Comprehensive Report & Analysis (c-f)

c. How the system handles conflicts when jobs are put into a waiting queue, and there are jobs still entering the system. Which job goes first?

When jobs enter the waiting queue:

- Jobs that cannot be immediately allocated to memory are placed in a waiting queue (using Python's `Queue` data structure.
- Each job records its queue entry time
- The queue maintains FIFO (First In, First Out) ordering

How do we resolve conflict?

1. **Jobs already in the waiting queue have priority** over newly arriving jobs
2. When a memory block becomes free, the system:
    - First attempts to allocate jobs from the waiting queue before considering new arrivals
    - Processes queued jobs in the order they entered the queue
    - Only if a queued job cannot fit does it remain in the queue and the next queued job is checked

Which job goes first:

- Jobs in the waiting queue are served before new incoming jobs
- Among waiting jobs: the job that entered the queue earliest gets priority (FCFS)
- If multiple jobs in the queue can fit in a freed block, the first one in the queue is selected
- New jobs entering the system only get allocated if no waiting jobs can fit the available memory

This prevents **starvation** and ensures fairness - a job that has been waiting will not be bypassed by newer arrivals.

```
================================================================================
                              RUNNING BEST-FIT
================================================================================

[t=0.0] Job 1 allocated to Block 2 (waste=1240, wait=0.0)
[t=0.0] Job 2 allocated to Block 3 (waste=310, wait=0.0)
[t=0.0] Job 3 allocated to Block 8 (waste=2210, wait=0.0)
[t=0.0] Job 4 allocated to Block 5 (waste=970, wait=0.0)
[t=0.0] Job 5 allocated to Block 4 (waste=5950, wait=0.0)
[t=0.0] Job 6 allocated to Block 6 (waste=2010, wait=0.0)
[t=0.0] Job 7 allocated to Block 1 (waste=560, wait=0.0)
[t=0.0] Job 8 allocated to Block 7 (waste=260, wait=0.0)
[t=0.0] Job 9 queued (queue size: 1)
[t=0.0] Job 10 queued (queue size: 2)
[t=0.0] Job 11 queued (queue size: 3)
[t=0.0] Job 12 queued (queue size: 4)
[t=0.0] Job 13 queued (queue size: 5)
[t=0.0] Job 14 allocated to Block 10 (waste=80, wait=0.0)
[t=0.0] Job 15 allocated to Block 9 (waste=1280, wait=0.0)
[t=0.0] Job 16 queued (queue size: 6)
[t=0.0] Job 17 queued (queue size: 7)
[t=0.0] Job 18 queued (queue size: 8)
[t=0.0] Job 19 queued (queue size: 9)
[t=0.0] Job 20 queued (queue size: 10)
[t=0.0] Job 21 queued (queue size: 11)
[t=0.0] Job 22 queued (queue size: 12)
[t=0.0] Job 23 queued (queue size: 13)
[t=0.0] Job 24 queued (queue size: 14)
[t=0.0] Job 25 queued (queue size: 15)
[t=2.0] Job 4 completed. Block 5 freed.
[t=2.0] Job 18 allocated to Block 5 (waste=1620, wait=2.0)
[t=2.0] Job 5 completed. Block 4 freed.
[t=2.0] Job 9 allocated to Block 4 (waste=4570, wait=2.0)
[t=3.0] Job 18 completed. Block 5 freed.
[t=3.0] Job 22 allocated to Block 5 (waste=290, wait=3.0)
[t=4.0] Job 2 completed. Block 3 freed.
[t=4.0] Job 12 allocated to Block 3 (waste=680, wait=4.0)
[t=5.0] Job 1 completed. Block 2 freed.
[t=5.0] Job 10 allocated to Block 2 (waste=110, wait=5.0)
[t=5.0] Job 22 completed. Block 5 freed.
[t=5.0] Job 25 allocated to Block 5 (waste=2240, wait=5.0)
[t=6.0] Job 6 completed. Block 6 freed.
[t=6.0] Job 11 allocated to Block 6 (waste=2420, wait=6.0)
[t=8.0] Job 3 completed. Block 8 freed.
[t=8.0] Job 17 allocated to Block 8 (waste=2290, wait=8.0)
[t=8.0] Job 7 completed. Block 1 freed.
[t=8.0] Job 13 allocated to Block 1 (waste=360, wait=8.0)
[t=9.0] Job 9 completed. Block 4 freed.
[t=9.0] Job 16 allocated to Block 4 (waste=960, wait=9.0)
[t=10.0] Job 8 completed. Block 7 freed.
[t=10.0] Job 14 completed. Block 10 freed.
[t=10.0] Job 15 completed. Block 9 freed.
[t=11.0] Job 10 completed. Block 2 freed.
[t=11.0] Job 20 allocated to Block 2 (waste=3390, wait=11.0)
[t=11.0] Job 11 completed. Block 6 freed.
[t=11.0] Job 21 allocated to Block 6 (waste=1460, wait=11.0)
[t=11.0] Job 17 completed. Block 8 freed.
[t=12.0] Job 12 completed. Block 3 freed.
[t=14.0] Job 20 completed. Block 2 freed.
[t=14.0] Job 24 allocated to Block 2 (waste=1050, wait=14.0)
[t=15.0] Job 25 completed. Block 5 freed.
[t=16.0] Job 16 completed. Block 4 freed.
[t=16.0] Job 23 allocated to Block 4 (waste=110, wait=16.0)
[t=17.0] Job 13 completed. Block 1 freed.
[t=18.0] Job 21 completed. Block 6 freed.
[t=19.0] Job 24 completed. Block 2 freed.
[t=24.0] Job 23 completed. Block 4 freed.

WARNING: 1 jobs remain in queue (could not be allocated)

============================================================
Simulation completed at t=24.0
============================================================
```

```
================================================================================
                    FIXED PARTITION MEMORY ALLOCATION SIMULATION
================================================================================


================================================================================
                              RUNNING FIRST-FIT
================================================================================

[t=0.0] Job 1 allocated to Block 1 (waste=3740, wait=0.0)
[t=0.0] Job 2 allocated to Block 2 (waste=2810, wait=0.0)
[t=0.0] Job 3 allocated to Block 3 (waste=1210, wait=0.0)
[t=0.0] Job 4 allocated to Block 4 (waste=6470, wait=0.0)
[t=0.0] Job 5 allocated to Block 5 (waste=450, wait=0.0)
[t=0.0] Job 6 allocated to Block 6 (waste=2010, wait=0.0)
[t=0.0] Job 7 queued (queue size: 1)
[t=0.0] Job 8 allocated to Block 7 (waste=260, wait=0.0)
[t=0.0] Job 9 allocated to Block 8 (waste=1570, wait=0.0)
[t=0.0] Job 10 queued (queue size: 2)
[t=0.0] Job 11 queued (queue size: 3)
[t=0.0] Job 12 queued (queue size: 4)
[t=0.0] Job 13 queued (queue size: 5)
[t=0.0] Job 14 allocated to Block 9 (waste=1080, wait=0.0)
[t=0.0] Job 15 allocated to Block 10 (waste=280, wait=0.0)
[t=0.0] Job 16 queued (queue size: 6)
[t=0.0] Job 17 queued (queue size: 7)
[t=0.0] Job 18 queued (queue size: 8)
[t=0.0] Job 19 queued (queue size: 9)
[t=0.0] Job 20 queued (queue size: 10)
[t=0.0] Job 21 queued (queue size: 11)
[t=0.0] Job 22 queued (queue size: 12)
[t=0.0] Job 23 queued (queue size: 13)
[t=0.0] Job 24 queued (queue size: 14)
[t=0.0] Job 25 queued (queue size: 15)
[t=2.0] Job 4 completed. Block 4 freed.
[t=2.0] Job 10 allocated to Block 4 (waste=1610, wait=2.0)
[t=2.0] Job 5 completed. Block 5 freed.
[t=2.0] Job 18 allocated to Block 5 (waste=1620, wait=2.0)
[t=3.0] Job 18 completed. Block 5 freed.
[t=3.0] Job 22 allocated to Block 5 (waste=290, wait=3.0)
[t=4.0] Job 2 completed. Block 2 freed.
[t=4.0] Job 11 allocated to Block 2 (waste=420, wait=4.0)
[t=5.0] Job 1 completed. Block 1 freed.
[t=5.0] Job 7 allocated to Block 1 (waste=560, wait=5.0)
[t=5.0] Job 22 completed. Block 5 freed.
[t=5.0] Job 25 allocated to Block 5 (waste=2240, wait=5.0)
[t=6.0] Job 6 completed. Block 6 freed.
[t=6.0] Job 12 allocated to Block 6 (waste=5180, wait=6.0)
[t=7.0] Job 9 completed. Block 8 freed.
[t=7.0] Job 17 allocated to Block 8 (waste=2290, wait=7.0)
[t=8.0] Job 3 completed. Block 3 freed.
[t=8.0] Job 20 allocated to Block 3 (waste=890, wait=8.0)
[t=8.0] Job 10 completed. Block 4 freed.
[t=8.0] Job 16 allocated to Block 4 (waste=960, wait=8.0)
[t=9.0] Job 11 completed. Block 2 freed.
[t=9.0] Job 24 allocated to Block 2 (waste=1050, wait=9.0)
[t=10.0] Job 8 completed. Block 7 freed.
[t=10.0] Job 14 completed. Block 9 freed.
[t=10.0] Job 15 completed. Block 10 freed.
[t=10.0] Job 17 completed. Block 8 freed.
[t=11.0] Job 20 completed. Block 3 freed.
[t=13.0] Job 7 completed. Block 1 freed.
[t=13.0] Job 13 allocated to Block 1 (waste=360, wait=13.0)
[t=14.0] Job 12 completed. Block 6 freed.
[t=14.0] Job 21 allocated to Block 6 (waste=1460, wait=14.0)
[t=14.0] Job 24 completed. Block 2 freed.
[t=15.0] Job 25 completed. Block 5 freed.
[t=15.0] Job 16 completed. Block 4 freed.
[t=15.0] Job 23 allocated to Block 4 (waste=110, wait=15.0)
[t=21.0] Job 21 completed. Block 6 freed.
[t=22.0] Job 13 completed. Block 1 freed.
[t=23.0] Job 23 completed. Block 4 freed.

WARNING: 1 jobs remain in queue (could not be allocated)

============================================================
Simulation completed at t=23.0
============================================================
```

d. How the system handles the "job clocks" and "wait clocks"

Our system uses **SimPy's discrete-event simulation framework** to handle timing:

Job Clocks (Execution Time Tracking):

- Each job has a `'time'` attribute representing CPU execution duration
- When a job is allocated:
  - SimPy automatically advances the simulation clock by this amount
  - The job remains in memory for its full execution time before deallocation
  - Current simulation time is tracked

Wait Clocks (Queue Wait Time Tracking):

- When a job enters the waiting queue, its exact simulation time is recorded
- When the job is finally allocated from the queue:

  ```
  wait_time = self.env.now - job['queue_entry_time']job['wait_time'] = wait_time
  ```

- This line of code above calculates the total time spent waiting

- Additionally, we have a class that accumulates and calculates total waiting time for all jobs: `MemorySimulatorMetrics`

e. How you define "event" and what happens when the event occurs

In our **event-driven simulation**, an event is any occurrence that changes the system state. Our implementation defines these key events:

Event Types:

1. **Job Arrival Event**
2. **Job Allocation Event**
3. **Job Completion Event**
4. **Job enters Waiting Queue**

We calculate and track the following:

- Memory block states (free ↔ occupied)
- Job states (waiting → queued → running → completed)
- Metrics (throughput, wait times, queue sizes)

f. Results comparison: First-Fit vs Best-Fit - Analysis and Recommendations

Based on our code implementation, we observed the following:

```
================================================================================
                    COMPREHENSIVE COMPARISON REPORT
================================================================================

1. THROUGHPUT (Jobs per time unit)
--------------------------------------------------------------------------------
   First-Fit:  1.0435 jobs/time unit
   Best-Fit:   1.0000 jobs/time unit
   Winner: First-Fit (by 4.35%)

2. STORAGE UTILIZATION
--------------------------------------------------------------------------------
   Blocks Never Used:
      First-Fit: 0 (0.0%)
      Best-Fit:  0 (0.0%)

   Average Block Usage:
      First-Fit: 2.40 times
      Best-Fit:  2.40 times

   Most Used Block:
      First-Fit: 4 times
      Best-Fit:  4 times

3. WAITING QUEUE METRICS
--------------------------------------------------------------------------------
   Maximum Queue Length:
      First-Fit: 15 jobs
      Best-Fit:  15 jobs

   Average Queue Length:
      First-Fit: 8.00 jobs
      Best-Fit:  8.00 jobs

   Jobs That Required Queuing:
      First-Fit: 15 jobs
      Best-Fit:  15 jobs

4. WAITING TIME IN QUEUE
--------------------------------------------------------------------------------
   Average Wait Time:
      First-Fit: 7.21 time units
      Best-Fit:  7.43 time units

   Maximum Wait Time:
      First-Fit: 15.00 time units
      Best-Fit:  16.00 time units
   Winner: First-Fit (lower by 0.21 time units)

5. INTERNAL FRAGMENTATION
--------------------------------------------------------------------------------
   Average Internal Fragmentation per Job:
      First-Fit: 1621.67 memory units
      Best-Fit:  1517.50 memory units

   Total Internal Fragmentation:
      First-Fit: 38920 memory units
      Best-Fit:  36420 memory units
   Winner: Best-Fit (reduced fragmentation by 6.42%)

6. OVERALL PERFORMANCE SUMMARY
--------------------------------------------------------------------------------
   Average Turnaround Time:
      First-Fit: 10.29 time units
      Best-Fit:  10.42 time units

   Simulation Duration:
      First-Fit: 23.00 time units
      Best-Fit:  24.00 time units

   Job Completion Rate:
      First-Fit: 96.0%
      Best-Fit:  96.0%
```

First-Fit Advantages

First-Fit proved faster at allocation time because it stopped at the first available block.

Even though it still has an **O(n)** time complexity, it filled memory quickly, which led to **higher internal fragmentation**.

Some blocks got used repeatedly, while others stayed idle, leaving many jobs waiting in the queue for earlier ones to finish.

Overall, First-Fit is **better suited for systems that prioritize speed over optimization**, especially when quick allocations are more important than perfect memory utilization.

Best-Fit Advantages

Best-Fit performed better in terms of **memory space efficiency and utilization**.

It also shares the same **O(n)** time complexity since it has to search the entire list, but it resulted in **much less internal fragmentation** compared to First-Fit.

However, the need to find the "most optimal" block increased its overhead, slightly reducing throughput.

In short, Best-Fit is **slower but more space-efficient**, making it a better choice when conserving memory is the main priority.

Recommendations

Use First-Fit when:

- Real-time or near-instant response is critical
- Memory is relatively abundant compared to job sizes
- Job sizes are mostly uniform
- The system needs to handle a high rate of job arrivals efficiently

Use Best-Fit when:

- Memory is limited and must be managed carefully
- Job sizes vary widely
- Long-term efficiency and stability are more important than immediate response
- Minimizing internal fragmentation is a key performance goal

Final Thoughts

This analysis doesn't hold universally for all systems.

While **Best-Fit** tends to produce **less fragmentation**, **First-Fit** is generally **faster and simpler** — though that speed can come at the cost of wasted memory over time.

Both algorithms have trade-offs depending on system goals and workload patterns, but for the given simulation, **First-Fit performed slightly better overall (3/5 vs 2/5)** in throughput and waiting time.