Delivery Vehicle Routing System

Multi-Agent System Implementation for CVRP

COS30018 - Intelligent Systems

Project Assignment - Option A (Topic 1)

https://github.com/ThanhLeDuc552/Delivery-Vehicle-Routing-Problem.git

Team Members: Do Tung Duong, Le Duc Thanh

November 20, 2025

1 Introduction

The Capacitated Vehicle Routing Problem (CVRP) requires designing efficient routes for delivery vehicles to serve customers while respecting capacity and maximum travel distance constraints. This project implements a Multi-Agent System using JADE framework with Google OR-Tools for optimization.

The system uses autonomous Delivery Agents (DAs) coordinated by a Master Routing Agent (MRA). Agents communicate via FIPA protocols and prioritize maximizing items delivered over minimizing distance. The system handles scenarios where demand exceeds capacity and new requests arrive during execution. Key features include request/route queuing, unserved customer tracking with persistent coordinate storage, and real-time vehicle movement simulation. A separate visualization system (not part of the MAS) provides web-based monitoring of agent activities.

2 Overall System Architecture

The system implements a Multi-Agent System using the JADE framework, where autonomous agents coordinate to solve the Capacitated Vehicle Routing Problem.

2.1 Agent Architecture

The MAS follows a hub-and-spoke architecture with two types of agents:

1. **Master Routing Agent (MRA):** The central coordinator that discovers available vehicles through JADE's Directory Facilitator (DF), assembles routing problems, solves them using OR-Tools, and assigns routes to Delivery Agents. The MRA maintains a request queue for processing customer delivery requests sequentially (using `isProcessingRequest` flag to prevent concurrent processing), tracks unserved customers for reprocessing, and uses `WaitForVehiclesBehaviour` with timeout (10 seconds) to ensure all vehicle information is collected before solving. The MRA uses adaptive polling to check for new requests: when requests are available, it polls every 2 seconds; when idle, it reduces to 10 seconds to minimize unnecessary operations. The MRA logs all conversations and solution results in JSON format.

2. **Delivery Agents (DAs):** Autonomous vehicle agents that know their own capacity and maximum distance limits (each vehicle can have different constraints). Each DA maintains its own route queue, allowing it to accept multiple route assignments even while executing a current route. DAs execute assigned routes through movement simulation, validate route assignments against their constraints before acceptance (sending REFUSE messages for invalid routes), and automatically return to depot when idle using `ReturnToDepotBehaviour`. DAs register with DF using retry logic (up to 5 attempts with exponential backoff) to ensure reliable service discovery.

2.2 Request Processing

The MRA processes customer requests through an adaptive polling mechanism. When requests are found, it polls every 2 seconds; when idle, it reduces to 10 seconds to minimize unnecessary operations:

```java
private class RequestPoller extends CyclicBehaviour {
    private static final long POLL_INTERVAL_ACTIVE = 2000;  // 2s when active
    private static final long POLL_INTERVAL_IDLE = 10000;  // 10s when idle
    private long currentPollInterval = POLL_INTERVAL_ACTIVE;

    @Override
    public void action() {
        CustomerRequest newRequest = pollForNewRequest();
        if (newRequest != null) {
            currentPollInterval = POLL_INTERVAL_ACTIVE;
            // Process request and add to queue
            RequestQueueItem queueItem = new RequestQueueItem(
                newRequest.requestId, newRequest.displayId,
                newRequest.customers, false);
            requestQueue.offer(queueItem);
        } else {
            currentPollInterval = POLL_INTERVAL_IDLE;
        }
        block(currentPollInterval);
    }
}
```

Listing 1: Adaptive Request Polling Implementation

Requests are stored in a queue structure that supports both regular user requests and synthetic requests for previously unserved customers:

```java
private static class RequestQueueItem {
    final String requestId;
    final String displayId;
    final List<CustomerInfo> customers;
    final boolean synthetic;  // true for unserved customer batches

    RequestQueueItem(String requestId, String displayId,
                     List<CustomerInfo> customers, boolean synthetic) {
        this.requestId = requestId;
        this.displayId = displayId;
        this.customers = new ArrayList<>(customers);
        this.synthetic = synthetic;
    }
}
```

Listing 2: Request Queue Structure

2.3 System Diagram

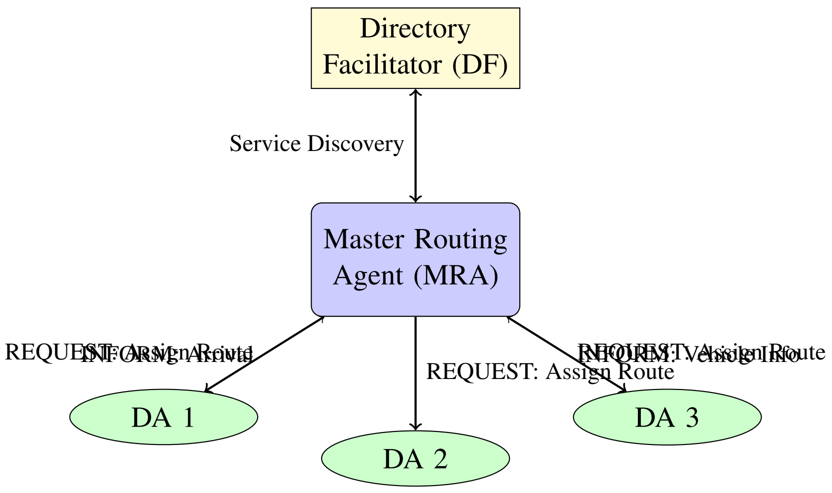

Figure 1: Hub-and-Spoke MAS Architecture

3 Implemented Interaction Protocols

All agent communications use the FIPA-Request Interaction Protocol to ensure standardized and reliable message exchange. Both the MasterRoutingAgent and DeliveryAgent classes implement this protocol consistently.

3.1 Phase 1: Service Discovery

Agents register themselves with JADE's Directory Facilitator (DF) using service types: Delivery Agents register as `da-service`, and the MRA registers as `mra-service`. The MRA dynamically discovers available vehicles by searching the DF for agents offering the `da-service`:

```java
private List<AID> findDeliveryAgentsViaDF() {
    List<AID> daAIDs = new ArrayList<>();
    DFAgentDescription dfd = new DFAgentDescription();
    ServiceDescription sd = new ServiceDescription();
    sd.setType("da-service");
    dfd.addServices(sd);
    DFAgentDescription[] results = DFService.search(this, dfd);
    for (DFAgentDescription result : results) {
        daAIDs.add(result.getName());
    }
    return daAIDs;
}
```

Listing 3: DF Service Discovery

3.2 Phase 2: Information Gathering (FIPA-Request)

The MRA queries each Delivery Agent for its vehicle information using a REQUEST message. The DA responds with an INFORM message containing its capacity, maximum distance, name, and current position:

```java
// MRA sends query
ACLMessage query = new ACLMessage(ACLMessage.REQUEST);
query.addReceiver(daAID);
query.setContent("QUERY_VEHICLE_INFO");
query.setProtocol(FIPANames.InteractionProtocol.FIPA_REQUEST);

// DA responds
ACLMessage reply = msg.createReply();
reply.setPerformative(ACLMessage.INFORM);
reply.setProtocol(FIPANames.InteractionProtocol.FIPA_REQUEST);
reply.setContent("CAPACITY:" + capacity + "|MAX_DISTANCE:" + maxDistance +
                 "|NAME:" + vehicleName + "|X:" + currentX + "|Y:" + currentY);
```

Listing 4: Vehicle Info Query and Response

3.3 Phase 3: Route Assignment (FIPA-Request)

When assigning routes, the MRA sends a REQUEST message with the ontology `route-assignment`. The DA's RouteAssignmentHandler validates the route against its constraints before accepting:

```java
private void handleRouteAssignment(ACLMessage routeAssignment) {
    // Parse route details
    int routeDemand = parseDemand(routeData);
    double routeDistance = parseDistance(routeData);

    // Validate constraints
    if (routeDemand > capacity) {
        sendRejectResponse(routeAssignment, routeId,
                           "CAPACITY_EXCEEDED",
                           "Demand exceeds capacity");
        return;
    }

    if (routeDistance > maxDistance) {
        sendRejectResponse(routeAssignment, routeId,
                           "DISTANCE_EXCEEDED",
                           "Distance exceeds max distance");
        return;
    }

    // Accept and queue route
    QueuedRoute queuedRoute = new QueuedRoute(routeId, routeData,
                                              routeDemand, routeDistance);
    routeQueue.offer(queuedRoute);

    // Send acceptance response
    ACLMessage response = routeAssignment.createReply();
    response.setPerformative(ACLMessage.INFORM);
    response.setContent("ROUTE_ACCEPTED:" + routeId + "|VEHICLE:" + vehicleName);
    send(response);
}
```

Listing 5: Route Assignment Validation

3.4 Phase 4: Request Queue Management

The MRA maintains a request queue that processes requests sequentially. The RequestQueueProcessor ensures only one request is processed at a time using the `isProcessingRequest` flag. When the accumulated unserved customer buffer reaches 6 customers, or when the request queue is empty, it creates a synthetic request to retry serving those customers:

```java
private class RequestQueueProcessor extends CyclicBehaviour {
    @Override
    public void action() {
        if (isProcessingRequest) {
            block(1000);
            return;
        }

        // Check if unserved customers should be processed
        boolean shouldProcessUnrouted = false;
        if (accumulatedUnroutedNodes.size() >= MAX_ACCUMULATED_UNROUTED) {
            shouldProcessUnrouted = true;
        } else if (requestQueue.isEmpty() && !accumulatedUnroutedNodes.isEmpty()) {
            shouldProcessUnrouted = true;
        }

        if (shouldProcessUnrouted) {
            // Create synthetic request for unserved customers
            RequestQueueItem unroutedRequest = new RequestQueueItem(
                lastUserRequestId, "unrouted-" + System.currentTimeMillis(),
                nodesToProcess, true);
            processRequest(unroutedRequest);
        } else if (!requestQueue.isEmpty()) {
            RequestQueueItem request = requestQueue.poll();
            processRequest(request);
        }
    }
}
```

Listing 6: Request Queue Processing Logic

3.5 Phase 5: Route Queue Management

Each Delivery Agent maintains its own route queue, allowing it to accept multiple route assignments even while executing a current route. Routes are validated and immediately enqueued, ensuring non-blocking assignment:

```java
private java.util.Queue<QueuedRoute> routeQueue;

// Route is validated and queued immediately
QueuedRoute queuedRoute = new QueuedRoute(routeId, routeData,
                                          routeDemand, routeDistance);
routeQueue.offer(queuedRoute);

// Process next route when current one completes
private void processNextRouteInQueue() {
    if (routeQueue.isEmpty()) return;
    QueuedRoute queuedRoute = routeQueue.poll();
    parseRouteAndStartMovement(queuedRoute.routeId, queuedRoute.routeData);
}
```

Listing 7: Route Queue Management in Delivery Agent

3.6 Phase 6: DA Arrival Notifications

When a Delivery Agent returns to the depot, it notifies the MRA so that the system knows the vehicle is available for new assignments:

```java
private void notifyMRAArrival() {
    // Find MRA via DF
    DFAgentDescription template = new DFAgentDescription();
    ServiceDescription sd = new ServiceDescription();
    sd.setType("mra-service");
    template.addServices(sd);

    DFAgentDescription[] results = DFService.search(this, template);
    if (results.length > 0) {
        ACLMessage notification = new ACLMessage(ACLMessage.INFORM);
        notification.addReceiver(results[0].getName());
        notification.setContent("DA_ARRIVED_AT_DEPOT");
        notification.setProtocol(FIPANames.InteractionProtocol.FIPA_REQUEST);
        send(notification);
    }
}
```

Listing 8: Arrival Notification

3.7 Sequence Diagram

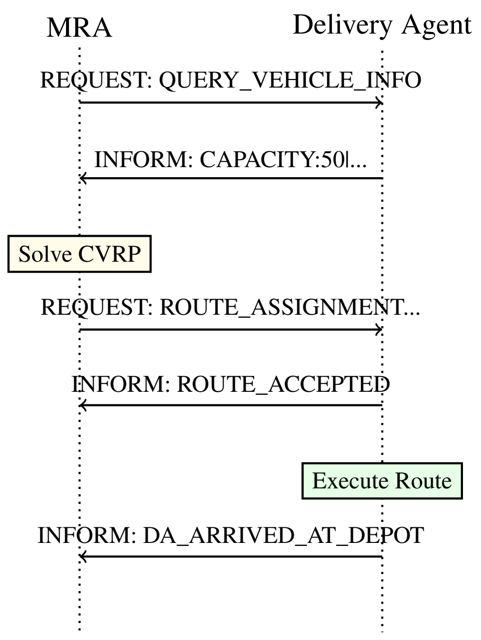

Figure 2: Agent Communication Workflow

4 Implemented Search and Optimization Techniques

The system uses Google OR-Tools Constraint Programming solver implemented in `ORToolsSolver.java`. The solver employs a hierarchical objective function that prioritizes maximizing the number of items delivered over minimizing total travel distance.

4.1 Hierarchical Objective Function

The primary objective is to maximize items delivered. This is achieved by applying a large penalty (1,000,000) to any customer node that is not visited. Since this penalty is much larger than any possible travel distance, the solver will always prefer visiting a customer over skipping it, unless constraints make it physically impossible. The secondary objective is to minimize total distance among all valid solutions that maximize deliveries:

```java
// Large penalty ensures maximizing deliveries is primary goal
private static final long UNVISITED_NODE_PENALTY = 1000000L;

// Add disjunction with penalty for each customer node
for (int node = 1; node < numNodes; node++) {
    long index = manager.nodeToIndex(node);
    routing.addDisjunction(new long[]{index}, UNVISITED_NODE_PENALTY);
}

// Set arc cost (distance) as secondary objective
routing.setArcCostEvaluatorOfAllVehicles(transitCallbackIndex);
```

Listing 9: Hierarchical Objective Implementation

4.2 Constraints Implementation

Two constraints are enforced: vehicle capacity and maximum travel distance. Both use OR-Tools dimension constraints with vehicle-specific limits:

```java
// Capacity constraint
final int demandCallbackIndex = routing.registerUnaryTransitCallback(
    (long fromIndex) -> {
        int fromNode = manager.indexToNode(fromIndex);
        return demand[fromNode];
    });

routing.addDimensionWithVehicleCapacity(
    demandCallbackIndex, 0, vehicleCapacitiesLong, true, "Capacity");
```

```java

// Maximum distance constraint
final int distanceCallbackIndex = routing.registerTransitCallback(
    (long fromIndex, long toIndex) -> {
        int fromNode = manager.indexToNode(fromIndex);
        int toNode = manager.indexToNode(toIndex);
        return distance[fromNode][toNode];
    });

routing.addDimensionWithVehicleCapacity(
    distanceCallbackIndex, 0, vehicleMaxDistancesLong, true, "Distance");
```

Listing 10: Constraint Implementation

4.3 Unserved Customer Management

The system maintains a persistent coordinate map to track unserved customers across multiple solve attempts. When customers cannot be served, their coordinates are stored and they are added to an accumulation buffer. When the buffer reaches 6 customers or when the request queue is empty, a synthetic request is created to retry serving them:

```java
private Map<Integer, CustomerInfo> customerCoordinateMap;
private List<CustomerInfo> accumulatedUnroutedNodes;
private static final int MAX_ACCUMULATED_UNROUTED = 6;

private void storeCustomerCoordinates(CustomerInfo customer) {
    if (customer != null && customer.id > 0) {
        CustomerInfo stored = new CustomerInfo(
            customer.id, customer.x, customer.y,
            customer.demand, customer.name);
        customerCoordinateMap.put(customer.id, stored);
    }
}

private void addToAccumulatedUnroutedNodes(
        List<CustomerInfo> newUnservedCustomers) {
    for (CustomerInfo newUnserved : newUnservedCustomers) {
        storeCustomerCoordinates(newUnserved);
        // Check for duplicates by ID
        boolean alreadyExists = false;
        for (CustomerInfo existing : accumulatedUnroutedNodes) {
            if (existing.id == newUnserved.id) {
                alreadyExists = true;
                break;
            }
        }
        if (!alreadyExists) {
            accumulatedUnroutedNodes.add(newUnserved);
        }
    }
}
```

Listing 11: Unserved Customer Tracking

5 Scenarios and Execution Analysis

We validated the system using four comprehensive test cases that demonstrate different aspects of the implementation.

5.1 Test Case 1: Basic Agent Communication

Scenario: 15 customers in clusters, 4 vehicles (capacity 50, max distance 1000.0). Total demand (75) < capacity (200). **Result:** All customers served via efficient loop routes. Logs confirm FIPA-Request protocol and successful assignments.

```
[2025-11-20 10:15:23.456] === Agent Logger Initialized ===
[2025-11-20 10:15:23.457] Agent: MRA
[2025-11-20 10:15:25.123] >>> SENT MESSAGE
[2025-11-20 10:15:25.123]   From: MRA@platform:1099/JADE
[2025-11-20 10:15:25.123]   To: DA-DA1@platform:1099/JADE
[2025-11-20 10:15:25.123]   Performative: REQUEST
[2025-11-20 10:15:25.123]   Protocol: FIPA_REQUEST
[2025-11-20 10:15:25.123]   Conversation ID: vehicle-info-query-DA1-1734686125123
[2025-11-20 10:15:25.123]   Content: QUERY_VEHICLE_INFO
[2025-11-20 10:15:25.123] ---
```

```
[2025-11-20 10:15:25.145] <<< RECEIVED MESSAGE
[2025-11-20 10:15:25.145]   From: DA-DA1@platform:1099/JADE
[2025-11-20 10:15:25.145]   To: MRA@platform:1099/JADE
[2025-11-20 10:15:25.145]   Performative: INFORM
[2025-11-20 10:15:25.145]   Protocol: FIPA_REQUEST
[2025-11-20 10:15:25.145]   Conversation ID: vehicle-info-query-DA1-1734686125123
[2025-11-20 10:15:25.145]   Content: CAPACITY:50|MAX_DISTANCE:1000.0|NAME:DA1|X:0.0|Y
    :0.0
[2025-11-20 10:15:25.145] ---
[2025-11-20 10:15:27.234] >>> SENT MESSAGE
[2025-11-20 10:15:27.234]   From: MRA@platform:1099/JADE
[2025-11-20 10:15:27.234]   To: DA-DA1@platform:1099/JADE
[2025-11-20 10:15:27.234]   Performative: REQUEST
[2025-11-20 10:15:27.234]   Protocol: FIPA_REQUEST
[2025-11-20 10:15:27.234]   Ontology: route-assignment
[2025-11-20 10:15:27.234]   Conversation ID: route-assignment-1-DA1-1734686127234
[2025-11-20 10:15:27.234]   Content: ROUTE_ASSIGNMENT:ROUTE:1|VEHICLE_ID:1|
    VEHICLE_NAME:DA1|CUSTOMERS:1,2,3|CUSTOMER_IDS:1,2,3|COORDS
    :15.0,10.0;25.0,15.0;35.0,12.0|DEMAND:30|DISTANCE:45.25|DEPOT_X:0.0|DEPOT_Y:0.0
[2025-11-20 10:15:27.234] ---
[2025-11-20 10:15:27.256] <<< RECEIVED MESSAGE
[2025-11-20 10:15:27.256]   From: DA-DA1@platform:1099/JADE
[2025-11-20 10:15:27.256]   To: MRA@platform:1099/JADE
[2025-11-20 10:15:27.256]   Performative: INFORM
[2025-11-20 10:15:27.256]   Protocol: FIPA_REQUEST
[2025-11-20 10:15:27.256]   Conversation ID: route-assignment-1-DA1-1734686127234
[2025-11-20 10:15:27.256]   Content: ROUTE_ACCEPTED:1|VEHICLE:DA1|STATUS:ACCEPTED|
    DEMAND:30|DISTANCE:45.25|CUSTOMERS:3
[2025-11-20 10:15:27.256] ---
[2025-11-20 10:18:45.678] <<< RECEIVED MESSAGE
[2025-11-20 10:18:45.678]   From: DA-DA1@platform:1099/JADE
[2025-11-20 10:18:45.678]   To: MRA@platform:1099/JADE
[2025-11-20 10:18:45.678]   Performative: INFORM
[2025-11-20 10:18:45.678]   Protocol: FIPA_REQUEST
[2025-11-20 10:18:45.678]   Conversation ID: da-arrival-DA1-1734686325678
[2025-11-20 10:18:45.678]   Content: DA_ARRIVED_AT_DEPOT
[2025-11-20 10:18:45.678] ---
```

Listing 12: Agent communication logs for Test Case 1

5.2 Test Case 2: Demand Exceeds Capacity

Scenario: 15 customers, demand 300, but only 4 vehicles with capacity 30 each (total 120). **Result:** Solver maximized deliveries in first round. Unserved customers detected via `updateUnservedCustomersTracking()`, coordinates stored in `customerCoordinateMap`, added to `accumulatedUnroutedNodes`. When queue empty, synthetic request created to retry unserved customers.

```
[2025-11-20 10:25:12.345] >>> SENT MESSAGE
[2025-11-20 10:25:12.345]   From: MRA@platform:1099/JADE
[2025-11-20 10:25:12.345]   To: DA-DA1@platform:1099/JADE
[2025-11-20 10:25:12.345]   Performative: REQUEST
[2025-11-20 10:25:12.345]   Protocol: FIPA_REQUEST
[2025-11-20 10:25:12.345]   Conversation ID: vehicle-info-query-DA1-1734686712345
[2025-11-20 10:25:12.345]   Content: QUERY_VEHICLE_INFO
[2025-11-20 10:25:12.345] ---
[2025-11-20 10:25:12.367] <<< RECEIVED MESSAGE
[2025-11-20 10:25:12.367]   From: DA-DA1@platform:1099/JADE
[2025-11-20 10:25:12.367]   To: MRA@platform:1099/JADE
[2025-11-20 10:25:12.367]   Performative: INFORM
[2025-11-20 10:25:12.367]   Protocol: FIPA_REQUEST
[2025-11-20 10:25:12.367]   Conversation ID: vehicle-info-query-DA1-1734686712345
[2025-11-20 10:25:12.367]   Content: CAPACITY:30|MAX_DISTANCE:1000.0|NAME:DA1|X:0.0|Y
    :0.0
[2025-11-20 10:25:12.367] ---
[2025-11-20 10:25:15.456] >>> SENT MESSAGE
```

```
[2025-11-20 10:25:15.456]   From: MRA@platform:1099/JADE
[2025-11-20 10:25:15.456]   To: DA-DA1@platform:1099/JADE
[2025-11-20 10:25:15.456]   Performative: REQUEST
[2025-11-20 10:25:15.456]   Protocol: FIPA_REQUEST
[2025-11-20 10:25:15.456]   Ontology: route-assignment
[2025-11-20 10:25:15.456]   Conversation ID: route-assignment-1-DA1-1734686715456
[2025-11-20 10:25:15.456]   Content: ROUTE_ASSIGNMENT:ROUTE:1|VEHICLE_ID:1|
    VEHICLE_NAME:DA1|CUSTOMERS:1,2|CUSTOMER_IDS:1,2|COORDS:15.0,10.0;25.0,15.0|DEMAND
    :40|DISTANCE:28.28|DEPOT_X:0.0|DEPOT_Y:0.0
[2025-11-20 10:25:15.456] ---
[2025-11-20 10:25:15.478] <<< RECEIVED MESSAGE
[2025-11-20 10:25:15.478]   From: DA-DA1@platform:1099/JADE
[2025-11-20 10:25:15.478]   To: MRA@platform:1099/JADE
[2025-11-20 10:25:15.478]   Performative: INFORM
[2025-11-20 10:25:15.478]   Protocol: FIPA_REQUEST
[2025-11-20 10:25:15.478]   Conversation ID: route-assignment-1-DA1-1734686715456
[2025-11-20 10:25:15.478]   Content: ROUTE_REJECTED:1|VEHICLE:DA1|STATUS:REJECTED|
    REASON:CAPACITY_EXCEEDED|DETAILS:Demand 40 exceeds capacity 30
[2025-11-20 10:25:15.478] ---
[2025-11-20 10:25:15.500] *** EVENT: Unserved customers detected: [Customer 1,
    Customer 2]
[2025-11-20 10:25:15.501] *** EVENT: Storing customer coordinates in
    customerCoordinateMap
[2025-11-20 10:25:15.502] *** EVENT: Adding to accumulatedUnroutedNodes (count: 2)
[2025-11-20 10:25:18.789] *** EVENT: Request queue empty, processing accumulated
    unrouted nodes (count: 8)
[2025-11-20 10:25:18.790] *** EVENT: Creating synthetic request for unserved
    customers
[2025-11-20 10:25:20.123] >>> SENT MESSAGE
[2025-11-20 10:25:20.123]   From: MRA@platform:1099/JADE
[2025-11-20 10:25:20.123]   To: DA-DA1@platform:1099/JADE
[2025-11-20 10:25:20.123]   Performative: REQUEST
[2025-11-20 10:25:20.123]   Protocol: FIPA_REQUEST
[2025-11-20 10:25:20.123]   Conversation ID: route-assignment-unrouted-DA1
    -1734686720123
[2025-11-20 10:25:20.123]   Content: ROUTE_ASSIGNMENT:ROUTE:1|VEHICLE_ID:1|
    VEHICLE_NAME:DA1|CUSTOMERS:5|CUSTOMER_IDS:5|COORDS:30.0,28.0|DEMAND:21|DISTANCE
    :37.48|DEPOT_X:0.0|DEPOT_Y:0.0
[2025-11-20 10:25:20.123] ---
[2025-11-20 10:25:20.145] <<< RECEIVED MESSAGE
[2025-11-20 10:25:20.145]   From: DA-DA1@platform:1099/JADE
[2025-11-20 10:25:20.145]   To: MRA@platform:1099/JADE
[2025-11-20 10:25:20.145]   Performative: INFORM
[2025-11-20 10:25:20.145]   Protocol: FIPA_REQUEST
[2025-11-20 10:25:20.145]   Conversation ID: route-assignment-unrouted-DA1
    -1734686720123
[2025-11-20 10:25:20.145]   Content: ROUTE_ACCEPTED:1|VEHICLE:DA1|STATUS:ACCEPTED|
    DEMAND:21|DISTANCE:37.48|CUSTOMERS:1
[2025-11-20 10:25:20.145] ---
```

Listing 13: Unserved customer handling logs for Test Case 2

5.3 Test Case 3: Maximum Distance Constraint

Scenario: Customer 6 at (100, 100), round-trip ≈ 282 units, but max distance 50. **Result:** Solver enforced distance constraint, served only nearby customers (1-5). Customer 6 added to unserved list with coordinates preserved in `customerCoordinateMap` for future attempts.

```
[2025-11-20 10:35:23.456] >>> SENT MESSAGE
[2025-11-20 10:35:23.456]   From: MRA@platform:1099/JADE
[2025-11-20 10:35:23.456]   To: DA-DA1@platform:1099/JADE
[2025-11-20 10:35:23.456]   Performative: REQUEST
[2025-11-20 10:35:23.456]   Protocol: FIPA_REQUEST
[2025-11-20 10:35:23.456]   Conversation ID: vehicle-info-query-DA1-1734687323456
[2025-11-20 10:35:23.456]   Content: QUERY_VEHICLE_INFO
```

```
[2025-11-20 10:35:23.456] ---
[2025-11-20 10:35:23.478] <<< RECEIVED MESSAGE
[2025-11-20 10:35:23.478]   From: DA-DA1@platform:1099/JADE
[2025-11-20 10:35:23.478]   To: MRA@platform:1099/JADE
[2025-11-20 10:35:23.478]   Performative: INFORM
[2025-11-20 10:35:23.478]   Protocol: FIPA_REQUEST
[2025-11-20 10:35:23.478]   Conversation ID: vehicle-info-query-DA1-1734687323456
[2025-11-20 10:35:23.478]   Content: CAPACITY:50|MAX_DISTANCE:50.0|NAME:DA1|X:0.0|Y
    :0.0
[2025-11-20 10:35:23.478] ---
[2025-11-20 10:35:25.567] >>> SENT MESSAGE
[2025-11-20 10:35:25.567]   From: MRA@platform:1099/JADE
[2025-11-20 10:35:25.567]   To: DA-DA1@platform:1099/JADE
[2025-11-20 10:35:25.567]   Performative: REQUEST
[2025-11-20 10:35:25.567]   Protocol: FIPA_REQUEST
[2025-11-20 10:35:25.567]   Ontology: route-assignment
[2025-11-20 10:35:25.567]   Conversation ID: route-assignment-1-DA1-1734687325567
[2025-11-20 10:35:25.567]   Content: ROUTE_ASSIGNMENT:ROUTE:1|VEHICLE_ID:1|
    VEHICLE_NAME:DA1|CUSTOMERS:1,2,3|CUSTOMER_IDS:1,2,3|COORDS
    :10.0,10.0;20.0,15.0;30.0,12.0|DEMAND:30|DISTANCE:32.14|DEPOT_X:0.0|DEPOT_Y:0.0
[2025-11-20 10:35:25.567] ---
[2025-11-20 10:35:25.589] <<< RECEIVED MESSAGE
[2025-11-20 10:35:25.589]   From: DA-DA1@platform:1099/JADE
[2025-11-20 10:35:25.589]   To: MRA@platform:1099/JADE
[2025-11-20 10:35:25.589]   Performative: INFORM
[2025-11-20 10:35:25.589]   Protocol: FIPA_REQUEST
[2025-11-20 10:35:25.589]   Conversation ID: route-assignment-1-DA1-1734687325567
[2025-11-20 10:35:25.589]   Content: ROUTE_ACCEPTED:1|VEHICLE:DA1|STATUS:ACCEPTED|
    DEMAND:30|DISTANCE:32.14|CUSTOMERS:3
[2025-11-20 10:35:25.589] ---
[2025-11-20 10:35:25.612] >>> SENT MESSAGE
[2025-11-20 10:35:25.612]   From: MRA@platform:1099/JADE
[2025-11-20 10:35:25.612]   To: DA-DA1@platform:1099/JADE
[2025-11-20 10:35:25.612]   Performative: REQUEST
[2025-11-20 10:35:25.612]   Protocol: FIPA_REQUEST
[2025-11-20 10:35:25.612]   Ontology: route-assignment
[2025-11-20 10:35:25.612]   Conversation ID: route-assignment-2-DA1-1734687325612
[2025-11-20 10:35:25.612]   Content: ROUTE_ASSIGNMENT:ROUTE:2|VEHICLE_ID:1|
    VEHICLE_NAME:DA1|CUSTOMERS:10|CUSTOMER_IDS:10|COORDS:100.0,100.0|DEMAND:12|
    DISTANCE:141.42|DEPOT_X:0.0|DEPOT_Y:0.0
[2025-11-20 10:35:25.612] ---
[2025-11-20 10:35:25.634] <<< RECEIVED MESSAGE
[2025-11-20 10:35:25.634]   From: DA-DA1@platform:1099/JADE
[2025-11-20 10:35:25.634]   To: MRA@platform:1099/JADE
[2025-11-20 10:35:25.634]   Performative: INFORM
[2025-11-20 10:35:25.634]   Protocol: FIPA_REQUEST
[2025-11-20 10:35:25.634]   Conversation ID: route-assignment-2-DA1-1734687325612
[2025-11-20 10:35:25.634]   Content: ROUTE_REJECTED:2|VEHICLE:DA1|STATUS:REJECTED|
    REASON:DISTANCE_EXCEEDED|DETAILS:Distance 141.42 exceeds max distance 50.0
[2025-11-20 10:35:25.634] ---
[2025-11-20 10:35:25.656] *** EVENT: Unserved customers detected: [Customer 10,
    Customer 14, Customer 15]
[2025-11-20 10:35:25.657] *** EVENT: Storing customer coordinates in
    customerCoordinateMap
[2025-11-20 10:35:25.658] *** EVENT: Customer 10 coordinates stored: (100.0, 100.0)
[2025-11-20 10:35:25.659] *** EVENT: Customer 14 coordinates stored: (90.0, 90.0)
[2025-11-20 10:35:25.660] *** EVENT: Customer 15 coordinates stored: (110.0, 110.0)
[2025-11-20 10:35:25.661] *** EVENT: Adding to accumulatedUnroutedNodes (count: 3)
```

Listing 14: Distance constraint enforcement logs for Test Case 3

5.4 Test Case 4: Multiple Requests and Queuing

Scenario: Request 2 arrives while DAs execute Request 1 routes. **Mechanism:** Request 1 processed immediately (`isProcessingRequest=true`). Request 2 enqueued via `requestQueue.offer()`. Unserved from Request 1 stored in coordinate map, added to `accumulatedUnroutedNodes`. DAs queue routes via `routeQueue.offer()` even during execution. `processNextRouteInQueue()` auto-called on depot return, ensuring continuous operation.

```
[2025-11-20 10:45:12.123] *** EVENT: Processing Request 1 (requestId: req-001)
[2025-11-20 10:45:12.124] *** EVENT: isProcessingRequest = true
[2025-11-20 10:45:12.125] >>> SENT MESSAGE
[2025-11-20 10:45:12.125]   From: MRA@platform:1099/JADE
[2025-11-20 10:45:12.125]   To: DA-DA1@platform:1099/JADE
[2025-11-20 10:45:12.125]   Performative: REQUEST
[2025-11-20 10:45:12.125]   Protocol: FIPA_REQUEST
[2025-11-20 10:45:12.125]   Conversation ID: route-assignment-1-DA1-1734687912125
[2025-11-20 10:45:12.125]   Content: ROUTE_ASSIGNMENT:ROUTE:1|VEHICLE_ID:1|
    VEHICLE_NAME:DA1|CUSTOMERS:1,3|CUSTOMER_IDS:1,3|COORDS:15.0,10.0;35.0,12.0|DEMAND
    :18|DISTANCE:25.50|DEPOT_X:0.0|DEPOT_Y:0.0
[2025-11-20 10:45:12.125] ---
[2025-11-20 10:45:12.147] <<< RECEIVED MESSAGE
[2025-11-20 10:45:12.147]   From: DA-DA1@platform:1099/JADE
[2025-11-20 10:45:12.147]   To: MRA@platform:1099/JADE
[2025-11-20 10:45:12.147]   Performative: INFORM
[2025-11-20 10:45:12.147]   Protocol: FIPA_REQUEST
[2025-11-20 10:45:12.147]   Conversation ID: route-assignment-1-DA1-1734687912125
[2025-11-20 10:45:12.147]   Content: ROUTE_ACCEPTED:1|VEHICLE:DA1|STATUS:ACCEPTED|
    DEMAND:18|DISTANCE:25.50|CUSTOMERS:2
[2025-11-20 10:45:12.147] ---
[2025-11-20 10:45:15.234] *** EVENT: Request 2 received (requestId: req-002) - adding
     to requestQueue
[2025-11-20 10:45:15.235] *** EVENT: Request queue size: 1
[2025-11-20 10:45:15.236] *** EVENT: isProcessingRequest = true, waiting for Request
    1 to complete
[2025-11-20 10:45:18.456] *** EVENT: Unserved customers from Request 1: [Customer 5]
[2025-11-20 10:45:18.457] *** EVENT: Adding to accumulatedUnroutedNodes (count: 1)
[2025-11-20 10:45:18.458] *** EVENT: Request 1 completed, isProcessingRequest = false
[2025-11-20 10:45:18.459] *** EVENT: Processing Request 2 from queue
[2025-11-20 10:45:18.460] *** EVENT: isProcessingRequest = true
[2025-11-20 10:45:18.567] >>> SENT MESSAGE
[2025-11-20 10:45:18.567]   From: MRA@platform:1099/JADE
[2025-11-20 10:45:18.567]   To: DA-DA1@platform:1099/JADE
[2025-11-20 10:45:18.567]   Performative: REQUEST
[2025-11-20 10:45:18.567]   Protocol: FIPA_REQUEST
[2025-11-20 10:45:18.567]   Conversation ID: vehicle-info-query-DA1-1734687918567
[2025-11-20 10:45:18.567]   Content: QUERY_VEHICLE_INFO
[2025-11-20 10:45:18.567] ---
[2025-11-20 10:45:18.589] <<< RECEIVED MESSAGE
[2025-11-20 10:45:18.589]   From: DA-DA1@platform:1099/JADE
[2025-11-20 10:45:18.589]   To: MRA@platform:1099/JADE
[2025-11-20 10:45:18.589]   Performative: INFORM
[2025-11-20 10:45:18.589]   Protocol: FIPA_REQUEST
[2025-11-20 10:45:18.589]   Conversation ID: vehicle-info-query-DA1-1734687918567
[2025-11-20 10:45:18.589]   Content: CAPACITY:50|MAX_DISTANCE:1000.0|NAME:DA1|X:15.5|
    Y:12.3
[2025-11-20 10:45:18.589] ---
[2025-11-20 10:45:20.678] >>> SENT MESSAGE
[2025-11-20 10:45:20.678]   From: MRA@platform:1099/JADE
[2025-11-20 10:45:20.678]   To: DA-DA1@platform:1099/JADE
[2025-11-20 10:45:20.678]   Performative: REQUEST
[2025-11-20 10:45:20.678]   Protocol: FIPA_REQUEST
[2025-11-20 10:45:20.678]   Ontology: route-assignment
[2025-11-20 10:45:20.678]   Conversation ID: route-assignment-2-DA1-1734687920678
[2025-11-20 10:45:20.678]   Content: ROUTE_ASSIGNMENT:ROUTE:2|VEHICLE_ID:1|
    VEHICLE_NAME:DA1|CUSTOMERS:7,8|CUSTOMER_IDS:7,8|COORDS:10.0,20.0;18.0,35.0|DEMAND
    :23|DISTANCE:35.20|DEPOT_X:0.0|DEPOT_Y:0.0
[2025-11-20 10:45:20.678] ---
[2025-11-20 10:45:20.700] <<< RECEIVED MESSAGE
[2025-11-20 10:45:20.700]   From: DA-DA1@platform:1099/JADE
[2025-11-20 10:45:20.700]   To: MRA@platform:1099/JADE
```

```
[2025-11-20 10:45:20.700]   Performative: INFORM
[2025-11-20 10:45:20.700]   Protocol: FIPA_REQUEST
[2025-11-20 10:45:20.700]   Conversation ID: route-assignment-2-DA1-1734687920678
[2025-11-20 10:45:20.700]   Content: ROUTE_ACCEPTED:2|VEHICLE:DA1|STATUS:ACCEPTED|
    DEMAND:23|DISTANCE:35.20|CUSTOMERS:2|NOTE:Route queued, will execute after
    current route
[2025-11-20 10:45:20.700] ---
[2025-11-20 10:45:22.789] <<< RECEIVED MESSAGE
[2025-11-20 10:45:22.789]   From: DA-DA1@platform:1099/JADE
[2025-11-20 10:45:22.789]   To: MRA@platform:1099/JADE
[2025-11-20 10:45:22.789]   Performative: INFORM
[2025-11-20 10:45:22.789]   Protocol: FIPA_REQUEST
[2025-11-20 10:45:22.789]   Conversation ID: da-arrival-DA1-1734687922789
[2025-11-20 10:45:22.789]   Content: DA_ARRIVED_AT_DEPOT
[2025-11-20 10:45:22.789] ---
[2025-11-20 10:45:22.790] *** EVENT: DA1 processing queued route (Route 2)
[2025-11-20 10:45:25.123] *** EVENT: accumulatedUnroutedNodes count: 6, triggering
    processing
[2025-11-20 10:45:25.124] *** EVENT: Creating synthetic request for accumulated
    unrouted nodes
```

Listing 15: Multiple request queuing logs for Test Case 4

6 Critical Analysis

6.1 Strengths

Key strengths include: (1) Robustness via `UNVISITED_NODE_PENALTY` ensuring valid solutions rather than failures; (2) Decoupled architecture enabling solver swapping without affecting other components; (3) Dynamic discovery through JADE DF supporting runtime vehicle changes; (4) Non-blocking queuing for concurrent operations; (5) Persistent coordinate map preserving customer locations across requests; (6) Autonomous agent behavior with constraint validation at the agent level.

6.2 Limitations and Future Work

Limitations include: (1) Unreachable customers accumulate indefinitely—need permanent drop logic after failed attempts; (2) Fleet stability assumption during 30s solve window—re-query before assignment would improve robustness; (3) Sequential request processing limits throughput—parallel solving for disjoint clusters could help; (4) Fixed threshold (`MAX_ACCUMULATED_UNROUTED=6`)—should be configurable; (5) No priority system—priority queue needed for express deliveries; (6) Movement simplification—constant speed acceptable but could add variable speeds/traffic modeling.

7 Conclusion

This project successfully implements a Multi-Agent System for CVRP using JADE and Google OR-Tools. The system prioritizes maximizing items delivered over distance minimization, enforces capacity and maximum travel distance constraints, and handles demand exceeding capacity. The MAS implementation includes request/route queuing, unserved customer accumulation/reprocessing with persistent coordinate preservation, and real-time movement simulation. The modular architecture, dynamic discovery through JADE's Directory Facilitator, and non-blocking patterns enable scalability while maintaining correctness. Agent-to-agent communication follows FIPA-Request protocols, ensuring standardized and reliable message exchange. The system meets all basic requirements and provides a foundation for future enhancements.