penalty = 0.0

    students_in_course: Dict[str, List[Student]] = {}
    lecturers_in_course: Dict[str, List[Lecturer]] = {}
    for student in students:
        for course_id in student.course_list:
            if course_id not in students_in_course:
                students_in_course[course_id] = []
            students_in_course[course_id].append(student)

    for lecturer in lecturers:
        for course_id in lecturer.course_list:
            if course_id not in lecturers_in_course:
                lecturers_in_course[course_id] = []
            lecturers_in_course[course_id].append(lecturer)

    for student in students:
        filled: Set[int] = set()

        for course_id in student.course_list:
            current_course_assignments: List[Assignment] = schedule.course_assignments[course_id]
            for assignment in current_course_assignments:
                hour_idx: int = assignment.time_slot.hour_index()
                if hour_idx in filled:
                    penalty += 1
                else:
                    filled.add(hour_idx)

    for lecturer in lecturers:
        filled: Set[int] = set()

        for course_id in lecturer.course_list:
            current_course_assignments: List[Assignment] = schedule.course_assignments[course_id]
            for assignment in current_course_assignments:
                hour_idx: int = assignment.time_slot.hour_index()
                if hour_idx in filled:
                    penalty += 1
                else:
                    filled.add(hour_idx)

    room_time_courses: Dict[Tuple[str, int], List[str]] = {}

    for assignment in schedule.assignments:
        room_id: str = assignment.room.room_id
        hour_index: int = assignment.time_slot.hour_index()
        course_id: str = assignment.course.course_id
        if (room_id, hour_index) not in room_time_courses:
            room_time_courses[(room_id, hour_index)] = []
        room_time_courses[(room_id, hour_index)].append(course_id)

    for courses_list in room_time_courses.values():
        if len(courses_list) == 1:
            continue

        for course_id in courses_list:
            students_list: List[Student] = students_in_course[course_id]
            for student in students_list:
                penalty += priority_weight(student.priority_map[course_id])

    return -penalty
```

Gambar 5 Screenshot Implementasi *Objective Function* secara Keseluruhan

2.2. Implementasi Algoritma Local Search

2.2.1. Hill-Climbing Algorithm

Hill-climbing search adalah sebuah algoritma pencarian lokal yang bekerja dengan pendekatan *greedy*. Prosesnya diibaratkan seperti "mendaki Everest dalam kabut tebal dengan amnesia", yang menggambarkan bahwa algoritma ini hanya melihat kondisi terdekatnya tanpa mengetahui gambaran keseluruhan. Pencarian dimulai dari sebuah *initial state* (kondisi awal) yang dibuat secara acak. Dari titik tersebut, algoritma akan secara terus-menerus bergerak ke arah yang nilainya meningkat (jika menggunakan *objective function*) atau menurun (jika menggunakan *cost function*). Proses ini akan berhenti ketika mencapai sebuah "puncak" atau yang disebut local maximum, yaitu sebuah kondisi di mana tidak ada lagi tetangga (*neighbor*) yang memiliki nilai lebih tinggi. Karena sifatnya yang hanya bergerak maju ke kondisi yang lebih baik, algoritma ini berisiko terjebak di puncak lokal dan gagal menemukan global maximum atau solusi terbaik yang sesungguhnya.

Hill-Climbing algorithm memiliki tiga bentuk pendekatan yaitu Steepest Ascent, Stochastic, Sideways Move, dan Random Restart.

1. Steepest Ascent

Algoritma *Steepest-Ascent Hill-Climbing* adalah metode pencarian *greedy* yang dimulai dari sebuah kondisi acak. Pada setiap langkah, ia akan memeriksa semua tetangga di sekitarnya dan selalu memilih satu tetangga dengan nilai paling tinggi (*highest-valued successor*). Algoritma akan terus bergerak ke tetangga terbaik ini selama nilainya meningkat. Pencarian akan langsung berhenti ketika mencapai "puncak", yaitu saat tidak ada satu pun tetangga yang memiliki nilai lebih tinggi dari kondisi saat ini.

Mekanisme Steepest Ascent bekerja dengan cara mengevaluasi sejumlah solusi tetangga pada setiap iterasinya. Pada tahap awal, sebuah jadwal acak dibuat sebagai solusi saat ini (*current state*). Kemudian, algoritma akan menghasilkan beberapa jadwal tetangga dengan melakukan perubahan kecil, seperti menukar slot waktu atau ruangan. Setiap jadwal tetangga tersebut dievaluasi menggunakan *objective function*. Dari semua tetangga yang dievaluasi, algoritma akan mengidentifikasi satu tetangga dengan nilai *objective function* tertinggi (terbaik). Jika nilai tetangga terbaik ini lebih baik daripada nilai solusi saat ini, maka algoritma akan pindah ke solusi tetangga tersebut dan mengulangi prosesnya. Namun, jika tidak ada satu pun tetangga yang memiliki nilai lebih baik, algoritma akan langsung berhenti, karena dianggap telah mencapai "puncak".

Langkah-langkah algoritma Steepest Ascent terdiri atas inisialisasi, evaluasi solusi awal, dan proses pencarian iteratif. Pada tahap inisiasi, sebuah solusi awal dibuat secara acak. Kemudian, dalam setiap iterasi, algoritma melakukan tiga langkah utama: pembangkitan dan evaluasi tetangga, pemilihan tetangga terbaik, dan keputusan pergantian solusi. Pada tahap pembangkitan, sejumlah solusi tetangga dibuat dari solusi saat ini. Setelah itu, pada tahap pemilihan, algoritma memilih satu tetangga dengan nilai *objective function* paling tinggi. Terakhir, pada tahap keputusan, jika nilai tetangga terbaik tersebut lebih tinggi dari solusi saat ini, maka ia akan menjadi solusi baru untuk iterasi berikutnya. Jika tidak, proses pencarian berhenti karena telah terjebak di puncak lokal.

Pseudocode

```
function HILL-CLIMBING(problem) returns a state  
that is a local maximum  
  
    current  $\leftarrow$  MAKE-NODE(problem.INITIAL-STATE)  
  
    loop do  
  
        neighbor  $\leftarrow$  a highest-valued successor of  
current  
  
        if neighbor.VALUE  $\leq$  current.VALUE then  
            return current.STATE  
  
        current  $\leftarrow$  neighbor
```

Kelebihan dan kekurangan dari algoritma ini dapat ditinjau dari tiga aspek. Pada eksplorasi ruang solusi, kelemahan utama algoritma ini adalah ketidakmampuannya untuk keluar dari *local optimum*. Namun, kelebihannya adalah waktu eksekusi yang sangat cepat karena tidak ada perhitungan probabilitas dan pencarinya yang langsung menuju ke "puncak" terdekat. Pada aspek fleksibilitas, algoritma ini sangat sederhana untuk diimplementasikan dan memiliki lebih sedikit parameter yang perlu diatur dibandingkan Simulated Annealing. Kemudian, pada aspek kualitas solusi, algoritma ini dapat menemukan solusi yang "cukup baik" dengan sangat efisien, tetapi jarang sekali mencapai solusi optimal global kecuali jika lanskap pencarian solusinya sederhana.

Implementasi algoritma Steepest Ascent Hill-Climbing pada sistem penjadwalan ini dilakukan melalui dua fungsi yang terdapat pada file

hill_climbing.py: `steepest_ascent_hill_climbing_sampling()` dan
`steepest_ascent_hill_climbing_full()`. Keduanya bertujuan mencari solusi
jadwal dengan total penalti minimum. Perbedaan utamanya terletak pada
cara mereka menjelajahi tetangga
`steepest_ascent_hill_climbing_sampling()` hanya memeriksa sejumlah
sampel tetangga secara acak (`neighbors_to_check`) untuk mempercepat
proses pencarian. Sedangkan, `steepest_ascent_hill_climbing_full()`
memeriksa semua kemungkinan tetangga yang bisa dihasilkan, sehingga
lebih lambat namun lebih teliti dalam menemukan langkah terbaik di
setiap iterasi.

Algoritma Steepest Ascent Hill-Climbing berupaya memaksimalkan nilai *objective function* (meminimalkan penalti) dengan selalu bergerak ke arah peningkatan tertinggi. Proses pencarian dimulai dari sebuah jadwal awal yang acak (*initial schedule*). Pada setiap iterasi, algoritma akan menghasilkan sejumlah jadwal baru (*neighbor schedule*) dengan perubahan kecil, lalu memilih satu tetangga yang memiliki nilai *objective function* paling tinggi. Jika nilai tetangga terbaik ini lebih baik daripada jadwal saat ini, maka ia akan menjadi jadwal baru untuk iterasi berikutnya. Jika tidak, pencarian akan berhenti karena telah mencapai puncak lokal.
Fungsi `steepest_ascent_hill_climbing()` menerima beberapa parameter penting, yaitu sebagai berikut.

Parameter	Keterangan
<code>courses</code> , <code>rooms</code> , <code>time_slots</code> , <code>students</code> , <code>lecturers</code>	Kumpulan data utama yang mendefinisikan masalah penjadwalan.
<code>max_iterations</code>	Jumlah iterasi maksimum yang akan dijalankan. Ini berfungsi sebagai batas waktu agar algoritma pasti berhenti.
<code>neighbors_to_check</code>	(Khusus varian <i>sampling</i>) Jumlah tetangga (<i>neighbor</i>) acak yang akan dievaluasi pada setiap iterasi untuk menemukan langkah terbaik.

Implementasi kode dalam hill_climbing.py:

```

def steepest_ascent_hill_climbing_sampling(courses: List[Course], rooms: List[Room], time_slots: List[TimeSlot], students: List[Student], lecturers: List[Lecturer], max_iterations: int, neighbors_to_check: int) -> Tuple[Schedule, List[float], int, float]:
    start_time: float = time.time()
    current_schedule: Schedule = generate_initial_schedule(courses, rooms, time_slots)
    current_objective: float = objective(current_schedule, students, lecturers)
    objective_history: List[float] = [current_objective]
    iterations: int = 0
    for i in range(max_iterations):
        iterations += 1
        best_neighbor: Schedule | None = None
        best_neighbor_objective: float = float('inf')
        objective_history.append(float('inf'))
        for _ in range(neighbors_to_check):
            neighbor: Schedule = generate_neighbor(current_schedule, rooms, time_slots)
            neighbor_objective: float = objective(neighbor, students, lecturers)
            if neighbor_objective < best_neighbor_objective:
                best_neighbor = neighbor
                best_neighbor_objective = neighbor_objective
        if best_neighbor_objective > current_objective:
            current_schedule = best_neighbor # type: ignore
            current_objective = best_neighbor_objective
            objective_history.append(current_objective)
        else:
            print(f">> Steepest Ascent: Local optimum reached at iteration {iterations}.")
            break
    duration: float = time.time() - start_time
    return current_schedule, objective_history, iterations, duration

def steepest_ascent_hill_climbing_full(courses: List[Course], rooms: List[Room], time_slots: List[TimeSlot], students: List[Student], lecturers: List[Lecturer], max_iterations: int) -> Tuple[Schedule, List[float], int, float]:
    start_time: float = time.time()
    current_schedule: Schedule = generate_initial_schedule(courses, rooms, time_slots)
    current_objective: float = objective(current_schedule, students, lecturers)
    objective_history: List[float] = [current_objective]
    iterations: int = 0
    for i in range(max_iterations):
        iterations += 1
        best_neighbor: Schedule | None = None
        best_neighbor_objective: float = float('inf')
        num_assignments: int = len(current_schedule.assignments)
        for ass_i in range(num_assignments):
            for ass_j in range(ass_i + 1, num_assignments):
                neighbor = copy.deepcopy(current_schedule)
                neighbor[ass_i].room.neighbor_assignments[ass_j].room = neighbor.assignments[ass_j].room
                neighbor[ass_i].time_slot.neighbor_assignments[ass_j].time_slot = neighbor.assignments[ass_j].time_slot
                neighbor[ass_i].time_slot.hour_index = neighbor.assignments[ass_j].time_slot.hour_index
                for room in time_slots:
                    for time_slot in room:
                        if room.room_id == original_room_id and time_slot.hour_index() == original_time_slot_idx:
                            continue
                        neighbor = copy.deepcopy(current_schedule)
                        neighbor[ass_i].room = room
                        neighbor[ass_i].time_slot = time_slot
                        neighbor_objective: float = objective(neighbor, students, lecturers)
                        if neighbor_objective > best_neighbor_objective:
                            best_neighbor = neighbor
                            best_neighbor_objective = neighbor_objective
                if best_neighbor_objective > current_objective:
                    current_schedule = best_neighbor # type: ignore
                    current_objective = best_neighbor_objective
                    objective_history.append(current_objective)
                else:
                    print(f">> Steepest Ascent (Full): local optimum reached at iteration {iterations}.")
                    break
    duration: float = time.time() - start_time
    return current_schedule, objective_history, iterations, duration

```

Gambar 7 Screenshot Implementasi Steepest Ascent Hill-Climbing
Penjelasan implementasi kode:

1. Inisialisasi Variabel Utama

```

current_schedule: Schedule =
generate_initial_schedule(courses, rooms,
time_slots)
current_objective: float =
objective(current_schedule, students,
lecturers)
objective_history: List[float] =
[current_objective]

```

Proses dimulai dengan membuat sebuah jadwal awal acak sebagai kondisi saat ini (current_schedule) dan menghitung nilai objective function-nya. Riwayat nilai (objective_history) juga diinisialisasi dengan nilai awal ini untuk melacak progres pencarian.

2. Inisialisasi Variabel Pelacak

```

iterations: int = 0

```

Dilakukan pengaturan awal untuk variabel iterations yang akan menghitung jumlah total langkah yang telah dijalankan oleh algoritma.

3. Perulangan Utama (Search Loop)

```
for i in range(max_iterations):
```

Proses pencarian iteratif akan terus berjalan selama belum mencapai batas max_iterations. Setiap iterasi merepresentasikan satu langkah pendakian untuk mencari solusi yang lebih baik.

4. Pembangkit dan Pemilihan Tetangga Terbaik (Sampling)

```
best_neighbor: Schedule | None = None
best_neighbor_objective: float = -float('inf')

for _ in range(neighbors_to_check):
    neighbor: Schedule =
    generate_neighbor(current_schedule, rooms,
    time_slots)
    neighbor_objective: float =
    objective(neighbor, students, lecturers)
    if neighbor_objective >
    best_neighbor_objective:
        best_neighbor = neighbor
        best_neighbor_objective =
    neighbor_objective
```

Pada setiap iterasi, algoritma mencari tetangga terbaik dengan cara mengambil sampel. Fungsi generate_neighbor() dipanggil sebanyak neighbors_to_check kali (misalnya 50) untuk menghasilkan jadwal tetangga acak. Setiap tetangga dievaluasi menggunakan objective(), dan yang memiliki nilai tertinggi akan disimpan sebagai best_neighbor.

5. Keputusan Pergantian Solusi

```
if best_neighbor_objective >
current_objective:
    current_schedule = best_neighbor
    current_objective =
best_neighbor_objective

objective_history.append(current_objective)
```

Ini adalah langkah greedy dari algoritma. Jika nilai objective dari best_neighbor yang ditemukan lebih tinggi daripada solusi saat ini (current_objective), maka algoritma akan pindah ke solusi tetangga tersebut.

6. Kondisi Berhenti

```
else:
    print(f"--> Steepest-Ascent: Local optimum
reached at iteration {iterations}.")
    break
```

Jika tidak ada tetangga dari sampel yang memiliki nilai lebih baik (best_neighbor_objective tidak lebih besar dari current_objective), maka algoritma dianggap telah mencapai puncak lokal (local optimum) dan proses pencarian akan dihentikan lebih awal.

7. Pengembalian Solusi Akhir

```
return current_schedule, objective_history,  
iterations, duration
```

Fungsi mengembalikan jadwal terakhir yang ditemukan, riwayat nilai *objective*, total iterasi yang dijalankan, dan durasi eksekusi.

Penjelasan implementasi kode:

1. Inisialisasi Variabel Utama

```
current_schedule: Schedule =  
generate_initial_schedule(courses, rooms,  
time_slots)  
current_objective: float =  
objective(current_schedule, students,  
lecturers)  
objective_history: List[float] =  
[current_objective]
```

Proses dimulai dengan membuat sebuah jadwal awal acak sebagai kondisi saat ini (current_schedule) dan menghitung nilai objective function-nya. Riwayat nilai (objective_history) juga diinisialisasi dengan nilai awal ini untuk melacak progres pencarian.

2. Inisialisasi Variabel Pelacak

```
iterations: int = 0
```

Dilakukan pengaturan awal untuk variabel iterations yang akan menghitung jumlah total langkah yang telah dijalankan oleh algoritma.

3. Perulangan Utama (Search Loop)

```
for i in range(max_iterations):
```

Perulangan utama juga berjalan hingga max_iterations, setiap iterasi adalah satu langkah pencarian penuh.

4. Pembangkit dan Pemilihan Tetangga Terbaik (Pencarian Penuh)

```
# Swap moves for ass_i in  
range(num_assignments): for ass_j in  
range(ass_i + 1, num_assignments): # ...  
generate and evaluate neighbor ... # Move  
moves for ass_i in range(num_assignments): for  
room in rooms: for time_slot in time_slots: #  
... generate and evaluate neighbor ...
```

Ini adalah perbedaan utama. Alih-alih mengambil sampel, fungsi ini secara sistematis menghasilkan semua kemungkinan tetangga. Ini dilakukan melalui dua proses besar yaitu swap moves dan move moves. Swap Moves adalah dua loop for bersarang digunakan untuk mencoba setiap kemungkinan pertukaran (swap) antara dua pertemuan (assignment) di dalam jadwal. Move Moves adalah tiga loop for bersarang digunakan untuk mencoba setiap kemungkinan pemindahan (move) dari setiap pertemuan ke setiap kombinasi slot waktu dan ruangan lainnya. Dari ribuan atau jutaan tetangga yang dihasilkan, algoritma akan memilih satu yang memiliki nilai objective absolut terbaik sebagai best_neighbor.

5. Keputusan Pergantian Solusi dan Kondisi Berhenti

```

        if best_neighbor_objective >
current_objective:
    assert best_neighbor is not None
    current_schedule = best_neighbor
    current_objective =
best_neighbor_objective

    objective_history.append(current_objective)
    sideways_moves_count = 0
    elif best_neighbor_objective ==
current_objective and sideways_moves_count <
max_sideways_moves:
    assert best_neighbor is not None
    current_schedule = best_neighbor
    current_objective =
best_neighbor_objective

    objective_history.append(current_objective)
    sideways_moves_count += 1
    else:
        print(f"--> Sideways-Move (Full):")
        Optimum reached or sideways limit exceeded at
iteration {iterations}.")
        break

```

Logika untuk memutuskan apakah akan bergerak naik (>), menyamping (==), atau berhenti (else) sama persis dengan versi sampling, berdasarkan best_neighbor yang telah ditemukan.

6. Pengembalian Solusi Akhir

```

return current_schedule, objective_history,
iterations, duration

```

Fungsi mengembalikan current_schedule, objective_history, iterations, dan duration, sama seperti versi sampling.

2. Stochastic

Algoritma *Stochastic Hill-climbing* merupakan varian dari pencarian Hill-climbing yang tidak bersifat *greedy*. Berbeda dengan Steepest Ascent, algoritma ini tidak mengevaluasi semua kemungkinan tetangga, melainkan hanya memilih satu tetangga (*successor*) secara acak pada setiap langkahnya. Jika tetangga yang dipilih secara acak tersebut memiliki nilai yang lebih baik dari kondisi saat ini, maka algoritma akan pindah ke tetangga tersebut. Proses ini akan terus diulang hingga mencapai batas iterasi maksimum (*nmax*) yang telah ditentukan, dan pada saat itu pula pencarian akan berhenti.

Pseudocode

```
function HILL-CLIMBING(problem) returns a state  
that is a local maximum  
    current ← MAKE-NODE(problem.INITIAL-STATE)  
    repeat nmax times  
        neighbor ← a random successor of current  
        if neighbor.VALUE > current.VALUE then  
            current ← neighbor
```

Kelebihan dan kekurangan dari algoritma ini dapat ditinjau dari tiga aspek. Pada eksplorasi ruang solusi, kelebihan dari algoritma ini adalah waktu eksekusi per iterasi yang sangat cepat karena hanya satu tetangga yang perlu dievaluasi. Akan tetapi, seperti varian Hill-Climbing lainnya, ia tetap rentan terjebak di *local optimum* dan tidak memiliki mekanisme untuk keluar darinya. Pada aspek fleksibilitas, algoritma ini sangat mudah diimplementasikan karena logikanya yang sederhana. Parameter yang perlu diatur pun relatif sedikit, yaitu *max_iterations* dan *max_stuck_iterations*. Kemudian, pada aspek kualitas solusi, hasilnya sangat bergantung pada "keberuntungan" dari jalur acak yang diambil. Meskipun bisa jadi menemukan solusi yang lebih baik daripada Steepest Ascent jika jalur acaknya menguntungkan, algoritma ini tidak memberikan jaminan konsistensi dan sering kali menghasilkan solusi yang kurang optimal.

Implementasi algoritma Stochastic Hill-Climbing pada sistem penjadwalan ini dilakukan melalui fungsi *stochastic_hill_climbing()* yang terdapat pada file *hill_climbing.py*. Algoritma ini digunakan untuk mencari solusi jadwal yang optimal dengan pendekatan yang lebih cepat per iterasi dibandingkan Steepest Ascent, meskipun dengan potensi kualitas solusi yang lebih bervariasi.

Algoritma Stochastic Hill-Climbing berupaya memaksimalkan nilai *objective function* dengan cara yang efisien. Proses pencarian dimulai dari sebuah jadwal acak (*initial schedule*). Pada setiap iterasi, algoritma ini hanya menghasilkan satu tetangga acak (*neighbor schedule*) dengan melakukan perubahan kecil pada jadwal saat ini. Jika tetangga acak tersebut memiliki nilai *objective function* yang lebih baik, maka ia akan diterima sebagai solusi baru untuk iterasi berikutnya. Proses ini akan berhenti jika tidak ditemukan peningkatan solusi setelah sejumlah iterasi tertentu (*max_stuck_iterations*) atau jika telah mencapai batas iterasi maksimum (*max_iterations*).

Parameter	Keterangan
courses, rooms, time_slots, students, lecturers	Kumpulan data utama yang mendefinisikan masalah penjadwalan dan digunakan dalam evaluasi <i>objective function</i> .
max_iterations	Jumlah iterasi maksimum yang akan dijalankan. Ini berfungsi sebagai batas akhir agar algoritma pasti berhenti.
max_stuck_iterations	Jumlah iterasi berturut-turut tanpa menemukan solusi yang lebih baik sebelum proses pencarian dihentikan lebih awal.

Implementasi Stochastic dalam hill_climbing.py.

```
def stochastic_hill_climbing(courses: List[Course], rooms: List[Room], time_slots: List[TimeSlot], students: List[Student], lecturers: List[Lecturer], max_iterations: int, max_stuck_iterations: int) -> Tuple[Schedule, List[float], int, float]:
    start_time: float = time.time()
    current_schedule: Schedule = generate_initial_schedule(courses, rooms, time_slots)
    current_objective: float = objective(current_schedule, students, lecturers)
    objective_history: List[float] = [current_objective]
    iterations: int = 0
    stuck_count: int = 0

    for i in range(max_iterations):
        iterations += 1
        neighbor: Schedule = generate_neighbor(current_schedule, rooms, time_slots)
        neighbor_objective: float = objective(neighbor, students, lecturers)

        if neighbor_objective > current_objective:
            current_schedule = neighbor
            current_objective = neighbor_objective
            objective_history.append(current_objective)
            stuck_count = 0
        else:
            stuck_count += 1

        if stuck_count == max_stuck_iterations:
            print(f"-> Stochastic: local optima reached after {stuck_count} iterations without improvement.")
            break

    duration: float = time.time() - start_time
    return current_schedule, objective_history, iterations, duration
```

Gambar 8 Screenshot Implementasi Stochastic Hill-Climbing
Penjelasan implementasi kode: stochastic_hill_climbing()

1. Inisialisasi Variabel Utama

```
current_schedule: Schedule =
generate_initial_schedule(courses, rooms,
time_slots)
```

```
current_objective: float =  
objective(current_schedule, students,  
lecturers)  
objective_history: List[float] =  
[current_objective]
```

Proses dimulai dengan membuat sebuah jadwal awal acak sebagai kondisi saat ini (current_schedule) dan menghitung nilai objective function-nya. Riwayat nilai (objective_history) juga diinisialisasi dengan nilai awal ini untuk melacak progres pencarian.

2. Inisialisasi Variabel Pelacak

```
iterations: int = 0  
stuck_count: int = 0
```

Dilakukan pengaturan awal untuk variabel pelacak. iterations akan menghitung jumlah total langkah yang telah dijalankan, sementara stuck_count digunakan untuk memantau berapa banyak iterasi berturut-turut algoritma tidak menemukan solusi yang lebih baik.

3. Perulangan Utama (Search Loop)

```
for i in range(max_iterations):
```

Proses pencarian iteratif akan terus berjalan selama belum mencapai batas max_iterations. Setiap iterasi merepresentasikan satu langkah pencarian acak untuk mencoba menemukan solusi yang lebih baik.

4. Pembangkit Solusi Tetangga (Stochastic)

```
neighbor: Schedule =  
generate_neighbor(current_schedule, rooms,  
time_slots)  
neighbor_objective: float =  
objective(neighbor, students, lecturers)
```

Pada setiap iterasi, hanya satu solusi tetangga (neighbor) yang dihasilkan secara acak menggunakan fungsi generate_neighbor(). Nilai objective function dari tetangga tunggal ini kemudian dihitung.

5. Keputusan Pergantian Solusi

```
if neighbor_objective > current_objective:  
    current_schedule = neighbor  
    current_objective = neighbor_objective  
    stuck_count = 0
```

Ini adalah langkah keputusan dari algoritma. Jika solusi tetangga yang dipilih secara acak memiliki nilai objective yang lebih tinggi daripada solusi saat ini, maka algoritma akan pindah ke solusi

tetangga tersebut. `stuck_count` juga di-reset ke 0 karena telah terjadi peningkatan.

6. Pemantauan Kondisi Stuck

```
else:  
    stuck_count += 1  
  
if stuck_count >= max_stuck_iterations:  
    print(f"--> Stochastic: Local optimum  
reached after {stuck_count} ...")  
    break
```

Jika solusi tetangga tidak lebih baik, `stuck_count` akan bertambah. Algoritma akan berhenti lebih awal jika `stuck_count` mencapai batas `max_stuck_iterations`, yang menandakan bahwa algoritma kemungkinan besar telah terjebak di sebuah puncak lokal dan tidak membuat kemajuan.

7. Pengembalian Solusi Akhir

```
return current_schedule, objective_history,  
iterations, duration
```

Setelah proses pencarian berhenti (baik karena mencapai `max_iterations` atau `max_stuck_iterations`), fungsi akan mengembalikan jadwal terakhir yang ditemukan, riwayat nilai objective, total iterasi yang dijalankan, dan total durasi eksekusi.

3. Sideways Move

Hill-climbing with Sideways Move adalah varian yang dirancang untuk mengatasi masalah "dataran" (*flat*) di mana beberapa *state* memiliki nilai yang sama. Berbeda dengan versi standar, algoritma ini diizinkan untuk bergerak ke *neighbor* yang nilainya sama dengan *state* saat ini, dan hanya akan berhenti jika telah mencapai puncak di mana semua *neighbor* di sekitarnya memiliki nilai yang lebih rendah. Meskipun varian ini terbukti dapat meningkatkan tingkat keberhasilan secara drastis, seperti pada masalah 8-queens yang suksesnya meningkat dari 14% menjadi 94%, namun cenderung bekerja lebih lambat karena membutuhkan lebih banyak langkah untuk mencapai solusi.

Pseudocode

```
function HILL-CLIMBING(problem) returns a state  
that is a local maximum  
    current  $\leftarrow$  MAKE-NODE(problem.INITIAL-STATE)  
    loop do  
        neighbor  $\leftarrow$  a highest-valued successor of  
        current
```

```
if neighbor.VALUE < current.VALUE then  
    return current.STATE  
    current ← neighbor
```

Kelebihan dan kekurangan dari algoritma ini dapat ditinjau dari tiga aspek. Pada eksplorasi ruang solusi, kelebihan utama algoritma ini adalah kemampuannya untuk terus bergerak meskipun tidak menemukan peningkatan, selama ia berada di area yang datar. Ini memberinya kesempatan untuk menyeberangi *plateau* dan menemukan "lereng naik" lain yang mungkin terlewati oleh Hill-Climbing biasa. Namun, ia tetap tidak bisa keluar dari *local optimum* sejati (karena tidak bisa menerima solusi yang lebih buruk) dan bisa menghabiskan banyak waktu jika *plateau*-nya sangat luas. Pada aspek fleksibilitas, algoritma ini mudah diimplementasikan karena hanya menambahkan satu kondisi pada Hill-Climbing standar, dengan parameter tambahan yang relatif sederhana (`max_sideways_moves`). Kemudian, pada aspek kualitas solusi, varian ini cenderung menghasilkan solusi yang lebih baik daripada Steepest Ascent standar karena tidak mudah menyerah, tetapi tetap tidak memberikan jaminan untuk menemukan solusi optimal global.

Implementasi algoritma Hill-Climbing with Sideways Moves pada sistem penjadwalan ini dilakukan melalui dua fungsi yang terdapat pada file `hill_climbing.py`: `hill_climbing_with_sideways_moves_sampling()` dan `hill_climbing_with_sideways_moves_full()`. Keduanya bertujuan untuk menemukan solusi jadwal yang optimal dengan kemampuan tambahan untuk menjelajahi area datar dalam ruang solusi.

- `_sampling()` hanya memeriksa sejumlah sampel tetangga untuk efisiensi waktu.
- `_full()` memeriksa semua kemungkinan tetangga untuk memastikan setiap langkah adalah yang terbaik.

Algoritma Hill-Climbing with Sideways Moves berupaya memaksimalkan nilai *objective function* dengan mengizinkan gerakan lateral. Prosesnya dimulai dari sebuah jadwal acak. Pada setiap iterasi, algoritma akan mencari tetangga dengan nilai terbaik. Jika nilai tetangga tersebut lebih baik dari solusi saat ini, algoritma akan pindah ke sana. Namun, jika nilai tetangga terbaik tersebut sama dengan solusi saat ini, algoritma tetap diizinkan untuk pindah, dan sebuah penghitung gerakan menyamping akan bertambah. Proses ini akan berhenti jika tidak ada lagi tetangga yang

nilainya lebih baik atau sama, atau jika jumlah gerakan menyamping berturut-turut telah mencapai batas maksimum (max_sideways_moves).

Fungsi hill_climbing_with_sideways_moves() menerima beberapa parameter penting, yaitu sebagai berikut.

Parameter	Keterangan
courses, rooms, time_slots, students, lecturers	Kumpulan data utama yang mendefinisikan masalah penjadwalan dan digunakan dalam evaluasi <i>objective function</i> .
max_iterations	Jumlah iterasi maksimum yang akan dijalankan. Ini berfungsi sebagai batas waktu agar algoritma pasti berhenti.
max_sideways_moves	Jumlah maksimum gerakan menyamping (ke solusi dengan nilai yang sama) yang diizinkan secara berturut-turut sebelum pencarian dihentikan.

Implementasi Sideways Move dalam hill_climbing.py.

```
def hill_climbing_with_sideways_moves_sampling(courses: List[Course], rooms: List[Room], time_slots: List[TimeSlot], students: List[Student], lecturers: List[Lecturer], max_iterations: int, max_sideways_moves: int) -> Tuple[Schedule, List[Float], int, float]:
    start_time = time.time()
    current_schedule: Schedule = generate_initial_schedule(courses, rooms, time_slots)
    current_objective: float = objective(current_schedule, students, lecturers)
    objective_history: List[float] = [current_objective]

    sideways_moves_count: int = 0
    iterations: int = 0
    for i in range(max_iterations):
        iterations += 1 + 1
        best_neighbor: Schedule | None = None
        best_neighbor_objective: float = float('inf')

        for _ in range(50):
            neighbor: Schedule = generate_neighbor(current_schedule, rooms, time_slots)
            neighbor_objective: float = objective(neighbor, students, lecturers)
            if neighbor_objective < best_neighbor_objective:
                best_neighbor = neighbor
                best_neighbor_objective = neighbor_objective

            if best_neighbor_objective == current_objective:
                assert best_neighbor is not None # make sure best_neighbor is not None before use
                current_objective = best_neighbor_objective
                objective_history.append(current_objective)
                sideways_moves_count += 1
                if sideways_moves_count == max_sideways_moves:
                    assert best_neighbor is not None # make sure best_neighbor is not None before use
                    current_objective = best_neighbor_objective
                    objective_history.append(current_objective)
                    sideways_moves_count += 1
                    break
            else:
                print(f"> sideways.Move: Optimum reached or sideways limit exceeded at iteration {iterations}.")
                break

    duration: float = time.time() - start_time
    return current_schedule, objective_history, iterations, duration
```

```

def hill_climbing_with_sideways_moves_full(courses: List[Course], rooms: List[Room], time_slots: List[TimeSlot], students: List[Student], lecturers: List[Lecturer], max_iterations: int, max_sideways_moves: int) -> Tuple[Schedule, List[float], int, float]:
    start_time = float = time.time()
    current_schedule: Schedule = generate_initial_schedule(courses, rooms, time_slots)
    current_objective: float = objective(current_schedule, students, lecturers)
    objective_history: List[float] = [current_objective]

    sideways_moves_count: int = 0
    iterations: int = 0
    for i in range(max_iterations):
        sideways_moves_count += 1
        best_neighbor: Schedule | None = None
        best_neighbor_objective: float = float("inf")

        num_assignments: int = len(current_schedule.assignments)

        for ass_1 in range(num_assignments):
            for ass_2 in range(ass_1 + 1, num_assignments):
                neighbor = copy.deepcopy(current_schedule)
                neighbor.assignments[ass_1].room = neighbor.assignments[ass_1].room + neighbor.assignments[ass_2].room
                neighbor.assignments[ass_1].time_slot = neighbor.assignments[ass_2].time_slot
                neighbor.assignments[ass_1].time_slot.hour_index()

                neighbor_objective: float = objective(neighbor, students, lecturers)
                if neighbor_objective < best_neighbor_objective:
                    best_neighbor = neighbor
                    best_neighbor_objective = neighbor_objective

        for room_id in range(len(rooms)):
            original_room_id = current_schedule.assignments[room_id].room.room_id
            original_time_slot_idx = current_schedule.assignments[room_id].time_slot_idx
            for room in rooms:
                for time_slot in time_slots:
                    if room.room_id == original_room_id and time_slot.hour_index() == original_time_slot_idx:
                        continue

                    neighbor = copy.deepcopy(current_schedule)
                    neighbor.assignments[original_time_slot_idx].room = room
                    neighbor.assignments[original_time_slot_idx].time_slot = time_slot

                    neighbor_objective: float = objective(neighbor, students, lecturers)
                    if neighbor_objective < best_neighbor_objective:
                        best_neighbor = neighbor
                        best_neighbor_objective = neighbor_objective

        if best_neighbor_objective < current_objective:
            assert best_neighbor is not None
            current_schedule = best_neighbor
            current_objective = best_neighbor_objective
            objective_history.append(best_neighbor_objective)
            sideways_moves_count += 1
        elif best_neighbor_objective == current_objective and sideways_moves_count > max_sideways_moves:
            print(f">> Sideways Move (Full): Optimum reached or sideways limit exceeded at iteration {iterations}.")
            break
        else:
            sideways_moves_count += 1
    duration: float = time.time() - start_time
    return current_schedule, objective_history, iterations, duration

```

Gambar 9 *Screenshot* Implementasi Sideways Move Hill-Climbing
Penjelasan implementasi kode:

hill_climbing_with_sideways_moves_sampling()

1. Inisialisasi Variabel Utama

```

current_schedule: Schedule =
generate_initial_schedule(courses, rooms,
time_slots)
current_objective: float =
objective(current_schedule, students,
lecturers)
objective_history: List[float] =
[current_objective]

```

Proses dimulai dengan membuat sebuah jadwal awal acak sebagai kondisi saat ini (current_schedule) dan menghitung nilai objective function-nya. Riwayat nilai (objective_history) juga diinisialisasi dengan nilai awal ini untuk melacak progres.

2. Inisialisasi Variabel Pelacak

```

sideways_moves_count: int = 0
iterations: int = 0

```

Dilakukan pengaturan awal untuk variabel pelacak. iterations menghitung jumlah total iterasi, sementara sideways_moves_count adalah variabel kunci yang menghitung berapa banyak langkah "menyamping" (ke solusi dengan nilai yang sama) yang telah dilakukan secara berturut-turut.

3. Perulangan Utama (Search Loop)

```

for i in range(max_iterations):

```

Proses pencarian iteratif akan terus berjalan selama belum mencapai batas max_iterations. Setiap iterasi merepresentasikan satu langkah pencarian untuk menemukan solusi yang lebih baik atau setara.

4. Pembangkit dan Pemilihan Tetangga Terbaik (Sampling)

```
best_neighbor: Schedule | None = None
best_neighbor_objective: float = -float('inf')

for _ in range(50):
    neighbor: Schedule =
    generate_neighbor(current_schedule, rooms,
                      time_slots)
    neighbor_objective: float =
    objective(neighbor, students, lecturers)
    if neighbor_objective >
    best_neighbor_objective:
        best_neighbor = neighbor
        best_neighbor_objective =
    neighbor_objective
```

Pada setiap iterasi, algoritma mencari tetangga terbaik dengan cara mengambil sampel. Fungsi generate_neighbor() dipanggil sebanyak 50 kali untuk menghasilkan 50 jadwal tetangga acak. Setiap tetangga dievaluasi, dan yang memiliki nilai objective tertinggi akan disimpan sebagai best_neighbor.

5. Keputusan Pergantian Solusi (Logika Inti Sideways Move)

```
if best_neighbor_objective >
current_objective:
    current_schedule = best_neighbor
    sideways_moves_count = 0
elif best_neighbor_objective ==
current_objective and sideways_moves_count <
max_sideways_moves:
    current_objective =
best_neighbor_objective
    sideways_moves_count += 1
```

Ini adalah logika inti dari algoritma. Jika best_neighbor memberikan peningkatan nilai, algoritma akan pindah ke sana dan sideways_moves_count di-reset ke 0. Jika best_neighbor memiliki nilai yang sama, algoritma akan melakukan "gerakan menyamping" dengan pindah ke sana, dan sideways_moves_count akan bertambah, selama belum mencapai batas max_sideways_moves.

6. Kondisi Berhenti

```

else:
    print(f"--> Sideways-Move: Optimum reached
or sideways limit exceeded...")
    break

```

Pencarian akan berhenti jika tidak ada lagi tetangga yang nilainya lebih baik atau sama, atau jika batas gerakan menyamping (`max_sideways_moves`) telah tercapai.

7. Pengembalian Solusi Akhir

```

return current_schedule, objective_history,
iterations, duration

```

Fungsi mengembalikan jadwal terakhir yang ditemukan, riwayat nilai *objective*, total iterasi, dan durasi eksekusi.

Penjelasan implementasi kode:

`hill_climbing_with_sideways_moves_full()`

7. Inisialisasi Variabel Utama

```

current_schedule: Schedule =
generate_initial_schedule(courses, rooms,
time_slots)
current_objective: float =
objective(current_schedule, students,
lecturers)
objective_history: List[float] =
[current_objective]

```

Proses dimulai dengan membuat sebuah jadwal awal acak sebagai kondisi saat ini (`current_schedule`) dan menghitung nilai objective function-nya. Riwayat nilai (`objective_history`) juga diinisialisasi dengan nilai awal ini untuk melacak progres.

8. Inisialisasi Variabel Pelacak

```

sideways_moves_count: int = 0
iterations: int = 0

```

Dilakukan pengaturan awal untuk variabel pelacak. `iterations` menghitung jumlah total iterasi, sementara `sideways_moves_count` adalah variabel kunci yang menghitung berapa banyak langkah "menyamping" (ke solusi dengan nilai yang sama) yang telah dilakukan secara berturut-turut.

9. Perulangan Utama (Search Loop)

```

for i in range(max_iterations):

```

Perulangan utama juga berjalan hingga `max_iterations`, setiap iterasi adalah satu langkah pencarian penuh.

10. Pembangkit dan Pemilihan Tetangga Terbaik (Pencarian Penuh)

```

# Swap moves for ass_i in
range(num_assignments): for ass_j in
range(ass_i + 1, num_assignments): # ...
generate and evaluate neighbor ... # Move
moves for ass_i in range(num_assignments): for
room in rooms: for time_slot in time_slots: #
... generate and evaluate neighbor ...

```

Ini adalah perbedaan utama. Alih-alih mengambil sampel, fungsi ini secara sistematis menghasilkan semua kemungkinan tetangga. Ini dilakukan melalui dua proses besar yaitu swap moves dan move moves. Swap Moves adalah dua loop for bersarang digunakan untuk mencoba setiap kemungkinan pertukaran (swap) antara dua pertemuan (assignment) di dalam jadwal. Move Moves adalah tiga loop for bersarang digunakan untuk mencoba setiap kemungkinan pemindahan (move) dari setiap pertemuan ke setiap kombinasi slot waktu dan ruangan lainnya. Dari ribuan atau jutaan tetangga yang dihasilkan, algoritma akan memilih satu yang memiliki nilai objective absolut terbaik sebagai best_neighbor.

11. Keputusan Pergantian Solusi dan Kondisi Berhenti

```

if best_neighbor_objective >
current_objective:
    assert best_neighbor is not None
    current_schedule = best_neighbor
    current_objective =
best_neighbor_objective

objective_history.append(current_objective)
sideways_moves_count = 0
elif best_neighbor_objective ==
current_objective and sideways_moves_count <
max_sideways_moves:
    assert best_neighbor is not None
    current_schedule = best_neighbor
    current_objective =
best_neighbor_objective

objective_history.append(current_objective)
sideways_moves_count += 1
else:
    print(f"--> Sideways-Move (Full):")
    Optimum reached or sideways limit exceeded at
iteration {iterations}.")
    break

```

Logika untuk memutuskan apakah akan bergerak naik (>), menyamping (==), atau berhenti (else) sama persis dengan versi sampling, berdasarkan best_neighbor yang telah ditemukan.

12. Pengembalian Solusi Akhir

```
return current_schedule, objective_history,  
iterations, duration
```

Fungsi mengembalikan `current_schedule`, `objective_history`, `iterations`, dan `duration`, sama seperti versi sampling.

4. Random Restart

Random Restart Hill-Climbing bukanlah algoritma pencarian baru, melainkan sebuah strategi metaheuristik yang dirancang untuk mengatasi kelemahan fundamental dari Hill-Climbing standar: ketidakmampuannya untuk keluar dari *local optimum*. Sesuai dengan namanya, strategi ini menjalankan algoritma Hill-Climbing secara berulang kali, di mana setiap eksekusi dimulai dari titik awal acak yang berbeda.

Penjelasan dan Formula

If at first you don't succeed, try, try again.

It conducts a series of hill climbing searches
(random initial states, until a goal is found)

Expected nb of restarts = $1/p \rightarrow p=0.14: 7$ restart

Expected nb of steps = $s+f (1-p) /p \rightarrow p=0.14, s=4, f=3: 22$ steps.

Kelebihan dan kekurangan dari strategi ini dapat ditinjau dari tiga aspek. Pada eksplorasi ruang solusi, kelebihan utamanya adalah kemampuannya untuk menjelajahi berbagai bagian dari lanskap solusi. Dengan memulai dari banyak titik acak, ia memiliki probabilitas yang jauh lebih tinggi untuk menemukan *global optimum*. Akan tetapi, kekurangannya adalah biaya komputasi yang lebih tinggi karena pada dasarnya ia menjalankan algoritma pencarian penuh beberapa kali. Pada aspek fleksibilitas, strategi ini sangat mudah diimplementasikan di atas algoritma Hill-Climbing yang sudah ada, dengan parameter yang intuitif (`num_restarts` dan `max_iter_per_restart`). Kemudian, pada aspek kualitas solusi, Random Restart secara signifikan meningkatkan kemungkinan untuk menemukan solusi berkualitas tinggi yang mendekati atau bahkan sama dengan *global optimum*, meskipun tetap tidak memberikan jaminan absolut karena keberhasilannya bergantung pada titik-titik awal yang dipilih secara acak.

Implementasi strategi Random Restart Hill-Climbing pada sistem penjadwalan ini dilakukan melalui fungsi `random_restart_hill_climbing()`

yang terdapat pada file hill_climbing.py. Fungsi ini bertindak sebagai "manajer" yang memanggil fungsi Hill-Climbing inti (steepest_ascent_hill_climbing_sampling) beberapa kali untuk menemukan solusi jadwal terbaik dari semua percobaan tersebut.

Algoritma Random Restart Hill-Climbing berupaya menemukan solusi optimal global dengan melakukan serangkaian pencarian independen. Prosesnya adalah sebuah *loop* yang berjalan sebanyak num_restarts. Di dalam setiap iterasi *loop*, sebuah pencarian Steepest Ascent Hill-Climbing penuh dijalankan dari awal hingga akhir, yang secara otomatis dimulai dari jadwal acak baru. Setelah setiap pencarian selesai, solusi yang ditemukannya akan dibandingkan dengan solusi terbaik yang pernah ditemukan sejauh ini (*global best*). Jika solusi baru lebih baik, maka ia akan disimpan sebagai *global best* yang baru. Setelah semua proses *restart* selesai, solusi *global best* terakhir inilah yang akan dikembalikan sebagai hasil akhir.

Parameter	Keterangan
courses, rooms, time_slots, students, lecturers	Kumpulan data utama yang mendefinisikan masalah penjadwalan dan digunakan dalam evaluasi <i>objective function</i> .
num_restarts	Jumlah total berapa kali algoritma Hill-Climbing akan dijalankan dari awal dengan <i>state</i> acak yang baru.
max_iter_per_restart	Batas iterasi maksimum untuk <i>setiap</i> pencarian Hill-Climbing individual yang dijalankan. Ini mencegah satu pencarian berjalan terlalu lama.

Implementasi Random Restart dalam hill_climbing.py.

```
def random_restart_hill_climbing(courses: List[Course], rooms: List[Room], time_slots: List[TimeSlot], students: List[Student], lecturers: List[Lecturer], num_restarts: int, max_iter_per_restart: int) -> Tuple[Schedule, List[float], int, float, int]:
    start_time: float = time.time()
    global_best_schedule: Schedule | None = None
    global_best_objective: float = float("inf")
    total_iterations: int = 0
    objective_history_per_restart: List[float] = []
    objective_history_global: List[float] = []

    print("Starting Random-Restart Hill-Climbing with {} restarts.".format(num_restarts))
    for i in range(num_restarts):
        print(" -> Restart #{} + 1)/({} restarts)...".format(i+1, num_restarts))

        schedule: Schedule | List[None] = None
        iterations: int = 0
        scheme: Iterations = steepest_ascent_hill_climbing(
            courses, rooms, time_slots, students, lecturers,
            max_iter_per_restart=max_iter_per_restart,
            objective_history_global=objective_history_global
        )

        current_objective: float = objective(schedule, student, lecturer)
        total_iterations += iterations

        if current_objective > global_best_objective:
            global_best_objective = current_objective
            global_best_schedule = schedule
            print(" -> New global best found with objective: ({}, {})".format(global_best_objective, iterations))
            objective_history_per_restart.append(global_best_objective)

    duration: float = time.time() - start_time
    return global_best_schedule, objective_history_per_restart, total_iterations, duration, num_restarts, objective_history_global
```

Gambar 10 Screenshot Implementasi Random Restart Hill-Climbing

Penjelasan implementasi kode: random_restart_hill_climbing()

1. Inisialisasi Variabel Global

```
global_best_schedule: Schedule | None = None
global_best_objective: float = -float('-inf')
total_iterations: int = 0
objective_history_per_restart: List[float] = []
```

Proses dimulai dengan menginisialisasi variabel untuk menyimpan solusi terbaik yang ditemukan di seluruh proses restart. `global_best_schedule` akan menyimpan jadwal terbaik, `global_best_objective` menyimpan nilai objective function terbaik, `total_iterations` mengakumulasi jumlah iterasi dari setiap run, dan `objective_history_per_restart` digunakan untuk mencatat progres penemuan solusi terbaik setelah setiap restart.

2. Perulangan Utama (Restart Loop)

```
for i in range(num_restarts):
```

Perulangan utama ini mengontrol strategi Random Restart. Proses pencarian Hill-Climbing akan dijalankan dari awal sebanyak `num_restarts` kali. Setiap iterasi dalam loop ini merepresentasikan satu pencarian independen dari titik awal acak yang baru.

3. Panggilan Algoritma Hill-Climbing Inti

```
schedule, _, iterations, _ =
steepest_ascent_hill_climbing_sampling(
    courses, rooms, time_slots, students,
    lecturers,
    max_iterations=max_iter_per_restart,
    neighbors_to_check=50
)
```

Di dalam setiap restart, fungsi `steepest_ascent_hill_climbing_sampling()` dipanggil. Fungsi ini menjalankan satu siklus penuh algoritma Hill-Climbing, yang secara otomatis dimulai dari jadwal acak yang baru. Pencarian ini akan berhenti ketika mencapai puncak lokal atau batas `max_iter_per_restart`. Hasilnya adalah jadwal terbaik yang ditemukan dalam satu siklus tersebut.

4. Evaluasi dan Pembaruan Solusi Global

```
current_objective: float = objective(schedule,
students, lecturers)
total_iterations += iterations

if current_objective > global_best_objective:
```

```

global_best_objective = current_objective
global_best_schedule = schedule

objective_history_per_restart.append(global_be
st_objective)

```

Setelah satu siklus Hill-Climbing selesai, nilai objective function dari jadwal yang dihasilkannya (`current_objective`) dievaluasi. Jika nilai ini lebih baik daripada `global_best_objective` yang pernah ditemukan sejauh ini, maka `global_best_schedule` dan `global_best_objective` akan diperbarui dengan solusi baru tersebut. Nilai `global_best_objective` saat ini kemudian dicatat ke dalam `objective_history_per_restart` untuk melacak kemajuan.

5. Pengembalian Solusi Akhir

```

return global_best_schedule,
objective_history_per_restart,
total_iterations, duration, num_restarts

```

Setelah semua siklus restart selesai, fungsi akan mengembalikan jadwal terbaik (`global_best_schedule`) yang ditemukan dari semua percobaan, riwayat nilai objektif terbaik per restart, total akumulasi iterasi, total durasi, dan jumlah restart yang dilakukan.

2.2.2. Simulated Annealing Algorithm

Simulated Annealing Algorithm adalah algoritma pencarian lokal berbasis probabilistik yang terinspirasi dari proses annealing dalam metalurgi, yaitu proses pemanasan dan pendinginan dari logam secara perlahan untuk mendapatkan struktur kristal yang lebih stabil (energi minimum). Dalam konteks optimasi penjadwalan, algoritma ini berusaha untuk mencari solusi terbaik (optimum global) dengan cara mengeksplorasi ruang solusi dan menghindari terjebak pada optimum lokal seperti yang sering terjadi pada algoritma *Hill Climbing*.

Simulated Annealing bekerja dengan mempertimbangkan temperatur atau suhu (T) sebagai parameter pengendali probabilitas untuk menerima solusi yang lebih buruk. Pada tahap awal, temperatur bernilai tinggi, sehingga algoritma cenderung lebih bebas berpindah ke solusi lain, termasuk solusi yang memiliki nilai *objective function* lebih buruk. Seiring berjalannya iterasi, suhu akan diturunkan secara bertahap sesuai dengan cooling schedule, sehingga peluang menerima solusi yang lebih buruk semakin kecil. Dengan mekanisme ini, algoritma dapat keluar dari optimum lokal dan berpotensi untuk menemukan optimum global.

Langkah-langkah algoritma Simulated Annealing terdiri atas inisialisasi, evaluasi solusi awal, pembangkitan neighbor, evaluasi neighbor, dan keputusan pergantian solusi. Pada tahap inisiasi, ditentukan solusi awal secara acak, kemudian ditetapkan juga nilai initial temperature (T_0), cooling rate (α), serta temperatur minimum (T_{min}). Pada tahap evaluasi solusi awal, dihitung nilai *objective function* dari solusi awal sebagai current cost. Kemudian, pada tahap pembangkitan neighbor, dibuat solusi baru dengan melakukan perubahan kecil pada jadwal saat ini, misalnya dengan menukar slot waktu atau ruangan antar mata kuliah. Setelah itu, pada bagian evaluasi neighbor, dihitung nilai objective function dari solusi tetangga (neighbor cost).

Jika solusi baru lebih baik ($\Delta < 0$, dimana $\Delta = \text{neighbor cost} - \text{current cost}$), maka solusi tersebut diterima. Jika solusi baru lebih buruk ($\Delta \geq 0$), maka solusi tetap dapat diterima dengan probabilitas $P=e^{-\Delta/T}$, dengan T adalah suhu saat ini. Probabilitas ini memungkinkan algoritma untuk sesekali menerima solusi yang lebih buruk agar dapat keluar dari perangkap *local optimum*.

Pada proses pendinginan, diturunkan suhu menggunakan rumus $T = \alpha \times T$, dengan $0 < \alpha < 1$, misalnya 0,95. Proses berhenti ketika suhu telah turun di bawah T_{min} atau ketika perubahan solusi sudah tidak lagi memberikan peningkatan yang signifikan.

Pseudocode
<pre> SimulatedAnnealing(initial_schedule, students, time_slots, rooms, initial_temp, cooling_rate, min_temp) : current = initial_schedule best = current T = initial_temp while T > min_temp: neighbor = generate_neighbor(current) Δ = fitness(neighbor) - fitness(current) if Δ < 0: current = neighbor else if random(0,1) < exp(-Δ / T): current = neighbor </pre>

```

        if fitness(current) < fitness(best) :
            best = current

            T = T * cooling_rate

        return best
    
```

Kelebihan dan kekurangan dari algoritma ini dalam ditinjau dari tiga aspek, yaitu eksplorasi ruang solusi, fleksibilitas, dan kualitas solusi. Pada eksplorasi ruang solusi, kelebihan dari algoritma ini adalah dapat keluar dari *local optimum* karena probabilitas penerimaan solusi yang lebih buruk dari *current state*. Akan tetapi, waktu eksekusinya lebih lama dibandingkan *Hill-Climbing* karena banyak iterasi dan perhitungan probabilitas. Pada aspek fleksibilitas, algoritma ini dapat diterapkan pada berbagai jenis permasalahan optimasi. Di sisi lain, pemilihan parameter (*initial_temp*, *cooling_rate*, *min_temp*) sangat mempengaruhi hasil akhir. Kemudian, pada aspek kualitas solusi, algoritma ini cenderung menghasilkan solusi yang lebih mendekati global optimum, tetapi tidak menjamin hasil terbaik secara absolut karena bersifat stokastik.

Implementasi algoritma Simulated Annealing pada sistem penjadwalan ini dilakukan melalui fungsi *simulated_annealing()* yang terdapat pada file *simulated_annealing.py*. Algoritma ini digunakan untuk mencari solusi jadwal perkuliahan yang optimal dengan mempertimbangkan berbagai kendala dan penalti yang sudah didefinisikan dalam *objective function*.

Algoritma Simulated Annealing berupaya meminimalkan total penalti dalam penjadwalan dengan mensimulasikan proses pendinginan logam (*annealing*). Proses pencarian dimulai dari jadwal awal (*initial schedule*) dan secara bertahap menghasilkan jadwal baru (*neighbor schedule*) dengan sedikit perubahan, seperti pertukaran slot waktu atau ruangan. Berbeda dengan Hill-Climbing yang selalu bergerak ke solusi lebih baik, Simulated Annealing masih dapat menerima solusi yang lebih buruk dengan probabilitas tertentu. Mekanisme ini memungkinkan algoritma untuk keluar dari local optimum dan menjelajahi ruang solusi yang lebih luas.

Fungsi *simulated_annealing()* menerima beberapa parameter penting, yaitu sebagai berikut.

Parameter	Keterangan
initial_schedule	Jadwal awal yang digunakan sebagai titik mulai pencarian solusi
student	Data mahasiswa yang digunakan dalam evaluasi <i>objective function</i>
time_slots	Daftar waktu yang tersedia untuk penjadwalan mata kuliah
rooms	Daftar ruangan yang dapat digunakan dalam penjadwalan
initial_temp	Suhu awal (biasanya nilai besar, misalnya 1000) yang menentukan probabilitas penerimaan solusi buruk di awal
cooling_rate	Tingkat penurunan suhu pada setiap iterasi, biasanya di antara 0.90 - 0.99
min_temp	Suhu minimum di mana proses pencarian akan berhenti

Implementasi kode:

```

def simulated_annealing(initial_schedule: Schedule, students: List[Student], lecturers: List[Lecturer], time_slots: List[TimeSlot],
                        rooms: List[Room], initial_temp: float = 1000, cooling_rate: float = 0.95,
                        min_temp: float = 1) -> Tuple[Schedule, float, List[float], List[int], int]:
    current: Schedule = deepcopy(initial_schedule)
    current_objective: float = objective(current, students, lecturers)
    best: Schedule = deepcopy(current)
    best_objective: float = current_objective

    temp: float = initial_temp
    iteration: int = 0

    acceptance_probabilities: List[float] = []
    iterations_list: List[int] = []
    stuck_count: int = 0
    last_improvement_iter: int = 0

    while temp > min_temp:
        neighbor: Schedule = generate_neighbor(current, rooms, time_slots)
        neighbor_objective: float = objective(neighbor, students, lecturers)
        delta: float = neighbor_objective - current_objective

        accept_prob: float
        if delta > 0:
            accept_prob = 1.0
            acceptance_probabilities.append(accept_prob)
        else:
            accept_prob = math.exp(delta / temp)
            acceptance_probabilities.append(accept_prob)

        iterations_list.append(iteration)

        if delta > 0 or random.random() < math.exp(delta / temp):
            current = neighbor
            current_objective = neighbor_objective
            if current_objective > best_objective:
                best = deepcopy(current)
                best_objective = current_objective
                last_improvement_iter = iteration

        if iteration - last_improvement_iter >= 100:
            stuck_count += 1
            last_improvement_iter = iteration

        temp *= cooling_rate
        iteration += 1

    return best, best_objective, acceptance_probabilities, iterations_list, stuck_count

```

Gambar 11 Screenshot Implementasi Simulated Annealing

Penjelasan implementasi kode:

1. Inisialisasi Variabel Utama

```

current = deepcopy(initial_schedule)
current_fitness = objective(current, students)
best = deepcopy(current)
best_fitness = current_fitness

```

Proses dimulai dengan menyalin jadwal awal (initial_schedule) sebagai kondisi saat ini (current) dan menghitung nilai fitness menggunakan fungsi objective(). Nilai terbaik (best) juga diinisialisasi dengan jadwal dan fitness yang sama.

2. Inisialisasi Parameter Suhu dan Iterasi

```
temp = initial_temp
iteration = 0
acceptance_probabilities = []
iterations_list = []
stuck_count = 0
last_improvement_iter = 0
```

Dilakukan pengaturan awal suhu dan variabel pelacak seperti jumlah iterasi, daftar probabilitas penerimaan, serta penghitung kondisi “stuck” (tidak ada perbaikan dalam 100 iterasi terakhir).

3. Perulangan Utama (Cooling Loop)

```
while temp > min_temp:
```

Proses iteratif akan terus berjalan selama suhu belum mencapai batas minimum (min_temp). Setiap iterasi merepresentasikan satu langkah pendinginan.

4. Pembangkit Solusi Tetangga

```
neighbor = generate_neighbor(current, rooms, time_slots)
neighbor_fitness = objective(neighbor, students)
delta = neighbor_fitness - current_fitness
```

Fungsi generate_neighbor() menghasilkan jadwal baru dengan sedikit perubahan dari jadwal saat ini. Setelah itu, dihitung juga perbedaan fitness antara jadwal tetangga dengan jadwal sebelumnya.

5. Perhitungan Probabilitas Penerimaan Solusi

```
if delta > 0:
    accept_prob = 1.0
else:
    accept_prob = math.exp(delta / temp)
```

Jika solusi baru lebih baik, maka langsung diterima, tetapi jika lebih buruk, maka solusi baru masih bisa diterima dengan probabilitas $P=e^{T/\Delta}$, dengan Δ adalah selisih fitness dan T adalah suhu saat ini.

6. Keputusan Pergantian Solusi

```
if delta > 0 or random.random() < math.exp(delta / temp):
    current = neighbor
    current_fitness = neighbor_fitness
```

Jika kondisi penerimaan terpenuhi, solusi akan berpindah ke neighbor, tetapi jika fitness baru lebih baik dari yang terbaik sejauh ini, maka solusi disimpan sebagai *best solution*.

7. Pemantauan Iterasi dan Kondisi Stuck

```
if iteration - last_improvement_iter >= 100:
    stuck_count += 1
    last_improvement_iter = iteration
```

Jika dalam 100 iterasi terakhir tidak ada perbaikan solusi terbaik, maka dianggap stuck, dan penghitung stuck_count bertambah satu. Hal ini digunakan untuk analisa performa algoritma.

8. Pendinginan Suhu (Cooling Schedule)

```
temp *= cooling_rate
iteration += 1
```

Setelah setiap iterasi, suhu diturunkan dengan mengalikan faktor cooling rate (misalnya 0.95). Proses ini membuat algoritma semakin “selektif” terhadap penerimaan solusi yang lebih buruk.

9. Pengembalian Solusi Akhir

```
return best, best_fitness, acceptance_probabilities,
iterations_list, stuck_count
```

Setelah suhu mencapai batas minimum, fungsi mengembalikan solusi terbaik yang ditemukan beserta nilai fitness-nya, daftar probabilitas penerimaan selama proses berjalan, dan jumlah kondisi *stuck*.

2.2.3. *Genetic Algorithm*

Genetic Algorithm (GA) adalah algoritma optimasi berbasis evolusi biologis yang meniru proses seleksi alam (*natural selection*) dalam teori Darwin. GA bekerja dengan membangkitkan populasi solusi (disebut individu atau kromosom), kemudian secara iteratif memperbaiki populasi melalui operasi seleksi, crossover, dan mutasi agar mendekati solusi optimal. Terdapat beberapa konsep dasar yang digunakan, yaitu sebagai berikut.

1. **Populasi**, sekumpulan solusi yang merepresentasikan jadwal (*Schedule*)
2. **Gen**, bagian terkecil dari kromosom yang menyimpan informasi (di sini, gen direpresentasikan sebagai *Assignment*, kombinasi antara mata kuliah, ruangan, dan waktu)
3. **Fitness Function** (*Objective Function*), fungsi penilai kualitas solusi (semakin tinggi nilai fitness, semakin baik jadwalnya).
4. **Seleksi**, memilih individu-individu terbaik berdasarkan fitness untuk menjadi “orang tua”.
5. **Crossover** (Rekombinasi), menggabungkan gen dari dua orang tua untuk membentuk keturunan baru.
6. **Mutasi**, memperkenalkan variasi acak untuk mencegah konvergensi prematur.

Dalam konteks penjadwalan perkuliahan otomatis, GA digunakan untuk mencari jadwal terbaik dengan meminimalkan penalti, seperti konflik waktu mahasiswa atau dosen, tabrakan penggunaan ruangan, dan kelebihan kapasitas ruangan. Output akhir berupa jadwal optimal yang memiliki nilai fitness tertinggi (penalti terendah).

1. **selection(population_objective)**

Seleksi digunakan untuk memilih individu terbaik yang akan dijadikan induk (parent) bagi generasi berikutnya. Metode yang digunakan adalah Roulette Wheel Selection, yaitu setiap individu diberi peluang dipilih sebanding dengan nilai fitness-nya dan individu dengan fitness tinggi memiliki peluang lebih besar untuk dipilih.

Pertama-tama, fungsi akan menyimpan nilai fitness semua indivisu, kemudian ditambahkan konstanta `max_absolute_objective + 1` untuk memastikan semua nilai positif (agar peluang proporsional). Selanjutnya, dihitung total fitness populasi dan dihasilkan bilangan acak pick untuk

menentukan titik pada “roda roulette”. Lalu, diakumulasikan fitness hingga mencapai pick * total_objective, dan pada akhirnya individu tersebut dipilih. Semakin tinggi fitness, semakin besar peluang untuk terpilih menjadi parent.

```
def selection(population_objective: List[Tuple[Schedule, float]]) -> Schedule:
    # using roulette wheel selection

    max_absolute_objective: float = 0.0
    objective_scores: List[float] = []
    for _, objective_value in population_objective:
        max_absolute_objective = max(max_absolute_objective, abs(objective_value))
        objective_scores.append(objective_value)

    total_objective: float = 0.0
    for i in range(len(objective_scores)):
        objective_scores[i] += max_absolute_objective + 1
        total_objective += objective_scores[i]

    pick: float = random.random()

    sum: float = 0.0
    for i in range(len(objective_scores)):
        sum += objective_scores[i]
        if sum >= pick * total_objective:
            return population_objective[i][0]

    assert False
```

Gambar 11 Screenshot Implementasi *Genetic Algorithm* selection()

2. crossover(parent1, parent2)

Crossover adalah proses pertukaran gen antar dua individu (*parent*) untuk membentuk dua individu baru. Hal ini meniru proses reproduksi biologis untuk menghasilkan variasi baru.

Pertama-tama, dilakukan duplikasi penuh (*deepcopy*) dari parent, kemudian ditentukan titik crossover acak. Setelah itu, ditukar sebagian gen (ruangan dan waktu) antara dua parent sampai titik tersebut. Kemudian, dihasilkan dua anak baru. Tujuan dari fungsi ini adalah menggabungkan karakteristik dua jadwal yang relatif baik agar menghasilkan solusi yang lebih optimal.

```
def crossover(parent1: Schedule, parent2: Schedule) -> Tuple[Schedule, Schedule]:
    child1: Schedule = copy.deepcopy(parent1)
    child2: Schedule = copy.deepcopy(parent2)

    crossover_point: int = random.randint(0, len(parent1.assignments) - 1)

    for i in range(crossover_point):
        child1.assignments[i].room, child2.assignments[i].room = child2.assignments[i].room, child1.assignments[i].room
        child1.assignments[i].time_slot, child2.assignments[i].time_slot = child2.assignments[i].time_slot, child1.assignments[i].time_slot

    return child1, child2
```

Gambar 12 Screenshot Implementasi *Genetic Algorithm* crossover()

3. mutation (schedule, rooms, time_slots)

Mutasi bertujuan memperkenalkan keragaman genetik agar algoritma tidak terjebak pada solusi lokal (*local optimum*).

generate_neighbor() memilih satu *Assignment* acak dan mengubah ruangan atau waktu kuliah secara acak atau menukar dua assignment dalam jadwal. Tujuannya adalah untuk menambah variasi kecil yang bisa memperbaiki jadwal tanpa mengubah struktur besar.

```
def mutation(schedule: Schedule, rooms: List[Room], time_slots: List[TimeSlot]) -> Schedule:  
    return generate_neighbor(schedule, rooms, time_slots)
```

Gambar 13 Screenshot Implementasi *Genetic Algorithm* mutation()

4. genetic_algorithm()

Fungsi ini adalah fungsi utama yang menjalankan seluruh proses evolusi populasi dalam beberapa generasi. Langkah-langkah implementasi dimulai dengan inisialisasi populasi, membuat populasi awal berisi sejumlah jadwal acak. Kemudian, dilanjutkan dengan evaluasi awal yang menghitung nilai fitness (objective) untuk setiap jadwal. Setelah itu, dilanjutkan dengan evolusi per generasi, dalam setiap iterasi (generasi): dihitung nilai maksimum dan rata-rata fitness untuk statistik, dilakukan seleksi dua parent, melakukan crossover untuk menghasilkan dua anak, melakukan mutasi terhadap anak-anak tersebut, serta menyusun populasi baru dari anak-anak yang dihasilkan.

Lalu, evaluasi populasi baru dan diukur fitness generasi baru. Setelahnya, update solusi terbaik dan jika ada individu dengan fitness lebih tinggi daripada solusi terbaik sejauh ini, maka di update. Kemudian, dilakukan penyimpanan statistik dan disimpan perkembangan nilai maksimum dan rata-rata fitness di setiap generasi, lalu dikembalikan jadwal terbaik dan statistik performa algoritma.

```

def genetic_algorithm(courses: List[Course], rooms: List[Room], time_slots: List[TimeSlot], students: List[Student], lecturers: List[Lecturer], population_size: int, generations: int) -> Tuple[Schedule, Dict[str, List[float]]]:
    population: List[Schedule] = initialize_population(courses, rooms, time_slots, population_size)
    population_objective: List[Tuple[Schedule, float]] = evaluate_population(population, students, lecturers)

    max_objective_history: List[float] = []
    avg_objective_history: List[float] = []

    best_schedule: Tuple[Schedule, float] = population_objective[0]

    for _ in range(generations):
        # Track statistics
        objective_values: List[float] = [obj for _, obj in population_objective]
        max_objective_history.append(max(objective_values))
        avg_objective_history.append(sum(objective_values) / len(objective_values))

        new_population: List[Schedule] = []
        for _ in range(population_size // 2):
            parent1: Schedule = selection(population_objective)
            parent2: Schedule = selection(population_objective)

            child1: Schedule
            child2: Schedule
            child1, child2 = crossover(parent1, parent2)

            child1 = mutation(child1, rooms, time_slots)
            child2 = mutation(child2, rooms, time_slots)

            new_population.append(child1)
            if len(new_population) < population_size:
                new_population.append(child2)

        population = new_population
        population_objective = evaluate_population(population, students, lecturers)

    for i in range(len(population_objective)):
        if best_schedule[1] < population_objective[i][1]:
            best_schedule = population_objective[i]

    statistics: Dict[str, List[float]] = {
        'max_objective': max_objective_history,
        'avg_objective': avg_objective_history
    }

    return best_schedule[0], statistics

```

Gambar 14 Screenshot Implementasi *Genetic Algorithm*
genetic_algorithm()

Secara sederhana, alur kerja keseluruhan Genetic Algorithm adalah bentik populasi awal secara acak, hitung fitness setiap individu, pilih parent menggunakan Roulette Wheel Selection, lakukan crossover antara parent dan menghasilkan offspring baru, lakukan mutasi untuk menjaga variasi, evaluasi offspring baru dan bentuk populasi generasi berikutnya, ulangi proses evolusi selama beberapa generasi, dan pada akhirnya ambil solusi terbaik dengan fitness tertinggi (penalti terendah).

2.3. Implementasi Lainnya

Berikut merupakan implementasi kode lainnya.

2.3.1. models.py

File ini berisi definisi semua kelas (struktur data) yang merepresentasikan entitas-entitas utama dalam masalah penjadwalan.

Class Course

- Atribut

Atribut	Deskripsi
course_id	Identifier unik untuk mata kuliah (contoh: "IF3170_K01")
num_students	Jumlah mahasiswa yang terdaftar pada mata kuliah ini.

credits	Jumlah SKS (Satuan Kredit Semester) dari mata kuliah
---------	------------------------------------------------------

- **Konstruktor**

Konstruksi	Deskripsi
<code>__init__(course_id, num_students, credits)</code>	Menginisialisasi objek Course dengan ID, jumlah mahasiswa, dan SKS

Class TimeSlot

- **Atribut**

Atribut	Deskripsi
day	Nama hari dalam seminggu (contoh: "Senin")
hour	Jam mulai dari slot waktu (contoh: 8 untuk 08:00-09:00)

- **Konstruktor**

Konstruksi	Deskripsi
<code>__init__(day, hour)</code>	Menginisialisasi objek TimeSlot dengan hari dan jam.

- **Metode**

Metode	Deskripsi
<code>hour_index()</code>	Mengonversi kombinasi hari dan jam menjadi sebuah indeks integer unik untuk mempermudah pengecekan konflik.

Class Room

- **Atribut**

Atribut	Deskripsi
<code>room_id</code>	Identifier unik untuk ruangan (contoh: "7609")
<code>capacity</code>	Kapasitas maksimal mahasiswa yang dapat ditampung ruangan.

- **Konstruktor**

Konstruksi	Deskripsi
<code>__init__(room_id, capacity)</code>	Menginisialisasi objek Room dengan ID dan kapasitasnya.

Class Student

- Atribut

Atribut	Deskripsi
<code>student_id</code>	Identifier unik untuk mahasiswa (NIM)
<code>course_list</code>	Daftar <i>ID</i> mata kuliah yang diambil oleh mahasiswa.
<code>priority</code>	Daftar nilai prioritas yang berkorespondensi dengan <code>course_list</code>
<code>priority_map</code>	Dictionary yang memetakan setiap <i>ID</i> mata kuliah ke nilai prioritasnya untuk pencarian yang lebih cepat.

- Konstruktor

Konstruksi	Deskripsi
<code>__init__(student_id, course_list, priority)</code>	Menginisialisasi objek Student dan secara otomatis membangun <code>priority_map</code> dari <code>course_list</code> dan <code>priority</code>

Class Assignment

- Atribut

Atribut	Deskripsi
<code>course</code>	Objek Course yang dijadwalkan
<code>time_slot</code>	Objek TimeSlot yang dialokasikan.
<code>room</code>	Objek Room yang digunakan.

- Konstruktor

Konstruksi	Deskripsi
<code>__init__(course, time_slot, room)</code>	Menginisialisasi objek Assignment dengan menggabungkan sebuah mata kuliah, slot waktu, dan ruangan.

Class Schedule

- Atribut

Atribut	Deskripsi
assignments	Daftar lengkap berisi semua objek Assignment yang membentuk jadwal.
course_assignments	Dictionary yang memetakan setiap ID mata kuliah ke daftar Assignment-nya untuk pencarian yang lebih cepat.

- Konstruktor

Konstruksi	Deskripsi
__init__(assignments)	Menginisialisasi objek Schedule dari sebuah daftar Assignment dan secara otomatis membangun course_assignments

Class Lecturer

- Atribut

Atribut	Deskripsi
lecturer_id	Identifier unik untuk dosen
course_list	Daftar <i>ID</i> mata kuliah yang diajar oleh dosen tersebut.

- Konstruktor

Konstruksi	Deskripsi
__init__(lecturer_id, course_list)	Menginisialisasi objek Lecturer dengan ID dan daftar mata kuliah yang diajar.

2.3.2. utils.py

- Atribut

Atribut	Deskripsi
load_data_from_json(file_path)	Bertanggung jawab untuk membaca file input.json dari file_path yang diberikan. Fungsi ini mem-parsing data JSON dan

	mengubahnya menjadi daftar objek Course, Room, Student, dan Lecturer. Fungsi ini juga dilengkapi penanganan error untuk kasus di mana file tidak ditemukan atau format JSON tidak valid.
schedule_table_for_room(schedule, room_id)	Menghasilkan dan mencetak satu tabel jadwal mingguan untuk room_id tertentu. Fungsi ini akan memformat jadwal ke dalam tabel berbasis teks yang menampilkan hari dan jam, serta mampu menangani bentrok jadwal dengan menampilkannya di baris yang berbeda dalam satu slot waktu.
visualize_schedule(schedule, rooms)	Fungsi utama untuk menampilkan visualisasi. Fungsi ini akan mengiterasi semua ruangan yang ada, memeriksa ruangan mana saja yang digunakan dalam schedule yang diberikan, lalu memanggil schedule_table_for_room untuk mencetak tabel jadwal bagi setiap ruangan yang tidak kosong.

2.3.3. runners.py

- Metode

Metode	Deskripsi
plot_objective_history(objective_history: List[float], title: str, xlabel: str = "Iteration", ylabel: str = "Objective Value")	Menghasilkan dan menampilkan grafik garis sederhana yang memvisualisasikan perubahan nilai objective function terhadap iterasi. Umumnya digunakan oleh varian Hill-Climbing.
plot_acceptance_probability(iterations: List[int], probabilities: List[float], title: str) -> None	Fungsi visualisasi khusus untuk Simulated Annealing. Menampilkan grafik probabilitas penerimaan solusi yang lebih buruk seiring berjalannya iterasi.
plot_genetic_statistics(max_objective: List[float], avg_objective: List[float], title: str) -> None	Fungsi visualisasi khusus untuk Genetic Algorithm. Menampilkan grafik nilai objective function maksimum dan rata-rata dari populasi di setiap generasi.
run_steepest_ascent(courses: List[Course], rooms: List[Room],	Menjalankan algoritma Steepest-Ascent Hill-Climbing dengan metode sampling

time_slots: List[TimeSlot], students: List[Student], lecturers: List[Lecturer]) -> None	(memeriksa sebagian tetangga). Setelah selesai, fungsi ini mencetak statistik akhir, memvisualisasikan jadwal, dan menampilkan grafik performanya.
run_steepest_ascent_full(courses: List[Course], rooms: List[Room], time_slots: List[TimeSlot], students: List[Student], lecturers: List[Lecturer]) -> None	Menjalankan Steepest-Ascent Hill-Climbing dengan memeriksa semua kemungkinan tetangga di setiap iterasi.
run_stochastic(courses: List[Course], rooms: List[Room], time_slots: List[TimeSlot], students: List[Student], lecturers: List[Lecturer]) -> None	Menjalankan algoritma Stochastic Hill-Climbing, mencetak hasil, dan menampilkan visualisasi tabel serta grafik.
run_sideways_moves(courses: List[Course], rooms: List[Room], time_slots: List[TimeSlot], students: List[Student], lecturers: List[Lecturer]) -> None	Menjalankan algoritma Hill-Climbing with Sideways Moves dengan metode sampling.
run_sideways_moves_full(courses: List[Course], rooms: List[Room], time_slots: List[TimeSlot], students: List[Student], lecturers: List[Lecturer]) -> None	Menjalankan algoritma Hill-Climbing with Sideways Moves dengan memeriksa semua kemungkinan tetangga.
run_random_restart(courses: List[Course], rooms: List[Room], time_slots: List[TimeSlot], students: List[Student], lecturers: List[Lecturer]) -> None	Menjalankan strategi Random-Restart Hill-Climbing, mencetak statistik agregat (jumlah restart, total iterasi), dan menampilkan grafik nilai objektif terbaik yang ditemukan per restart.
run_genetic_algorithm(courses: List[Course], rooms: List[Room], time_slots: List[TimeSlot], students: List[Student], lecturers: List[Lecturer]) -> None	Menjalankan Genetic Algorithm, mencetak statistik akhir, dan menampilkan grafik performa maksimum serta rata-rata populasi.
run_simulated_annealing(courses: List[Course], rooms: List[Room], time_slots: List[TimeSlot], students: List[Student], lecturers: List[Lecturer]) -> None	Menjalankan algoritma Simulated Annealing, mencetak hasilnya, dan menampilkan grafik probabilitas penerimaan.

2.3.4. scheduler.py

- Metode

Metode	Deskripsi
generate_initial_schedule(courses, rooms, time_slots)	Membuat satu buah jadwal awal (<i>initial state</i>) yang sepenuhnya acak. Fungsi ini mengiterasi setiap mata kuliah dan untuk setiap SKS-nya, ia akan memberikan satu slot waktu dan satu ruangan secara acak.
priority_weight(priority)	Fungsi pembantu sederhana yang mengubah nilai prioritas numerik (misalnya 1, 2, 3) menjadi bobot <i>float</i> yang sesuai (1.75, 1.5, 1.25) untuk digunakan dalam perhitungan penalti.
objective(schedule, students, lecturers)	Merupakan objective function utama yang mengevaluasi kualitas sebuah schedule. Fungsi ini menghitung total skor penalti berdasarkan tiga aturan: (1) konflik waktu untuk setiap mahasiswa, (2) konflik waktu untuk setiap dosen, dan (3) bentrok penggunaan ruangan yang bobotnya dihitung dari prioritas mahasiswa. Nilai yang dikembalikan adalah negatif dari total penalti, sehingga nilai yang lebih tinggi menandakan jadwal yang lebih baik.
generate_neighbor(schedule, rooms, time_slots)	Mengambil sebuah jadwal dan menghasilkan satu "tetangga" (<i>neighbor</i>), yaitu jadwal baru yang hanya sedikit berbeda. Fungsi ini secara acak memilih salah satu dari dua operasi: (1) menukar posisi (waktu dan ruangan) dari dua pertemuan, atau (2) memindahkan satu pertemuan ke slot waktu dan ruangan yang baru secara acak.
initialize_population(courses: List[Course], rooms: List[Room],	Fungsi khusus untuk Genetic Algorithm. Fungsi ini memanggil

time_slots: List[TimeSlot], population_size: int) -> List[Schedule]	generate_initial_schedule berulang kali untuk membuat sebuah "populasi" awal yang terdiri dari population_size jadwal acak.
evaluate_population(population: List[Schedule], students: List[Student], lecturers: List[Lecturer]) -> List[Tuple[Schedule, float]]	Fungsi khusus untuk Genetic Algorithm. Fungsi ini mengambil seluruh populasi jadwal, lalu memanggil objective untuk setiap jadwal di dalamnya guna menghitung skor masing-masing.

2.3.5. test_generator.py

- Atribut

Atribut	Deskripsi
days: List[str]	Daftar hari yang digunakan untuk membuat kombinasi time slot (misalnya: ["Senin", "Selasa", "Rabu", ...]).
hours: List[int]	Daftar jam yang digunakan untuk membuat kombinasi time slot (misalnya: [7, 9, 11, 13, 15]).
num_courses: int	Jumlah total mata kuliah yang akan digenerasikan.
num_rooms: int	Jumlah total ruangan yang tersedia untuk penjadwalan.
num_students: int	Jumlah total mahasiswa yang akan dibuat dalam data uji.
num_lecturers: int	Jumlah total dosen yang akan dibuat dalam data uji.
min_credits: int	Jumlah minimal SKS yang harus diambil setiap mahasiswa (default: 2 SKS).
max_credits: int	Jumlah maksimal SKS yang dapat diambil mahasiswa (default: 24 SKS).
file_path: str	Lokasi file JSON tempat data hasil generator akan disimpan.
courses: List[Course]	Daftar objek Course yang menyimpan informasi setiap mata kuliah (kode, jumlah mahasiswa, SKS).

rooms: List[Room]	Daftar objek Room yang menyimpan informasi ruangan dan kapasitasnya.
students: List[Student]	Daftar objek Student yang menyimpan data mahasiswa, daftar mata kuliah yang diambil, dan prioritasnya.
lecturers: List[Lecturer]	Daftar objek Lecturer yang menyimpan data dosen beserta daftar mata kuliah yang diajar.

- **Konstruktor**

Tidak terdapat konstruktor karena seluruh fungsi dalam file ini bersifat fungsi utilitas (standalone functions) yang dipanggil secara langsung.

- **Metode**

Metode	Deskripsi
generate_time_slots(days: List[str], hours: List[int]) -> List[TimeSlot]	Menghasilkan daftar time slot dari kombinasi hari dan jam yang diberikan. Setiap kombinasi hari dan jam dibuat menjadi satu objek TimeSlot.
generate_test_data(num_courses: int, num_rooms: int, num_students: int, num_lecturers: int) -> Tuple[List[Course], List[Room], List[Student], List[Lecturer]]	Membuat dataset uji acak yang berisi daftar mata kuliah, ruangan, mahasiswa, dan dosen. Setiap data dihasilkan secara acak dengan nilai-nilai realistik seperti kapasitas ruang, jumlah SKS, dan daftar mata kuliah per mahasiswa/dosen.
save_data_to_json(courses: List[Course], rooms: List[Room], students: List[Student], lecturers: List[Lecturer], file_path: str) -> None	Menyimpan hasil data uji (courses, rooms, students, lecturers) dalam format JSON ke file dengan struktur data yang telah disesuaikan untuk kebutuhan penjadwalan.
__main__	Menjalankan fungsi test data generator untuk membuat dan menyimpan data uji berukuran kecil dan besar. Contohnya: membuat dataset small_test.json dengan 15 mata kuliah, 10 ruangan, 50 mahasiswa, dan 6 dosen.

2.3.6. main.py

- **Atribut**

Atribut	Deskripsi
courses: List[Course]	Menyimpan daftar objek Course yang berisi data mata kuliah, termasuk kode, jumlah mahasiswa, dan jumlah SKS.
rooms: List[Room]	Menyimpan daftar objek Room yang berisi informasi ruangan, termasuk kapasitas dan kode ruangan.
students: List[Student]	Menyimpan daftar objek Student yang berisi data mahasiswa, termasuk NIM, daftar mata kuliah, dan prioritas.
lecturers: List[Lecturer]	Menyimpan daftar objek Lecturer yang berisi data dosen beserta daftar mata kuliah yang diajarkan.
time_slots: List[TimeSlot]	Menyimpan daftar objek TimeSlot yang merepresentasikan kombinasi hari dan jam kuliah yang tersedia.
initial_schedule: Schedule	Jadwal awal yang dihasilkan sebelum dilakukan optimisasi dengan algoritma pencarian lokal.
choice: str	Input dari pengguna untuk memilih algoritma yang akan dijalankan melalui menu utama.

- **Konstruktor**

Tidak terdapat konstruktor.

- **Metode**

Metode	Deskripsi
main()	Fungsi utama program yang mengatur alur kerja. Memuat data dari file JSON, menghasilkan jadwal awal, menampilkan menu interaktif, serta menjalankan algoritma penjadwalan berdasarkan pilihan pengguna.

2.4. Hasil Eksperimen dan Analisis

Berikut merupakan hasil eksperimen dan analisis dari pencarian solusi penjadwalan kelas mingguan dengan *local search* menggunakan *Hill-Climbing Algorithm*, *Simulated Annealing Algorithm*, dan *Genetic Algorithm*.

2.4.1. Hasil Eksperimen

1. Test Case Input dari Spesifikasi (input.json)

Initial State						
Initial State:						
Initial Objective: -0.00						
Ruangan 7609:						
	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00				IF3110_K02		
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00 IF3071_K01						
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00		IF3140_K01				

Ruangan 7606:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00				IF3071_K01		
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00					IF3140_K01	
13:00 - 14:00						
14:00 - 15:00	IF3130_K01					
15:00 - 16:00						
16:00 - 17:00					IF3110_K02	

Ruangan multimedia:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00				IF3130_K01		
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00				IF3110_K02		
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00					IF3071_K01	

Tampilan pertama saat program baru dijalankan

```
~~~~~  
Please select the input file to use for this session:  
~~~~~
```

1. Standard Test (input.json)
 2. Semi-Large Test (semi_large_test.json)
 3. Large Test (large_test.json)
 4. Enter a custom file path
 5. Exit Program
- ```
~~~~~
```

```
Enter your choice (1-5): 1
```

```
-> File '../data/input.json' loaded successfully.
```

```
~~~~~  
Solving the Weekly Class Scheduling Problem with Local Search Algorithms
~~~~~
```

```
Developed by: Tubes1 - K1 (13523019, 13523059, 13523067)  
~~~~~
```

1. Steepest-Ascent Hill-Climbing (Sampling)
  2. Steepest-Ascent Hill-Climbing (Full)
  3. Stochastic Hill-Climbing
  4. Hill-Climbing with Sideways Moves (Sampling)
  5. Hill-Climbing with Sideways Moves (Full)
  6. Random-Restart Hill-Climbing
  7. Genetic Algorithm
  8. Simulated Annealing
  9. Run All Algorithms Sequentially
  10. Exit
- ```
~~~~~
```

Output penyelesaian program menggunakan
Steepest Ascent Sampling Hill-Climbing Algorithm pada terminal

Enter your choice (1-10): 1

1. Steepest-Ascent Hill-Climbing (Sampling)
-> Steepest-Ascent: Local optimum reached at iteration 2.

Final Result:

- Final objective: -0.00
- Iterations until stop: 2
- Search Duration: 0.0163 seconds

Ruangan 7609:

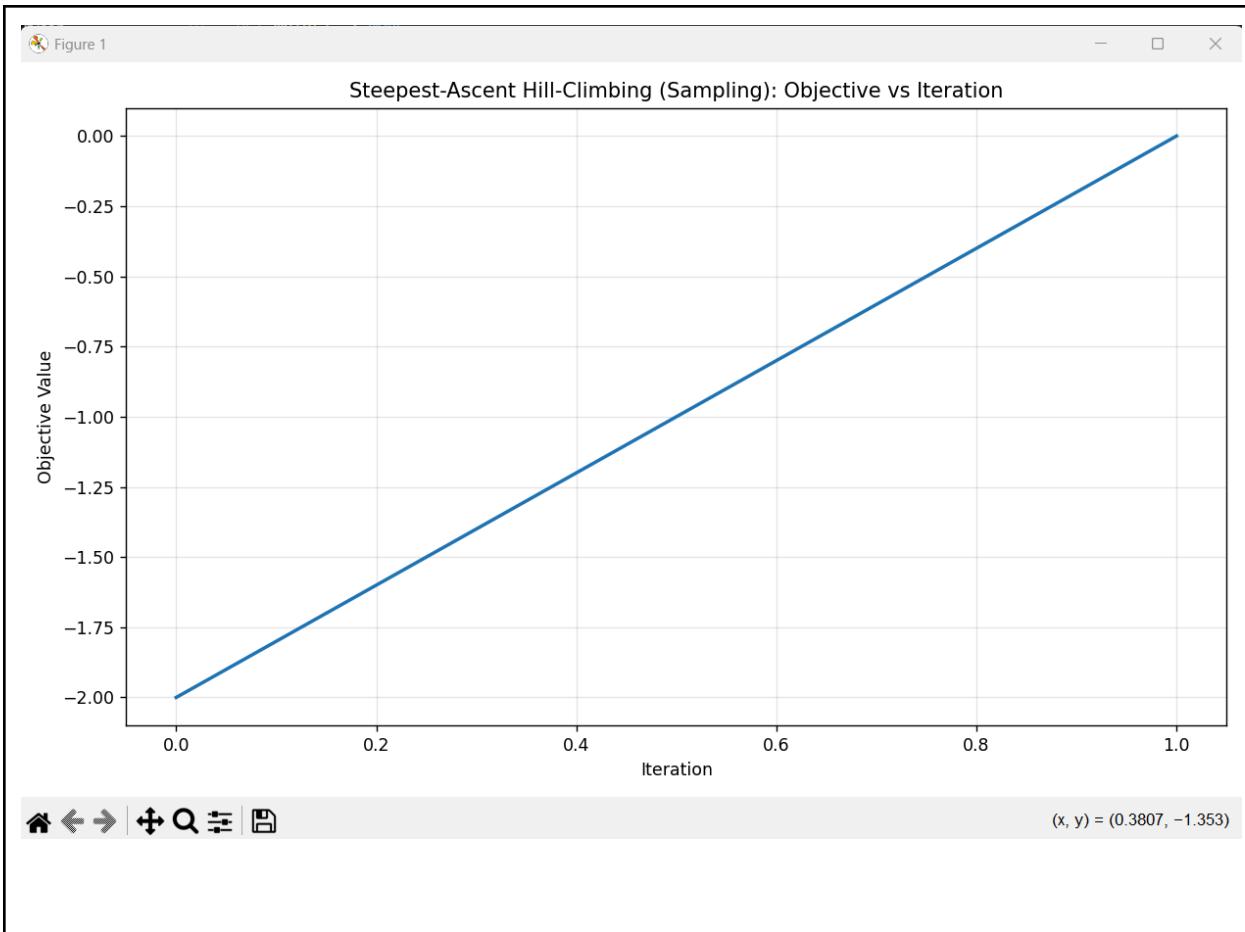
	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00	IF3110_K02			IF3130_K01		
09:00 - 10:00			IF3110_K02			
10:00 - 11:00				IF3110_K02		
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						

Ruangan 7606:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						IF3071_K01
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00			IF3140_K01			
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						

Ruangan multimedia:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00				IF3140_K01		
09:00 - 10:00						
10:00 - 11:00				IF3071_K01		
11:00 - 12:00					IF3071_K01	
12:00 - 13:00						
13:00 - 14:00					IF3130_K01	
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						



Output penyelesaian program menggunakan
Steepest Ascent Full Hill-Climbing Algorithm pada terminal

1b. Steepest-Ascent Hill-Climbing (Full)

-> Steepest-Ascent (Full): Local optimum reached at iteration 1.

Final Result:

- Final objective: -0.00
- Iterations until stop: 1
- Search Duration: 0.1990 seconds

Ruangan 7609:

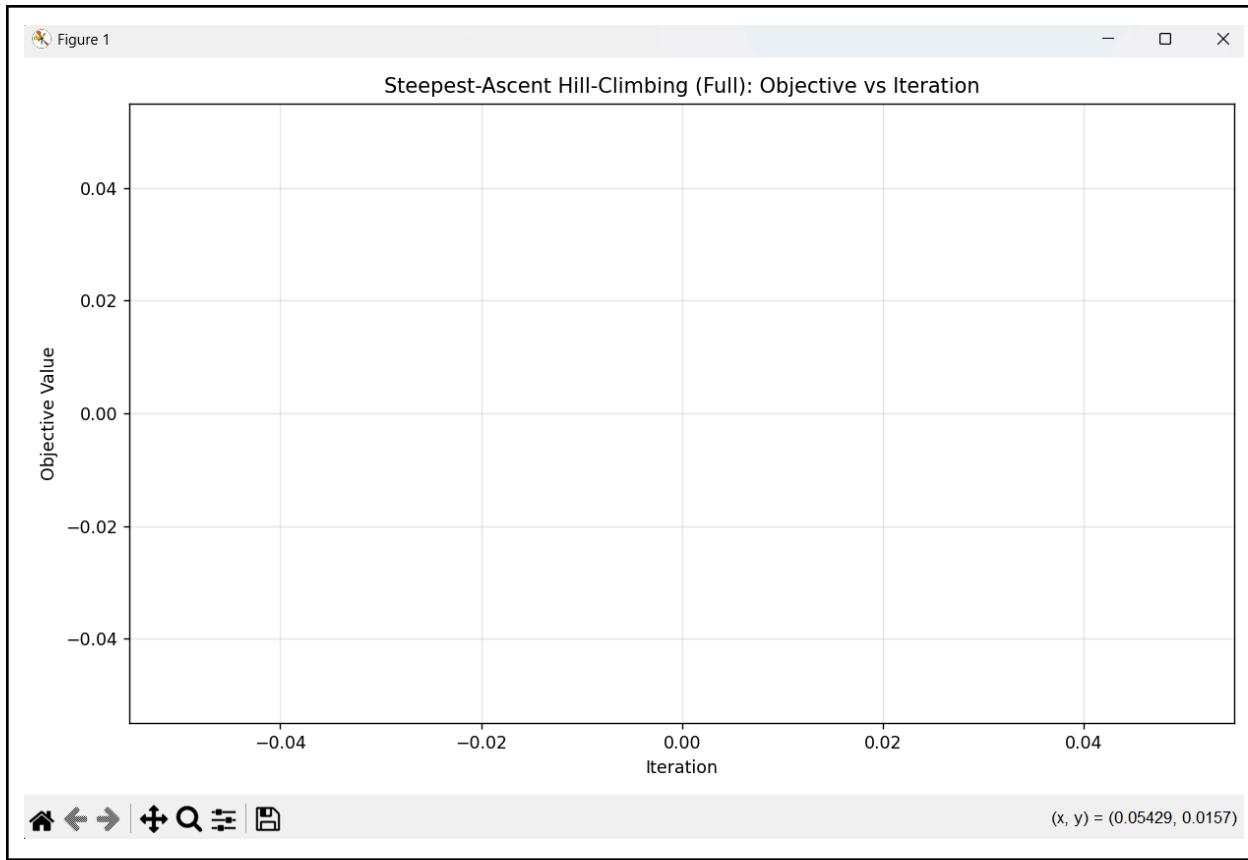
	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00		IF3140_K01				
09:00 - 10:00				IF3130_K01		
10:00 - 11:00						IF3071_K01
11:00 - 12:00						
12:00 - 13:00		IF3071_K01				
13:00 - 14:00						
14:00 - 15:00			IF3110_K02			IF3110_K02
15:00 - 16:00						
16:00 - 17:00						

Ruangan 7606:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00			IF3140_K01			

Ruangan multimedia:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00			IF3130_K01			
13:00 - 14:00			IF3110_K02			
14:00 - 15:00	IF3071_K01					
15:00 - 16:00						
16:00 - 17:00						



Output penyelesaian program menggunakan
Stochastic Hill-Climbing Algorithm pada terminal

```
Enter your choice (1-10): 3
```

2. Stochastic Hill-Climbing

-> Stochastic: Local optimum reached after 100 iterations without improvement.

Final Result:

- Final objective: -0.00
- Iterations until stop: 109
- Search Duration: 0.0186 seconds

Ruangan 7609:

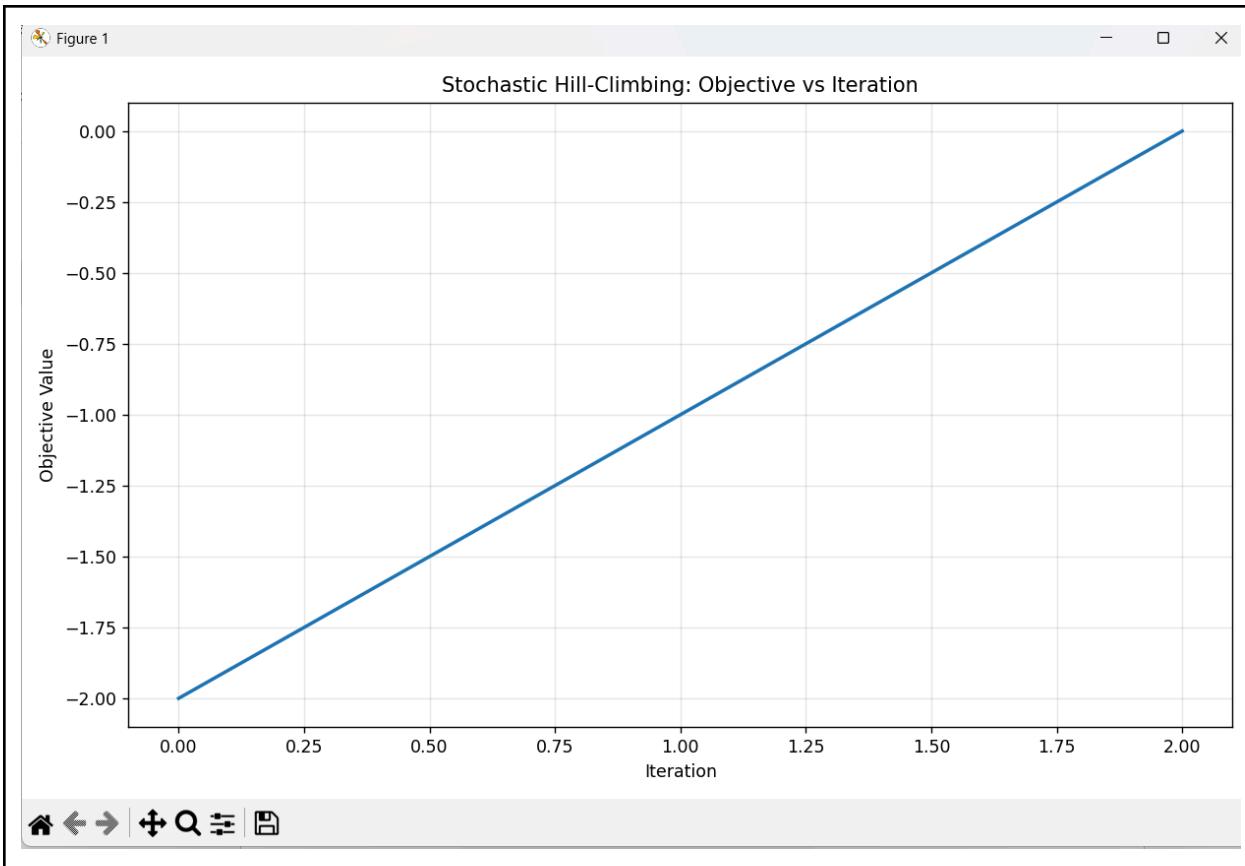
	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00		IF3110_K02				
12:00 - 13:00						
13:00 - 14:00				IF3140_K01		
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						

Ruangan 7606:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						IF3130_K01
09:00 - 10:00			IF3130_K01	IF3110_K02		
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00				IF3110_K02		
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						

Ruangan multimedia:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00	IF3071_K01					
10:00 - 11:00						
11:00 - 12:00			IF3071_K01			
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00			IF3140_K01			
15:00 - 16:00			IF3071_K01			
16:00 - 17:00						



Output penyelesaian program menggunakan
Sideways Move Sampling Hill-Climbing Algorithm pada terminal

3. Hill-Climbing with Sideways Moves (Sampling)

-> Sideways-Move: Optimum reached or sideways limit exceeded at iteration 101.

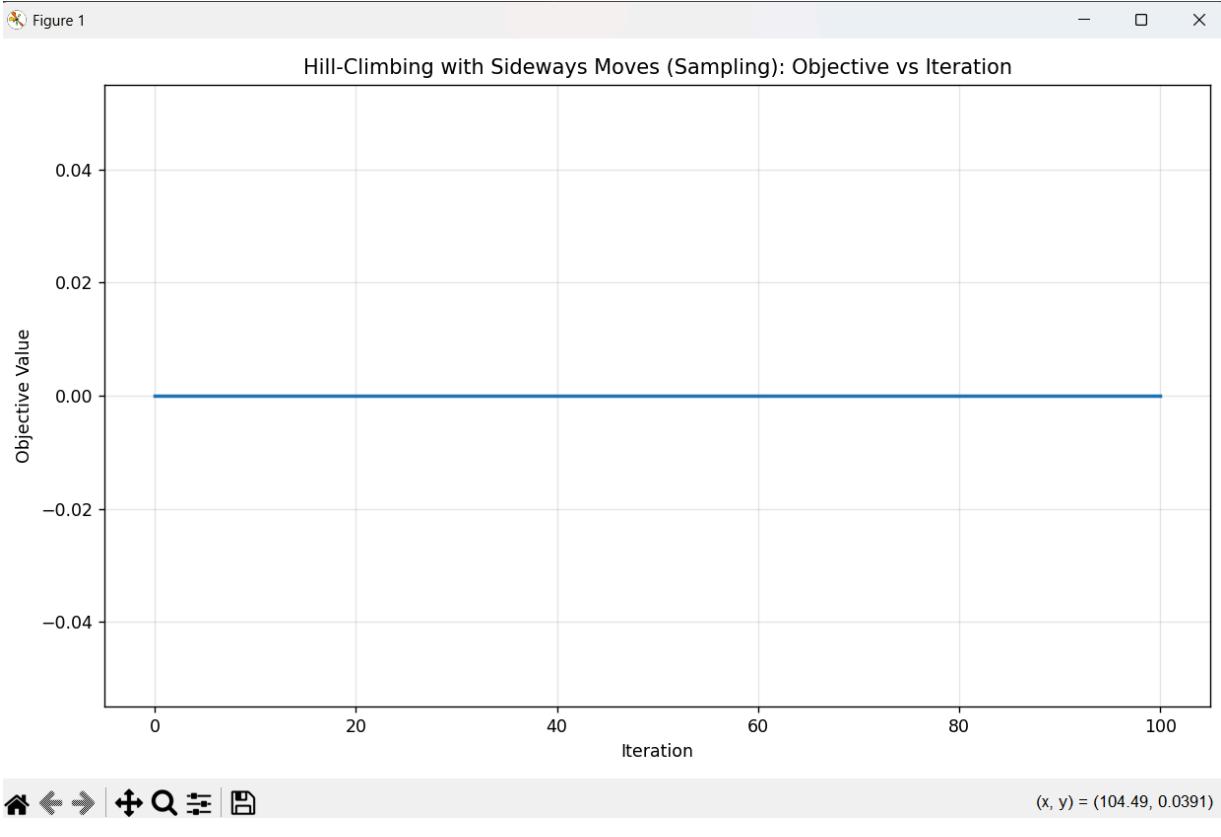
Final Result:

- Final objective: -0.00
- Iterations until stop: 101
- Search Duration: 0.6717 seconds

Ruangan 7606:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00	IF3110_K02					
11:00 - 12:00					IF3110_K02	
12:00 - 13:00						
13:00 - 14:00					IF3130_K01	
14:00 - 15:00		IF3110_K02				
15:00 - 16:00						
16:00 - 17:00						

Ruangan multimedia:	Senin	Selasa	Rabu	Kamis	Jumat
08:00 - 09:00	IF3071_K01				
09:00 - 10:00					
10:00 - 11:00					
11:00 - 12:00					
12:00 - 13:00	IF3071_K01				
13:00 - 14:00		IF3130_K01			
14:00 - 15:00			IF3071_K01		
15:00 - 16:00		IF3140_K01			
16:00 - 17:00				IF3140_K01	



Output penyelesaian program menggunakan
Sideways Move Full Hill-Climbing Algorithm pada terminal

```
Enter your choice (1-10): 5
```

```
3b. Hill-Climbing with Sideways Moves (Full)
```

```
-> Sideways-Move (Full): Optimum reached or sideways limit exceeded at iteration 101.
```

```
Final Result:
```

- Final objective: -0.00
- Iterations until stop: 101
- Search Duration: 20.6970 seconds

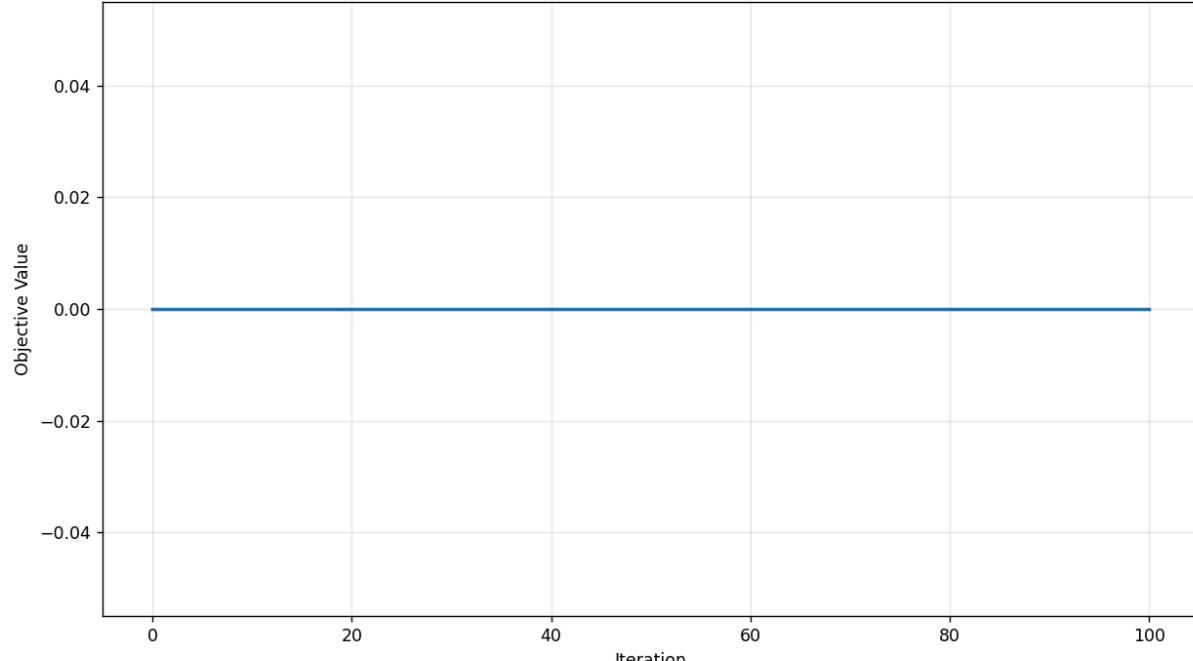
```
Ruangan 7609:
```

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						IF3130_K01
09:00 - 10:00	IF3140_K01					
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00			IF3071_K01	IF3071_K01		
14:00 - 15:00						
15:00 - 16:00				IF3140_K01		
16:00 - 17:00						

Ruangan multimedia:	Senin	Selasa	Rabu	Kamis	Jumat
08:00 - 09:00			IF3110_K02		
09:00 - 10:00					
10:00 - 11:00			IF3110_K02		
11:00 - 12:00		IF3071_K01			
12:00 - 13:00					
13:00 - 14:00					
14:00 - 15:00					IF3130_K01
15:00 - 16:00		IF3110_K02			
16:00 - 17:00					

Figure 1

Hill-Climbing with Sideways Moves (Full): Objective vs Iteration



Output penyelesaian program menggunakan

Random Restart Hill-Climbing Algorithm pada terminal

```
4. Random-Restart Hill-Climbing
Starting Random-Restart Hill-Climbing with 20 restarts.
-> Restart #1/20...
-> Steepest-Ascent: Local optimum reached at iteration 2.
-> New global best found with objective: -0.00
-> Restart #2/20...
-> Steepest-Ascent: Local optimum reached at iteration 2.
-> Restart #3/20...
-> Steepest-Ascent: Local optimum reached at iteration 1.
-> Restart #4/20...
-> Steepest-Ascent: Local optimum reached at iteration 3.
-> Restart #5/20...
-> Steepest-Ascent: Local optimum reached at iteration 1.
-> Restart #6/20...
-> Steepest-Ascent: Local optimum reached at iteration 2.
-> Restart #7/20...
-> Steepest-Ascent: Local optimum reached at iteration 1.
-> Restart #8/20...
-> Steepest-Ascent: Local optimum reached at iteration 2.
-> Restart #9/20...
-> Steepest-Ascent: Local optimum reached at iteration 2.
-> Restart #10/20...
-> Steepest-Ascent: Local optimum reached at iteration 1.
-> Restart #11/20...
-> Steepest-Ascent: Local optimum reached at iteration 2.
-> Restart #12/20...
-> Steepest-Ascent: Local optimum reached at iteration 1.
-> Restart #13/20...
-> Steepest-Ascent: Local optimum reached at iteration 2.
-> Restart #14/20...
-> Steepest-Ascent: Local optimum reached at iteration 2.
-> Restart #15/20...
```

```

    > RESTART #14/20...
-> Steepest-Ascent: Local optimum reached at iteration 2.
-> Restart #15/20...
-> Steepest-Ascent: Local optimum reached at iteration 1.
-> Restart #16/20...
-> Steepest-Ascent: Local optimum reached at iteration 1.
-> Restart #17/20...
-> Steepest-Ascent: Local optimum reached at iteration 2.
-> Restart #18/20...
-> Steepest-Ascent: Local optimum reached at iteration 2.
-> Restart #19/20...
-> Steepest-Ascent: Local optimum reached at iteration 2.
-> Restart #20/20...
-> Steepest-Ascent: Local optimum reached at iteration 1.

```

Final Result:

- Global Best objective: -0.00
- Number of Restarts: 20
- Total Iterations (sum over all restarts): 33
- Search Duration: 0.2785 seconds

Ruangan 7609:

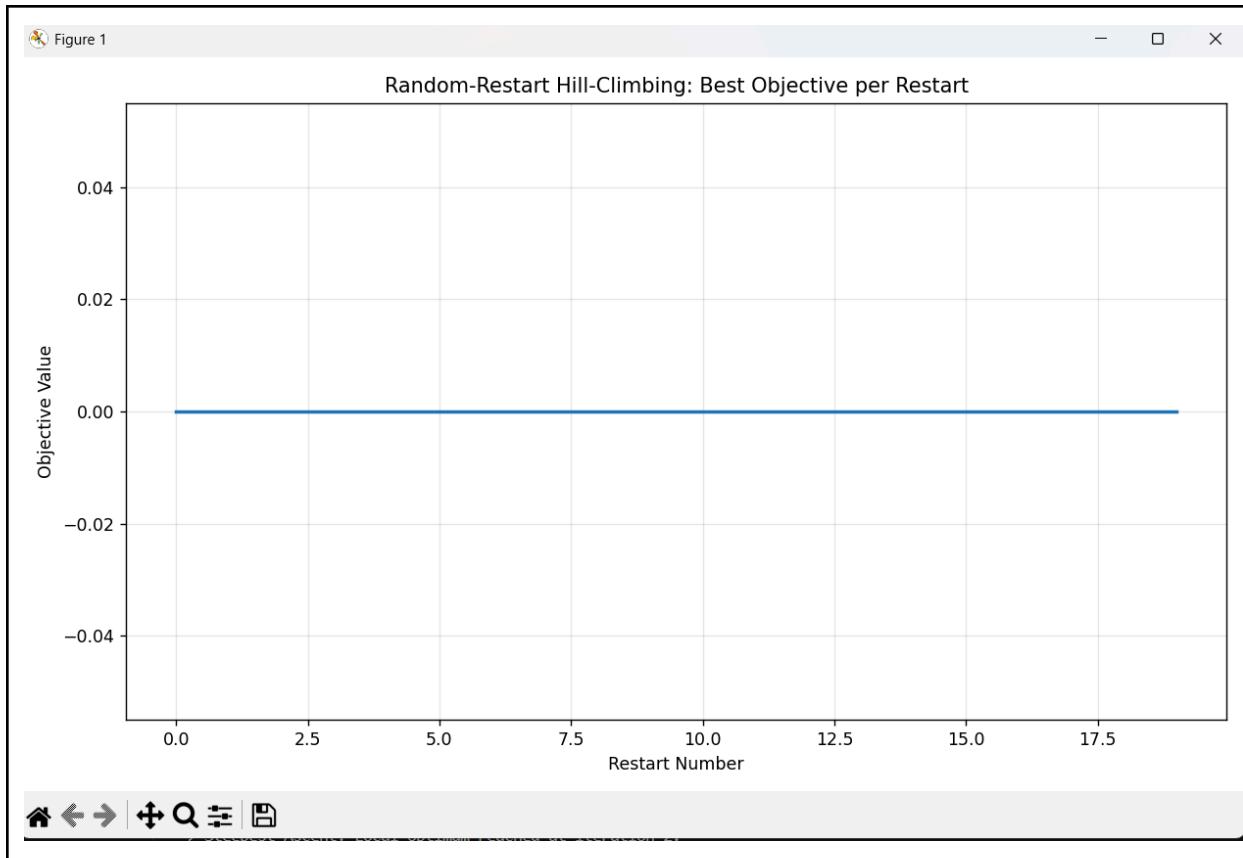
	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00	IF3140_K01					
10:00 - 11:00				IF3071_K01		
11:00 - 12:00						
12:00 - 13:00	IF3130_K01					
13:00 - 14:00			IF3071_K01			
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						

Ruangan 7606:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00				IF3110_K02		
12:00 - 13:00						IF3071_K01
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						

Ruangan multimedia:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00		IF3130_K01				
10:00 - 11:00						
11:00 - 12:00	IF3110_K02		IF3140_K01			
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00	IF3110_K02					
16:00 - 17:00						



Output penyelesaian program menggunakan
Genetic Algorithm pada terminal

5. Genetic Algorithm

Final Result:

- Final objective: -0.00
- Population Size: 100, Generations: 100
- Search Duration: 3.4477 seconds

Ruangan 7609:

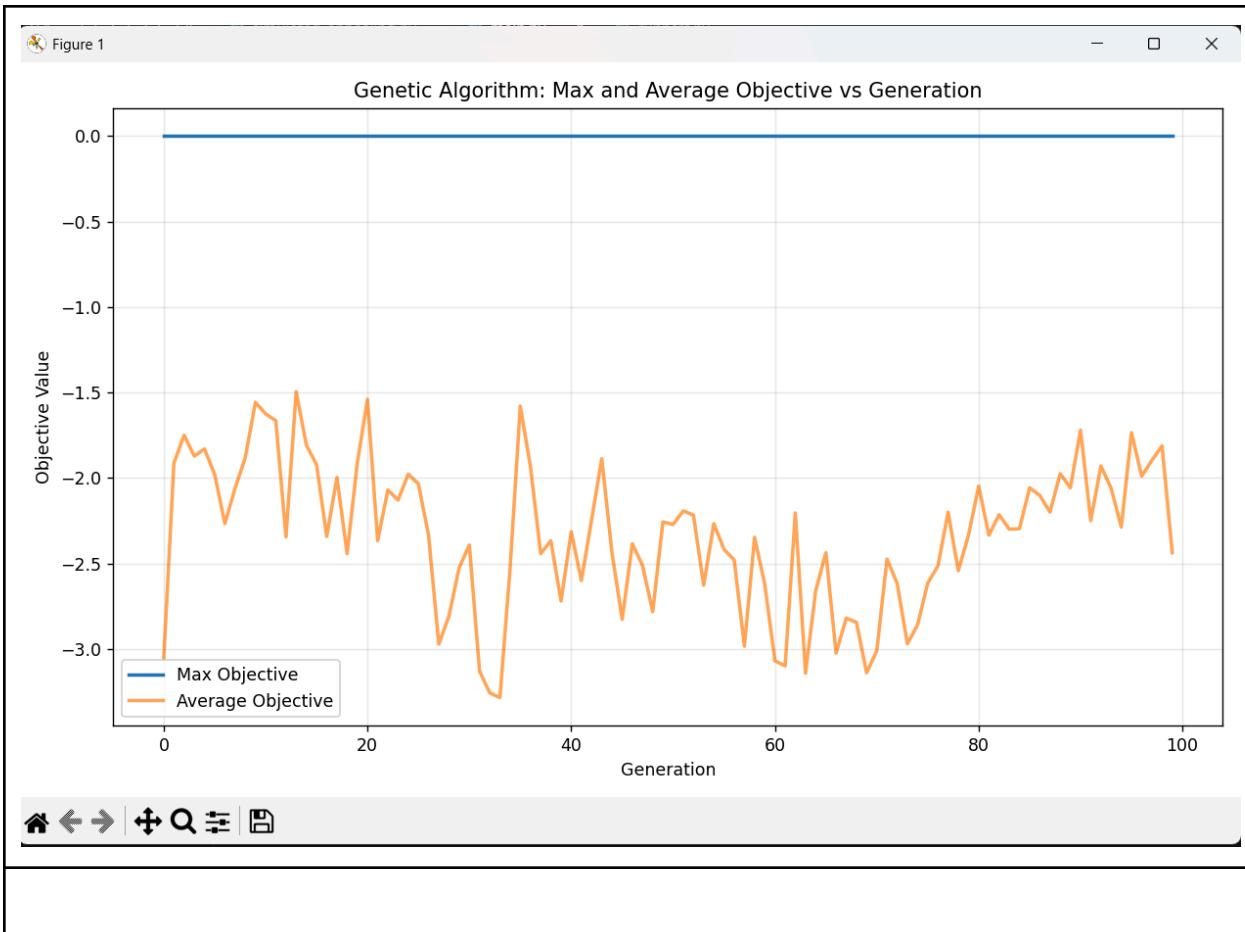
	Senin	Selasa	Rabu	Kamis	Jumat
08:00 - 09:00					
09:00 - 10:00					
10:00 - 11:00					IF3130_K01
11:00 - 12:00				IF3071_K01	
12:00 - 13:00					
13:00 - 14:00					
14:00 - 15:00					
15:00 - 16:00					
16:00 - 17:00					

Ruangan 7606:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00			IF3130_K01	IF3110_K02		
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00						IF3140_K01
15:00 - 16:00						
16:00 - 17:00						

Ruangan multimedia:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00	IF3110_K02					
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00			IF3140_K01			
14:00 - 15:00			IF3071_K01			
15:00 - 16:00			IF3110_K02			
16:00 - 17:00		IF3071_K01				



Output penyelesaian program menggunakan
Simulated Annealing Algorithm pada terminal

6. Simulated Annealing

Final Result:

- Final objective: -0.00
- Frequency of 'stuck' at local optima: 1
- Search Duration: 0.0265 seconds

Ruangan 7609:

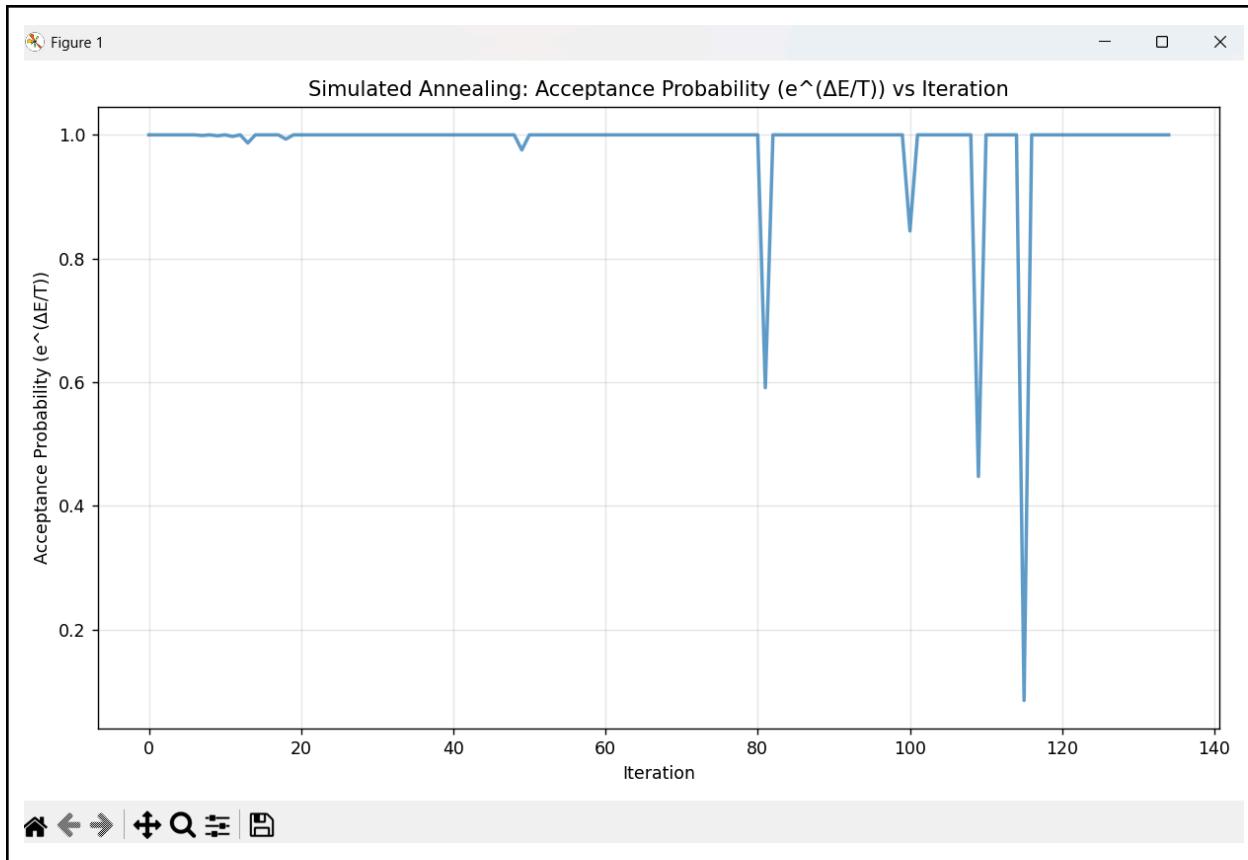
	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00 IF3071_K01						
10:00 - 11:00 IF3130_K01						
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						

Ruangan 7606:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00			IF3130_K01			
15:00 - 16:00						
16:00 - 17:00						

Ruangan multimedia:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00		IF3140_K01				
10:00 - 11:00			IF3071_K01			
11:00 - 12:00	IF3110_K02	IF3110_K02				
12:00 - 13:00				IF3071_K01		
13:00 - 14:00						
14:00 - 15:00				IF3110_K02		
15:00 - 16:00	IF3140_K01					
16:00 - 17:00						



2. Test Case Semi Large (semi_large_test.json)

Initial State

Initial State:

Initial Objective: -551.75

Ruangan R-7600:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00		IF3000_K02				
16:00 - 17:00	IF3000_K02			IF3008_K03		

Ruangan R-7601:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00	IF3009_K01					
09:00 - 10:00						
10:00 - 11:00		IF3006_K03				
11:00 - 12:00						
12:00 - 13:00						IF3008_K03
13:00 - 14:00	IF3000_K02	IF3009_K01				
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						

Ruangan R-7602:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00				IF3008_K03		
13:00 - 14:00						
14:00 - 15:00					IF3000_K02	
15:00 - 16:00						
16:00 - 17:00						

Ruangan R-7603:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00			IF3007_K01			
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						

Ruangan R-7604:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00		IF3003_K01				
16:00 - 17:00						

Ruangan R-7605:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00			IF3006_K03			
09:00 - 10:00	IF3002_K01					
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00			IF3009_K01			
15:00 - 16:00			IF3005_K03			
16:00 - 17:00						

Ruangan R-7606:

	Senin	Selasa	Rabu	Kamis	Jumat
08:00 - 09:00					
09:00 - 10:00					
10:00 - 11:00					
11:00 - 12:00					
12:00 - 13:00					IF3002_K01 IF3004_K01
13:00 - 14:00		IF3001_K01			IF3001_K01
14:00 - 15:00					
15:00 - 16:00					
16:00 - 17:00					

Ruangan R-7608:

	Senin	Selasa	Rabu	Kamis	Jumat
08:00 - 09:00					
09:00 - 10:00					
10:00 - 11:00				IF3005_K03	
11:00 - 12:00					
12:00 - 13:00					
13:00 - 14:00			IF3003_K01		
14:00 - 15:00					
15:00 - 16:00					
16:00 - 17:00					

Ruangan R-7609:

	Senin	Selasa	Rabu	Kamis	Jumat
08:00 - 09:00					
09:00 - 10:00			IF3006_K03		
10:00 - 11:00					
11:00 - 12:00				IF3007_K01	
12:00 - 13:00					IF3002_K01
13:00 - 14:00					
14:00 - 15:00		IF3006_K03			
15:00 - 16:00				IF3004_K01	
16:00 - 17:00					

Tampilan pertama saat program baru dijalankan

```
PS D:\Semester 5\IF3170 - Intelegensi Artifisial\IF3170_Tubes1\src> python main.py

~~~~~
Please select the input file to use for this session:
~~~~~

1. Standard Test (input.json)
2. Semi-Large Test (semi_large_test.json)
3. Large Test (large_test.json)
4. Enter a custom file path
5. Exit Program

~~~~~
Enter your choice (1-5): 2
-> File '../data/semi_large_test.json' loaded successfully.

~~~~~
Solving the Weekly Class Scheduling Problem with Local Search Algorithms
Developed by: Tubes1 - K1 (13523019, 13523059, 13523067)

~~~~~

1. Steepest-Ascent Hill-Climbing (Sampling)
2. Steepest-Ascent Hill-Climbing (Full)
3. Stochastic Hill-Climbing
4. Hill-Climbing with Sideways Moves (Sampling)
5. Hill-Climbing with Sideways Moves (Full)
6. Random-Restart Hill-Climbing
7. Genetic Algorithm
8. Simulated Annealing
9. Run All Algorithms Sequentially
10. Exit
```

Output penyelesaian program menggunakan
Steepest Ascent Sampling Hill-Climbing Algorithm pada terminal

```
Enter your choice (1-10): 1
```

```
1. Steepest-Ascent Hill-climbing (Sampling)
-> Steepest-Ascent: Local optimum reached at iteration 8.
```

```
Final Result:
```

- Final objective: -0.00
- Iterations until stop: 8
- Search Duration: 0.4026 seconds

```
Ruangan R-7600:
```

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00	IF3005_K03					
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						IF3000_K02

Ruangan R-7601:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00				IF3007_K01		
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						

Ruangan R-7602:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00					IF3006_K03	
12:00 - 13:00					IF3002_K01	
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00				IF3008_K03		

Ruangan R-7603:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00				IF3002_K01		
12:00 - 13:00						
13:00 - 14:00				IF3009_K01		
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00			IF3007_K01			

Ruangan R-7604:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00					IF3008_K03	
11:00 - 12:00		IF3004_K01				
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00					IF3006_K03	
15:00 - 16:00						
16:00 - 17:00						

Ruangan R-7605:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00		IF3001_K01				
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00	IF3009_K01					
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00	IF3004_K01					

Ruangan R-7606:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00			IF3000_K02			
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00	IF3002_K01					
15:00 - 16:00		IF3008_K03				
16:00 - 17:00						

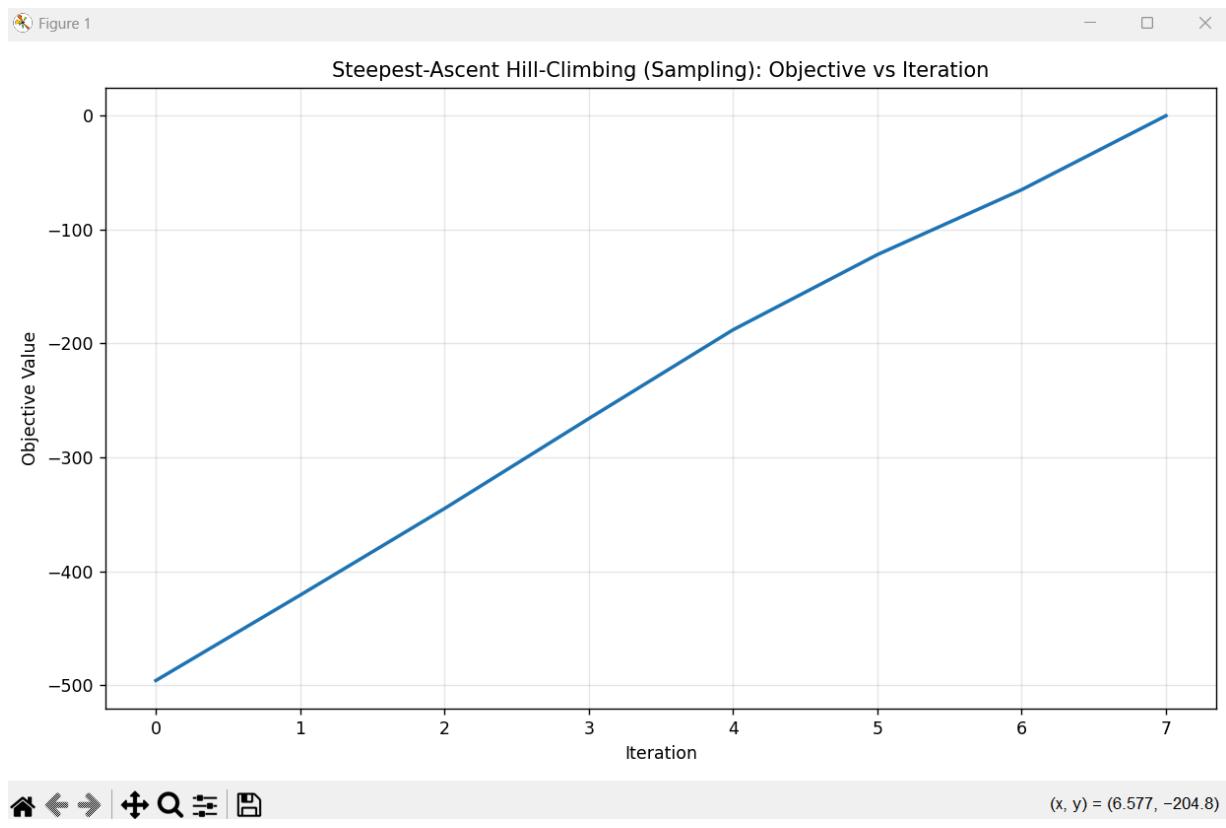
Ruangan R-7607:

	Senin	Selasa	Rabu	Kamis	Jumat
08:00 - 09:00					IF3003_K01
09:00 - 10:00					
10:00 - 11:00					
11:00 - 12:00					
12:00 - 13:00				IF3006_K03	
13:00 - 14:00		IF3009_K01	IF3005_K03		
14:00 - 15:00					
15:00 - 16:00					
16:00 - 17:00					

Ruangan R-7608:

	Senin	Selasa	Rabu	Kamis	Jumat
08:00 - 09:00					
09:00 - 10:00					IF3000_K02
10:00 - 11:00					
11:00 - 12:00					
12:00 - 13:00					
13:00 - 14:00					IF3000_K02
14:00 - 15:00					
15:00 - 16:00				IF3006_K03	IF3001_K01
16:00 - 17:00					

Ruangan R-7609:	Senin	Selasa	Rabu	Kamis	Jumat
08:00 - 09:00					
09:00 - 10:00					
10:00 - 11:00					
11:00 - 12:00					
12:00 - 13:00		IF3003_K01			
13:00 - 14:00					
14:00 - 15:00					
15:00 - 16:00					
16:00 - 17:00					



Output penyelesaian program menggunakan
Steepest Ascent Full Hill-Climbing Algorithm pada terminal

1b. Steepest-Ascent Hill-Climbing (Full)
-> Steepest-Ascent (Full): Local optimum reached at iteration 6.

Final Result:

- Final objective: -0.00
- Iterations until stop: 6
- Search Duration: 70.0659 seconds

Ruangan R-7600:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00	IF3008_K03	IF3000_K02				IF3009_K01
09:00 - 10:00	IF3002_K01					
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00					IF3006_K03	
13:00 - 14:00	IF3001_K01					
14:00 - 15:00	IF3007_K01					
15:00 - 16:00	IF3000_K02					
16:00 - 17:00	IF3004_K01					

Ruangan R-7601:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00	IF3004_K01					
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00						IF3008_K03
16:00 - 17:00			IF3000_K02			IF3006_K03

Ruangan R-7603:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00			IF3006_K03			
10:00 - 11:00			IF3005_K03			
11:00 - 12:00						
12:00 - 13:00	IF3005_K03					
13:00 - 14:00			IF3002_K01			
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						

Ruangan R-7605:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00		IF3007_K01				
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						

Ruangan R-7606:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00				IF3009_K01		
10:00 - 11:00 IF3006_K03						
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00			IF3000_K02			
16:00 - 17:00						

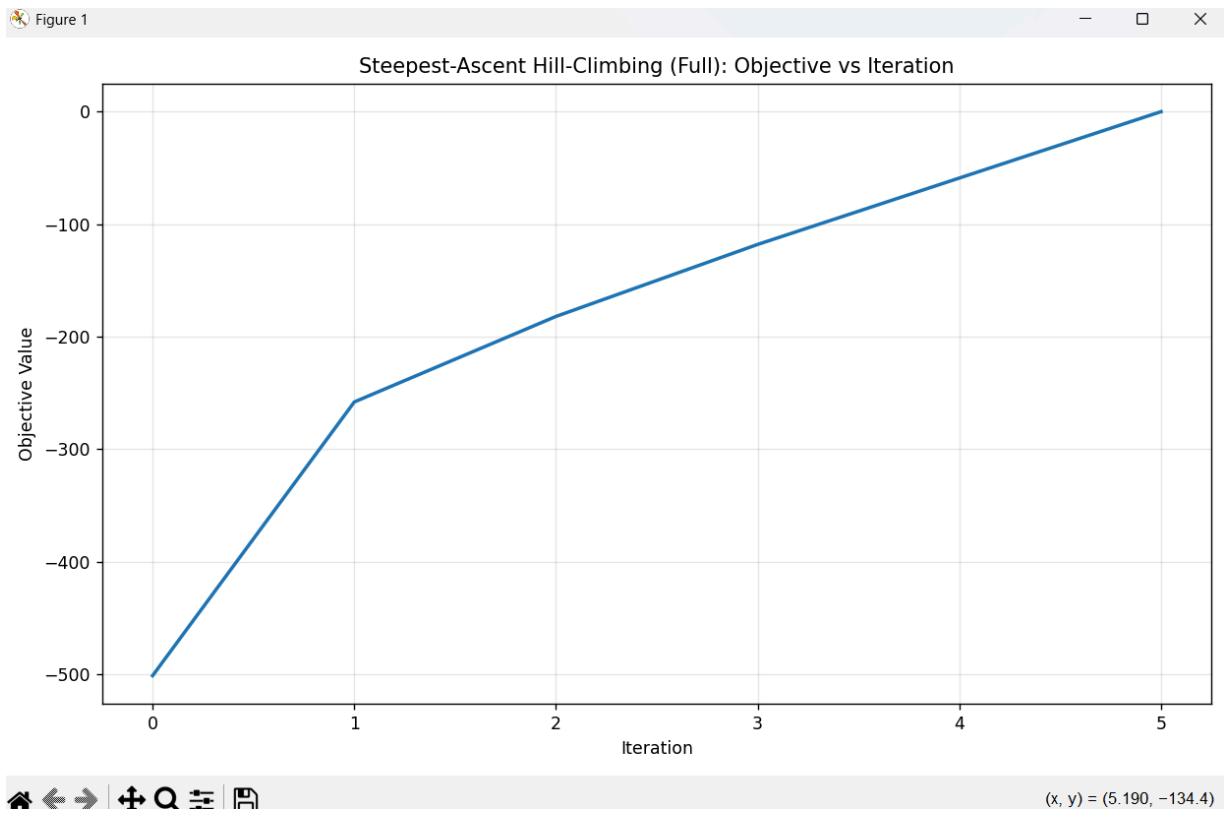
Ruangan R-7607:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00				IF3003_K01		
09:00 - 10:00					IF3001_K01	
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00				IF3003_K01		
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						

Ruangan R-7608:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00					IF3008_K03	
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00		IF3009_K01				
16:00 - 17:00						

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00		IF3002_K01				
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						



Output penyelesaian program menggunakan

Stochastic Hill-Climbing Algorithm pada terminal

```
Enter your choice (1-10): 3
```

2. Stochastic Hill-Climbing

-> Stochastic: Local optimum reached after 100 iterations without improvement.

Final Result:

- Final objective: -0.00
- Iterations until stop: 211
- Search Duration: 0.2317 seconds

Ruangan R-7600:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00			IF3000_K02			
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						

Ruangan R-7601:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00				IF3004_K01		
10:00 - 11:00						
11:00 - 12:00 IF3003_K01						
12:00 - 13:00					IF3000_K02	
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00 IF3009_K01						
16:00 - 17:00						

Ruangan R-7602:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00		IF3008_K03				
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						

Ruangan R-7603:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00				IF3006_K03		
09:00 - 10:00						
10:00 - 11:00						IF3008_K03
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00		IF3002_K01				
15:00 - 16:00						
16:00 - 17:00			IF3004_K01			

Ruangan R-7604:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00		IF3008_K03		IF3007_K01		
11:00 - 12:00		IF3002_K01				
12:00 - 13:00		IF3006_K03				
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00				IF3001_K01		
16:00 - 17:00						

Ruangan R-7605:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00	IF3003_K01					
09:00 - 10:00	IF3000_K02					
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						

Ruangan R-7606:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00	IF3006_K03					

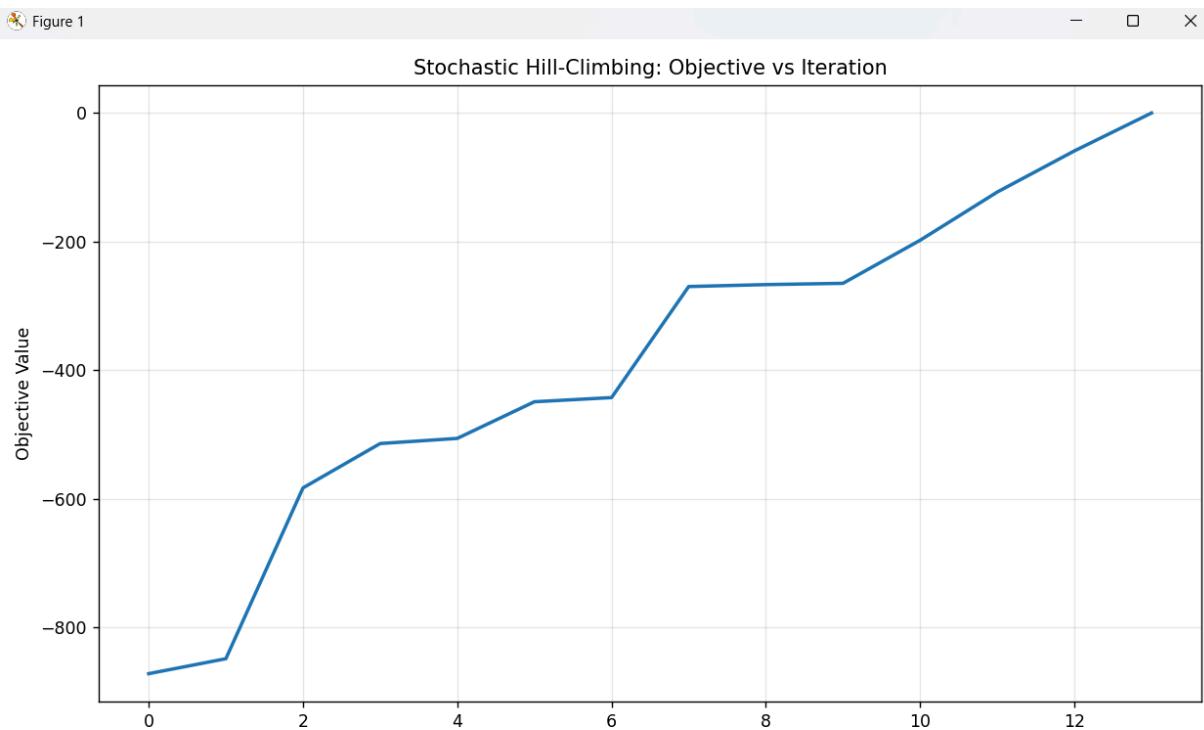
Ruangan R-7607:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00						IF3009_K01
14:00 - 15:00				IF3001_K01		
15:00 - 16:00						
16:00 - 17:00						

Ruangan R-7608:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						IF3009_K01
10:00 - 11:00			IF3005_K03			
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00 IF3000_K02						
15:00 - 16:00		IF3007_K01				
16:00 - 17:00						

Ruangan R-7609:	Senin	Selasa	Rabu	Kamis	Jumat
08:00 - 09:00					
09:00 - 10:00					
10:00 - 11:00					
11:00 - 12:00			IF3002_K01	IF3005_K03	
12:00 - 13:00					
13:00 - 14:00			IF3006_K03		
14:00 - 15:00					
15:00 - 16:00					
16:00 - 17:00					



Output penyelesaian program menggunakan

Sideways Move Sampling Hill-Climbing Algorithm pada terminal

3. Hill-Climbing with Sideways Moves (Sampling)

-> Sideways-Move: Optimum reached or sideways limit exceeded at iteration 110.

Final Result:

- Final objective: -0.00
- Iterations until stop: 110
- Search Duration: 5.2682 seconds

Ruangan R-7600:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00			IF3007_K01			
09:00 - 10:00						
10:00 - 11:00						IF3002_K01
11:00 - 12:00				IF3004_K01		
12:00 - 13:00			IF3000_K02			
13:00 - 14:00		IF3000_K02				
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00		IF3001_K01		IF3009_K01		

Ruangan R-7601:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00		IF3000_K02				
16:00 - 17:00	IF3003_K01					

Ruangan R-7602:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						IF3007_K01
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00					IF3008_K03	
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						

Ruangan R-7603:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00			IF3001_K01			
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00 IF3008_K03						
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						IF3003_K01

Ruangan R-7604:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00				IF3006_K03		
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00			IF3000_K02			
14:00 - 15:00 IF3006_K03						
15:00 - 16:00				IF3002_K01		
16:00 - 17:00						

Ruangan R-7605:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00		IF3004_K01				
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00						IF3006_K03
16:00 - 17:00						

Ruangan R-7607:

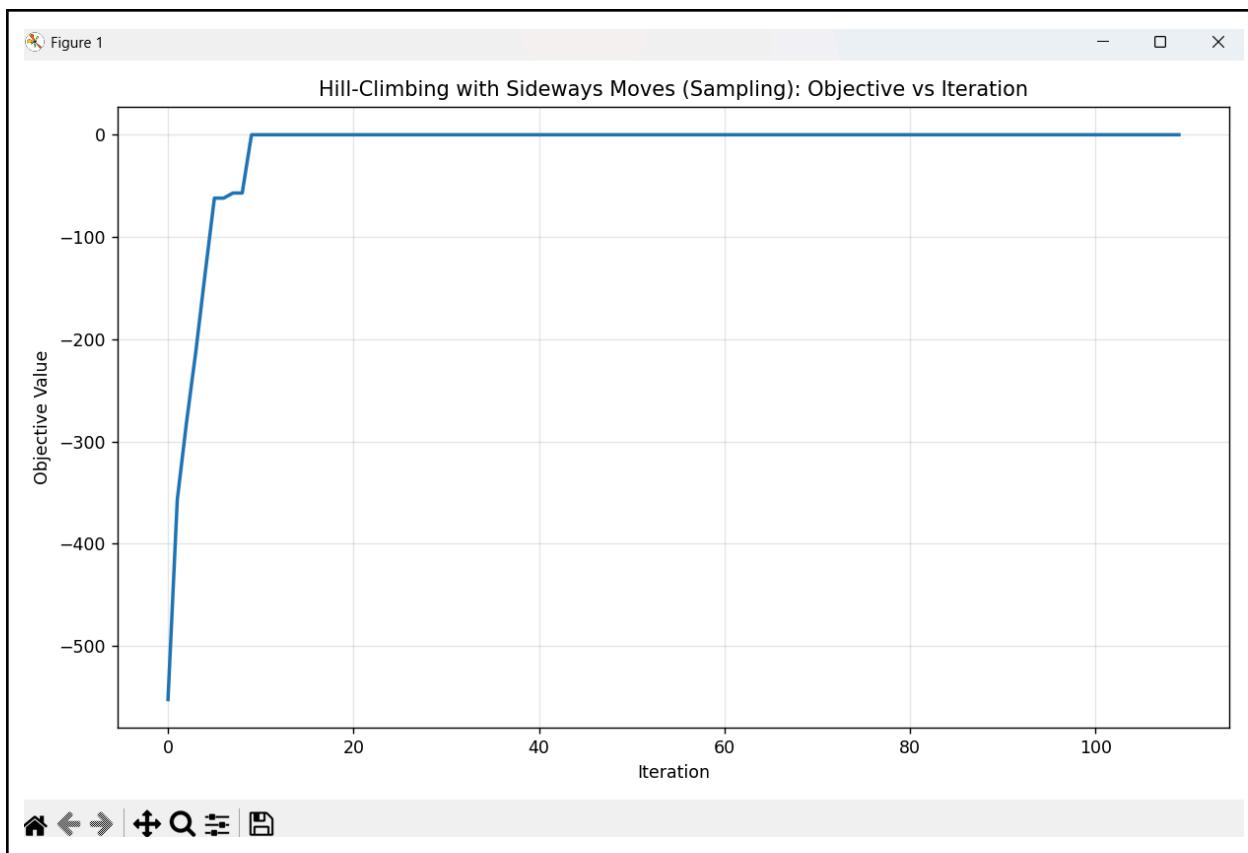
	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00		IF3002_K01				
15:00 - 16:00						
16:00 - 17:00						

Ruangan R-7608:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00		IF3008_K03				
10:00 - 11:00		IF3009_K01				
11:00 - 12:00					IF3005_K03	
12:00 - 13:00						
13:00 - 14:00	IF3005_K03					
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						

Ruangan R-7609:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00					IF3006_K03	
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00					IF3009_K01	
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						



Output penyelesaian program menggunakan
Sideways Move Full Hill-Climbing Algorithm pada terminal

Enter your choice (1-10): 5

3b. Hill-Climbing with Sideways Moves (Full)

-> Sideways-Move (Full): Optimum reached or sideways limit exceeded at iteration 108.

Final Result:

- Final objective: -0.00
- Iterations until stop: 108
- Search Duration: 1407.4437 seconds

Ruangan R-7600:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00	IF3007_K01					
09:00 - 10:00		IF3007_K01				
10:00 - 11:00	IF3001_K01					
11:00 - 12:00	IF3002_K01	IF3000_K02				
12:00 - 13:00						
13:00 - 14:00		IF3006_K03				
14:00 - 15:00	IF3006_K03					
15:00 - 16:00						
16:00 - 17:00	IF3000_K02					

Ruangan R-7601:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00						IF3002_K01
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00						IF3009_K01
15:00 - 16:00						
16:00 - 17:00						

Ruangan R-7602:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00	IF3008_K03					
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						

Ruangan R-7603:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00 IF3004_K01						
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						

Ruangan R-7604:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00				IF3006_K03		
16:00 - 17:00						

Ruangan R-7605:

	Senin	Selasa	Rabu	Kamis	Jumat
08:00 - 09:00		IF3000_K02			IF3006_K03
09:00 - 10:00					
10:00 - 11:00		IF3008_K03			
11:00 - 12:00					
12:00 - 13:00					
13:00 - 14:00				IF3003_K01	IF3008_K03
14:00 - 15:00					
15:00 - 16:00					
16:00 - 17:00					

Ruangan R-7606:

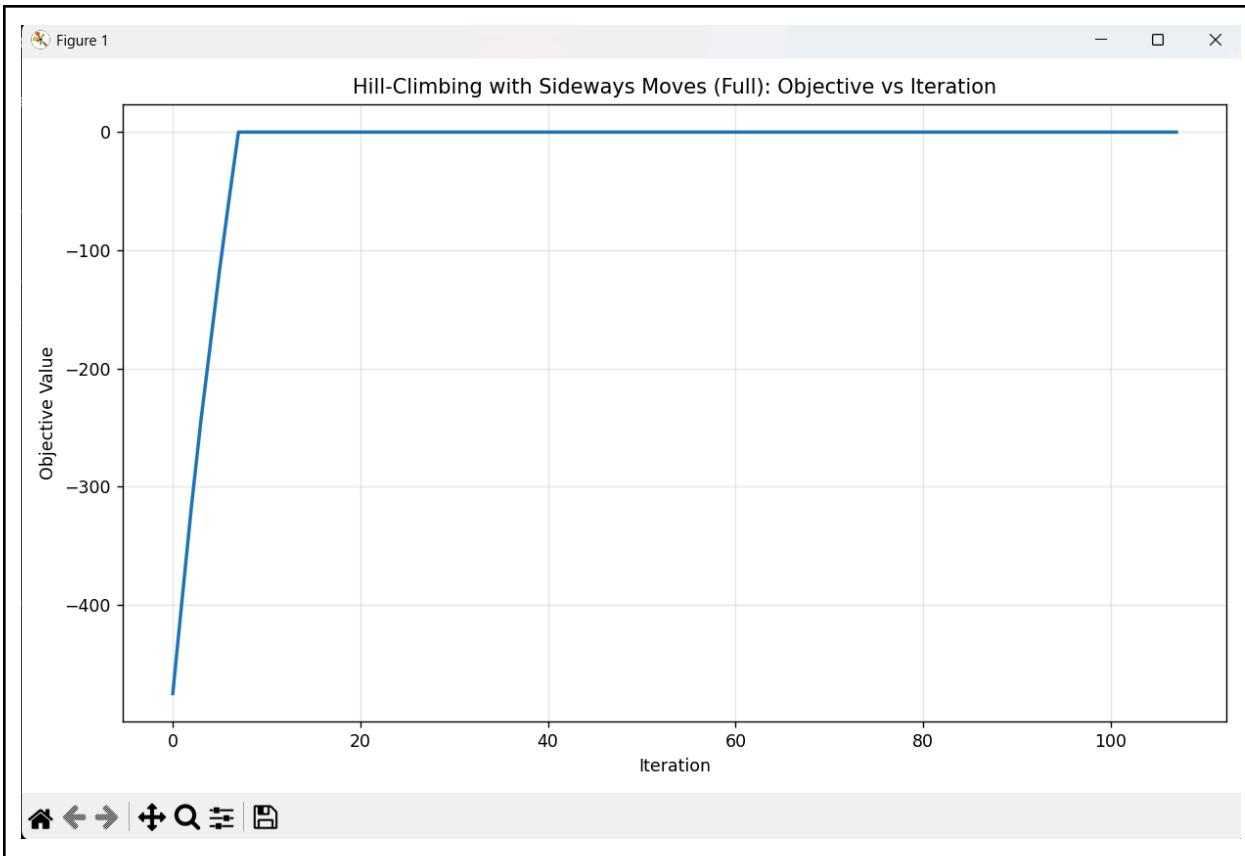
	Senin	Selasa	Rabu	Kamis	Jumat
08:00 - 09:00					
09:00 - 10:00					
10:00 - 11:00					
11:00 - 12:00					
12:00 - 13:00				IF3009_K01	
13:00 - 14:00					
14:00 - 15:00				IF3003_K01	
15:00 - 16:00					
16:00 - 17:00					

Ruangan R-7607:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						IF3000_K02
11:00 - 12:00						
12:00 - 13:00		IF3005_K03				
13:00 - 14:00						
14:00 - 15:00		IF3009_K01				
15:00 - 16:00						
16:00 - 17:00						

Ruangan R-7609:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00	IF3005_K03		IF3004_K01			
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00	IF3002_K01					IF3001_K01
16:00 - 17:00						



Output penyelesaian program menggunakan
Random Restart Hill-Climbing Algorithm pada terminal

```
Enter your choice (1-10): 6
```

```
4. Random-Restart Hill-Climbing
```

```
Starting Random-Restart Hill-Climbing with 20 restarts.
```

```
-> Restart #1/20...
-> Steepest-Ascent: Local optimum reached at iteration 4.
-> New global best found with objective: -119.00
-> Restart #2/20...
-> Steepest-Ascent: Local optimum reached at iteration 8.
-> New global best found with objective: -0.00
-> Restart #3/20...
-> Steepest-Ascent: Local optimum reached at iteration 7.
-> Restart #4/20...
-> Steepest-Ascent: Local optimum reached at iteration 10.
-> Restart #5/20...
-> Steepest-Ascent: Local optimum reached at iteration 9.
-> Restart #6/20...
-> Steepest-Ascent: Local optimum reached at iteration 4.
-> Restart #7/20...
-> Steepest-Ascent: Local optimum reached at iteration 9.
-> Restart #8/20...
-> Steepest-Ascent: Local optimum reached at iteration 6.
-> Restart #9/20...
-> Steepest-Ascent: Local optimum reached at iteration 8.
-> Restart #10/20...
-> Steepest-Ascent: Local optimum reached at iteration 6.
-> Restart #11/20...
-> Steepest-Ascent: Local optimum reached at iteration 8.
-> Restart #12/20...
-> Steepest-Ascent: Local optimum reached at iteration 10.
-> Restart #13/20...
-> Steepest-Ascent: Local optimum reached at iteration 7.
-> Restart #14/20...
```

```
-> Steepest-Ascent: Local optimum reached at iteration 10.  
-> Restart #15/20...  
-> Steepest-Ascent: Local optimum reached at iteration 10.  
-> Restart #16/20...  
-> Steepest-Ascent: Local optimum reached at iteration 9.  
-> Restart #17/20...  
-> Steepest-Ascent: Local optimum reached at iteration 8.  
-> Restart #18/20...  
-> Steepest-Ascent: Local optimum reached at iteration 7.  
-> Restart #19/20...  
-> Steepest-Ascent: Local optimum reached at iteration 11.  
-> Restart #20/20...  
-> Steepest-Ascent: Local optimum reached at iteration 7.
```

Final Result:

- Global Best objective: -0.00
- Number of Restarts: 20
- Total Iterations (sum over all restarts): 158
- Search Duration: 7.7176 seconds

Ruangan R-7600:	Senin	Selasa	Rabu	Kamis	Jumat
08:00 - 09:00					
09:00 - 10:00					
10:00 - 11:00					
11:00 - 12:00					IF3000_K02
12:00 - 13:00					
13:00 - 14:00					
14:00 - 15:00					
15:00 - 16:00					
16:00 - 17:00					IF3001_K01

Ruangan R-7601:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						IF3007_K01
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00			IF3000_K02			
14:00 - 15:00						
15:00 - 16:00				IF3008_K03		
16:00 - 17:00						

Ruangan R-7602:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						IF3009_K01
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00						IF3008_K03
16:00 - 17:00	IF3007_K01	IF3009_K01				

Ruangan R-7603:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						IF3004_K01
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00					IF3009_K01	
15:00 - 16:00						
16:00 - 17:00				IF3005_K03		

Ruangan R-7604:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00		IF3006_K03				
09:00 - 10:00	IF3008_K03					
10:00 - 11:00						
11:00 - 12:00					IF3001_K01	
12:00 - 13:00		IF3002_K01				
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						

Ruangan R-7605:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00		IF3003_K01				
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00		IF3002_K01				
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						

Ruangan R-7606:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00		IF3006_K03	IF3003_K01			
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00				IF3000_K02		
14:00 - 15:00					IF3006_K03	
15:00 - 16:00						
16:00 - 17:00						

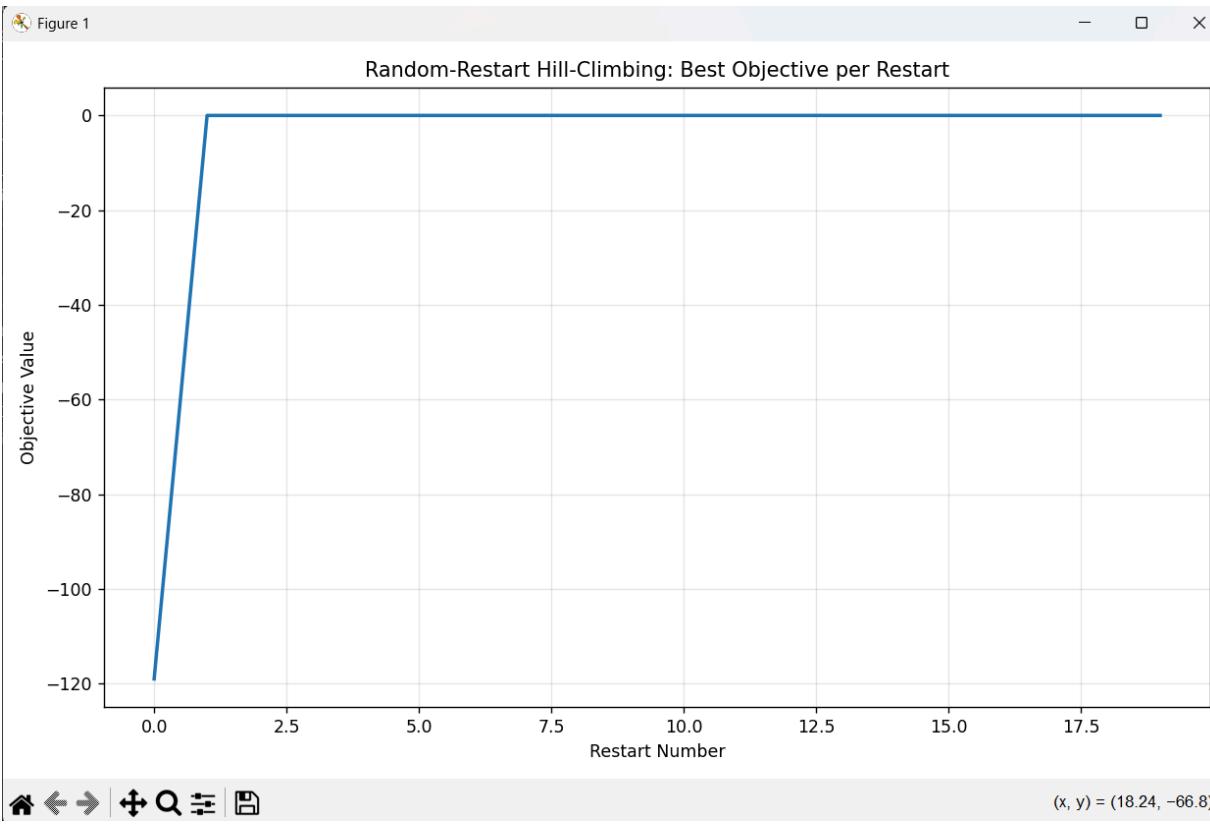
Ruangan R-7607:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00	IF3006_K03					
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00		IF3005_K03				
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						

Ruangan R-7608:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00						IF3000_K02
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00		IF3002_K01				
16:00 - 17:00						

Ruangan R-7609:	Senin	Selasa	Rabu	Kamis	Jumat
08:00 - 09:00					
09:00 - 10:00					
10:00 - 11:00					
11:00 - 12:00					
12:00 - 13:00 IF3004_K01					
13:00 - 14:00					
14:00 - 15:00					
15:00 - 16:00					
16:00 - 17:00					



Output penyelesaian program menggunakan
Genetic Algorithm pada terminal

```
Enter your choice (1-10): 7
```

5. Genetic Algorithm

Final Result:

- Final objective: -0.00
- Population Size: 100, Generations: 100
- Search Duration: 15.0603 seconds

Ruangan R-7600:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						IF3008_K03
10:00 - 11:00		IF3007_K01				
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00			IF3009_K01			
15:00 - 16:00		IF3007_K01				
16:00 - 17:00						

Ruangan R-7601:

	Senin	Selasa	Rabu	Kamis	Jumat
08:00 - 09:00			IF3001_K01		
09:00 - 10:00			IF3000_K02		
10:00 - 11:00					
11:00 - 12:00					
12:00 - 13:00					
13:00 - 14:00					
14:00 - 15:00					
15:00 - 16:00					
16:00 - 17:00					

Ruangan R-7602:

	Senin	Selasa	Rabu	Kamis	Jumat
08:00 - 09:00					
09:00 - 10:00				IF3000_K02	
10:00 - 11:00					
11:00 - 12:00					
12:00 - 13:00					
13:00 - 14:00					
14:00 - 15:00					
15:00 - 16:00	IF3003_K01				IF3002_K01
16:00 - 17:00					

Ruangan R-7603:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00						IF3008_K03
12:00 - 13:00						
13:00 - 14:00 IF3005_K03						
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						

Ruangan R-7605:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00		IF3005_K03				
10:00 - 11:00						
11:00 - 12:00			IF3004_K01			
12:00 - 13:00						IF3003_K01
13:00 - 14:00						
14:00 - 15:00				IF3001_K01		
15:00 - 16:00						
16:00 - 17:00				IF3002_K01		

Ruangan R-7606:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00	IF3008_K03	IF3006_K03				
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00	IF3000_K02					
12:00 - 13:00				IF3006_K03		
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00		IF3002_K01				

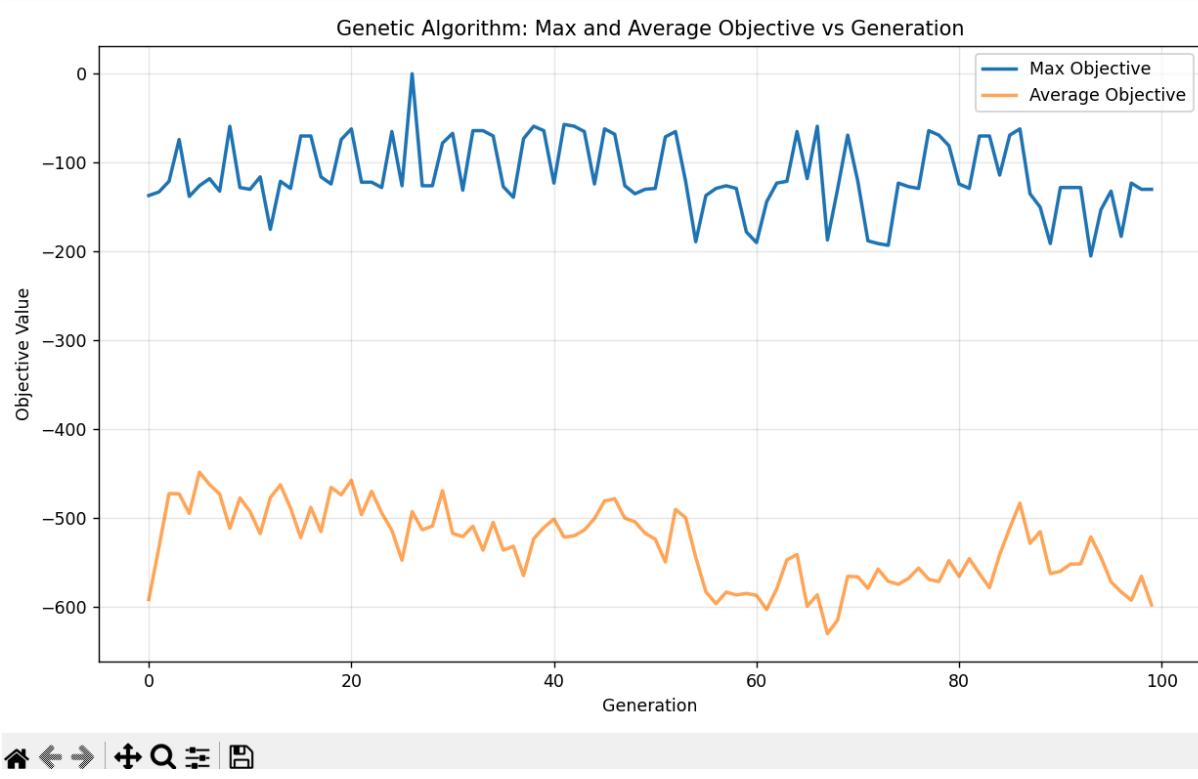
Ruangan R-7607:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00				IF3000_K02	IF3006_K03	
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						

Ruangan R-7608:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00				IF3004_K01		
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00		IF3009_K01		IF3006_K03		
14:00 - 15:00						
15:00 - 16:00				IF3009_K01		
16:00 - 17:00						

Figure 1



Output penyelesaian program menggunakan
Simulated Annealing Algorithm pada terminal

```
Enter your choice (1-10): 8
```

6. Simulated Annealing

Final Result:

- Final objective: -139.00
- Frequency of 'stuck' at local optima: 1
- Search Duration: 0.1642 seconds

Ruangan R-7600:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00	IF3006_K03					
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00						IF3004_K01
15:00 - 16:00						
16:00 - 17:00						

Ruangan R-7601:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00		IF3007_K01	IF3000_K02			
15:00 - 16:00						
16:00 - 17:00						

Ruangan R-7602:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00					IF3009_K01	
12:00 - 13:00	IF3006_K03					
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						

Ruangan R-7603:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00				IF3002_K01		
10:00 - 11:00			IF3000_K02			
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00		IF3000_K02				
16:00 - 17:00			IF3001_K01			IF3009_K01

Ruangan R-7604:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						IF3004_K01
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00				IF3008_K03		
15:00 - 16:00			IF3009_K01			IF3005_K03
16:00 - 17:00						

Ruangan R-7605:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00		IF3008_K03				
09:00 - 10:00						IF3005_K03
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						

Ruangan R-7606:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00			IF3003_K01			
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						

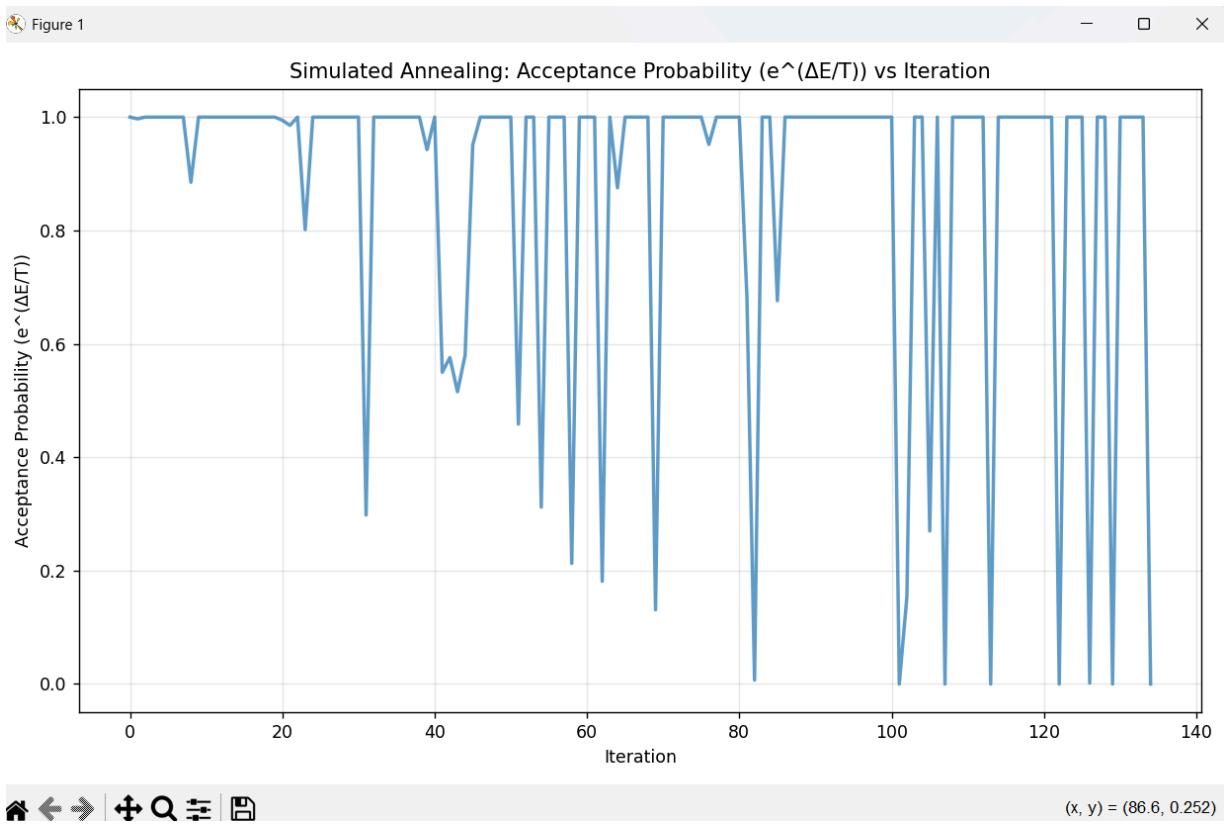
Ruangan R-7607:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						IF3003_K01
11:00 - 12:00			IF3002_K01			
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						

Ruangan R-7608:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00	IF3007_K01		IF3001_K01			
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						

Ruangan R-7609:		Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00							
09:00 - 10:00				IF3006_K03			
10:00 - 11:00							
11:00 - 12:00							
12:00 - 13:00	IF3000_K02	IF3006_K03			IF3002_K01		
13:00 - 14:00							
14:00 - 15:00							
15:00 - 16:00	IF3008_K03						
16:00 - 17:00							



3. Test Case Large (large_test.json)

Initial State						
Initial State:						
Initial Objective: -658.50						
Ruangan R-7600:						
	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						IF3011_K02
09:00 - 10:00				IF3011_K02	IF3010_K01	
10:00 - 11:00						
11:00 - 12:00				IF3011_K02		
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00						IF3001_K02
15:00 - 16:00			IF3005_K01			IF3006_K01
16:00 - 17:00	IF3002_K01	IF3027_K02				

Ruangan R-7601:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00		IF3001_K02		IF3007_K02		
		IF3023_K02				
09:00 - 10:00						
10:00 - 11:00					IF3028_K02	
11:00 - 12:00			IF3013_K01	IF3028_K02	IF3003_K02	
12:00 - 13:00				IF3022_K01		
13:00 - 14:00						
14:00 - 15:00	IF3012_K01		IF3000_K01	IF3025_K01		
15:00 - 16:00						
16:00 - 17:00	IF3027_K02					

Ruangan R-7602:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00			IF3013_K01			
10:00 - 11:00						
11:00 - 12:00						
12:00 - 13:00				IF3018_K03		
13:00 - 14:00		IF3018_K03				
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00		IF3019_K03				
		IF3025_K01				

Ruangan R-7603:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00	IF3007_K02		IF3026_K02			
09:00 - 10:00					IF3025_K01	
10:00 - 11:00					IF3014_K02	
11:00 - 12:00						
12:00 - 13:00				IF3024_K01		
13:00 - 14:00				IF3018_K03		
14:00 - 15:00				IF3023_K02	IF3019_K03	
15:00 - 16:00						
16:00 - 17:00						

Ruangan R-7604:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00	IF3017_K02					
	IF3029_K02					
09:00 - 10:00			IF3019_K03			
10:00 - 11:00		IF3028_K02				
11:00 - 12:00						
12:00 - 13:00						
13:00 - 14:00						
14:00 - 15:00			IF3008_K01			
15:00 - 16:00	IF3017_K02		IF3002_K01			
16:00 - 17:00						

Ruangan R-7605:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00		IF3024_K01	IF3014_K02	IF3015_K02		
			IF3014_K02			
			IF3019_K03			
10:00 - 11:00						
11:00 - 12:00			IF3005_K01			
12:00 - 13:00						
13:00 - 14:00				IF3018_K03	IF3027_K02	
14:00 - 15:00				IF3017_K02	IF3015_K02	
15:00 - 16:00						
16:00 - 17:00						

Ruangan R-7606:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						IF3000_K01
09:00 - 10:00				IF3013_K01		
10:00 - 11:00				IF3021_K02		
11:00 - 12:00	IF3024_K01		IF3020_K01			IF3016_K01
	IF3025_K01					IF3016_K01
12:00 - 13:00						IF3003_K02
13:00 - 14:00						
14:00 - 15:00		IF3009_K03				
15:00 - 16:00						
16:00 - 17:00						

Ruangan R-7607:						
	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00						
09:00 - 10:00		IF3024_K01	IF3021_K02			IF3009_K03
10:00 - 11:00						
11:00 - 12:00		IF3008_K01				
12:00 - 13:00						IF3010_K01
13:00 - 14:00						
14:00 - 15:00				IF3003_K02		
15:00 - 16:00		IF3007_K02	IF3020_K01			
16:00 - 17:00						

Ruangan R-7608:						
	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00	IF3006_K01					
09:00 - 10:00		IF3029_K02				
10:00 - 11:00	IF3008_K01					
11:00 - 12:00	IF3028_K02	IF3013_K01				
12:00 - 13:00				IF3010_K01		
13:00 - 14:00	IF3006_K01	IF3010_K01	IF3001_K02	IF3005_K01		
14:00 - 15:00	IF3004_K03		IF3002_K01	IF3003_K02	IF3026_K02	
				IF3011_K02		
15:00 - 16:00						
16:00 - 17:00	IF3022_K01					

Ruangan R-7609:

	Senin	Selasa	Rabu	Kamis	Jumat	
08:00 - 09:00	IF3001_K02			IF3012_K01	IF3006_K01	
09:00 - 10:00						
10:00 - 11:00						
11:00 - 12:00	IF3015_K02			IF3020_K01	IF3022_K01	
12:00 - 13:00	IF3008_K01			IF3007_K02		
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00		IF3012_K01		IF3004_K03	IF3017_K02	IF3027_K02

Tampilan pertama saat program baru dijalankan

```
~~~~~  
Please select the input file to use for this session:  
~~~~~
```

1. Standard Test (input.json)
 2. Semi-Large Test (semi_large_test.json)
 3. Large Test (large_test.json)
 4. Enter a custom file path
 5. Exit Program
- ```
~~~~~
```

```
Enter your choice (1-5): 3
```

```
-> File '../data/large_test.json' loaded successfully.
```

```
~~~~~  
Solving the Weekly Class Scheduling Problem with Local Search Algorithms
Developed by: Tubes1 - K1 (13523019, 13523059, 13523067)
~~~~~
```

1. Steepest-Ascent Hill-Climbing (Sampling)
  2. Steepest-Ascent Hill-Climbing (Full)
  3. Stochastic Hill-Climbing
  4. Hill-Climbing with Sideways Moves (Sampling)
  5. Hill-Climbing with Sideways Moves (Full)
  6. Random-Restart Hill-Climbing
  7. Genetic Algorithm
  8. Simulated Annealing
  9. Run All Algorithms Sequentially
  10. Exit
- ```
~~~~~
```

```
Enter your choice (1-10): |
```

```
Output penyelesaian program menggunakan
Steepest Ascent Sampling Hill-Climbing Algorithm pada terminal
```

Enter your choice (1-10): 1

1. Steepest-Ascent Hill-Climbing (Sampling)  
-> Steepest-Ascent: Local optimum reached at iteration 42.

Final Result:

- Final objective: -114.00
- Iterations until stop: 42
- Search Duration: 2.9310 seconds

Ruangan R-7600:

|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat      |            |
|---------------|------------|------------|------------|------------|------------|------------|
| 08:00 - 09:00 | IF3006_K01 | IF3017_K02 |            |            |            |            |
| 09:00 - 10:00 | IF3010_K01 |            | IF3028_K02 | IF3015_K02 | IF3028_K02 |            |
| 10:00 - 11:00 | IF3023_K02 |            |            |            |            |            |
| 11:00 - 12:00 |            |            | IF3011_K02 |            |            |            |
| 12:00 - 13:00 |            | IF3026_K02 |            | IF3013_K01 |            |            |
| 13:00 - 14:00 |            |            |            |            | IF3001_K02 |            |
| 14:00 - 15:00 |            | IF3027_K02 |            |            |            |            |
| 15:00 - 16:00 |            | IF3020_K01 |            |            |            |            |
| 16:00 - 17:00 |            |            |            |            |            | IF3018_K03 |

| Ruangan R-7601: |            |            |            |            |            |  |
|-----------------|------------|------------|------------|------------|------------|--|
|                 | Senin      | Selasa     | Rabu       | Kamis      | Jumat      |  |
| 08:00 - 09:00   |            | IF3010_K01 |            |            |            |  |
| 09:00 - 10:00   | IF3007_K02 | IF3017_K02 |            |            |            |  |
| 10:00 - 11:00   |            | IF3018_K03 |            |            | IF3008_K01 |  |
| 11:00 - 12:00   |            |            |            |            |            |  |
| 12:00 - 13:00   |            |            |            |            |            |  |
| 13:00 - 14:00   |            |            | IF3012_K01 |            |            |  |
| 14:00 - 15:00   |            |            | IF3003_K02 | IF3014_K02 | IF3006_K01 |  |
| 15:00 - 16:00   |            |            |            |            |            |  |
| 16:00 - 17:00   | IF3007_K02 |            |            | IF3001_K02 | IF3027_K02 |  |
| Ruangan R-7602: |            |            |            |            |            |  |
|                 | Senin      | Selasa     | Rabu       | Kamis      | Jumat      |  |
| 08:00 - 09:00   |            |            |            |            |            |  |
| 09:00 - 10:00   |            |            |            |            |            |  |
| 10:00 - 11:00   |            |            | IF3014_K02 |            |            |  |
| 11:00 - 12:00   |            |            |            |            | IF3000_K01 |  |
| 12:00 - 13:00   |            |            | IF3023_K02 | IF3009_K03 |            |  |
| 13:00 - 14:00   |            |            |            |            |            |  |
| 14:00 - 15:00   |            |            |            |            |            |  |
| 15:00 - 16:00   |            |            |            |            |            |  |
| 16:00 - 17:00   |            |            | IF3000_K01 |            |            |  |

**Ruangan R-7603:**

|               | Senin      | Selasa | Rabu       | Kamis      | Jumat |  |
|---------------|------------|--------|------------|------------|-------|--|
| 08:00 - 09:00 |            |        |            | IF3003_K02 |       |  |
| 09:00 - 10:00 | IF3024_K01 |        | IF3010_K01 |            |       |  |
| 10:00 - 11:00 |            |        |            | IF3002_K01 |       |  |
| 11:00 - 12:00 |            |        |            |            |       |  |
| 12:00 - 13:00 |            |        |            |            |       |  |
| 13:00 - 14:00 | IF3014_K02 |        |            |            |       |  |
| 14:00 - 15:00 |            |        |            |            |       |  |
| 15:00 - 16:00 | IF3003_K02 |        | IF3017_K02 |            |       |  |
| 16:00 - 17:00 |            |        |            |            |       |  |

**Ruangan R-7604:**

|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat |            |
|---------------|------------|------------|------------|------------|-------|------------|
| 08:00 - 09:00 |            | IF3008_K01 |            |            |       | IF3019_K03 |
| 09:00 - 10:00 |            |            |            |            |       |            |
| 10:00 - 11:00 |            |            |            |            |       |            |
| 11:00 - 12:00 | IF3024_K01 | IF3017_K02 |            |            |       |            |
| 12:00 - 13:00 | IF3012_K01 |            | IF3019_K03 |            |       | IF3011_K02 |
| 13:00 - 14:00 |            |            |            |            |       | IF3004_K03 |
| 14:00 - 15:00 |            |            |            | IF3029_K02 |       |            |
| 15:00 - 16:00 | IF3011_K02 |            |            |            |       |            |
| 16:00 - 17:00 |            |            |            |            |       |            |

**Ruangan R-7605:**

|               | Senin      | Selasa     | Rabu | Kamis      | Jumat |            |
|---------------|------------|------------|------|------------|-------|------------|
| 08:00 - 09:00 |            |            |      |            |       |            |
| 09:00 - 10:00 |            | IF3015_K02 |      |            |       |            |
| 10:00 - 11:00 |            |            |      |            |       |            |
| 11:00 - 12:00 |            |            |      |            |       | IF3008_K01 |
| 12:00 - 13:00 | IF3016_K01 |            |      |            |       | IF3022_K01 |
| 13:00 - 14:00 |            |            |      | IF3004_K03 |       |            |
| 14:00 - 15:00 | IF3012_K01 |            |      |            |       |            |
| 15:00 - 16:00 | IF3006_K01 | IF3007_K02 |      | IF3010_K01 |       |            |
| 16:00 - 17:00 |            |            |      |            |       |            |

**Ruangan R-7606:**

|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat |            |
|---------------|------------|------------|------------|------------|-------|------------|
| 08:00 - 09:00 | IF3026_K02 |            | IF3005_K01 | IF3022_K01 |       |            |
| 09:00 - 10:00 | IF3025_K01 |            |            | IF3028_K02 |       |            |
| 10:00 - 11:00 |            |            |            |            |       | IF3001_K02 |
| 11:00 - 12:00 |            |            | IF3018_K03 | IF3002_K01 |       |            |
| 12:00 - 13:00 |            |            |            |            |       |            |
| 13:00 - 14:00 | IF3007_K02 | IF3022_K01 |            |            |       |            |
| 14:00 - 15:00 | IF3013_K01 |            |            |            |       | IF3002_K01 |
| 15:00 - 16:00 | IF3009_K03 |            | IF3005_K01 |            |       |            |
| 16:00 - 17:00 | IF3021_K02 |            |            |            |       |            |

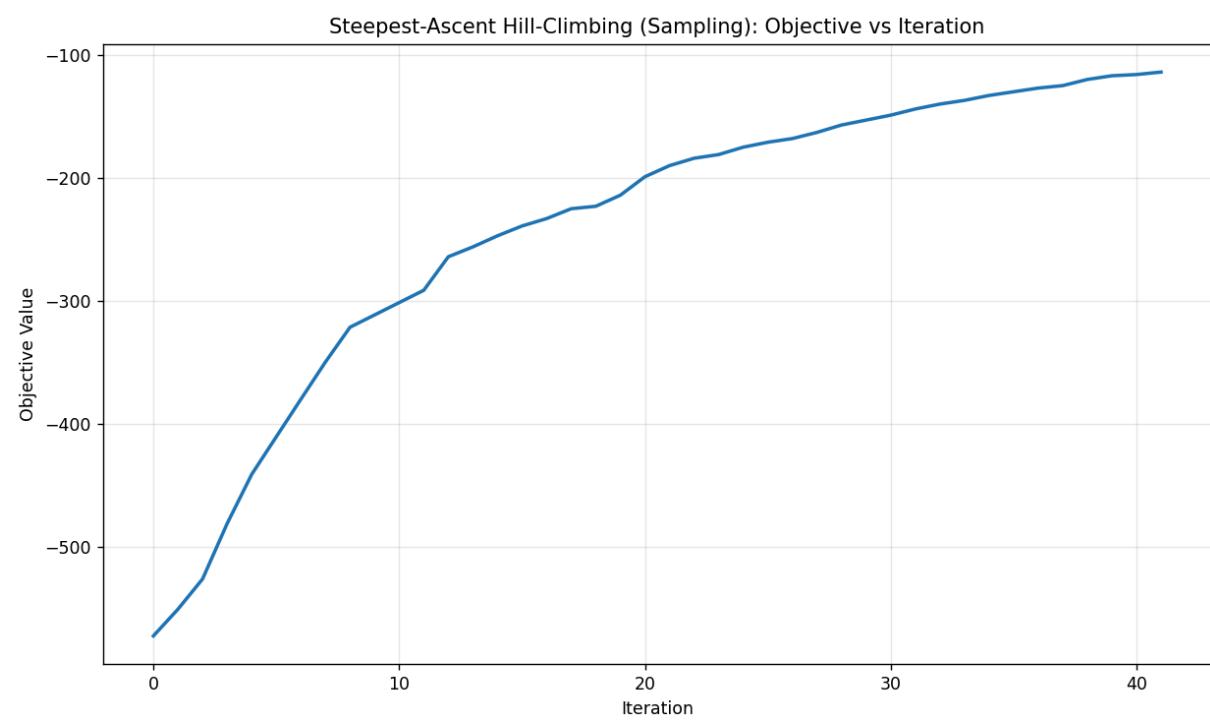
**Ruangan R-7607:**

|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat |            |
|---------------|------------|------------|------------|------------|-------|------------|
| 08:00 - 09:00 |            |            |            |            |       | IF3025_K01 |
| 09:00 - 10:00 |            |            |            |            |       |            |
| 10:00 - 11:00 | IF3008_K01 |            |            |            |       |            |
| 11:00 - 12:00 | IF3018_K03 | IF3025_K01 | IF3020_K01 |            |       |            |
| 12:00 - 13:00 |            |            |            |            |       |            |
| 13:00 - 14:00 |            |            |            |            |       | IF3027_K02 |
| 14:00 - 15:00 |            | IF3016_K01 |            | IF3005_K01 |       |            |
| 15:00 - 16:00 |            |            |            | IF3006_K01 |       |            |
| 16:00 - 17:00 |            | IF3013_K01 | IF3015_K02 | IF3029_K02 |       |            |

**Ruangan R-7608:**

|               | Senin | Selasa     | Rabu       | Kamis      | Jumat |            |
|---------------|-------|------------|------------|------------|-------|------------|
| 08:00 - 09:00 |       |            |            |            |       | IF3013_K01 |
| 09:00 - 10:00 |       |            |            |            |       | IF3025_K01 |
| 10:00 - 11:00 |       |            |            |            |       |            |
| 11:00 - 12:00 |       |            |            |            |       |            |
| 12:00 - 13:00 |       |            |            |            |       |            |
| 13:00 - 14:00 |       | IF3019_K03 | IF3027_K02 | IF3024_K01 |       |            |
| 14:00 - 15:00 |       |            |            |            |       |            |
| 15:00 - 16:00 |       |            |            |            |       |            |
| 16:00 - 17:00 |       | IF3019_K03 |            |            |       |            |

| Ruang R-7609: |            |            |            |            |       |            |
|---------------|------------|------------|------------|------------|-------|------------|
|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat |            |
| 08:00 - 09:00 |            |            |            | IF3028_K02 |       |            |
| 09:00 - 10:00 |            |            |            |            |       |            |
| 10:00 - 11:00 | IF3003_K02 |            |            |            |       |            |
| 11:00 - 12:00 |            | IF3020_K01 |            |            |       |            |
| 12:00 - 13:00 |            |            |            | IF3001_K02 |       |            |
| 13:00 - 14:00 |            |            |            |            |       |            |
| 14:00 - 15:00 |            |            |            |            |       |            |
| 15:00 - 16:00 |            | IF3011_K02 |            |            |       | IF3021_K02 |
| 16:00 - 17:00 |            |            | IF3024_K01 |            |       |            |



Output penyelesaian program menggunakan  
*Steepest Ascent Full Hill-Climbing Algorithm* pada terminal

Enter your choice (1-10): 2

1b. Steepest-Ascent Hill-climbing (Full)

-> Steepest-Ascent (Full): Local optimum reached at iteration 51.

Final Result:

- Final objective: -50.00
- Iterations until stop: 51
- Search Duration: 3547.8379 seconds

Ruangan R-7600:

|               | Senin                                                          | Selasa                  | Rabu                                 | Kamis | Jumat      |  |
|---------------|----------------------------------------------------------------|-------------------------|--------------------------------------|-------|------------|--|
| 08:00 - 09:00 | IF3029_K02   IF3013_K01   IF3003_K02   IF3013_K01   IF3008_K01 |                         |                                      |       |            |  |
| 09:00 - 10:00 | IF3015_K02   IF3017_K02   IF3020_K01                           |                         |                                      |       |            |  |
| 10:00 - 11:00 |                                                                | IF3007_K02   IF3004_K03 |                                      |       | IF3022_K01 |  |
| 11:00 - 12:00 | IF3020_K01   IF3013_K01   IF3018_K03                           |                         |                                      |       |            |  |
| 12:00 - 13:00 |                                                                |                         | IF3019_K03   IF3005_K01   IF3001_K02 |       |            |  |
| 13:00 - 14:00 | IF3025_K01                                                     |                         | IF3004_K03                           |       | IF3002_K01 |  |
| 14:00 - 15:00 | IF3008_K01                                                     |                         | IF3011_K02                           |       | IF3014_K02 |  |
| 15:00 - 16:00 | IF3011_K02                                                     |                         |                                      |       | IF3003_K02 |  |
| 16:00 - 17:00 | IF3021_K02   IF3006_K01   IF3009_K03                           |                         |                                      |       | IF3022_K01 |  |

**Ruangan R-7601:**

|               | Senin      | Selasa     | Rabu       | Kamis | Jumat |            |
|---------------|------------|------------|------------|-------|-------|------------|
| 08:00 - 09:00 | IF3015_K02 |            | IF3024_K01 |       |       |            |
| 09:00 - 10:00 |            |            | IF3008_K01 |       |       |            |
| 10:00 - 11:00 |            | IF3002_K01 |            |       |       | IF3003_K02 |
| 11:00 - 12:00 |            |            |            |       |       |            |
| 12:00 - 13:00 |            |            | IF3023_K02 |       |       |            |
| 13:00 - 14:00 | IF3015_K02 |            |            |       |       | IF3014_K02 |
| 14:00 - 15:00 | IF3010_K01 |            |            |       |       | IF3005_K01 |
| 15:00 - 16:00 |            |            |            |       |       |            |
| 16:00 - 17:00 |            |            |            |       |       | IF3025_K01 |

**Ruangan R-7602:**

|               | Senin      | Selasa     | Rabu | Kamis | Jumat |            |
|---------------|------------|------------|------|-------|-------|------------|
| 08:00 - 09:00 |            |            |      |       |       | IF3029_K02 |
| 09:00 - 10:00 | IF3006_K01 | IF3008_K01 |      |       |       |            |
| 10:00 - 11:00 | IF3014_K02 |            |      |       |       |            |
| 11:00 - 12:00 |            |            |      |       |       | IF3005_K01 |
| 12:00 - 13:00 |            | IF3011_K02 |      |       |       |            |
| 13:00 - 14:00 |            |            |      |       |       |            |
| 14:00 - 15:00 | IF3006_K01 |            |      |       |       |            |
| 15:00 - 16:00 |            |            |      |       |       |            |
| 16:00 - 17:00 |            |            |      |       |       |            |

**Ruangan R-7603:**

|               | Senin      | Selasa | Rabu       | Kamis      | Jumat      |  |
|---------------|------------|--------|------------|------------|------------|--|
| 08:00 - 09:00 |            |        |            |            |            |  |
| 09:00 - 10:00 |            |        |            | IF3024_K01 |            |  |
| 10:00 - 11:00 |            |        | IF3027_K02 |            |            |  |
| 11:00 - 12:00 |            |        |            |            |            |  |
| 12:00 - 13:00 | IF3017_K02 |        |            | IF3010_K01 | IF3003_K02 |  |
| 13:00 - 14:00 |            |        |            |            |            |  |
| 14:00 - 15:00 |            |        | IF3018_K03 | IF3010_K01 |            |  |
| 15:00 - 16:00 |            |        | IF3026_K02 |            |            |  |
| 16:00 - 17:00 |            |        |            |            |            |  |

**Ruangan R-7604:**

|               | Senin      | Selasa     | Rabu | Kamis      | Jumat      |  |
|---------------|------------|------------|------|------------|------------|--|
| 08:00 - 09:00 |            |            |      |            |            |  |
| 09:00 - 10:00 |            |            |      | IF3028_K02 | IF3006_K01 |  |
| 10:00 - 11:00 | IF3002_K01 |            |      |            |            |  |
| 11:00 - 12:00 | IF3016_K01 |            |      |            |            |  |
| 12:00 - 13:00 |            |            |      |            |            |  |
| 13:00 - 14:00 |            |            |      |            |            |  |
| 14:00 - 15:00 |            |            |      |            |            |  |
| 15:00 - 16:00 |            |            |      |            |            |  |
| 16:00 - 17:00 |            | IF3010_K01 |      | IF3017_K02 |            |  |

**Ruangan R-7605:**

|               | Senin | Selasa | Rabu | Kamis      | Jumat      |  |
|---------------|-------|--------|------|------------|------------|--|
| 08:00 - 09:00 |       |        |      | IF3025_K01 |            |  |
| 09:00 - 10:00 |       |        |      |            |            |  |
| 10:00 - 11:00 |       |        |      |            |            |  |
| 11:00 - 12:00 |       |        |      |            | IF3028_K02 |  |
| 12:00 - 13:00 |       |        |      |            |            |  |
| 13:00 - 14:00 |       |        |      |            |            |  |
| 14:00 - 15:00 |       |        |      | IF3028_K02 |            |  |
| 15:00 - 16:00 |       |        |      |            |            |  |
| 16:00 - 17:00 |       |        |      |            |            |  |

**Ruangan R-7606:**

|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat |  |
|---------------|------------|------------|------------|------------|-------|--|
| 08:00 - 09:00 |            |            |            |            |       |  |
| 09:00 - 10:00 |            |            |            |            |       |  |
| 10:00 - 11:00 | IF3007_K02 |            |            | IF3000_K01 |       |  |
| 11:00 - 12:00 |            |            |            |            |       |  |
| 12:00 - 13:00 |            |            |            |            |       |  |
| 13:00 - 14:00 |            | IF3024_K01 | IF3012_K01 |            |       |  |
| 14:00 - 15:00 |            | IF3000_K01 |            |            |       |  |
| 15:00 - 16:00 |            | IF3023_K02 |            | IF3001_K02 |       |  |
| 16:00 - 17:00 | IF3016_K01 |            | IF3012_K01 |            |       |  |

**Ruangan R-7607:**

|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat |  |
|---------------|------------|------------|------------|------------|-------|--|
| 08:00 - 09:00 |            |            | IF3022_K01 |            |       |  |
| 09:00 - 10:00 |            |            | IF3017_K02 |            |       |  |
| 10:00 - 11:00 |            |            |            | IF3027_K02 |       |  |
| 11:00 - 12:00 |            |            | IF3024_K01 |            |       |  |
| 12:00 - 13:00 | IF3001_K02 |            |            |            |       |  |
| 13:00 - 14:00 | IF3027_K02 |            |            |            |       |  |
| 14:00 - 15:00 |            |            |            |            |       |  |
| 15:00 - 16:00 |            |            |            |            |       |  |
| 16:00 - 17:00 |            | IF3009_K03 |            |            |       |  |

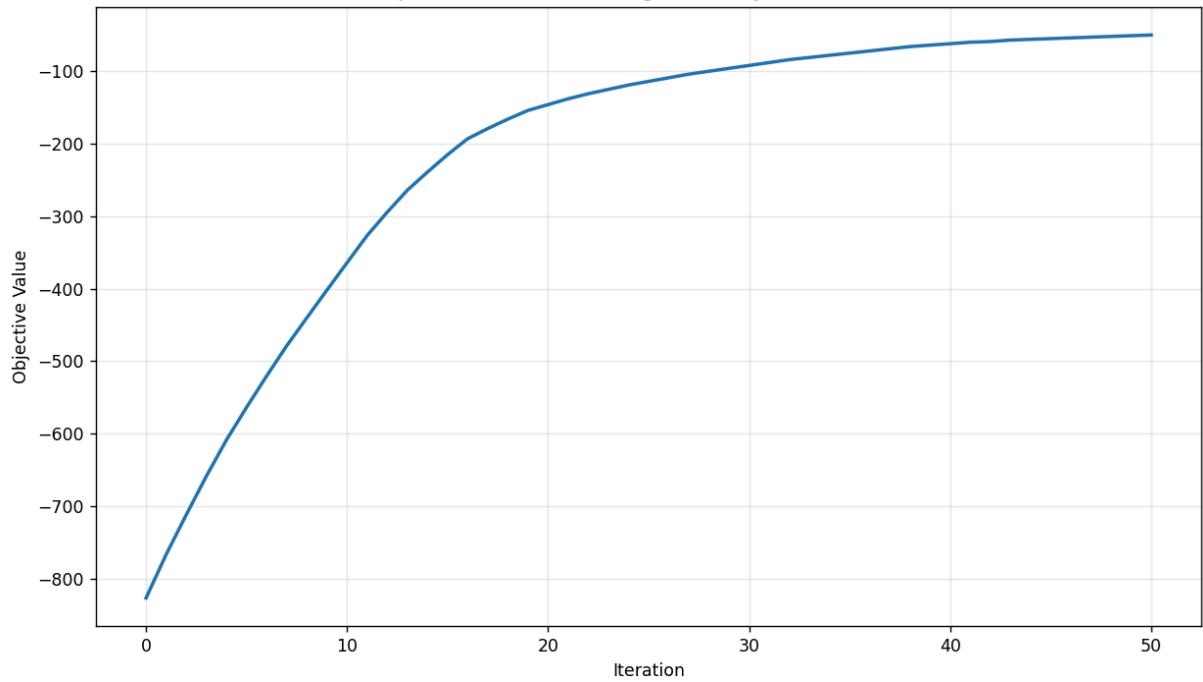
**Ruangan R-7608:**

|               | Senin | Selasa     | Rabu       | Kamis      | Jumat |            |
|---------------|-------|------------|------------|------------|-------|------------|
| 08:00 - 09:00 |       |            |            |            |       |            |
| 09:00 - 10:00 |       |            |            |            |       | IF3026_K02 |
| 10:00 - 11:00 |       |            | IF3025_K01 |            |       |            |
| 11:00 - 12:00 |       | IF3019_K03 |            | IF3018_K03 |       |            |
| 12:00 - 13:00 |       | IF3007_K02 |            |            |       |            |
| 13:00 - 14:00 |       |            |            | IF3011_K02 |       |            |
| 14:00 - 15:00 |       | IF3027_K02 |            |            |       |            |
| 15:00 - 16:00 |       | IF3019_K03 |            | IF3013_K01 |       |            |
| 16:00 - 17:00 |       |            |            | IF3012_K01 |       |            |

**Ruangan R-7609:**

|               | Senin      | Selasa     | Rabu | Kamis      | Jumat |            |
|---------------|------------|------------|------|------------|-------|------------|
| 08:00 - 09:00 |            | IF3019_K03 |      |            |       |            |
| 09:00 - 10:00 |            |            |      |            |       |            |
| 10:00 - 11:00 |            |            |      |            |       |            |
| 11:00 - 12:00 | IF3021_K02 |            |      |            |       |            |
| 12:00 - 13:00 |            |            |      |            |       |            |
| 13:00 - 14:00 |            | IF3028_K02 |      | IF3018_K03 |       |            |
| 14:00 - 15:00 |            |            |      |            |       |            |
| 15:00 - 16:00 | IF3007_K02 |            |      |            |       | IF3001_K02 |
| 16:00 - 17:00 | IF3020_K01 |            |      |            |       |            |

Steepest-Ascent Hill-Climbing (Full): Objective vs Iteration



Output penyelesaian program menggunakan  
*Stochastic Hill-Climbing Algorithm* pada terminal

Enter your choice (1-10): 3

2. Stochastic Hill-Climbing

-> Stochastic: Local optimum reached after 100 iterations without improvement.

Final Result:

- Final objective: -95.00
- Iterations until stop: 675
- Search Duration: 1.0524 seconds

Ruangan R-7600:

|               | Senin      | Selasa     | Rabu       | Kamis | Jumat      |            |
|---------------|------------|------------|------------|-------|------------|------------|
| 08:00 - 09:00 | IF3006_K01 |            |            |       |            |            |
| 09:00 - 10:00 |            |            |            |       |            |            |
| 10:00 - 11:00 |            |            | IF3012_K01 |       |            |            |
| 11:00 - 12:00 |            |            |            |       |            |            |
| 12:00 - 13:00 | IF3016_K01 |            |            |       |            |            |
| 13:00 - 14:00 |            |            |            |       |            |            |
| 14:00 - 15:00 |            | IF3007_K02 | IF3021_K02 |       | IF3008_K01 |            |
| 15:00 - 16:00 |            |            |            |       |            | IF3014_K02 |
| 16:00 - 17:00 |            |            | IF3012_K01 |       |            |            |

**Ruangan R-7601:**

|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat |  |
|---------------|------------|------------|------------|------------|-------|--|
| 08:00 - 09:00 | IF3013_K01 | IF3005_K01 |            | IF3015_K02 |       |  |
| 09:00 - 10:00 |            |            |            |            |       |  |
| 10:00 - 11:00 | IF3014_K02 |            |            |            |       |  |
| 11:00 - 12:00 |            |            | IF3022_K01 |            |       |  |
| 12:00 - 13:00 |            | IF3018_K03 |            |            |       |  |
| 13:00 - 14:00 | IF3017_K02 |            |            |            |       |  |
| 14:00 - 15:00 |            |            |            |            |       |  |
| 15:00 - 16:00 |            |            |            |            |       |  |
| 16:00 - 17:00 |            |            |            |            |       |  |

**Ruangan R-7602:**

|               | Senin      | Selasa | Rabu       | Kamis      | Jumat      |  |
|---------------|------------|--------|------------|------------|------------|--|
| 08:00 - 09:00 |            |        | IF3003_K02 |            |            |  |
| 09:00 - 10:00 |            |        | IF3010_K01 | IF3023_K02 |            |  |
| 10:00 - 11:00 |            |        |            |            | IF3014_K02 |  |
| 11:00 - 12:00 |            |        |            |            |            |  |
| 12:00 - 13:00 |            |        |            |            | IF3019_K03 |  |
| 13:00 - 14:00 |            |        |            |            | IF3022_K01 |  |
| 14:00 - 15:00 | IF3026_K02 |        | IF3020_K01 |            | IF3026_K02 |  |
| 15:00 - 16:00 |            |        |            |            | IF3027_K02 |  |
| 16:00 - 17:00 |            |        |            | IF3008_K01 | IF3025_K01 |  |

**Ruangan R-7603:**

|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat |  |
|---------------|------------|------------|------------|------------|-------|--|
| 08:00 - 09:00 | IF3008_K01 | IF3010_K01 |            | IF3027_K02 |       |  |
| 09:00 - 10:00 |            |            |            |            |       |  |
| 10:00 - 11:00 |            | IF3028_K02 |            |            |       |  |
| 11:00 - 12:00 |            |            |            | IF3001_K02 |       |  |
| 12:00 - 13:00 |            |            |            |            |       |  |
| 13:00 - 14:00 |            |            | IF3013_K01 | IF3024_K01 |       |  |
| 14:00 - 15:00 |            |            |            | IF3010_K01 |       |  |
| 15:00 - 16:00 |            |            |            |            |       |  |
| 16:00 - 17:00 | IF3017_K02 |            |            |            |       |  |

**Ruangan R-7604:**

|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat |  |
|---------------|------------|------------|------------|------------|-------|--|
| 08:00 - 09:00 |            |            |            |            |       |  |
| 09:00 - 10:00 |            |            |            |            |       |  |
| 10:00 - 11:00 |            |            |            |            |       |  |
| 11:00 - 12:00 | IF3002_K01 |            |            |            |       |  |
| 12:00 - 13:00 |            |            | IF3023_K02 | IF3027_K02 |       |  |
| 13:00 - 14:00 | IF3000_K01 |            |            |            |       |  |
| 14:00 - 15:00 |            | IF3011_K02 |            | IF3013_K01 |       |  |
| 15:00 - 16:00 | IF3006_K01 | IF3009_K03 |            |            |       |  |
| 16:00 - 17:00 |            |            |            |            |       |  |

Ruangan R-7605:

|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat      |            |
|---------------|------------|------------|------------|------------|------------|------------|
| 08:00 - 09:00 |            |            |            | IF3000_K01 | IF3025_K01 |            |
| 09:00 - 10:00 | IF3024_K01 |            |            |            |            |            |
| 10:00 - 11:00 |            |            |            | IF3004_K03 |            |            |
| 11:00 - 12:00 |            | IF3027_K02 | IF3028_K02 |            |            | IF3018_K03 |
| 12:00 - 13:00 |            |            |            |            |            |            |
| 13:00 - 14:00 |            | IF3029_K02 |            |            |            |            |
| 14:00 - 15:00 |            |            |            |            |            |            |
| 15:00 - 16:00 |            |            | IF3020_K01 | IF3011_K02 |            |            |
| 16:00 - 17:00 |            |            |            |            |            | IF3017_K02 |

Ruangan R-7606:

|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat |            |
|---------------|------------|------------|------------|------------|-------|------------|
| 08:00 - 09:00 |            |            |            |            |       |            |
| 09:00 - 10:00 | IF3015_K02 | IF3005_K01 |            |            |       | IF3015_K02 |
| 10:00 - 11:00 |            |            |            | IF3007_K02 |       |            |
| 11:00 - 12:00 |            | IF3018_K03 |            |            |       |            |
| 12:00 - 13:00 | IF3025_K01 |            | IF3019_K03 |            |       |            |
| 13:00 - 14:00 |            |            |            |            |       |            |
| 14:00 - 15:00 |            |            |            |            |       |            |
| 15:00 - 16:00 | IF3007_K02 |            |            | IF3009_K03 |       |            |
| 16:00 - 17:00 |            |            |            |            |       |            |

**Ruangan R-7607:**

|               | Senin      | Selasa     | Rabu | Kamis      | Jumat |            |
|---------------|------------|------------|------|------------|-------|------------|
| 08:00 - 09:00 |            |            |      |            |       | IF3006_K01 |
| 09:00 - 10:00 | IF3002_K01 |            |      | IF3025_K01 |       |            |
| 10:00 - 11:00 |            |            |      |            |       | IF3007_K02 |
| 11:00 - 12:00 |            |            |      |            |       |            |
| 12:00 - 13:00 |            |            |      |            |       |            |
| 13:00 - 14:00 |            | IF3008_K01 |      |            |       |            |
| 14:00 - 15:00 |            |            |      | IF3005_K01 |       |            |
| 15:00 - 16:00 |            |            |      |            |       |            |
| 16:00 - 17:00 | IF3020_K01 | IF3001_K02 |      |            |       |            |

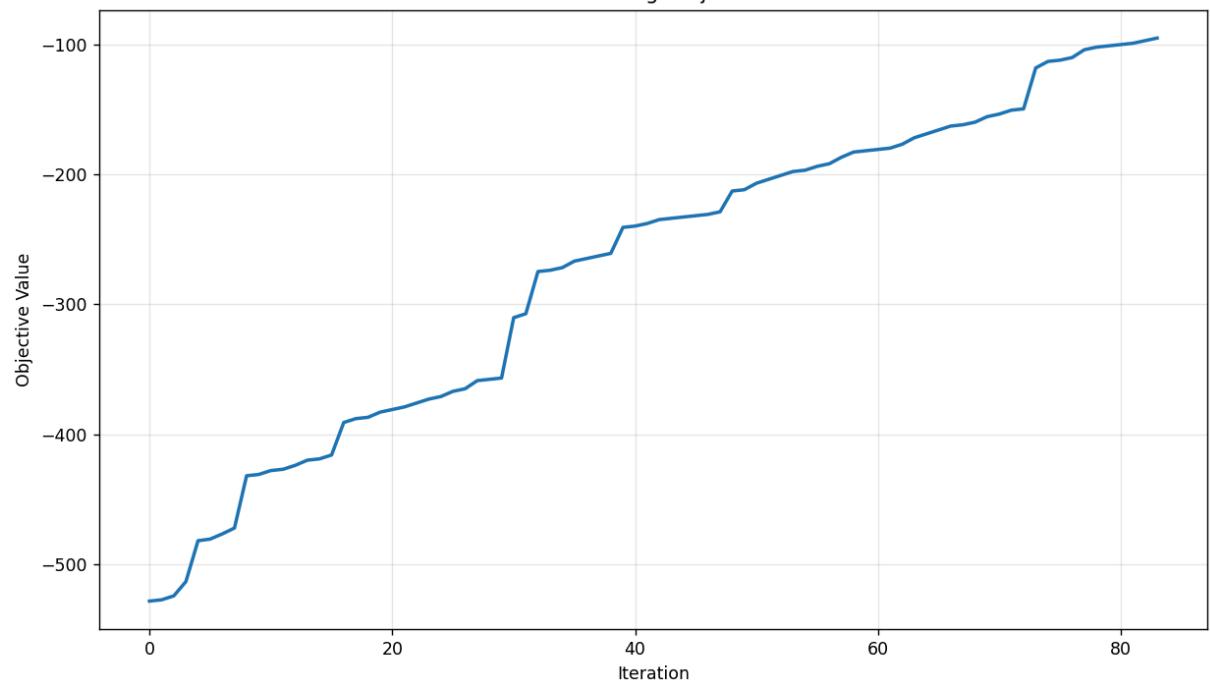
**Ruangan R-7608:**

|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat      |  |
|---------------|------------|------------|------------|------------|------------|--|
| 08:00 - 09:00 |            |            |            |            |            |  |
| 09:00 - 10:00 |            |            | IF3022_K01 | IF3010_K01 | IF3029_K02 |  |
| 10:00 - 11:00 | IF3012_K01 |            | IF3016_K01 |            |            |  |
| 11:00 - 12:00 |            |            |            | IF3003_K02 |            |  |
| 12:00 - 13:00 |            |            |            | IF3018_K03 |            |  |
| 13:00 - 14:00 |            | IF3024_K01 | IF3019_K03 | IF3003_K02 |            |  |
| 14:00 - 15:00 |            |            |            |            |            |  |
| 15:00 - 16:00 |            | IF3028_K02 |            |            |            |  |
| 16:00 - 17:00 |            |            |            |            |            |  |

Ruangan R-7609:

|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat      |
|---------------|------------|------------|------------|------------|------------|
| 08:00 - 09:00 |            |            |            |            | IF3004_K03 |
| 09:00 - 10:00 |            | IF3028_K02 |            |            |            |
| 10:00 - 11:00 |            |            |            |            |            |
| 11:00 - 12:00 |            |            |            |            |            |
| 12:00 - 13:00 | IF3013_K01 | IF3011_K02 |            |            |            |
| 13:00 - 14:00 | IF3024_K01 |            |            |            | IF3003_K02 |
| 14:00 - 15:00 | IF3021_K02 |            | IF3019_K03 |            | IF3006_K01 |
| 15:00 - 16:00 | IF3002_K01 |            | IF3001_K02 |            |            |
| 16:00 - 17:00 | IF3001_K02 | IF3017_K02 |            | IF3011_K02 |            |

Stochastic Hill-Climbing: Objective vs Iteration



Output penyelesaian program menggunakan  
*Sideways Move Sampling Hill-Climbing Algorithm* pada terminal

Enter your choice (1-10): 4

3. Hill-Climbing with Sideways Moves (Sampling)

-> Sideways-Move: Optimum reached or sideways limit exceeded at iteration 52.

Final Result:

- Final objective: -84.00
- Iterations until stop: 52
- Search Duration: 3.7359 seconds

Ruangan R-7600:

|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat      |  |
|---------------|------------|------------|------------|------------|------------|--|
| 08:00 - 09:00 |            |            |            |            |            |  |
| 09:00 - 10:00 | IF3006_K01 |            |            |            |            |  |
| 10:00 - 11:00 |            | IF3001_K02 | IF3012_K01 |            |            |  |
| 11:00 - 12:00 |            |            |            |            | IF3019_K03 |  |
| 12:00 - 13:00 |            | IF3011_K02 | IF3012_K01 |            |            |  |
| 13:00 - 14:00 |            |            |            | IF3023_K02 |            |  |
| 14:00 - 15:00 |            |            |            |            |            |  |
| 15:00 - 16:00 |            |            | IF3024_K01 | IF3028_K02 |            |  |
| 16:00 - 17:00 |            |            |            |            |            |  |

Ruangan R-7601:

|               | Senin      | Selasa     | Rabu | Kamis      | Jumat      |            |
|---------------|------------|------------|------|------------|------------|------------|
| 08:00 - 09:00 |            | IF3008_K01 |      | IF3000_K01 |            |            |
| 09:00 - 10:00 | IF3010_K01 |            |      |            |            |            |
| 10:00 - 11:00 |            |            |      |            |            | IF3013_K01 |
| 11:00 - 12:00 | IF3001_K02 |            |      |            |            |            |
| 12:00 - 13:00 |            | IF3008_K01 |      |            |            |            |
| 13:00 - 14:00 | IF3007_K02 |            |      |            |            |            |
| 14:00 - 15:00 |            |            |      | IF3014_K02 | IF3015_K02 |            |
| 15:00 - 16:00 |            |            |      | IF3015_K02 |            |            |
| 16:00 - 17:00 | IF3008_K01 |            |      |            |            |            |

Ruangan R-7602:

|               | Senin      | Selasa     | Rabu       | Kamis | Jumat |            |
|---------------|------------|------------|------------|-------|-------|------------|
| 08:00 - 09:00 |            | IF3002_K01 |            |       |       |            |
| 09:00 - 10:00 |            |            |            |       |       | IF3008_K01 |
| 10:00 - 11:00 |            |            |            |       |       |            |
| 11:00 - 12:00 |            | IF3003_K02 |            |       |       |            |
| 12:00 - 13:00 |            |            |            |       |       |            |
| 13:00 - 14:00 |            |            |            |       |       |            |
| 14:00 - 15:00 |            |            | IF3025_K01 |       |       | IF3027_K02 |
| 15:00 - 16:00 |            |            | IF3010_K01 |       |       |            |
| 16:00 - 17:00 | IF3011_K02 |            |            |       |       |            |

Ruangan R-7603:

|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat      |            |
|---------------|------------|------------|------------|------------|------------|------------|
| 08:00 - 09:00 |            |            | IF3018_K03 |            |            |            |
| 09:00 - 10:00 |            |            | IF3006_K01 |            |            |            |
| 10:00 - 11:00 |            |            |            |            |            |            |
| 11:00 - 12:00 |            |            | IF3029_K02 | IF3020_K01 |            |            |
| 12:00 - 13:00 | IF3021_K02 | IF3025_K01 |            |            | IF3003_K02 |            |
| 13:00 - 14:00 |            |            |            |            |            |            |
| 14:00 - 15:00 |            |            |            |            |            |            |
| 15:00 - 16:00 | IF3026_K02 | IF3014_K02 | IF3028_K02 |            |            |            |
| 16:00 - 17:00 |            |            |            |            |            | IF3018_K03 |

Ruangan R-7604:

|               | Senin | Selasa     | Rabu | Kamis      | Jumat |            |
|---------------|-------|------------|------|------------|-------|------------|
| 08:00 - 09:00 |       |            |      |            |       | IF3021_K02 |
| 09:00 - 10:00 |       |            |      | IF3027_K02 |       |            |
| 10:00 - 11:00 |       |            |      | IF3018_K03 |       |            |
| 11:00 - 12:00 |       |            |      |            |       |            |
| 12:00 - 13:00 |       |            |      |            |       |            |
| 13:00 - 14:00 |       | IF3019_K03 |      |            |       | IF3002_K01 |
| 14:00 - 15:00 |       |            |      |            |       | IF3016_K01 |
| 15:00 - 16:00 |       |            |      |            |       |            |
| 16:00 - 17:00 |       | IF3002_K01 |      | IF3024_K01 |       |            |

**Ruangan R-7605:**

|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat      |
|---------------|------------|------------|------------|------------|------------|
| 08:00 - 09:00 |            |            |            |            | IF3016_K01 |
| 09:00 - 10:00 |            | IF3003_K02 |            |            |            |
| 10:00 - 11:00 |            |            |            | IF3027_K02 |            |
| 11:00 - 12:00 |            | IF3022_K01 |            |            |            |
| 12:00 - 13:00 |            |            |            | IF3001_K02 |            |
| 13:00 - 14:00 |            |            | IF3000_K01 |            |            |
| 14:00 - 15:00 | IF3024_K01 |            | IF3006_K01 |            |            |
| 15:00 - 16:00 |            |            |            |            |            |
| 16:00 - 17:00 |            | IF3011_K02 | IF3023_K02 |            |            |

**Ruangan R-7606:**

|               | Senin      | Selasa     | Rabu | Kamis      | Jumat      |
|---------------|------------|------------|------|------------|------------|
| 08:00 - 09:00 | IF3009_K03 |            |      |            |            |
| 09:00 - 10:00 | IF3005_K01 |            |      |            |            |
| 10:00 - 11:00 |            |            |      |            |            |
| 11:00 - 12:00 |            |            |      | IF3017_K02 |            |
| 12:00 - 13:00 | IF3019_K03 |            |      |            | IF3026_K02 |
| 13:00 - 14:00 | IF3022_K01 | IF3009_K03 |      | IF3025_K01 |            |
| 14:00 - 15:00 |            |            |      |            |            |
| 15:00 - 16:00 |            |            |      |            | IF3027_K02 |
| 16:00 - 17:00 |            |            |      |            | IF3011_K02 |

**Ruangan R-7607:**

|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat      |            |
|---------------|------------|------------|------------|------------|------------|------------|
| 08:00 - 09:00 |            |            | IF3020_K01 | IF3020_K01 | IF3024_K01 |            |
| 09:00 - 10:00 |            |            |            |            |            |            |
| 10:00 - 11:00 | IF3028_K02 | IF3018_K03 |            |            |            | IF3005_K01 |
| 11:00 - 12:00 |            |            |            |            |            |            |
| 12:00 - 13:00 |            |            |            |            |            |            |
| 13:00 - 14:00 |            |            |            |            |            | IF3007_K02 |
| 14:00 - 15:00 | IF3028_K02 | IF3010_K01 |            | IF3005_K01 |            |            |
| 15:00 - 16:00 |            |            |            |            |            |            |
| 16:00 - 17:00 |            |            |            |            |            |            |

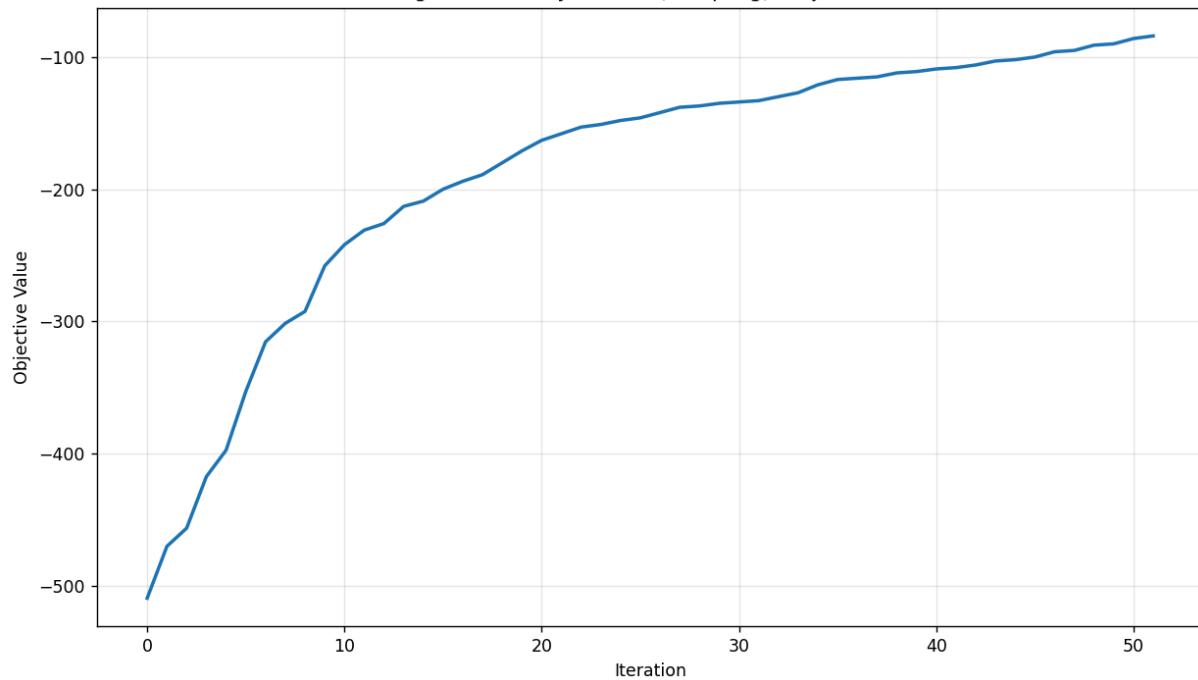
**Ruangan R-7608:**

|               | Senin | Selasa     | Rabu       | Kamis | Jumat |            |
|---------------|-------|------------|------------|-------|-------|------------|
| 08:00 - 09:00 |       |            |            |       |       |            |
| 09:00 - 10:00 |       |            | IF3007_K02 |       |       |            |
| 10:00 - 11:00 |       |            | IF3004_K03 |       |       |            |
| 11:00 - 12:00 |       |            | IF3001_K02 |       |       | IF3017_K02 |
| 12:00 - 13:00 |       |            | IF3004_K03 |       |       | IF3013_K01 |
| 13:00 - 14:00 |       |            |            |       |       | IF3014_K02 |
| 14:00 - 15:00 |       |            | IF3010_K01 |       |       |            |
| 15:00 - 16:00 |       |            |            |       |       |            |
| 16:00 - 17:00 |       | IF3007_K02 |            |       |       |            |

**Ruangan R-7609:**

|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat      |  |
|---------------|------------|------------|------------|------------|------------|--|
| 08:00 - 09:00 | IF3012_K01 |            |            |            |            |  |
| 09:00 - 10:00 |            | IF3006_K01 |            | IF3017_K02 | IF3029_K02 |  |
| 10:00 - 11:00 |            |            |            |            |            |  |
| 11:00 - 12:00 | IF3022_K01 |            |            | IF3025_K01 |            |  |
| 12:00 - 13:00 | IF3013_K01 |            |            |            |            |  |
| 13:00 - 14:00 |            |            |            |            |            |  |
| 14:00 - 15:00 |            | IF3013_K01 |            |            |            |  |
| 15:00 - 16:00 | IF3017_K02 |            |            |            | IF3015_K02 |  |
| 16:00 - 17:00 |            |            | IF3019_K03 | IF3003_K02 |            |  |

Hill-Climbing with Sideways Moves (Sampling): Objective vs Iteration



Output penyelesaian program menggunakan  
*Sideways Move Full Hill-Climbing Algorithm* pada terminal

```
Enter your choice (1-10): 5
```

### 3b. Hill-Climbing with Sideways Moves (Full)

```
-> Sideways-Move (Full): Optimum reached or sideways limit exceeded at iteration 145.
```

#### Final Result:

- Final objective: -53.00
- Iterations until stop: 145
- Search Duration: 10426.5089 seconds

Ruangan R-7600:

|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat      |
|---------------|------------|------------|------------|------------|------------|
| 08:00 - 09:00 |            | IF3001_K02 |            | IF3015_K02 | IF3014_K02 |
| 09:00 - 10:00 | IF3012_K01 | IF3015_K02 | IF3020_K01 |            | IF3004_K03 |
| 10:00 - 11:00 | IF3003_K02 | IF3013_K01 |            | IF3012_K01 |            |
| 11:00 - 12:00 |            | IF3012_K01 |            |            |            |
| 12:00 - 13:00 | IF3026_K02 |            | IF3002_K01 |            | IF3008_K01 |
| 13:00 - 14:00 | IF3016_K01 | IF3025_K01 | IF3005_K01 | IF3027_K02 |            |
| 14:00 - 15:00 | IF3017_K02 |            |            | IF3011_K02 | IF3024_K01 |
| 15:00 - 16:00 | IF3028_K02 | IF3008_K01 | IF3009_K03 |            |            |
| 16:00 - 17:00 | IF3010_K01 | IF3015_K02 |            |            |            |

**Ruangan R-7601:**

|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat      |            |
|---------------|------------|------------|------------|------------|------------|------------|
| 08:00 - 09:00 |            |            | IF3009_K03 |            |            |            |
| 09:00 - 10:00 |            |            | IF3021_K02 |            |            |            |
| 10:00 - 11:00 | IF3001_K02 |            |            |            |            | IF3019_K03 |
| 11:00 - 12:00 |            |            |            |            |            |            |
| 12:00 - 13:00 | IF3006_K01 |            | IF3007_K02 | IF3027_K02 | IF3006_K01 |            |
| 13:00 - 14:00 |            |            |            |            |            |            |
| 14:00 - 15:00 |            | IF3022_K01 | IF3018_K03 |            |            |            |
| 15:00 - 16:00 | IF3010_K01 | IF3020_K01 |            |            |            | IF3026_K02 |
| 16:00 - 17:00 |            | IF3006_K01 |            |            |            |            |

**Ruangan R-7602:**

|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat      |  |
|---------------|------------|------------|------------|------------|------------|--|
| 08:00 - 09:00 |            |            |            | IF3027_K02 |            |  |
| 09:00 - 10:00 | IF3017_K02 |            |            |            |            |  |
| 10:00 - 11:00 |            |            |            |            |            |  |
| 11:00 - 12:00 |            |            |            |            |            |  |
| 12:00 - 13:00 |            |            | IF3014_K02 |            |            |  |
| 13:00 - 14:00 |            |            |            | IF3008_K01 | IF3005_K01 |  |
| 14:00 - 15:00 |            | IF3003_K02 | IF3011_K02 |            |            |  |
| 15:00 - 16:00 |            |            |            |            |            |  |
| 16:00 - 17:00 | IF3024_K01 |            |            |            |            |  |

**Ruangan R-7603:**

|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat |            |
|---------------|------------|------------|------------|------------|-------|------------|
| 08:00 - 09:00 |            |            |            |            |       |            |
| 09:00 - 10:00 |            |            |            |            |       |            |
| 10:00 - 11:00 |            |            |            |            |       |            |
| 11:00 - 12:00 |            |            | IF3019_K03 |            |       | IF3020_K01 |
| 12:00 - 13:00 |            | IF3002_K01 |            |            |       |            |
| 13:00 - 14:00 | IF3003_K02 |            | IF3028_K02 |            |       |            |
| 14:00 - 15:00 | IF3001_K02 |            |            |            |       |            |
| 15:00 - 16:00 |            |            |            | IF3023_K02 |       |            |
| 16:00 - 17:00 |            |            |            |            |       |            |

**Ruangan R-7604:**

|               | Senin      | Selasa     | Rabu | Kamis | Jumat      |            |
|---------------|------------|------------|------|-------|------------|------------|
| 08:00 - 09:00 |            |            |      |       |            | IF3007_K02 |
| 09:00 - 10:00 |            | IF3029_K02 |      |       | IF3008_K01 |            |
| 10:00 - 11:00 |            |            |      |       |            |            |
| 11:00 - 12:00 | IF3003_K02 |            |      |       |            |            |
| 12:00 - 13:00 |            |            |      |       |            |            |
| 13:00 - 14:00 | IF3024_K01 |            |      |       |            |            |
| 14:00 - 15:00 |            |            |      |       |            | IF3017_K02 |
| 15:00 - 16:00 |            |            |      |       |            |            |
| 16:00 - 17:00 |            |            |      |       |            |            |

**Ruangan R-7605:**

|               | Senin      | Selasa | Rabu       | Kamis      | Jumat      |            |
|---------------|------------|--------|------------|------------|------------|------------|
| 08:00 - 09:00 | IF3019_K03 |        |            |            |            |            |
| 09:00 - 10:00 |            |        |            |            |            | IF3018_K03 |
| 10:00 - 11:00 |            |        |            |            |            |            |
| 11:00 - 12:00 |            |        |            | IF3028_K02 | IF3021_K02 |            |
| 12:00 - 13:00 |            |        |            |            |            |            |
| 13:00 - 14:00 |            |        |            |            |            | IF3028_K02 |
| 14:00 - 15:00 |            |        |            |            |            |            |
| 15:00 - 16:00 |            |        | IF3011_K02 | IF3019_K03 |            |            |
| 16:00 - 17:00 | IF3025_K01 |        |            |            |            | IF3014_K02 |

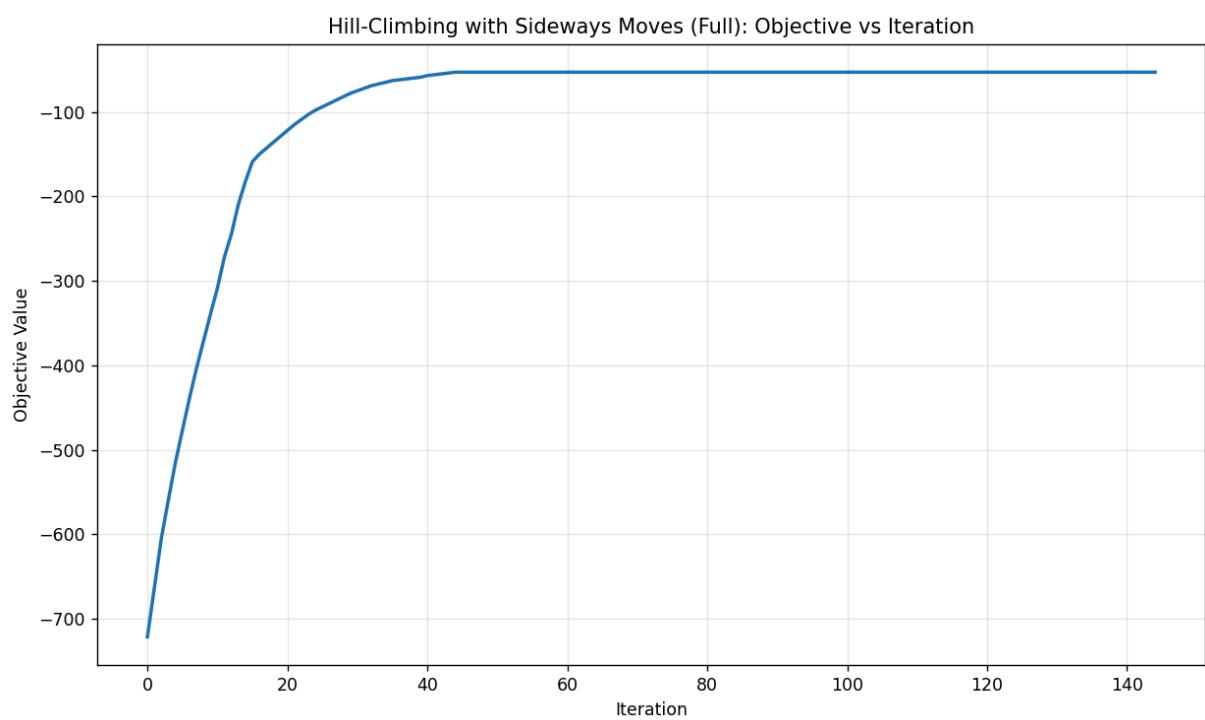
**Ruangan R-7606:**

|               | Senin | Selasa     | Rabu       | Kamis      | Jumat |  |
|---------------|-------|------------|------------|------------|-------|--|
| 08:00 - 09:00 |       |            | IF3024_K01 |            |       |  |
| 09:00 - 10:00 |       |            |            |            |       |  |
| 10:00 - 11:00 |       | IF3025_K01 | IF3011_K02 | IF3004_K03 |       |  |
| 11:00 - 12:00 |       | IF3022_K01 |            |            |       |  |
| 12:00 - 13:00 |       |            |            |            |       |  |
| 13:00 - 14:00 |       |            |            | IF3000_K01 |       |  |
| 14:00 - 15:00 |       |            |            |            |       |  |
| 15:00 - 16:00 |       |            |            |            |       |  |
| 16:00 - 17:00 |       |            | IF3027_K02 |            |       |  |

| Ruangan R-7607: |       |            |            |            |            |  |
|-----------------|-------|------------|------------|------------|------------|--|
|                 | Senin | Selasa     | Rabu       | Kamis      | Jumat      |  |
| 08:00 - 09:00   |       |            |            | IF3025_K01 |            |  |
| 09:00 - 10:00   |       |            |            |            |            |  |
| 10:00 - 11:00   |       |            | IF3007_K02 |            |            |  |
| 11:00 - 12:00   |       |            |            |            | IF3016_K01 |  |
| 12:00 - 13:00   |       |            |            |            |            |  |
| 13:00 - 14:00   |       |            |            |            |            |  |
| 14:00 - 15:00   |       |            |            |            |            |  |
| 15:00 - 16:00   |       | IF3017_K02 |            |            |            |  |
| 16:00 - 17:00   |       |            | IF3000_K01 |            |            |  |

| Ruangan R-7608: |            |            |      |            |       |            |
|-----------------|------------|------------|------|------------|-------|------------|
|                 | Senin      | Selasa     | Rabu | Kamis      | Jumat |            |
| 08:00 - 09:00   | IF3013_K01 |            |      |            |       |            |
| 09:00 - 10:00   |            |            |      | IF3029_K02 |       |            |
| 10:00 - 11:00   |            |            |      |            |       | IF3013_K01 |
| 11:00 - 12:00   | IF3001_K02 |            |      |            |       |            |
| 12:00 - 13:00   |            | IF3023_K02 |      | IF3018_K03 |       |            |
| 13:00 - 14:00   |            | IF3022_K01 |      |            |       |            |
| 14:00 - 15:00   |            |            |      | IF3007_K02 |       |            |
| 15:00 - 16:00   |            |            |      |            |       |            |
| 16:00 - 17:00   |            |            |      | IF3010_K01 |       |            |

| Ruang R-7609: |  | Senin | Selasa     | Rabu       | Kamis      | Jumat      |
|---------------|--|-------|------------|------------|------------|------------|
| 08:00 - 09:00 |  |       | IF3018_K03 |            |            | IF3002_K01 |
| 09:00 - 10:00 |  |       |            |            |            |            |
| 10:00 - 11:00 |  |       |            |            |            |            |
| 11:00 - 12:00 |  |       |            | IF3013_K01 | IF3010_K01 |            |
| 12:00 - 13:00 |  |       |            |            |            |            |
| 13:00 - 14:00 |  |       |            |            |            |            |
| 14:00 - 15:00 |  |       |            |            |            |            |
| 15:00 - 16:00 |  |       |            |            |            |            |
| 16:00 - 17:00 |  |       |            |            | IF3006_K01 | IF3005_K01 |



Output penyelesaian program menggunakan  
*Random Restart Hill-Climbing Algorithm* pada terminal

```
Enter your choice (1-10): 6
```

```
4. Random-Restart Hill-climbing
Starting Random-Restart Hill-Climbing with 20 restarts.
-> Restart #1/20...
-> Steepest-Ascent: Local optimum reached at iteration 51.
-> New global best found with objective: -88.00
-> Restart #2/20...
-> Steepest-Ascent: Local optimum reached at iteration 49.
-> Restart #3/20...
-> Steepest-Ascent: Local optimum reached at iteration 53.
-> Restart #4/20...
-> Steepest-Ascent: Local optimum reached at iteration 49.
-> Restart #5/20...
-> Steepest-Ascent: Local optimum reached at iteration 37.
-> Restart #6/20...
-> Steepest-Ascent: Local optimum reached at iteration 42.
-> Restart #7/20...
-> Steepest-Ascent: Local optimum reached at iteration 42.
-> Restart #8/20...
-> Steepest-Ascent: Local optimum reached at iteration 37.
-> Restart #9/20...
-> Steepest-Ascent: Local optimum reached at iteration 41.
-> Restart #10/20...
-> Steepest-Ascent: Local optimum reached at iteration 46.
-> Restart #11/20...
-> Steepest-Ascent: Local optimum reached at iteration 52.
-> Restart #12/20...
-> Steepest-Ascent: Local optimum reached at iteration 40.
-> Restart #13/20...
-> Steepest-Ascent: Local optimum reached at iteration 47.
-> Restart #14/20...
-> Steepest-Ascent: Local optimum reached at iteration 57.
-> Restart #15/20...
-> Steepest-Ascent: Local optimum reached at iteration 39.
-> Restart #16/20...
-> Steepest-Ascent: Local optimum reached at iteration 48.
-> Restart #17/20...
-> Steepest-Ascent: Local optimum reached at iteration 39.
-> Restart #18/20...
```

```
-> Steepest-Ascent: Local optimum reached at iteration 54.
-> Restart #19/20...
-> Steepest-Ascent: Local optimum reached at iteration 35.
-> Restart #20/20...
-> Steepest-Ascent: Local optimum reached at iteration 51.
```

Final Result:

- Global Best objective: -88.00
- Number of Restarts: 20
- Total Iterations (sum over all restarts): 909
- Search Duration: 65.2876 seconds

Ruangan R-7600:

|               | Senin      | Selasa     | Rabu | Kamis      | Jumat |  |
|---------------|------------|------------|------|------------|-------|--|
| 08:00 - 09:00 |            | IF3026_K02 |      |            |       |  |
| 09:00 - 10:00 |            |            |      | IF3008_K01 |       |  |
| 10:00 - 11:00 |            |            |      |            |       |  |
| 11:00 - 12:00 |            |            |      |            |       |  |
| 12:00 - 13:00 | IF3012_K01 |            |      | IF3011_K02 |       |  |
| 13:00 - 14:00 |            |            |      |            |       |  |
| 14:00 - 15:00 |            |            |      |            |       |  |
| 15:00 - 16:00 |            |            |      | IF3002_K01 |       |  |
| 16:00 - 17:00 |            | IF3005_K01 |      |            |       |  |

Ruangan R-7601:

|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat |            |
|---------------|------------|------------|------------|------------|-------|------------|
| 08:00 - 09:00 |            | IF3008_K01 |            |            |       |            |
| 09:00 - 10:00 |            | IF3000_K01 | IF3007_K02 |            |       | IF3001_K02 |
| 10:00 - 11:00 |            | IF3014_K02 |            |            |       |            |
| 11:00 - 12:00 |            |            |            |            |       | IF3009_K03 |
| 12:00 - 13:00 |            |            |            | IF3007_K02 |       |            |
| 13:00 - 14:00 |            | IF3001_K02 |            | IF3001_K02 |       |            |
| 14:00 - 15:00 |            |            |            | IF3013_K01 |       |            |
| 15:00 - 16:00 | IF3004_K03 |            |            |            |       |            |
| 16:00 - 17:00 |            | IF3028_K02 | IF3003_K02 | IF3011_K02 |       |            |

Ruangan R-7602:

|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat      |            |
|---------------|------------|------------|------------|------------|------------|------------|
| 08:00 - 09:00 | IF3020_K01 |            |            | IF3025_K01 | IF3027_K02 |            |
| 09:00 - 10:00 |            |            |            |            |            |            |
| 10:00 - 11:00 | IF3028_K02 |            | IF3024_K01 | IF3017_K02 | IF3022_K01 |            |
| 11:00 - 12:00 | IF3020_K01 |            |            |            |            |            |
| 12:00 - 13:00 | IF3017_K02 |            |            |            |            |            |
| 13:00 - 14:00 |            | IF3020_K01 | IF3012_K01 | IF3003_K02 |            |            |
| 14:00 - 15:00 |            | IF3018_K03 |            |            |            |            |
| 15:00 - 16:00 |            |            |            |            |            | IF3004_K03 |
| 16:00 - 17:00 |            |            | IF3005_K01 |            |            | IF3025_K01 |

Ruangan R-7603:

|               | Senin      | Selasa | Rabu       | Kamis      | Jumat      |            |
|---------------|------------|--------|------------|------------|------------|------------|
| 08:00 - 09:00 | IF3013_K01 |        |            | IF3024_K01 | IF3016_K01 |            |
| 09:00 - 10:00 |            |        |            | IF3028_K02 |            |            |
| 10:00 - 11:00 |            |        |            |            |            |            |
| 11:00 - 12:00 |            |        |            | IF3012_K01 |            |            |
| 12:00 - 13:00 |            |        |            |            |            | IF3022_K01 |
| 13:00 - 14:00 |            |        | IF3013_K01 |            |            |            |
| 14:00 - 15:00 | IF3021_K02 |        | IF3015_K02 |            |            | IF3015_K02 |
| 15:00 - 16:00 | IF3025_K01 |        |            |            |            |            |
| 16:00 - 17:00 |            |        |            |            |            | IF3006_K01 |

Ruangan R-7604:

|               | Senin      | Selasa | Rabu | Kamis      | Jumat      |            |
|---------------|------------|--------|------|------------|------------|------------|
| 08:00 - 09:00 |            |        |      | IF3010_K01 |            |            |
| 09:00 - 10:00 | IF3010_K01 |        |      |            |            |            |
| 10:00 - 11:00 |            |        |      |            |            |            |
| 11:00 - 12:00 | IF3017_K02 |        |      |            |            |            |
| 12:00 - 13:00 |            |        |      |            | IF3002_K01 |            |
| 13:00 - 14:00 | IF3023_K02 |        |      |            |            | IF3009_K03 |
| 14:00 - 15:00 |            |        |      | IF3019_K03 | IF3024_K01 |            |
| 15:00 - 16:00 |            |        |      |            |            |            |
| 16:00 - 17:00 | IF3017_K02 |        |      |            |            |            |

**Ruangan R-7605:**

|               | Senin      | Selasa     | Rabu       | Kamis | Jumat |  |
|---------------|------------|------------|------------|-------|-------|--|
| 08:00 - 09:00 | IF3001_K02 |            |            |       |       |  |
| 09:00 - 10:00 |            |            |            |       |       |  |
| 10:00 - 11:00 |            |            |            |       |       |  |
| 11:00 - 12:00 |            | IF3014_K02 |            |       |       |  |
| 12:00 - 13:00 |            |            | IF3019_K03 |       |       |  |
| 13:00 - 14:00 |            |            |            |       |       |  |
| 14:00 - 15:00 |            | IF3027_K02 |            |       |       |  |
| 15:00 - 16:00 |            | IF3010_K01 |            |       |       |  |
| 16:00 - 17:00 |            |            |            |       |       |  |

**Ruangan R-7606:**

|               | Senin | Selasa     | Rabu       | Kamis | Jumat |  |
|---------------|-------|------------|------------|-------|-------|--|
| 08:00 - 09:00 |       |            |            |       |       |  |
| 09:00 - 10:00 |       |            | IF3006_K01 |       |       |  |
| 10:00 - 11:00 |       |            |            |       |       |  |
| 11:00 - 12:00 |       |            |            |       |       |  |
| 12:00 - 13:00 |       |            |            |       |       |  |
| 13:00 - 14:00 |       |            |            |       |       |  |
| 14:00 - 15:00 |       | IF3021_K02 |            |       |       |  |
| 15:00 - 16:00 |       | IF3006_K01 | IF3029_K02 |       |       |  |
| 16:00 - 17:00 |       |            |            |       |       |  |

**Ruangan R-7607:**

|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat |            |
|---------------|------------|------------|------------|------------|-------|------------|
| 08:00 - 09:00 |            |            |            |            |       |            |
| 09:00 - 10:00 |            | IF3013_K01 |            |            |       |            |
| 10:00 - 11:00 |            |            |            |            |       |            |
| 11:00 - 12:00 |            |            | IF3011_K02 | IF3016_K01 |       |            |
| 12:00 - 13:00 |            |            |            |            |       |            |
| 13:00 - 14:00 |            |            |            |            |       | IF3019_K03 |
| 14:00 - 15:00 | IF3018_K03 |            |            |            |       | IF3028_K02 |
| 15:00 - 16:00 |            |            |            |            |       |            |
| 16:00 - 17:00 |            |            |            | IF3007_K02 |       |            |

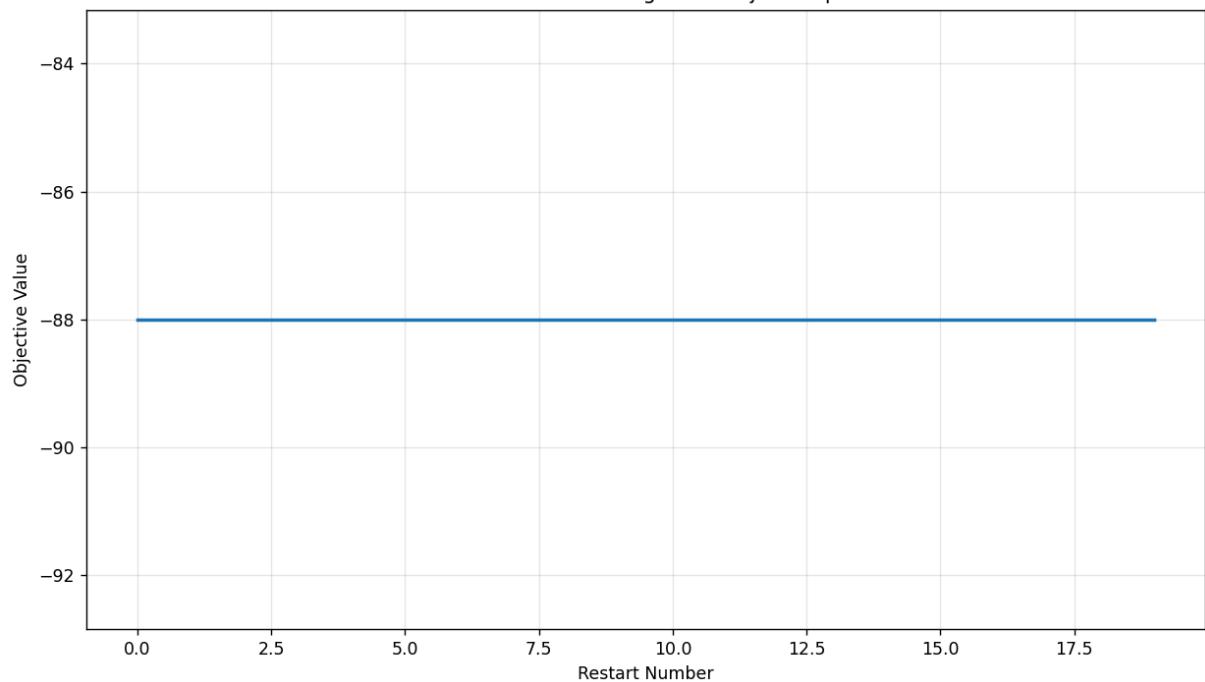
**Ruangan R-7608:**

|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat      |            |
|---------------|------------|------------|------------|------------|------------|------------|
| 08:00 - 09:00 |            |            | IF3018_K03 |            |            | IF3008_K01 |
| 09:00 - 10:00 |            |            |            |            |            |            |
| 10:00 - 11:00 | IF3005_K01 |            | IF3010_K01 | IF3014_K02 | IF3003_K02 |            |
| 11:00 - 12:00 |            |            |            |            |            | IF3006_K01 |
| 12:00 - 13:00 |            |            |            |            |            | IF3019_K03 |
| 13:00 - 14:00 |            |            |            |            |            |            |
| 14:00 - 15:00 |            |            | IF3029_K02 |            |            |            |
| 15:00 - 16:00 |            | IF3007_K02 |            |            |            | IF3027_K02 |
| 16:00 - 17:00 |            |            |            |            |            | IF3008_K01 |

**Ruangan R-7609:**

|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat      |            |
|---------------|------------|------------|------------|------------|------------|------------|
| 08:00 - 09:00 |            |            | IF3002_K01 |            |            |            |
| 09:00 - 10:00 | IF3024_K01 |            |            |            |            | IF3003_K02 |
| 10:00 - 11:00 |            |            |            |            |            |            |
| 11:00 - 12:00 |            | IF3026_K02 | IF3018_K03 |            |            |            |
| 12:00 - 13:00 |            | IF3022_K01 | IF3023_K02 |            |            |            |
| 13:00 - 14:00 | IF3000_K01 |            |            |            |            |            |
| 14:00 - 15:00 |            |            | IF3027_K02 |            |            |            |
| 15:00 - 16:00 |            |            | IF3015_K02 | IF3011_K02 | IF3025_K01 |            |
| 16:00 - 17:00 |            |            |            |            |            |            |

Random-Restart Hill-Climbing: Best Objective per Restart



Output penyelesaian program menggunakan  
*Genetic Algorithm* pada terminal

```
Enter your choice (1-10): 7
```

## 5. Genetic Algorithm

Final Result:

- Final objective: -212.00
- Population Size: 100, Generations: 100
- Search Duration: 38.1044 seconds

Ruangan R-7600:

|               | Senin | Selasa     | Rabu       | Kamis      | Jumat      |
|---------------|-------|------------|------------|------------|------------|
| 08:00 - 09:00 |       |            | IF3010_K01 |            | IF3025_K01 |
| 09:00 - 10:00 |       |            |            |            |            |
| 10:00 - 11:00 |       |            | IF3027_K02 |            | IF3000_K01 |
| 11:00 - 12:00 |       |            |            |            |            |
| 12:00 - 13:00 |       | IF3008_K01 |            |            |            |
| 13:00 - 14:00 |       |            |            | IF3015_K02 |            |
| 14:00 - 15:00 |       |            |            | IF3021_K02 |            |
| 15:00 - 16:00 |       |            |            |            | IF3018_K03 |
| 16:00 - 17:00 |       |            |            |            |            |

| Ruang R-7601: |            |            |            |            |       |            |
|---------------|------------|------------|------------|------------|-------|------------|
|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat |            |
| 08:00 - 09:00 |            |            |            |            |       |            |
| 09:00 - 10:00 | IF3013_K01 |            |            |            |       | IF3019_K03 |
| 10:00 - 11:00 |            |            |            |            |       |            |
| 11:00 - 12:00 |            |            |            |            |       |            |
| 12:00 - 13:00 |            |            |            |            |       |            |
| 13:00 - 14:00 | IF3017_K02 |            |            |            |       |            |
| 14:00 - 15:00 |            |            |            | IF3006_K01 |       |            |
| 15:00 - 16:00 |            | IF3013_K01 |            |            |       |            |
| 16:00 - 17:00 |            |            | IF3012_K01 | IF3009_K03 |       |            |

| Ruang R-7602: |            |            |            |            |            |            |
|---------------|------------|------------|------------|------------|------------|------------|
|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat      |            |
| 08:00 - 09:00 |            |            | IF3000_K01 |            |            | IF3028_K02 |
| 09:00 - 10:00 | IF3012_K01 | IF3016_K01 |            | IF3001_K02 |            |            |
| 10:00 - 11:00 |            |            |            |            |            |            |
| 11:00 - 12:00 | IF3027_K02 |            |            |            |            |            |
| 12:00 - 13:00 |            | IF3003_K02 | IF3017_K02 |            |            |            |
| 13:00 - 14:00 |            | IF3019_K03 |            | IF3003_K02 | IF3024_K01 |            |
| 14:00 - 15:00 |            |            |            |            | IF3025_K01 |            |
| 15:00 - 16:00 | IF3027_K02 |            |            | IF3008_K01 |            |            |
| 16:00 - 17:00 |            |            |            |            |            | IF3007_K02 |

| Ruangan R-7603: |            |            |      |            |            |            |
|-----------------|------------|------------|------|------------|------------|------------|
|                 | Senin      | Selasa     | Rabu | Kamis      | Jumat      |            |
| 08:00 - 09:00   | IF3022_K01 |            |      |            |            |            |
| 09:00 - 10:00   |            |            |      |            |            |            |
| 10:00 - 11:00   | IF3009_K03 |            |      |            |            | IF3027_K02 |
| 11:00 - 12:00   |            | IF3017_K02 |      |            |            | IF3010_K01 |
| 12:00 - 13:00   |            |            |      |            |            |            |
| 13:00 - 14:00   |            |            |      | IF3020_K01 | IF3025_K01 |            |
| 14:00 - 15:00   |            |            |      |            |            | IF3018_K03 |
| 15:00 - 16:00   |            |            |      |            |            | IF3011_K02 |
| 16:00 - 17:00   |            |            |      |            |            | IF3014_K02 |

| Ruangan R-7604: |            |            |      |            |       |            |
|-----------------|------------|------------|------|------------|-------|------------|
|                 | Senin      | Selasa     | Rabu | Kamis      | Jumat |            |
| 08:00 - 09:00   |            | IF3008_K01 |      | IF3023_K02 |       |            |
| 09:00 - 10:00   |            |            |      |            |       | IF3022_K01 |
| 10:00 - 11:00   |            | IF3024_K01 |      |            |       |            |
| 11:00 - 12:00   |            |            |      |            |       |            |
| 12:00 - 13:00   |            |            |      |            |       |            |
| 13:00 - 14:00   | IF3014_K02 | IF3002_K01 |      |            |       |            |
| 14:00 - 15:00   | IF3002_K01 |            |      |            |       |            |
| 15:00 - 16:00   |            |            |      |            |       |            |
| 16:00 - 17:00   |            | IF3016_K01 |      | IF3012_K01 |       |            |

Ruangan R-7605:

|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat      |  |
|---------------|------------|------------|------------|------------|------------|--|
| 08:00 - 09:00 |            |            |            | IF3010_K01 | IF3015_K02 |  |
| 09:00 - 10:00 |            |            |            | IF3018_K03 |            |  |
| 10:00 - 11:00 | IF3008_K01 | IF3019_K03 |            |            |            |  |
| 11:00 - 12:00 |            | IF3007_K02 |            | IF3026_K02 |            |  |
| 12:00 - 13:00 |            |            |            | IF3006_K01 | IF3005_K01 |  |
| 13:00 - 14:00 |            |            | IF3010_K01 |            |            |  |
| 14:00 - 15:00 |            |            |            |            |            |  |
| 15:00 - 16:00 |            | IF3013_K01 | IF3029_K02 |            |            |  |
| 16:00 - 17:00 | IF3002_K01 |            |            |            |            |  |

Ruangan R-7606:

|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat      |            |
|---------------|------------|------------|------------|------------|------------|------------|
| 08:00 - 09:00 |            |            |            |            |            | IF3001_K02 |
| 09:00 - 10:00 | IF3029_K02 |            |            |            |            |            |
| 10:00 - 11:00 | IF3011_K02 |            |            | IF3017_K02 | IF3011_K02 |            |
| 11:00 - 12:00 |            |            |            |            |            |            |
| 12:00 - 13:00 | IF3024_K01 |            |            |            |            |            |
| 13:00 - 14:00 |            |            |            |            |            |            |
| 14:00 - 15:00 |            |            |            |            |            |            |
| 15:00 - 16:00 |            |            | IF3021_K02 |            |            |            |
| 16:00 - 17:00 | IF3028_K02 | IF3007_K02 |            |            |            |            |

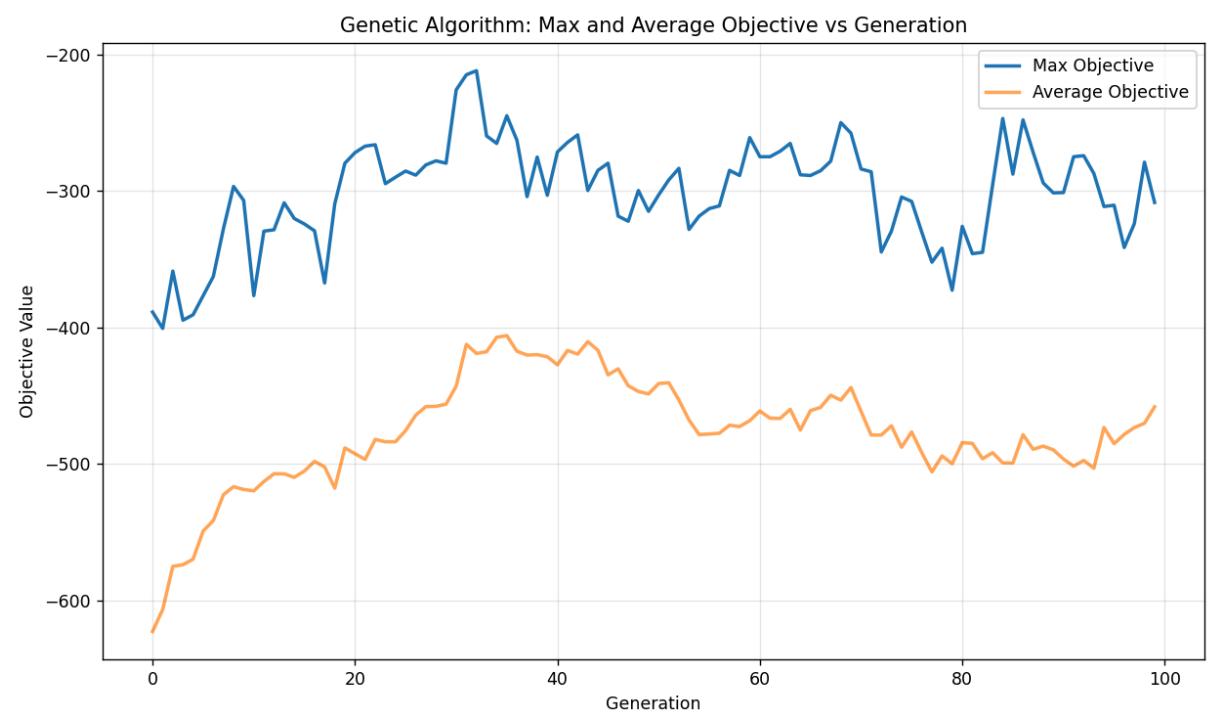
**Ruangan R-7607:**

|               | Senin | Selasa     | Rabu       | Kamis      | Jumat      |  |
|---------------|-------|------------|------------|------------|------------|--|
| 08:00 - 09:00 |       |            |            | IF3005_K01 |            |  |
| 09:00 - 10:00 |       |            |            |            |            |  |
| 10:00 - 11:00 |       |            |            |            | IF3019_K03 |  |
| 11:00 - 12:00 |       |            | IF3018_K03 |            |            |  |
| 12:00 - 13:00 |       |            |            |            |            |  |
| 13:00 - 14:00 |       |            |            |            |            |  |
| 14:00 - 15:00 |       | IF3026_K02 |            | IF3005_K01 |            |  |
| 15:00 - 16:00 |       | IF3006_K01 |            |            | IF3024_K01 |  |
| 16:00 - 17:00 |       |            |            |            |            |  |

**Ruangan R-7608:**

|               | Senin | Selasa     | Rabu       | Kamis      | Jumat      |  |
|---------------|-------|------------|------------|------------|------------|--|
| 08:00 - 09:00 |       |            | IF3006_K01 |            |            |  |
| 09:00 - 10:00 |       | IF3022_K01 | IF3001_K02 |            |            |  |
| 10:00 - 11:00 |       | IF3015_K02 |            | IF3003_K02 |            |  |
| 11:00 - 12:00 |       |            |            |            | IF3004_K03 |  |
| 12:00 - 13:00 |       |            |            |            |            |  |
| 13:00 - 14:00 |       |            |            | IF3025_K01 |            |  |
| 14:00 - 15:00 |       |            |            | IF3020_K01 |            |  |
| 15:00 - 16:00 |       |            | IF3014_K02 |            |            |  |
| 16:00 - 17:00 |       |            |            |            | IF3013_K01 |  |

| Ruangan R-7609: | Senin      | Selasa     | Rabu       | Kamis      | Jumat |
|-----------------|------------|------------|------------|------------|-------|
| 08:00 - 09:00   | IF3028_K02 |            |            | IF3020_K01 |       |
| 09:00 - 10:00   | IF3023_K02 |            |            |            |       |
| 10:00 - 11:00   |            |            |            |            |       |
| 11:00 - 12:00   | IF3001_K02 |            |            |            |       |
| 12:00 - 13:00   | IF3011_K02 |            | IF3028_K02 |            |       |
| 13:00 - 14:00   |            |            |            |            |       |
| 14:00 - 15:00   |            |            |            |            |       |
| 15:00 - 16:00   | IF3004_K03 | IF3007_K02 |            |            |       |
| 16:00 - 17:00   |            |            |            | IF3003_K02 |       |



Output penyelesaian program menggunakan  
*Simulated Algorithm* pada terminal

```
Enter your choice (1-10): 8
```

## 6. Simulated Annealing

### Final Result:

- Final objective: -391.25
- Frequency of 'stuck' at local optima: 1
- Search Duration: 0.2024 seconds

### Ruangan R-7600:

|               | Senin | Selasa     | Rabu       | Kamis | Jumat      |            |
|---------------|-------|------------|------------|-------|------------|------------|
| 08:00 - 09:00 |       | IF3002_K01 |            |       |            |            |
| 09:00 - 10:00 |       |            |            |       |            |            |
| 10:00 - 11:00 |       |            |            |       |            |            |
| 11:00 - 12:00 |       |            |            |       |            | IF3011_K02 |
| 12:00 - 13:00 |       |            |            |       |            |            |
| 13:00 - 14:00 |       |            |            |       | IF3000_K01 |            |
| 14:00 - 15:00 |       |            |            |       | IF3018_K03 |            |
| 15:00 - 16:00 |       |            |            |       |            |            |
| 16:00 - 17:00 |       |            | IF3000_K01 |       |            |            |

Ruangan R-7601:

|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat      |  |
|---------------|------------|------------|------------|------------|------------|--|
| 08:00 - 09:00 | IF3017_K02 | IF3005_K01 |            |            |            |  |
| 09:00 - 10:00 | IF3007_K02 |            |            | IF3028_K02 |            |  |
| 10:00 - 11:00 |            |            |            |            |            |  |
| 11:00 - 12:00 | IF3002_K01 |            | IF3025_K01 |            | IF3007_K02 |  |
| 12:00 - 13:00 |            | IF3020_K01 | IF3028_K02 |            |            |  |
| 13:00 - 14:00 |            |            |            |            |            |  |
| 14:00 - 15:00 |            |            | IF3020_K01 |            |            |  |
| 15:00 - 16:00 |            | IF3013_K01 |            |            |            |  |
| 16:00 - 17:00 | IF3017_K02 |            | IF3008_K01 |            |            |  |

Ruangan R-7602:

|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat      |            |
|---------------|------------|------------|------------|------------|------------|------------|
| 08:00 - 09:00 |            |            | IF3018_K03 |            |            | IF3022_K01 |
| 09:00 - 10:00 |            | IF3019_K03 |            | IF3016_K01 |            |            |
| 10:00 - 11:00 | IF3006_K01 |            |            |            |            |            |
| 11:00 - 12:00 | IF3014_K02 |            |            | IF3010_K01 | IF3021_K02 |            |
| 12:00 - 13:00 |            |            | IF3026_K02 | IF3019_K03 | IF3015_K02 | IF3028_K02 |
| 13:00 - 14:00 |            |            | IF3001_K02 |            |            |            |
|               |            |            | IF3013_K01 |            |            |            |
| 14:00 - 15:00 | IF3004_K03 |            |            |            |            |            |
| 15:00 - 16:00 |            | IF3026_K02 | IF3014_K02 |            |            |            |
| 16:00 - 17:00 |            |            | IF3008_K01 |            | IF3003_K02 |            |

**Ruangan R-7603:**

|               | Senin      | Selasa                   | Rabu       | Kamis      | Jumat      |            |
|---------------|------------|--------------------------|------------|------------|------------|------------|
| 08:00 - 09:00 |            |                          |            | IF3003_K02 |            |            |
| 09:00 - 10:00 |            | IF3005_K01<br>IF3017_K02 |            |            |            |            |
| 10:00 - 11:00 |            |                          | IF3016_K01 | IF3007_K02 |            |            |
| 11:00 - 12:00 |            |                          | IF3002_K01 |            | IF3027_K02 |            |
| 12:00 - 13:00 | IF3015_K02 | IF3025_K01               |            |            |            | IF3021_K02 |
| 13:00 - 14:00 |            |                          |            | IF3009_K03 |            |            |
| 14:00 - 15:00 | IF3010_K01 |                          |            | IF3019_K03 |            |            |
| 15:00 - 16:00 |            |                          |            |            |            |            |
| 16:00 - 17:00 |            |                          |            |            |            |            |

**Ruangan R-7604:**

|               | Senin | Selasa | Rabu       | Kamis      | Jumat      |  |
|---------------|-------|--------|------------|------------|------------|--|
| 08:00 - 09:00 |       |        |            |            |            |  |
| 09:00 - 10:00 |       |        | IF3001_K02 |            | IF3012_K01 |  |
| 10:00 - 11:00 |       |        |            |            |            |  |
| 11:00 - 12:00 |       |        | IF3011_K02 | IF3006_K01 | IF3005_K01 |  |
| 12:00 - 13:00 |       |        |            |            |            |  |
| 13:00 - 14:00 |       |        |            |            |            |  |
| 14:00 - 15:00 |       |        |            | IF3001_K02 |            |  |
| 15:00 - 16:00 |       |        |            | IF3010_K01 |            |  |
| 16:00 - 17:00 |       |        |            |            |            |  |

**Ruangan R-7605:**

|               | Senin      | Selasa     | Rabu | Kamis      | Jumat      |  |
|---------------|------------|------------|------|------------|------------|--|
| 08:00 - 09:00 |            |            |      |            |            |  |
| 09:00 - 10:00 |            |            |      | IF3011_K02 |            |  |
| 10:00 - 11:00 |            |            |      | IF3029_K02 |            |  |
| 11:00 - 12:00 |            |            |      |            | IF3004_K03 |  |
| 12:00 - 13:00 |            | IF3025_K01 |      | IF3013_K01 |            |  |
| 13:00 - 14:00 |            | IF3001_K02 |      |            |            |  |
| 14:00 - 15:00 |            |            |      |            |            |  |
| 15:00 - 16:00 | IF3003_K02 |            |      |            |            |  |
| 16:00 - 17:00 |            |            |      |            |            |  |

**Ruangan R-7606:**

|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat |  |
|---------------|------------|------------|------------|------------|-------|--|
| 08:00 - 09:00 | IF3020_K01 | IF3006_K01 | IF3025_K01 |            |       |  |
| 09:00 - 10:00 |            |            |            |            |       |  |
| 10:00 - 11:00 | IF3022_K01 |            | IF3009_K03 |            |       |  |
| 11:00 - 12:00 |            |            |            |            |       |  |
| 12:00 - 13:00 | IF3017_K02 |            |            |            |       |  |
| 13:00 - 14:00 |            |            | IF3007_K02 | IF3023_K02 |       |  |
| 14:00 - 15:00 |            | IF3022_K01 |            |            |       |  |
| 15:00 - 16:00 |            |            |            |            |       |  |
| 16:00 - 17:00 |            |            |            |            |       |  |

Ruangan R-7607:

|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat      |                          |
|---------------|------------|------------|------------|------------|------------|--------------------------|
| 08:00 - 09:00 |            | IF3011_K02 |            |            |            |                          |
| 09:00 - 10:00 |            |            |            |            |            | IF3023_K02               |
| 10:00 - 11:00 |            |            |            |            |            | IF3008_K01<br>IF3028_K02 |
| 11:00 - 12:00 |            | IF3012_K01 |            |            |            |                          |
| 12:00 - 13:00 |            |            | IF3024_K01 |            |            |                          |
| 13:00 - 14:00 |            |            |            |            |            |                          |
| 14:00 - 15:00 |            |            | IF3027_K02 | IF3018_K03 | IF3024_K01 |                          |
| 15:00 - 16:00 | IF3018_K03 |            |            |            |            |                          |
| 16:00 - 17:00 |            |            |            |            |            |                          |

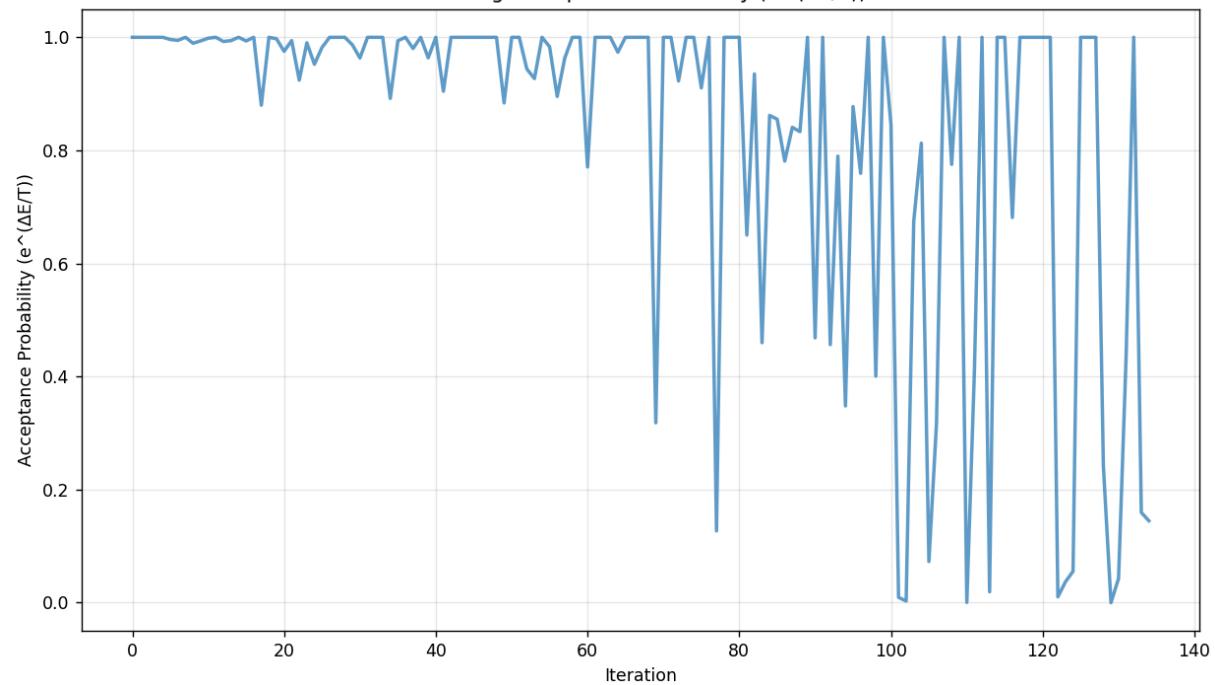
Ruangan R-7608:

|               | Senin      | Selasa     | Rabu       | Kamis      | Jumat      |  |
|---------------|------------|------------|------------|------------|------------|--|
| 08:00 - 09:00 |            |            |            |            |            |  |
| 09:00 - 10:00 |            |            |            | IF3029_K02 | IF3010_K01 |  |
| 10:00 - 11:00 | IF3012_K01 | IF3013_K01 |            |            |            |  |
| 11:00 - 12:00 |            |            |            |            |            |  |
| 12:00 - 13:00 |            |            |            |            |            |  |
| 13:00 - 14:00 |            |            |            |            |            |  |
| 14:00 - 15:00 |            |            |            |            |            |  |
| 15:00 - 16:00 |            |            |            |            | IF3019_K03 |  |
| 16:00 - 17:00 |            |            | IF3003_K02 |            | IF3008_K01 |  |

Ruangan R-7609:

|                            | Senin | Selasa     | Rabu       | Kamis      | Jumat      |  |
|----------------------------|-------|------------|------------|------------|------------|--|
| 08:00 - 09:00              |       |            | IF3015_K02 |            |            |  |
| 09:00 - 10:00              |       |            |            | IF3024_K01 |            |  |
| 10:00 - 11:00              |       |            |            |            | IF3027_K02 |  |
| 11:00 - 12:00              |       |            |            |            |            |  |
| 12:00 - 13:00   IF3027_K02 |       |            |            |            | IF3024_K01 |  |
| 13:00 - 14:00              |       |            |            |            |            |  |
| 14:00 - 15:00              |       | IF3006_K01 |            |            | IF3014_K02 |  |
| 15:00 - 16:00              |       |            |            |            |            |  |
| 16:00 - 17:00              |       |            |            |            |            |  |

Simulated Annealing: Acceptance Probability ( $e^{-(\Delta E/T)}$ ) vs Iteration



#### 2.4.2. Analisis

Pada tahap pengujian, program diuji dengan tiga jenis skenario data, yaitu test case dari spesifikasi (input.json), test case semi besar (semi\_large\_test.json), dan test case besar (large\_test.json). Tujuan dari pengujian ini adalah untuk menilai efektivitas dan efisiensi berbagai algoritma local search yang diimplementasikan dalam menemukan solusi penjadwalan yang optimal atau mendekati optimal. Algoritma yang diuji meliputi *Steepest Ascent Sampling Hill Climbing*, *Steepest Ascent Full Hill Climbing*, *Stochastic Hill Climbing*, *Sideways Move Sampling*, *Sideways Move Full*, *Random Restart Hill Climbing*, *Genetic Algorithm*, dan *Simulated Annealing*.

Pengujian awal dilakukan menggunakan dataset input.json, yang merepresentasikan skenario penjadwalan kelas mingguan berskala kecil hingga menengah. Dataset ini digunakan untuk memastikan bahwa seluruh algoritma local search berfungsi dengan benar serta mampu menghasilkan jadwal yang valid tanpa konflik besar. Pada tahap ini, ruang solusi relatif sempit dan kompleksitasnya masih dapat ditangani dengan baik oleh hampir semua algoritma yang diuji.

Hasil pengujian menunjukkan bahwa *Steepest-Ascent Hill-Climbing* (baik varian sampling maupun full) secara konsisten mampu menemukan solusi yang baik dengan jumlah konflik yang relatif sedikit. Varian full melakukan pencarian yang lebih menyeluruh di setiap iterasi, sehingga cenderung menghasilkan nilai objektif yang lebih baik, meskipun dengan waktu eksekusi yang lebih lama. Sebaliknya, varian sampling lebih efisien dari sisi waktu karena hanya mempertimbangkan sebagian tetangga, namun terkadang berhenti lebih cepat akibat terjebak di local optimum.

*Stochastic Hill-Climbing* memperlihatkan performa yang sangat efisien pada skenario ini. Dengan hanya mengevaluasi satu tetangga per iterasi, algoritma ini mampu bergerak lebih cepat dalam ruang solusi dan menunjukkan peningkatan kualitas solusi yang signifikan dalam waktu yang relatif singkat. Hasil akhirnya cukup kompetitif dibandingkan *Steepest-Ascent*, dengan waktu eksekusi jauh lebih singkat.

Varian *Hill-Climbing with Sideways Moves* menunjukkan keunggulan pada kondisi nilai objektif konstan, yaitu saat banyak tetangga memiliki nilai objektif serupa. Varian sampling dari algoritma ini bekerja dengan sangat baik, mampu menemukan solusi yang stabil dengan waktu eksekusi menengah, sementara varian full menghasilkan kualitas solusi sedikit lebih baik, tetapi dengan peningkatan waktu komputasi yang cukup signifikan. Ini menunjukkan bahwa kemampuan melakukan gerakan menyamping membantu algoritma keluar dari *local optimum* dangkal, namun biaya waktu meningkat seiring luasnya ruang pencarian.

Sementara itu, *Random-Restart Hill-Climbing* menunjukkan kemampuan eksplorasi global yang lebih baik. Dengan melakukan beberapa restart acak terhadap solusi awal, algoritma ini mampu menghindari jebakan local optimum yang sering dialami oleh metode *Hill-Climbing* biasa. Namun, untuk dataset kecil seperti input.json, keuntungan dari restart tidak terlalu signifikan, karena ruang solusi tidak terlalu kompleks sehingga solusi optimal sering kali ditemukan dengan cepat tanpa banyak percobaan ulang.

*Genetic Algorithm* pada tahap ini menghasilkan solusi yang cukup baik namun tidak menonjol. Karena ukuran populasi dan jumlah generasi masih terbatas, konvergensi belum optimal untuk ukuran dataset kecil, tetapi hasilnya tetap stabil. Sementara itu, *Simulated Annealing* tampil sebagai salah satu algoritma tercepat. Meskipun solusi akhirnya belum sebaik *Hill-Climbing* full, efisiensinya dalam menyeimbangkan eksplorasi dan eksploitasi sudah terlihat menjanjikan pada skenario sederhana ini.

Untuk pengujian dengan dataset semi\_large\_test.json, kompleksitas meningkat secara signifikan. Dataset ini berisi lebih banyak mata kuliah, ruangan, dan dosen dengan batasan yang lebih beragam, sehingga memperluas ruang solusi dan memperbanyak kemungkinan konflik. Pada tahap ini, perbedaan performa antar algoritma menjadi lebih jelas.

*Steepest-Ascent Full* kembali menonjol dalam hal kualitas solusi, menghasilkan nilai objektif yang baik dan stabil, menandakan kemampuan pencarian menyeluruhnya masih efektif pada skala menengah. Namun, waktu eksekusinya meningkat drastis karena jumlah tetangga yang perlu dievaluasi bertambah besar. *Steepest-Ascent Sampling*, sebaliknya, tetap menjadi pilihan yang efisien waktu, tetapi hasil akhirnya tidak sebaik varian full, menunjukkan kecenderungan cepat berhenti di local optimum yang dangkal.

*Stochastic Hill-Climbing* dan *Sideways Move Sampling* menunjukkan kinerja yang lebih seimbang antara kecepatan dan kualitas. Stochastic mampu menjelajahi ruang solusi dengan cepat karena sifat acaknya, sementara *Sideways Move* mampu memperbaiki solusi *Hill-Climbing* konvensional dengan keluar dari kondisi nilai objektif konstan. Kombinasi keduanya memberikan performa baik untuk dataset yang lebih besar tanpa biaya waktu yang terlalu besar.

Pada skala ini, *Random-Restart Hill-Climbing* mulai menunjukkan keunggulan nyata. Dengan melakukan pencarian berulang dari berbagai titik awal, algoritma ini lebih sering menemukan solusi yang lebih baik dibanding *Hill-Climbing* biasa. Hasilnya menunjukkan bahwa mekanisme restart efektif dalam mengatasi jebakan local optimum, meskipun waktu komputasi meningkat secara proporsional dengan jumlah restart.

*Genetic Algorithm* menunjukkan hasil yang beragam: kualitas solusinya cukup baik, namun waktu eksekusi meningkat signifikan karena kompleksitas

populasi dan proses evolusi. Hal ini menunjukkan bahwa parameter algoritma seperti ukuran populasi dan jumlah generasi perlu disesuaikan untuk mencapai keseimbangan optimal antara eksplorasi dan efisiensi waktu.

Terakhir, *Simulated Annealing* tetap menjadi algoritma dengan waktu eksekusi tercepat di antara semuanya. Meskipun kualitas solusinya belum setara dengan varian *Hill-Climbing full*, performanya stabil dan efisien, terutama ketika digunakan dengan parameter suhu awal dan laju pendinginan yang tepat.

Pengujian menggunakan `large_test.json` berisi data penjadwalan yang komprehensif, dimulai dengan 10 ruangan yang tersedia dengan kapasitas yang bervariasi antara 40 hingga 120 mahasiswa. Terdapat 30 mata kuliah yang perlu dijadwalkan, masing-masing memiliki jumlah mahasiswa yang beragam, mulai dari 21 hingga 97 orang, serta beban SKS antara 2 hingga 4. Data ini juga mencakup informasi mengenai 56 mahasiswa, di mana setiap mahasiswa mengambil antara 6 hingga 9 mata kuliah dengan tingkat prioritas yang berbeda-beda. Selain itu, terdapat 20 dosen yang tercatat, masing-masing bertanggung jawab untuk mengajar satu atau beberapa mata kuliah yang berbeda, yang menambah lapisan batasan pada penyusunan jadwal.

Algoritma Steepest-Ascent (Full) berhasil mencapai nilai objektif terbaik di antara semua pengujian, yaitu -50.00. Pencarian penuhnya yang teliti memastikan ia menemukan puncak lokal yang sangat berkualitas. Namun, kelemahannya sangat jelas: durasi pencarian yang sangat lama, mencapai 3547 detik (hampir 1 jam), meskipun hanya membutuhkan 51 iterasi untuk berhenti. Ini menunjukkan adanya *trade-off* yang ekstrim antara kualitas solusi dan waktu.

Algoritma Hill-Climbing with Sideways Moves (Full) Varian ini juga menghasilkan solusi berkualitas sangat tinggi dengan nilai objektif -53.00, sedikit di bawah Steepest-Ascent (Full). Kemampuannya untuk melakukan gerakan menyamping membuatnya berjalan lebih lama dalam hal iterasi (145 iterasi) dan waktu, dengan durasi pencarian yang luar biasa lama yaitu 10426 detik (hampir 3 jam). Ini menunjukkan bahwa algoritma menghabiskan banyak waktu untuk menjelajahi kondisi nilai objektif konstan setelah mencapai puncak awal.

Algoritma Hill-Climbing with Sideways Moves (Sampling) Dengan pendekatan sampling, algoritma ini jauh lebih cepat, hanya membutuhkan 3.7 detik untuk 52 iterasi. Hasil akhirnya adalah nilai objektif -84.00, yang lebih baik daripada varian Hill-Climbing lainnya yang menggunakan sampling. Ini menunjukkan bahwa kemampuan bergerak menyamping memberikan keuntungan bahkan dengan pencarian tetangga yang terbatas.

Algoritma Random-Restart Hill-Climbing Strategi ini menemukan solusi terbaiknya, -88.00, pada *restart* pertama dan tidak pernah menemukan solusi yang lebih baik selama 20 kali percobaan. Total durasinya adalah 65 detik untuk 909 iterasi gabungan. Hasil ini mengindikasikan bahwa lanskap solusi kemungkinan

memiliki banyak puncak lokal dengan kualitas serupa di sekitar -88.00, di mana algoritma Hill-Climbing (Sampling) yang menjadi intinya secara konsisten terjebak.

Algoritma Stochastic Hill-Climbing Algoritma ini terbukti sangat efisien. Dalam waktu hanya 1.05 detik, ia berhasil mencapai nilai objektif -95.00. Meskipun membutuhkan 675 iterasi sebelum berhenti, setiap iterasinya sangat cepat karena hanya mengevaluasi satu tetangga. Performanya dari segi kualitas lebih baik daripada Steepest-Ascent (Sampling).

Algoritma Steepest-Ascent (Sampling) Varian ini adalah salah satu yang tercepat dengan durasi 2.9 detik dan berhenti setelah 42 iterasi. Namun, kualitas solusinya adalah yang terendah di antara varian Hill-Climbing, dengan nilai objektif akhir -114.00. Ini menunjukkan bahwa dengan hanya mengambil sampel tetangga, ia terlalu cepat terjebak di puncak lokal pertama yang ditemuinya.

Genetic Algorithm Dengan durasi 38 detik, algoritma ini menghasilkan solusi yang kurang kompetitif, dengan nilai objektif akhir -212.00. Hasil ini jauh di bawah varian Hill-Climbing, yang mengindikasikan bahwa jumlah populasi (100) dan generasi (100) kemungkinan belum cukup untuk melakukan konvergensi ke solusi yang lebih baik pada dataset yang besar ini.

Algoritma Simulated Annealing adalah yang tercepat dari semua algoritma, selesai hanya dalam 0.2 detik. Namun, ia menghasilkan solusi dengan kualitas paling buruk, yaitu -391.25. Hasil ini menunjukkan bahwa parameter (suhu awal dan laju pendinginan) kemungkinan tidak diatur dengan baik, menyebabkan algoritma "mendingin" terlalu cepat sebelum sempat menjelajahi ruang solusi secara efektif.

Secara keseluruhan, varian Hill-Climbing yang menggunakan pencarian penuh (*Full*) menunjukkan performa terbaik dalam menemukan solusi berkualitas tinggi, meskipun dengan biaya komputasi yang sangat tinggi. Algoritma *greedy* seperti Hill-Climbing menunjukkan peningkatan yang sangat cepat di awal, sementara algoritma yang lebih eksploratif seperti Genetic Algorithm dan Simulated Annealing menunjukkan perilaku yang lebih kompleks dan, dengan parameter saat ini, menghasilkan solusi yang kurang optimal.

## **BAB III**

### **KESIMPULAN DAN SARAN**

#### **3.1. Kesimpulan**

Berdasarkan hasil pengujian yang telah dilakukan, dapat disimpulkan bahwa terdapat *trade-off* yang jelas antara kualitas solusi dan efisiensi waktu komputasi di antara algoritma *local search* yang diimplementasikan. Varian yang menggunakan pencarian penuh, yaitu Steepest-Ascent (Full) dan Hill-Climbing with Sideways Moves (Full), secara konsisten menghasilkan kualitas solusi terbaik dengan nilai objektif mencapai -50.00 dan -53.00 pada dataset besar. Namun, keunggulan ini harus dibayar dengan waktu eksekusi yang sangat tidak praktis, masing-masing memakan waktu hampir satu dan tiga jam. Di sisi lain, algoritma yang lebih efisien seperti Stochastic Hill-Climbing dan Hill-Climbing with Sideways Moves (Sampling) menawarkan keseimbangan yang sangat baik, mampu mencapai solusi yang kompetitif (masing-masing -95.00 dan -84.00) dalam waktu eksekusi yang sangat singkat, yaitu hanya beberapa detik. Sementara itu, algoritma yang lebih eksploratif seperti Genetic Algorithm dan Simulated Annealing menunjukkan performa yang kurang optimal pada pengujian data besar, dengan hasil akhir -212.00 dan -391.25, yang mengindikasikan bahwa parameter internalnya kemungkinan belum disesuaikan secara optimal untuk menangani kompleksitas masalah. Dengan demikian, tidak ada satu algoritma yang unggul di semua aspek; untuk kualitas solusi absolut, metode pencarian penuh adalah yang terbaik, namun untuk aplikasi praktis yang membutuhkan keseimbangan antara kecepatan dan hasil, Stochastic Hill-Climbing dan Sideways Move (Sampling) terbukti menjadi pilihan yang paling efisien dan efektif.

#### **3.2. Saran**

Berdasarkan hasil eksperimen yang telah dilakukan, terdapat beberapa saran yang dapat dipertimbangkan untuk pengembangan lebih lanjut dari sistem penjadwalan ini agar dapat menghasilkan solusi yang lebih optimal dan efisien.

1. Optimasi parameter untuk algoritma probabilistik.

Algoritma Simulated Annealing dan Genetic Algorithm menunjukkan performa yang kurang kompetitif dibandingkan dengan varian Hill-Climbing. Hal ini kemungkinan besar disebabkan oleh parameter yang belum optimal untuk dataset yang kompleks. Eksperimen lebih lanjut dengan berbagai nilai `initial_temp` dan `cooling_rate` untuk Simulated Annealing, serta `population_size`, `generations`, dan laju mutasi untuk Genetic Algorithm, berpotensi meningkatkan kualitas solusi yang dihasilkan secara signifikan.

2. Implementasi pendekatan hybrid dapat menjadi solusi untuk menyeimbangkan kualitas dan kecepatan.

Varian Hill-Climbing (Full) menghasilkan solusi terbaik namun dengan waktu eksekusi yang sangat lama, pendekatan hybrid dapat diterapkan. Sebagai contoh, proses pencarian dapat dimulai dengan algoritma yang cepat seperti Stochastic Hill-Climbing atau Simulated Annealing untuk menemukan solusi "cukup baik" dalam waktu singkat. Kemudian, solusi tersebut dapat digunakan sebagai titik awal (*initial state*) untuk algoritma yang lebih teliti seperti Steepest-Ascent (Full) yang dijalankan dalam beberapa iterasi terbatas untuk melakukan *fine-tuning*.

3. Gunakan algoritma inti yang lebih kuat atau meningkatkan jumlah *restart*.

Hal tersebut bertujuan meningkatkan efektivitas Random Restart Hill-Climbing, disarankan untuk menggunakan algoritma inti yang lebih kuat atau meningkatkan jumlah *restart*. Hasil eksperimen menunjukkan bahwa algoritma ini secara konsisten terjebak pada puncak lokal dengan kualitas yang sama. Dengan mengganti mesin pencarian internalnya dari versi *sampling* ke versi *full* (dengan `max_iter_per_restart` yang lebih kecil) atau dengan meningkatkan jumlah `num_restarts` secara drastis, kemungkinan penemuan puncak yang lebih tinggi dapat ditingkatkan.

4. Penambahan batasan pada *objective function* dapat membuat model lebih realistik.

Untuk pengembangan di masa depan, *objective function* dapat diperkaya dengan aturan penalti tambahan seperti ketersediaan jam mengajar dosen, preferensi ruangan tertentu, atau batasan lainnya yang sering ditemui dalam penyusunan jadwal di dunia nyata.

## REFERENSI

Khodra, M. L., & Maulidevi, N. U. (2025). *Modul 3: Beyond Classical Search* [Materi Kuliah]. Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung.

Russell, S. J., & Norvig, P. (2022). *Artificial intelligence: A modern approach* (4th ed.). Prentice-Hall International, Inc. Diperoleh dari <http://aima.cs.berkeley.edu/>

## LAMPIRAN

Tautan Repository Github : [https://github.com/BP04/IF3170\\_Tubes1](https://github.com/BP04/IF3170_Tubes1)

Tabel Pembagian Tugas

| No. | Pembagian Tugas                                                                                                                  | NIM      |
|-----|----------------------------------------------------------------------------------------------------------------------------------|----------|
| 1.  | Implementasi kode dan penjelasan Algoritma Hill-Climbing, laporan, eksperimen dan analisis large testing, kesimpulan dan saran   | 13523019 |
| 2.  | Implementasi kode dan penjelasan Algoritma Simulated Annealing, laporan, eksperimen dan analisis input spek, semi-large testing. | 13523059 |
| 3.  | Implementasi <i>utilities</i> , <i>objective function</i> , <i>neighbor generator</i> , dan <i>genetic algorithm</i> .           | 13523067 |