<routes>

    <vType id="car" accel="2.6" decel="4.5" sigma="0.5" length="5" maxSpeed="70" color="0.8,0.8,0.8"/>

    <vType id="ambulance" accel="4.0" decel="6.0" sigma="0" length="7" maxSpeed="90" vClass="emergency" color="1,0,0"/>

    <route id="start\_amb\_1" edges="e5"/>

    <route id="start\_amb\_2" edges="e10"/>

    <route id="start\_amb\_3" edges="e1"/>

    <vehicle id="amb\_1" type="ambulance" route="start\_amb\_1" depart="0" />

    <vehicle id="amb\_2" type="ambulance" route="start\_amb\_2" depart="0" />

    <vehicle id="amb\_3" type="ambulance" route="start\_amb\_3" depart="0" />

    <flow id="bg\_traffic" type="car" begin="0" end="1000" number="100" from="e6" to="e2"/>

</routes>

Hexagon.rou.xml

<configuration>

<input>

<net-file value="hexagon.net.xml"/>

<route-files value="hexagon.rou.xml"/>

<additional-files value="hospitals.add.xml ,patients.add.xml"/>

</input>

<time>

<begin value="0"/>

<end value="1000"/>

</time>

<output>

<tripinfo-output value="tripinfo.xml"/>

</output>

</configuration>

Hexagon.sumocfg

<additional>

    <poi id="P01" type="patient" color="red" layer="6" x="100" y="340"/>

    <poi id="P02" type="patient" color="red" layer="6" x="390" y="173"/>

    <poi id="P03" type="patient" color="red" layer="6" x="295" y="10"/>

</additional>

Patients.add.xml

import os

import sys

import traci

import heapq

import random

import xml.etree.ElementTree as ET

from collections import Counter

# --- GA CONFIGURATION ---

GA\_CONFIG = {

    "population\_size": 50,

    "generations": 30,

    "mutation\_rate": 0.1,

    "crossover\_rate": 0.8,

    "tournament\_size": 5

}

# --- SIMULATION DATA ---

PATIENTS = {

    "P01": {"name": "Ravi Kumar", "condition": "Cardiac Arrest", "keywords": ["Cardiology"], "start\_edge": "e6"},

    "P02": {"name": "Sita Devi", "condition": "Multiple Fractures", "keywords": ["Trauma Care", "Orthopedics"], "start\_edge": "e1"},

    "P03": {"name": "Arjun Singh", "condition": "Severe Lacerations", "keywords": ["Emergency Care"], "start\_edge": "e3\_rev"},

}

HOSPITALS = {

    "H-01": {"name": "City\_General", "specialties": ["General Medicine", "Emergency Care"], "available\_beds": 12, "dest\_edge": "e6"},

    "H-02": {"name": "Green\_Heart", "specialties": ["Cardiology"], "available\_beds": 5, "dest\_edge": "e2"},

    "H-04": {"name": "Tumakuru\_Trauma", "specialties": ["Trauma Care", "Orthopedics"], "available\_beds": 3, "dest\_edge": "e4"}

}

AMBULANCES = ["amb\_1", "amb\_2", "amb\_3"]

# --- Dijkstra Pathfinding Class (UPDATED) ---

class DijkstraForSUMO:

    def \_\_init\_\_(self, net\_file):

        self.net\_file = net\_file

        self.graph, self.edge\_to\_junctions, self.junction\_pair\_to\_edge = {}, {}, {}

        self.\_build\_graph()

    def \_build\_graph(self):

        tree = ET.parse(self.net\_file)

        for edge in tree.getroot().findall('edge'):

            if edge.get('function') != 'internal':

                edge\_id, from\_node, to\_node = edge.get('id'), edge.get('from'), edge.get('to')

                lane = edge.find('lane')

                if lane is not None:

                    speed = float(lane.get('speed'))

                    travel\_time = float(lane.get('length')) / speed if speed > 0 else float('inf')

                    if from\_node not in self.graph:

                        self.graph[from\_node] = {}

                    self.graph[from\_node][to\_node] = travel\_time

                    self.edge\_to\_junctions[edge\_id] = (from\_node, to\_node)

                    self.junction\_pair\_to\_edge[(from\_node, to\_node)] = edge\_id

    def find\_shortest\_path(self, start\_edge, end\_edge):

        if start\_edge == end\_edge:

            return [start\_edge], 0

        if start\_edge not in self.edge\_to\_junctions or end\_edge not in self.edge\_to\_junctions:

            return None, float('inf')

        start\_node = self.edge\_to\_junctions[start\_edge][1]

        end\_node = self.edge\_to\_junctions[end\_edge][1]

        distances = {node: float('inf') for node in self.graph}

        distances[start\_node] = 0

        previous\_nodes = {node: None for node in self.graph}

        pq = [(0, start\_node)]

        while pq:

            dist, current\_node = heapq.heappop(pq)

            if dist > distances[current\_node]:

                continue

            if current\_node == end\_node:

                break

            if current\_node in self.graph:

                for neighbor, weight in self.graph[current\_node].items():

                    distance = dist + weight

                    if distance < distances[neighbor]:

                        distances[neighbor] = distance

                        previous\_nodes[neighbor] = current\_node

                        heapq.heappush(pq, (distance, neighbor))

        path\_nodes = []

        current = end\_node

        while current is not None:

            path\_nodes.insert(0, current)

            current = previous\_nodes[current]

        if not path\_nodes or path\_nodes[0] != start\_node:

            return None, float('inf')

        path\_edges = [self.junction\_pair\_to\_edge.get((path\_nodes[i], path\_nodes[i+1])) for i in range(len(path\_nodes) - 1)]

        final\_path = [start\_edge] + [edge for edge in path\_edges if edge]

        return final\_path, distances[end\_node]

    def find\_shortest\_path\_time(self, start\_edge, end\_edge):

        \_, time = self.find\_shortest\_path(start\_edge, end\_edge)

        return time

# --- Genetic Algorithm Functions ---

def create\_chromosome():

    hospital\_ids = list(HOSPITALS.keys())

    return [random.choice(hospital\_ids) for \_ in PATIENTS]

def calculate\_fitness(chromosome, router):

    total\_travel\_time = 0

    penalty = 0

    available\_ambulances = list(AMBULANCES)

    hospital\_assignments = Counter(chromosome)

    for h\_id, count in hospital\_assignments.items():

        if count > HOSPITALS[h\_id]["available\_beds"]:

            penalty += 10000 \* (count - HOSPITALS[h\_id]["available\_beds"])

    # This logic is complex because it has to find the best ambulance for each patient in the plan

    patients\_to\_assign = list(PATIENTS.keys())

    temp\_ambulances = list(AMBULANCES)

    # Create a temporary assignment of ambulances to patients to calculate total time

    assignments = {} # patient\_id -> ambulance\_id

    for \_ in range(len(patients\_to\_assign)):

        best\_amb, best\_pat, min\_time = None, None, float('inf')

        for amb in temp\_ambulances:

            for pat in patients\_to\_assign:

                time = router.find\_shortest\_path\_time(traci.vehicle.getRoadID(amb), PATIENTS[pat]["start\_edge"])

                if time < min\_time:

                    min\_time, best\_amb, best\_pat = time, amb, pat

        if best\_amb:

            assignments[best\_pat] = best\_amb

            patients\_to\_assign.remove(best\_pat)

            temp\_ambulances.remove(best\_amb)

    for i, patient\_id in enumerate(PATIENTS.keys()):

        hospital\_id = chromosome[i]

        hospital\_edge = HOSPITALS[hospital\_id]["dest\_edge"]

        patient\_edge = PATIENTS[patient\_id]["start\_edge"]

        if patient\_id in assignments:

            assigned\_amb = assignments[patient\_id]

            amb\_edge = traci.vehicle.getRoadID(assigned\_amb)

            time\_to\_patient = router.find\_shortest\_path\_time(amb\_edge, patient\_edge)

            time\_to\_hospital = router.find\_shortest\_path\_time(patient\_edge, hospital\_edge)

            total\_travel\_time += time\_to\_patient + time\_to\_hospital

    return total\_travel\_time + penalty

def selection(population, fitnesses):

    tournament = random.sample(list(zip(population, fitnesses)), GA\_CONFIG["tournament\_size"])

    return min(tournament, key=lambda x: x[1])[0]

def crossover(parent1, parent2):

    if random.random() < GA\_CONFIG["crossover\_rate"]:

        point = random.randint(1, len(parent1) - 1)

        child1 = parent1[:point] + parent2[point:]

        child2 = parent2[:point] + parent1[point:]

        return child1, child2

    return parent1, parent2

def mutate(chromosome):

    for i in range(len(chromosome)):

        if random.random() < GA\_CONFIG["mutation\_rate"]:

            chromosome[i] = random.choice(list(HOSPITALS.keys()))

    return chromosome

# --- Main Simulation Logic ---

def run\_simulation():

    sumo\_cmd = [os.path.join(os.environ.get("SUMO\_HOME", "."), "bin", "sumo-gui"), "-c", "hexagon.sumocfg", "--tripinfo-output", "tripinfo\_results.xml"]

    traci.start(sumo\_cmd)

    router = DijkstraForSUMO('hexagon.net.xml')

    plan\_executed = False

    while traci.simulation.getMinExpectedNumber() > 0:

        traci.simulationStep()

        if not plan\_executed and traci.simulation.getTime() >= 1:

            print("--- Running Genetic Algorithm to Find Optimal Plan ---")

            population = [create\_chromosome() for \_ in range(GA\_CONFIG["population\_size"])]

            for gen in range(GA\_CONFIG["generations"]):

                fitnesses = [calculate\_fitness(chrom, router) for chrom in population]

                new\_population = []

                for \_ in range(GA\_CONFIG["population\_size"] // 2):

                    parent1 = selection(population, fitnesses)

                    parent2 = selection(population, fitnesses)

                    child1, child2 = crossover(parent1, parent2)

                    new\_population.extend([mutate(child1), mutate(child2)])

                population = new\_population

                print(f"Generation {gen+1}, Best Time: {min(fitnesses):.2f}s")

            final\_fitnesses = [calculate\_fitness(chrom, router) for chrom in population]

            best\_plan = min(zip(population, final\_fitnesses), key=lambda x: x[1])[0]

            print("\n--- Optimal Plan Found! Dispatching Ambulances ---")

            # Execute the best plan found by the GA

            patients\_to\_assign = list(PATIENTS.keys())

            temp\_ambulances = list(AMBULANCES)

            assignments = {}

            for \_ in range(len(patients\_to\_assign)):

                best\_amb, best\_pat, min\_time = None, None, float('inf')

                for amb in temp\_ambulances:

                    for pat in patients\_to\_assign:

                        time = router.find\_shortest\_path\_time(traci.vehicle.getRoadID(amb), PATIENTS[pat]["start\_edge"])

                        if time < min\_time:

                            min\_time, best\_amb, best\_pat = time, amb, pat

                if best\_amb:

                    assignments[best\_pat] = best\_amb

                    patients\_to\_assign.remove(best\_pat)

                    temp\_ambulances.remove(best\_amb)

            for patient\_id, amb\_id in assignments.items():

                patient\_index = list(PATIENTS.keys()).index(patient\_id)

                hospital\_id = best\_plan[patient\_index]

                current\_amb\_edge = traci.vehicle.getRoadID(amb\_id)

                patient\_edge = PATIENTS[patient\_id]["start\_edge"]

                hospital\_edge = HOSPITALS[hospital\_id]["dest\_edge"]

                path\_to\_patient, \_ = router.find\_shortest\_path(current\_amb\_edge, patient\_edge)

                path\_to\_hospital, \_ = router.find\_shortest\_path(patient\_edge, hospital\_edge)

                if path\_to\_patient and path\_to\_hospital:

                    full\_route = path\_to\_patient + path\_to\_hospital[1:] if len(path\_to\_hospital) > 1 else path\_to\_patient

                    traci.vehicle.setRoute(amb\_id, full\_route)

                    print(f"Dispatching '{amb\_id}' to patient '{patient\_id}' and then to hospital '{HOSPITALS[hospital\_id]['name']}'.")

                else:

                    print(f"ERROR: Could not find a full path for '{amb\_id}' to serve patient '{patient\_id}'.")

            plan\_executed = True

    traci.close()

if \_\_name\_\_ == "\_\_main\_\_":

    if "SUMO\_HOME" not in os.environ:

        sys.exit("Please declare the environment variable 'SUMO\_HOME'.")

    run\_simulation()

traci\_runner.py

<for implementing ga>

========================================

Emergency Dispatch Summary

========================================

--- Mission 1 ---

Ambulance: amb\_1

Patient Details:

- ID: P01

- Name: Ravi Kumar

- Condition: Cardiac Arrest (Severity: high)

- Location: Edge 'e6'

Assigned Hospital:

- ID: H-02

- Name: Green\_Heart

- Specialties:Cardiology

- Destination:Edge 'e2'

----------------------------------------

--- Mission 2 ---

Ambulance: amb\_3

Patient Details:

- ID: P02

- Name: Sita Devi

- Condition: Multiple Fractures (Severity: high)

- Location: Edge 'e1'

Assigned Hospital:

- ID: H-04

- Name: Tumakuru\_Trauma

- Specialties:Trauma Care, Orthopedics

- Destination:Edge 'e4'

----------------------------------------

--- Mission 3 ---

Ambulance: amb\_7

Patient Details:

- ID: P05

- Name: Amit Patel

- Condition: Head Injury (Severity: high)

- Location: Edge 'e9'

Assigned Hospital:

- ID: H-04

- Name: Tumakuru\_Trauma

- Specialties:Trauma Care, Orthopedics

- Destination:Edge 'e4'

----------------------------------------

--- Mission 4 ---

Ambulance: amb\_8

Patient Details:

- ID: P07

- Name: Vikram Rathod

- Condition: Chest Pain (Severity: high)

- Location: Edge 'e8'

Assigned Hospital:

- ID: H-02

- Name: Green\_Heart

- Specialties:Cardiology

- Destination:Edge 'e2'

----------------------------------------

--- Mission 5 ---

Ambulance: amb\_2

Patient Details:

- ID: P03

- Name: Arjun Singh

- Condition: Severe Lacerations (Severity: medium)

- Location: Edge 'e3\_rev'

Assigned Hospital:

- ID: H-01

- Name: City\_General

- Specialties:General Medicine, Emergency Care

- Destination:Edge 'e6'

----------------------------------------

--- Mission 6 ---

Ambulance: amb\_6

Patient Details:

- ID: P04

- Name: Priya Sharma

- Condition: Breathing Difficulty (Severity: medium)

- Location: Edge 'e5'

Assigned Hospital:

- ID: H-01

- Name: City\_General

- Specialties:General Medicine, Emergency Care

- Destination:Edge 'e6'

----------------------------------------

--- Mission 7 ---

Ambulance: amb\_4

Patient Details:

- ID: P06

- Name: Anjali Gupta

- Condition: Minor Burn (Severity: low)

- Location: Edge 'e12\_rev'

Assigned Hospital:

- ID: H-01

- Name: City\_General

- Specialties:General Medicine, Emergency Care

- Destination:Edge 'e6'

----------------------------------------

--- Mission 8 ---

Ambulance: amb\_5

Patient Details:

- ID: P08

- Name: Meera Iyer

- Condition: Sprained Ankle (Severity: low)

- Location: Edge 'e2\_rev'

Assigned Hospital:

- ID: H-04

- Name: Tumakuru\_Trauma

- Specialties:Trauma Care, Orthopedics

- Destination:Edge 'e4'

----------------------------------------

Generating Detailed Dispatch Report...

=============================================

MISSION DETAILS FOR: P01 (serviced by amb\_1)

=============================================

Timeline:

- Journey Started (Depart Time): 0.00 s

- Reached Hospital (Arrival Time): 48.00 s

Time Comparison (Full Trip):

- Estimated Ideal Time: 65.12 s

- Actual Time Taken: 48.00 s

Distance Comparison (Full Trip):

- Estimated Route Distance: 904.55 m

- Actual Distance Traveled: 609.49 m

=============================================

MISSION DETAILS FOR: P02 (serviced by amb\_3)

=============================================

Timeline:

- Journey Started (Depart Time): 0.00 s

- Reached Hospital (Arrival Time): 122.00 s

Time Comparison (Full Trip):

- Estimated Ideal Time: 62.82 s

- Actual Time Taken: 122.00 s

Distance Comparison (Full Trip):

- Estimated Route Distance: 872.55 m

- Actual Distance Traveled: 750.02 m

=============================================

MISSION DETAILS FOR: P05 (serviced by amb\_7)

=============================================

Timeline:

- Journey Started (Depart Time): 0.00 s

- Reached Hospital (Arrival Time): 106.00 s

Time Comparison (Full Trip):

- Estimated Ideal Time: 61.67 s

- Actual Time Taken: 106.00 s

Distance Comparison (Full Trip):

- Estimated Route Distance: 856.55 m

- Actual Distance Traveled: 642.12 m

=============================================

MISSION DETAILS FOR: P07 (serviced by amb\_8)

=============================================

Timeline:

- Journey Started (Depart Time): 0.00 s

- Reached Hospital (Arrival Time): 42.00 s

Time Comparison (Full Trip):

- Estimated Ideal Time: 48.64 s

- Actual Time Taken: 42.00 s

Distance Comparison (Full Trip):

- Estimated Route Distance: 675.64 m

- Actual Distance Traveled: 530.24 m

=============================================

MISSION DETAILS FOR: P03 (serviced by amb\_2)

=============================================

Timeline:

- Journey Started (Depart Time): 0.00 s

- Reached Hospital (Arrival Time): 71.00 s

Time Comparison (Full Trip):

- Estimated Ideal Time: 74.69 s

- Actual Time Taken: 71.00 s

Distance Comparison (Full Trip):

- Estimated Route Distance: 1037.46 m

- Actual Distance Traveled: 832.98 m

=============================================

MISSION DETAILS FOR: P04 (serviced by amb\_6)

=============================================

Timeline:

- Journey Started (Depart Time): 0.00 s

- Reached Hospital (Arrival Time): 44.00 s

Time Comparison (Full Trip):

- Estimated Ideal Time: 52.10 s

- Actual Time Taken: 44.00 s

Distance Comparison (Full Trip):

- Estimated Route Distance: 723.64 m

- Actual Distance Traveled: 566.20 m

=============================================

MISSION DETAILS FOR: P06 (serviced by amb\_4)

=============================================

Timeline:

- Journey Started (Depart Time): 0.00 s

- Reached Hospital (Arrival Time): 150.00 s

Time Comparison (Full Trip):

- Estimated Ideal Time: 85.41 s

- Actual Time Taken: 150.00 s

Distance Comparison (Full Trip):

- Estimated Route Distance: 1186.37 m

- Actual Distance Traveled: 1011.46 m

=============================================

MISSION DETAILS FOR: P08 (serviced by amb\_5)

=============================================

Timeline:

- Journey Started (Depart Time): 0.00 s

- Reached Hospital (Arrival Time): 125.00 s

Time Comparison (Full Trip):

- Estimated Ideal Time: 88.87 s

- Actual Time Taken: 125.00 s

Distance Comparison (Full Trip):

- Estimated Route Distance: 1234.37 m

- Actual Distance Traveled: 1044.62 m

20th oct 2025(Sunday mid night(home))

<?xml version="1.0" encoding="UTF-8"?>

<!-- generated on 2025-10-09 18:21:57 by Eclipse SUMO netconvert Version 1.24.0

<netconvertConfiguration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/netconvertConfiguration.xsd">

<input>

<node-files value="hexagon.nod.xml"/>

<edge-files value="hexagon.edg.xml"/>

</input>

<output>

<output-file value="hexagon.net.xml"/>

</output>

</netconvertConfiguration>

-->

<net version="1.20" junctionCornerDetail="5" limitTurnSpeed="5.50" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/net\_file.xsd">

<location netOffset="200.00,173.20" convBoundary="0.00,0.00,400.00,346.40" origBoundary="-200.00,-173.20,200.00,173.20" projParameter="!"/>

<edge id=":n1\_0" function="internal">

<lane id=":n1\_0\_0" index="0" speed="11.73" length="19.04" shape="306.16,338.94 303.21,342.90 299.62,345.73 295.36,347.43 290.46,348.00"/>

</edge>

<edge id=":n1\_1" function="internal">

<lane id=":n1\_1\_0" index="0" speed="7.40" length="7.21" shape="306.16,338.94 303.08,342.94 301.09,343.80"/>

</edge>

<edge id=":n1\_2" function="internal">

<lane id=":n1\_2\_0" index="0" speed="3.65" length="1.44" shape="306.16,338.94 304.86,339.58"/>

</edge>

<edge id=":n1\_9" function="internal">

<lane id=":n1\_9\_0" index="0" speed="7.40" length="9.59" shape="301.09,343.80 300.00,344.27 296.92,342.93 293.84,338.94"/>

</edge>

<edge id=":n1\_10" function="internal">

<lane id=":n1\_10\_0" index="0" speed="3.65" length="3.23" shape="304.86,339.58 303.97,339.52 303.48,338.78 303.39,337.34"/>

</edge>

<edge id=":n1\_3" function="internal">

<lane id=":n1\_3\_0" index="0" speed="5.67" length="9.24" shape="296.61,337.34 298.31,339.53 300.00,340.27 301.69,339.54 303.39,337.34"/>

</edge>

<edge id=":n1\_4" function="internal">

<lane id=":n1\_4\_0" index="0" speed="7.40" length="16.80" shape="296.61,337.34 298.54,342.00 298.15,345.33 295.46,347.33 290.46,348.00"/>

</edge>

<edge id=":n1\_5" function="internal">

<lane id=":n1\_5\_0" index="0" speed="3.65" length="4.67" shape="296.61,337.34 296.52,338.78 296.03,339.52 295.14,339.58 293.84,338.94"/>

</edge>

<edge id=":n1\_6" function="internal">

<lane id=":n1\_6\_0" index="0" speed="5.67" length="9.24" shape="290.46,344.80 293.21,344.43 294.69,343.33 294.90,341.50 293.84,338.94"/>

</edge>

<edge id=":n1\_7" function="internal">

<lane id=":n1\_7\_0" index="0" speed="10.69" length="15.68" shape="290.46,344.80 294.50,344.33 298.00,342.93 300.96,340.60 303.39,337.34"/>

</edge>

<edge id=":n1\_8" function="internal">

<lane id=":n1\_8\_0" index="0" speed="3.65" length="1.32" shape="290.46,344.80 291.56,345.53"/>

</edge>

<edge id=":n1\_11" function="internal">

<lane id=":n1\_11\_0" index="0" speed="3.65" length="3.35" shape="291.56,345.53 291.66,345.60 292.06,346.40 291.66,347.20 290.46,348.00"/>

</edge>

<edge id=":n2\_0" function="internal">

<lane id=":n2\_0\_0" index="0" speed="11.73" length="19.04" shape="396.61,164.14 398.58,168.67 399.23,173.20 398.58,177.73 396.61,182.26"/>

</edge>

<edge id=":n2\_1" function="internal">

<lane id=":n2\_1\_0" index="0" speed="7.40" length="7.22" shape="396.61,164.14 398.54,168.80 398.29,170.96"/>

</edge>

<edge id=":n2\_2" function="internal">

<lane id=":n2\_2\_0" index="0" speed="3.65" length="1.44" shape="396.61,164.14 396.52,165.58"/>

</edge>

<edge id=":n2\_9" function="internal">

<lane id=":n2\_9\_0" index="0" speed="7.40" length="9.59" shape="398.29,170.96 398.15,172.13 395.46,174.13 390.46,174.80"/>

</edge>

<edge id=":n2\_10" function="internal">

<lane id=":n2\_10\_0" index="0" speed="3.65" length="3.23" shape="396.52,165.58 396.03,166.32 395.14,166.38 393.84,165.74"/>

</edge>

<edge id=":n2\_3" function="internal">

<lane id=":n2\_3\_0" index="0" speed="5.67" length="9.24" shape="390.46,171.60 393.21,171.23 394.69,170.13 394.90,168.30 393.84,165.74"/>

</edge>

<edge id=":n2\_4" function="internal">

<lane id=":n2\_4\_0" index="0" speed="7.40" length="16.80" shape="390.46,171.60 395.46,172.27 398.15,174.27 398.54,177.60 396.61,182.26"/>

</edge>

<edge id=":n2\_5" function="internal">

<lane id=":n2\_5\_0" index="0" speed="3.65" length="4.67" shape="390.46,171.60 391.66,172.40 392.06,173.20 391.66,174.00 390.46,174.80"/>

</edge>

<edge id=":n2\_6" function="internal">

<lane id=":n2\_6\_0" index="0" speed="5.67" length="9.24" shape="393.84,180.66 394.90,178.10 394.69,176.27 393.21,175.17 390.46,174.80"/>

</edge>

<edge id=":n2\_7" function="internal">

<lane id=":n2\_7\_0" index="0" speed="10.68" length="15.68" shape="393.84,180.66 395.46,176.93 396.00,173.20 395.46,169.47 393.84,165.74"/>

</edge>

<edge id=":n2\_8" function="internal">

<lane id=":n2\_8\_0" index="0" speed="3.65" length="1.32" shape="393.84,180.66 395.03,180.08"/>

</edge>

<edge id=":n2\_11" function="internal">

<lane id=":n2\_11\_0" index="0" speed="3.65" length="3.35" shape="395.03,180.08 395.14,180.02 396.03,180.08 396.52,180.82 396.61,182.26"/>

</edge>

<edge id=":n3\_0" function="internal">

<lane id=":n3\_0\_0" index="0" speed="5.67" length="9.24" shape="303.39,9.06 301.69,6.86 300.00,6.13 298.31,6.87 296.61,9.06"/>

</edge>

<edge id=":n3\_1" function="internal">

<lane id=":n3\_1\_0" index="0" speed="10.69" length="15.68" shape="303.39,9.06 300.96,5.80 298.00,3.47 294.50,2.07 290.46,1.60"/>

</edge>

<edge id=":n3\_2" function="internal">

<lane id=":n3\_2\_0" index="0" speed="3.65" length="1.32" shape="303.39,9.06 303.47,7.74"/>

</edge>

<edge id=":n3\_9" function="internal">

<lane id=":n3\_9\_0" index="0" speed="3.65" length="3.35" shape="303.47,7.74 303.48,7.62 303.97,6.88 304.86,6.82 306.16,7.46"/>

</edge>

<edge id=":n3\_3" function="internal">

<lane id=":n3\_3\_0" index="0" speed="11.73" length="19.04" shape="290.46,-1.60 295.36,-1.03 299.62,0.67 303.21,3.50 306.16,7.46"/>

</edge>

<edge id=":n3\_4" function="internal">

<lane id=":n3\_4\_0" index="0" speed="7.40" length="7.21" shape="290.46,-1.60 295.46,-0.93 297.20,0.36"/>

</edge>

<edge id=":n3\_5" function="internal">

<lane id=":n3\_5\_0" index="0" speed="3.65" length="1.44" shape="290.46,-1.60 291.66,-0.80"/>

</edge>

<edge id=":n3\_10" function="internal">

<lane id=":n3\_10\_0" index="0" speed="7.40" length="9.59" shape="297.20,0.36 298.15,1.07 298.54,4.40 296.61,9.06"/>

</edge>

<edge id=":n3\_11" function="internal">

<lane id=":n3\_11\_0" index="0" speed="3.65" length="3.23" shape="291.66,-0.80 292.06,0.00 291.66,0.80 290.46,1.60"/>

</edge>

<edge id=":n3\_6" function="internal">

<lane id=":n3\_6\_0" index="0" speed="5.67" length="9.24" shape="293.84,7.46 294.90,4.90 294.69,3.07 293.21,1.97 290.46,1.60"/>

</edge>

<edge id=":n3\_7" function="internal">

<lane id=":n3\_7\_0" index="0" speed="7.40" length="16.80" shape="293.84,7.46 296.92,3.47 300.00,2.13 303.08,3.46 306.16,7.46"/>

</edge>

<edge id=":n3\_8" function="internal">

<lane id=":n3\_8\_0" index="0" speed="3.65" length="4.67" shape="293.84,7.46 295.14,6.82 296.03,6.88 296.52,7.62 296.61,9.06"/>

</edge>

<edge id=":n4\_0" function="internal">

<lane id=":n4\_0\_0" index="0" speed="5.67" length="9.24" shape="103.39,9.06 101.69,6.87 100.00,6.13 98.31,6.86 96.61,9.06"/>

</edge>

<edge id=":n4\_1" function="internal">

<lane id=":n4\_1\_0" index="0" speed="7.40" length="16.80" shape="103.39,9.06 101.46,4.40 101.85,1.07 104.54,-0.93 109.54,-1.60"/>

</edge>

<edge id=":n4\_2" function="internal">

<lane id=":n4\_2\_0" index="0" speed="3.65" length="4.67" shape="103.39,9.06 103.48,7.62 103.97,6.88 104.86,6.82 106.16,7.46"/>

</edge>

<edge id=":n4\_3" function="internal">

<lane id=":n4\_3\_0" index="0" speed="5.67" length="9.24" shape="109.54,1.60 106.79,1.97 105.31,3.07 105.10,4.90 106.16,7.46"/>

</edge>

<edge id=":n4\_4" function="internal">

<lane id=":n4\_4\_0" index="0" speed="10.69" length="15.68" shape="109.54,1.60 105.50,2.07 102.00,3.47 99.04,5.80 96.61,9.06"/>

</edge>

<edge id=":n4\_5" function="internal">

<lane id=":n4\_5\_0" index="0" speed="3.65" length="1.32" shape="109.54,1.60 108.44,0.87"/>

</edge>

<edge id=":n4\_9" function="internal">

<lane id=":n4\_9\_0" index="0" speed="3.65" length="3.35" shape="108.44,0.87 108.34,0.80 107.94,0.00 108.34,-0.80 109.54,-1.60"/>

</edge>

<edge id=":n4\_6" function="internal">

<lane id=":n4\_6\_0" index="0" speed="11.73" length="19.04" shape="93.84,7.46 96.79,3.50 100.38,0.67 104.64,-1.03 109.54,-1.60"/>

</edge>

<edge id=":n4\_7" function="internal">

<lane id=":n4\_7\_0" index="0" speed="7.40" length="7.21" shape="93.84,7.46 96.92,3.46 98.91,2.60"/>

</edge>

<edge id=":n4\_8" function="internal">

<lane id=":n4\_8\_0" index="0" speed="3.65" length="1.44" shape="93.84,7.46 95.14,6.82"/>

</edge>

<edge id=":n4\_10" function="internal">

<lane id=":n4\_10\_0" index="0" speed="7.40" length="9.59" shape="98.91,2.60 100.00,2.13 103.08,3.47 106.16,7.46"/>

</edge>

<edge id=":n4\_11" function="internal">

<lane id=":n4\_11\_0" index="0" speed="3.65" length="3.23" shape="95.14,6.82 96.03,6.88 96.52,7.62 96.61,9.06"/>

</edge>

<edge id=":n5\_0" function="internal">

<lane id=":n5\_0\_0" index="0" speed="11.73" length="19.04" shape="3.39,182.26 1.42,177.73 0.77,173.20 1.42,168.67 3.39,164.14"/>

</edge>

<edge id=":n5\_1" function="internal">

<lane id=":n5\_1\_0" index="0" speed="7.40" length="7.22" shape="3.39,182.26 1.46,177.60 1.71,175.44"/>

</edge>

<edge id=":n5\_2" function="internal">

<lane id=":n5\_2\_0" index="0" speed="3.65" length="1.44" shape="3.39,182.26 3.48,180.82"/>

</edge>

<edge id=":n5\_9" function="internal">

<lane id=":n5\_9\_0" index="0" speed="7.40" length="9.59" shape="1.71,175.44 1.85,174.27 4.54,172.27 9.54,171.60"/>

</edge>

<edge id=":n5\_10" function="internal">

<lane id=":n5\_10\_0" index="0" speed="3.65" length="3.23" shape="3.48,180.82 3.97,180.08 4.86,180.02 6.16,180.66"/>

</edge>

<edge id=":n5\_3" function="internal">

<lane id=":n5\_3\_0" index="0" speed="5.67" length="9.24" shape="9.54,174.80 6.79,175.17 5.31,176.27 5.10,178.10 6.16,180.66"/>

</edge>

<edge id=":n5\_4" function="internal">

<lane id=":n5\_4\_0" index="0" speed="7.40" length="16.80" shape="9.54,174.80 4.54,174.13 1.85,172.13 1.46,168.80 3.39,164.14"/>

</edge>

<edge id=":n5\_5" function="internal">

<lane id=":n5\_5\_0" index="0" speed="3.65" length="4.67" shape="9.54,174.80 8.34,174.00 7.94,173.20 8.34,172.40 9.54,171.60"/>

</edge>

<edge id=":n5\_6" function="internal">

<lane id=":n5\_6\_0" index="0" speed="5.67" length="9.24" shape="6.16,165.74 5.10,168.30 5.31,170.13 6.79,171.23 9.54,171.60"/>

</edge>

<edge id=":n5\_7" function="internal">

<lane id=":n5\_7\_0" index="0" speed="10.68" length="15.68" shape="6.16,165.74 4.54,169.47 4.00,173.20 4.54,176.93 6.16,180.66"/>

</edge>

<edge id=":n5\_8" function="internal">

<lane id=":n5\_8\_0" index="0" speed="3.65" length="1.32" shape="6.16,165.74 4.97,166.32"/>

</edge>

<edge id=":n5\_11" function="internal">

<lane id=":n5\_11\_0" index="0" speed="3.65" length="3.35" shape="4.97,166.32 4.86,166.38 3.97,166.32 3.48,165.58 3.39,164.14"/>

</edge>

<edge id=":n6\_0" function="internal">

<lane id=":n6\_0\_0" index="0" speed="11.73" length="19.04" shape="109.54,348.00 104.64,347.43 100.38,345.73 96.79,342.90 93.84,338.94"/>

</edge>

<edge id=":n6\_1" function="internal">

<lane id=":n6\_1\_0" index="0" speed="7.40" length="7.21" shape="109.54,348.00 104.54,347.33 102.80,346.04"/>

</edge>

<edge id=":n6\_2" function="internal">

<lane id=":n6\_2\_0" index="0" speed="3.65" length="1.44" shape="109.54,348.00 108.34,347.20"/>

</edge>

<edge id=":n6\_9" function="internal">

<lane id=":n6\_9\_0" index="0" speed="7.40" length="9.59" shape="102.80,346.04 101.85,345.33 101.46,342.00 103.39,337.34"/>

</edge>

<edge id=":n6\_10" function="internal">

<lane id=":n6\_10\_0" index="0" speed="3.65" length="3.23" shape="108.34,347.20 107.94,346.40 108.34,345.60 109.54,344.80"/>

</edge>

<edge id=":n6\_3" function="internal">

<lane id=":n6\_3\_0" index="0" speed="5.67" length="9.24" shape="106.16,338.94 105.10,341.50 105.31,343.33 106.79,344.43 109.54,344.80"/>

</edge>

<edge id=":n6\_4" function="internal">

<lane id=":n6\_4\_0" index="0" speed="7.40" length="16.80" shape="106.16,338.94 103.08,342.93 100.00,344.27 96.92,342.94 93.84,338.94"/>

</edge>

<edge id=":n6\_5" function="internal">

<lane id=":n6\_5\_0" index="0" speed="3.65" length="4.67" shape="106.16,338.94 104.86,339.58 103.97,339.52 103.48,338.78 103.39,337.34"/>

</edge>

<edge id=":n6\_6" function="internal">

<lane id=":n6\_6\_0" index="0" speed="5.67" length="9.24" shape="96.61,337.34 98.31,339.54 100.00,340.27 101.69,339.53 103.39,337.34"/>

</edge>

<edge id=":n6\_7" function="internal">

<lane id=":n6\_7\_0" index="0" speed="10.69" length="15.68" shape="96.61,337.34 99.04,340.60 102.00,342.93 105.50,344.33 109.54,344.80"/>

</edge>

<edge id=":n6\_8" function="internal">

<lane id=":n6\_8\_0" index="0" speed="3.65" length="1.32" shape="96.61,337.34 96.53,338.66"/>

</edge>

<edge id=":n6\_11" function="internal">

<lane id=":n6\_11\_0" index="0" speed="3.65" length="3.35" shape="96.53,338.66 96.52,338.78 96.03,339.52 95.14,339.58 93.84,338.94"/>

</edge>

<edge id=":n\_center\_0" function="internal">

<lane id=":n\_center\_0\_0" index="0" speed="9.88" length="31.07" shape="211.39,196.12 205.69,188.73 200.00,186.26 194.31,188.73 188.61,196.12"/>

</edge>

<edge id=":n\_center\_1" function="internal">

<lane id=":n\_center\_1\_0" index="0" speed="13.89" length="44.78" shape="211.39,196.12 204.46,186.79 196.00,180.13 186.00,176.13 174.46,174.80"/>

</edge>

<edge id=":n\_center\_2" function="internal">

<lane id=":n\_center\_2\_0" index="0" speed="13.89" length="51.09" shape="211.39,196.12 185.84,151.88"/>

</edge>

<edge id=":n\_center\_3" function="internal">

<lane id=":n\_center\_3\_0" index="0" speed="13.89" length="6.15" shape="211.39,196.12 208.94,190.47"/>

</edge>

<edge id=":n\_center\_4" function="internal">

<lane id=":n\_center\_4\_0" index="0" speed="10.63" length="3.79" shape="211.39,196.12 210.15,192.54"/>

</edge>

<edge id=":n\_center\_5" function="internal">

<lane id=":n\_center\_5\_0" index="0" speed="3.65" length="1.44" shape="211.39,196.12 211.48,194.68"/>

</edge>

<edge id=":n\_center\_36" function="internal">

<lane id=":n\_center\_36\_0" index="0" speed="13.89" length="41.98" shape="208.94,190.47 206.42,184.66 204.77,173.20 206.42,161.74 211.39,150.28"/>

</edge>

<edge id=":n\_center\_37" function="internal">

<lane id=":n\_center\_37\_0" index="0" speed="10.63" length="32.37" shape="210.15,192.54 208.07,186.54 209.62,178.75 215.59,173.52 225.54,171.60"/>

</edge>

<edge id=":n\_center\_38" function="internal">

<lane id=":n\_center\_38\_0" index="0" speed="3.65" length="3.23" shape="211.48,194.68 211.97,193.93 212.86,193.88 214.16,194.52"/>

</edge>

<edge id=":n\_center\_6" function="internal">

<lane id=":n\_center\_6\_0" index="0" speed="9.88" length="31.07" shape="225.54,174.80 216.29,176.03 211.31,179.73 210.60,185.89 214.16,194.52"/>

</edge>

<edge id=":n\_center\_7" function="internal">

<lane id=":n\_center\_7\_0" index="0" speed="13.89" length="44.78" shape="225.54,174.80 214.00,176.13 204.00,180.13 195.54,186.79 188.61,196.12"/>

</edge>

<edge id=":n\_center\_8" function="internal">

<lane id=":n\_center\_8\_0" index="0" speed="13.89" length="51.09" shape="225.54,174.80 174.46,174.80"/>

</edge>

<edge id=":n\_center\_9" function="internal">

<lane id=":n\_center\_9\_0" index="0" speed="13.89" length="6.15" shape="225.54,174.80 219.43,174.09"/>

</edge>

<edge id=":n\_center\_10" function="internal">

<lane id=":n\_center\_10\_0" index="0" speed="10.63" length="3.79" shape="225.54,174.80 221.82,174.08"/>

</edge>

<edge id=":n\_center\_11" function="internal">

<lane id=":n\_center\_11\_0" index="0" speed="3.65" length="1.44" shape="225.54,174.80 224.34,174.00"/>

</edge>

<edge id=":n\_center\_39" function="internal">

<lane id=":n\_center\_39\_0" index="0" speed="13.89" length="41.98" shape="219.43,174.09 213.14,173.37 202.38,169.07 193.29,161.91 185.84,151.88"/>

</edge>

<edge id=":n\_center\_40" function="internal">

<lane id=":n\_center\_40\_0" index="0" speed="10.63" length="32.37" shape="221.82,174.08 215.59,172.88 209.62,167.65 208.07,159.86 211.39,150.28"/>

</edge>

<edge id=":n\_center\_41" function="internal">

<lane id=":n\_center\_41\_0" index="0" speed="3.65" length="3.23" shape="224.34,174.00 223.94,173.20 224.34,172.40 225.54,171.60"/>

</edge>

<edge id=":n\_center\_12" function="internal">

<lane id=":n\_center\_12\_0" index="0" speed="9.88" length="31.07" shape="214.16,151.88 210.60,160.51 211.31,166.67 216.29,170.37 225.54,171.60"/>

</edge>

<edge id=":n\_center\_13" function="internal">

<lane id=":n\_center\_13\_0" index="0" speed="13.89" length="44.77" shape="214.16,151.88 209.54,162.54 208.00,173.20 209.54,183.86 214.16,194.52"/>

</edge>

<edge id=":n\_center\_14" function="internal">

<lane id=":n\_center\_14\_0" index="0" speed="13.89" length="51.09" shape="214.16,151.88 188.61,196.12"/>

</edge>

<edge id=":n\_center\_15" function="internal">

<lane id=":n\_center\_15\_0" index="0" speed="13.89" length="6.15" shape="214.16,151.88 210.49,156.82"/>

</edge>

<edge id=":n\_center\_16" function="internal">

<lane id=":n\_center\_16\_0" index="0" speed="10.63" length="3.79" shape="214.16,151.88 211.68,154.74"/>

</edge>

<edge id=":n\_center\_17" function="internal">

<lane id=":n\_center\_17\_0" index="0" speed="3.65" length="1.44" shape="214.16,151.88 212.86,152.52"/>

</edge>

<edge id=":n\_center\_42" function="internal">

<lane id=":n\_center\_42\_0" index="0" speed="13.89" length="41.98" shape="210.49,156.82 206.71,161.91 197.62,169.07 186.86,173.37 174.46,174.80"/>

</edge>

<edge id=":n\_center\_43" function="internal">

<lane id=":n\_center\_43\_0" index="0" speed="10.63" length="32.37" shape="211.68,154.74 207.52,159.54 200.00,162.10 192.48,159.54 185.84,151.88"/>

</edge>

<edge id=":n\_center\_44" function="internal">

<lane id=":n\_center\_44\_0" index="0" speed="3.65" length="3.23" shape="212.86,152.52 211.97,152.47 211.48,151.72 211.39,150.28"/>

</edge>

<edge id=":n\_center\_18" function="internal">

<lane id=":n\_center\_18\_0" index="0" speed="9.88" length="31.07" shape="188.61,150.28 194.31,157.67 200.00,160.14 205.69,157.67 211.39,150.28"/>

</edge>

<edge id=":n\_center\_19" function="internal">

<lane id=":n\_center\_19\_0" index="0" speed="13.89" length="44.78" shape="188.61,150.28 195.54,159.61 204.00,166.27 214.00,170.27 225.54,171.60"/>

</edge>

<edge id=":n\_center\_20" function="internal">

<lane id=":n\_center\_20\_0" index="0" speed="13.89" length="51.09" shape="188.61,150.28 214.16,194.52"/>

</edge>

<edge id=":n\_center\_21" function="internal">

<lane id=":n\_center\_21\_0" index="0" speed="13.89" length="6.15" shape="188.61,150.28 191.06,155.93"/>

</edge>

<edge id=":n\_center\_22" function="internal">

<lane id=":n\_center\_22\_0" index="0" speed="10.63" length="3.79" shape="188.61,150.28 189.85,153.86"/>

</edge>

<edge id=":n\_center\_23" function="internal">

<lane id=":n\_center\_23\_0" index="0" speed="3.65" length="1.44" shape="188.61,150.28 188.52,151.72"/>

</edge>

<edge id=":n\_center\_45" function="internal">

<lane id=":n\_center\_45\_0" index="0" speed="13.89" length="41.98" shape="191.06,155.93 193.58,161.74 195.23,173.20 193.58,184.66 188.61,196.12"/>

</edge>

<edge id=":n\_center\_46" function="internal">

<lane id=":n\_center\_46\_0" index="0" speed="10.63" length="32.37" shape="189.85,153.86 191.93,159.86 190.38,167.65 184.41,172.88 174.46,174.80"/>

</edge>

<edge id=":n\_center\_47" function="internal">

<lane id=":n\_center\_47\_0" index="0" speed="3.65" length="3.23" shape="188.52,151.72 188.03,152.47 187.14,152.52 185.84,151.88"/>

</edge>

<edge id=":n\_center\_24" function="internal">

<lane id=":n\_center\_24\_0" index="0" speed="9.88" length="31.07" shape="174.46,171.60 183.71,170.37 188.69,166.67 189.40,160.51 185.84,151.88"/>

</edge>

<edge id=":n\_center\_25" function="internal">

<lane id=":n\_center\_25\_0" index="0" speed="13.89" length="44.78" shape="174.46,171.60 186.00,170.27 196.00,166.27 204.46,159.61 211.39,150.28"/>

</edge>

<edge id=":n\_center\_26" function="internal">

<lane id=":n\_center\_26\_0" index="0" speed="13.89" length="51.09" shape="174.46,171.60 225.54,171.60"/>

</edge>

<edge id=":n\_center\_27" function="internal">

<lane id=":n\_center\_27\_0" index="0" speed="13.89" length="6.15" shape="174.46,171.60 180.57,172.31"/>

</edge>

<edge id=":n\_center\_28" function="internal">

<lane id=":n\_center\_28\_0" index="0" speed="10.63" length="3.79" shape="174.46,171.60 178.18,172.32"/>

</edge>

<edge id=":n\_center\_29" function="internal">

<lane id=":n\_center\_29\_0" index="0" speed="3.65" length="1.44" shape="174.46,171.60 175.66,172.40"/>

</edge>

<edge id=":n\_center\_48" function="internal">

<lane id=":n\_center\_48\_0" index="0" speed="13.89" length="41.98" shape="180.57,172.31 186.86,173.03 197.62,177.33 206.71,184.49 214.16,194.52"/>

</edge>

<edge id=":n\_center\_49" function="internal">

<lane id=":n\_center\_49\_0" index="0" speed="10.63" length="32.37" shape="178.18,172.32 184.41,173.52 190.38,178.75 191.93,186.54 188.61,196.12"/>

</edge>

<edge id=":n\_center\_50" function="internal">

<lane id=":n\_center\_50\_0" index="0" speed="3.65" length="3.23" shape="175.66,172.40 176.06,173.20 175.66,174.00 174.46,174.80"/>

</edge>

<edge id=":n\_center\_30" function="internal">

<lane id=":n\_center\_30\_0" index="0" speed="9.88" length="31.07" shape="185.84,194.52 189.40,185.89 188.69,179.73 183.71,176.03 174.46,174.80"/>

</edge>

<edge id=":n\_center\_31" function="internal">

<lane id=":n\_center\_31\_0" index="0" speed="13.89" length="44.77" shape="185.84,194.52 190.46,183.86 192.00,173.20 190.46,162.54 185.84,151.88"/>

</edge>

<edge id=":n\_center\_32" function="internal">

<lane id=":n\_center\_32\_0" index="0" speed="13.89" length="51.09" shape="185.84,194.52 211.39,150.28"/>

</edge>

<edge id=":n\_center\_33" function="internal">

<lane id=":n\_center\_33\_0" index="0" speed="13.89" length="6.15" shape="185.84,194.52 189.51,189.58"/>

</edge>

<edge id=":n\_center\_34" function="internal">

<lane id=":n\_center\_34\_0" index="0" speed="10.63" length="3.79" shape="185.84,194.52 188.32,191.66"/>

</edge>

<edge id=":n\_center\_35" function="internal">

<lane id=":n\_center\_35\_0" index="0" speed="3.65" length="1.44" shape="185.84,194.52 187.14,193.88"/>

</edge>

<edge id=":n\_center\_51" function="internal">

<lane id=":n\_center\_51\_0" index="0" speed="13.89" length="41.98" shape="189.51,189.58 193.29,184.49 202.38,177.33 213.14,173.03 225.54,171.60"/>

</edge>

<edge id=":n\_center\_52" function="internal">

<lane id=":n\_center\_52\_0" index="0" speed="10.63" length="32.37" shape="188.32,191.66 192.48,186.86 200.00,184.30 207.52,186.86 214.16,194.52"/>

</edge>

<edge id=":n\_center\_53" function="internal">

<lane id=":n\_center\_53\_0" index="0" speed="3.65" length="3.23" shape="187.14,193.88 188.03,193.93 188.52,194.68 188.61,196.12"/>

</edge>

<edge id="e1" from="n1" to="n2" priority="-1">

<lane id="e1\_0" index="0" speed="13.89" length="180.91" shape="303.39,337.34 393.84,180.66"/>

</edge>

<edge id="e10" from="n\_center" to="n4" priority="-1">

<lane id="e10\_0" index="0" speed="13.89" length="164.91" shape="185.84,151.88 103.39,9.06"/>

</edge>

<edge id="e10\_rev" from="n4" to="n\_center" priority="-1">

<lane id="e10\_rev\_0" index="0" speed="13.89" length="164.91" shape="106.16,7.46 188.61,150.28"/>

</edge>

<edge id="e11" from="n\_center" to="n5" priority="-1">

<lane id="e11\_0" index="0" speed="13.89" length="164.91" shape="174.46,174.80 9.54,174.80"/>

</edge>

<edge id="e11\_rev" from="n5" to="n\_center" priority="-1">

<lane id="e11\_rev\_0" index="0" speed="13.89" length="164.91" shape="9.54,171.60 174.46,171.60"/>

</edge>

<edge id="e12" from="n\_center" to="n6" priority="-1">

<lane id="e12\_0" index="0" speed="13.89" length="164.91" shape="188.61,196.12 106.16,338.94"/>

</edge>

<edge id="e12\_rev" from="n6" to="n\_center" priority="-1">

<lane id="e12\_rev\_0" index="0" speed="13.89" length="164.91" shape="103.39,337.34 185.84,194.52"/>

</edge>

<edge id="e1\_rev" from="n2" to="n1" priority="-1">

<lane id="e1\_rev\_0" index="0" speed="13.89" length="180.91" shape="396.61,182.26 306.16,338.94"/>

</edge>

<edge id="e2" from="n2" to="n3" priority="-1">

<lane id="e2\_0" index="0" speed="13.89" length="180.91" shape="393.84,165.74 303.39,9.06"/>

</edge>

<edge id="e2\_rev" from="n3" to="n2" priority="-1">

<lane id="e2\_rev\_0" index="0" speed="13.89" length="180.91" shape="306.16,7.46 396.61,164.14"/>

</edge>

<edge id="e3" from="n3" to="n4" priority="-1">

<lane id="e3\_0" index="0" speed="13.89" length="180.91" shape="290.46,1.60 109.54,1.60"/>

</edge>

<edge id="e3\_rev" from="n4" to="n3" priority="-1">

<lane id="e3\_rev\_0" index="0" speed="13.89" length="180.91" shape="109.54,-1.60 290.46,-1.60"/>

</edge>

<edge id="e4" from="n4" to="n5" priority="-1">

<lane id="e4\_0" index="0" speed="13.89" length="180.91" shape="96.61,9.06 6.16,165.74"/>

</edge>

<edge id="e4\_rev" from="n5" to="n4" priority="-1">

<lane id="e4\_rev\_0" index="0" speed="13.89" length="180.91" shape="3.39,164.14 93.84,7.46"/>

</edge>

<edge id="e5" from="n5" to="n6" priority="-1">

<lane id="e5\_0" index="0" speed="13.89" length="180.91" shape="6.16,180.66 96.61,337.34"/>

</edge>

<edge id="e5\_rev" from="n6" to="n5" priority="-1">

<lane id="e5\_rev\_0" index="0" speed="13.89" length="180.91" shape="93.84,338.94 3.39,182.26"/>

</edge>

<edge id="e6" from="n6" to="n1" priority="-1">

<lane id="e6\_0" index="0" speed="13.89" length="180.91" shape="109.54,344.80 290.46,344.80"/>

</edge>

<edge id="e6\_rev" from="n1" to="n6" priority="-1">

<lane id="e6\_rev\_0" index="0" speed="13.89" length="180.91" shape="290.46,348.00 109.54,348.00"/>

</edge>

<edge id="e7" from="n\_center" to="n1" priority="-1">

<lane id="e7\_0" index="0" speed="13.89" length="164.91" shape="214.16,194.52 296.61,337.34"/>

</edge>

<edge id="e7\_rev" from="n1" to="n\_center" priority="-1">

<lane id="e7\_rev\_0" index="0" speed="13.89" length="164.91" shape="293.84,338.94 211.39,196.12"/>

</edge>

<edge id="e8" from="n\_center" to="n2" priority="-1">

<lane id="e8\_0" index="0" speed="13.89" length="164.91" shape="225.54,171.60 390.46,171.60"/>

</edge>

<edge id="e8\_rev" from="n2" to="n\_center" priority="-1">

<lane id="e8\_rev\_0" index="0" speed="13.89" length="164.91" shape="390.46,174.80 225.54,174.80"/>

</edge>

<edge id="e9" from="n\_center" to="n3" priority="-1">

<lane id="e9\_0" index="0" speed="13.89" length="164.91" shape="211.39,150.28 293.84,7.46"/>

</edge>

<edge id="e9\_rev" from="n3" to="n\_center" priority="-1">

<lane id="e9\_rev\_0" index="0" speed="13.89" length="164.91" shape="296.61,9.06 214.16,151.88"/>

</edge>

<tlLogic id="n\_center" type="static" programID="0" offset="0">

<phase duration="26" state="rrrrrrGGGgggrrrrrrrrrrrrGGGgggrrrrrr"/>

<phase duration="3" state="rrrrrryyyyyyrrrrrrrrrrrryyyyyyrrrrrr"/>

<phase duration="1" state="rrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr"/>

<phase duration="26" state="rrrrrrrrrrrrGGGgggrrrrrrrrrrrrGGGggg"/>

<phase duration="3" state="rrrrrrrrrrrryyyyyyrrrrrrrrrrrryyyyyy"/>

<phase duration="1" state="rrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr"/>

<phase duration="26" state="GGGgggrrrrrrrrrrrrGGGgggrrrrrrrrrrrr"/>

<phase duration="3" state="yyyyyyrrrrrrrrrrrryyyyyyrrrrrrrrrrrr"/>

<phase duration="1" state="rrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr"/>

</tlLogic>

<junction id="n1" type="priority" x="300.00" y="346.40" incLanes="e1\_rev\_0 e7\_0 e6\_0" intLanes=":n1\_0\_0 :n1\_9\_0 :n1\_10\_0 :n1\_3\_0 :n1\_4\_0 :n1\_5\_0 :n1\_6\_0 :n1\_7\_0 :n1\_11\_0" shape="307.54,339.74 302.00,336.54 300.67,338.08 300.00,338.27 299.33,338.08 298.67,337.50 298.00,336.54 292.46,339.74 293.12,341.66 292.96,342.33 292.46,342.82 291.62,343.10 290.46,343.20 290.46,349.60 294.10,349.33 297.42,348.50 300.42,347.13 303.11,345.22 305.49,342.75">

<request index="0" response="000000000" foes="100010000" cont="0"/>

<request index="1" response="011000000" foes="011110000" cont="1"/>

<request index="2" response="010001000" foes="010001000" cont="1"/>

<request index="3" response="010000000" foes="010000100" cont="0"/>

<request index="4" response="010000011" foes="110000011" cont="0"/>

<request index="5" response="001000010" foes="001000010" cont="0"/>

<request index="6" response="000000000" foes="000100010" cont="0"/>

<request index="7" response="000000000" foes="000011110" cont="0"/>

<request index="8" response="000010001" foes="000010001" cont="1"/>

</junction>

<junction id="n2" type="priority" x="400.00" y="173.20" incLanes="e2\_rev\_0 e8\_0 e1\_0" intLanes=":n2\_0\_0 :n2\_9\_0 :n2\_10\_0 :n2\_3\_0 :n2\_4\_0 :n2\_5\_0 :n2\_6\_0 :n2\_7\_0 :n2\_11\_0" shape="398.00,163.34 392.46,166.54 393.12,168.46 392.96,169.13 392.46,169.62 391.62,169.90 390.46,170.00 390.46,176.40 392.46,176.78 392.96,177.27 393.12,177.94 392.96,178.81 392.46,179.86 398.00,183.06 399.58,179.78 400.53,176.49 400.85,173.20 400.53,169.91 399.58,166.62">

<request index="0" response="000000000" foes="100010000" cont="0"/>

<request index="1" response="011000000" foes="011110000" cont="1"/>

<request index="2" response="010001000" foes="010001000" cont="1"/>

<request index="3" response="010000000" foes="010000100" cont="0"/>

<request index="4" response="010000011" foes="110000011" cont="0"/>

<request index="5" response="001000010" foes="001000010" cont="0"/>

<request index="6" response="000000000" foes="000100010" cont="0"/>

<request index="7" response="000000000" foes="000011110" cont="0"/>

<request index="8" response="000010001" foes="000010001" cont="1"/>

</junction>

<junction id="n3" type="priority" x="300.00" y="0.00" incLanes="e2\_0 e3\_rev\_0 e9\_0" intLanes=":n3\_0\_0 :n3\_1\_0 :n3\_9\_0 :n3\_3\_0 :n3\_10\_0 :n3\_11\_0 :n3\_6\_0 :n3\_7\_0 :n3\_8\_0" shape="302.00,9.86 307.54,6.66 305.49,3.65 303.11,1.18 300.42,-0.73 297.42,-2.10 294.10,-2.93 290.46,-3.20 290.46,3.20 292.46,3.58 292.96,4.07 293.12,4.74 292.96,5.61 292.46,6.66 298.00,9.86 299.33,8.32 300.00,8.13 300.67,8.32 301.33,8.90">

<request index="0" response="000000000" foes="100010000" cont="0"/>

<request index="1" response="000000000" foes="011110000" cont="0"/>

<request index="2" response="010001000" foes="010001000" cont="1"/>

<request index="3" response="000000000" foes="010000100" cont="0"/>

<request index="4" response="000000011" foes="110000011" cont="1"/>

<request index="5" response="001000010" foes="001000010" cont="1"/>

<request index="6" response="000000010" foes="000100010" cont="0"/>

<request index="7" response="000011010" foes="000011110" cont="0"/>

<request index="8" response="000010001" foes="000010001" cont="0"/>

</junction>

<junction id="n4" type="priority" x="100.00" y="0.00" incLanes="e10\_0 e3\_0 e4\_rev\_0" intLanes=":n4\_0\_0 :n4\_1\_0 :n4\_2\_0 :n4\_3\_0 :n4\_4\_0 :n4\_9\_0 :n4\_6\_0 :n4\_10\_0 :n4\_11\_0" shape="102.00,9.86 107.54,6.66 106.88,4.74 107.04,4.07 107.54,3.58 108.38,3.30 109.54,3.20 109.54,-3.20 105.90,-2.93 102.58,-2.10 99.58,-0.73 96.89,1.18 94.51,3.65 92.46,6.66 98.00,9.86 99.33,8.32 100.00,8.13 100.67,8.32 101.33,8.90">

<request index="0" response="000010000" foes="100010000" cont="0"/>

<request index="1" response="011010000" foes="011110000" cont="0"/>

<request index="2" response="010001000" foes="010001000" cont="0"/>

<request index="3" response="000000000" foes="010000100" cont="0"/>

<request index="4" response="000000000" foes="110000011" cont="0"/>

<request index="5" response="001000010" foes="001000010" cont="1"/>

<request index="6" response="000000000" foes="000100010" cont="0"/>

<request index="7" response="000011000" foes="000011110" cont="1"/>

<request index="8" response="000010001" foes="000010001" cont="1"/>

</junction>

<junction id="n5" type="priority" x="0.00" y="173.20" incLanes="e5\_rev\_0 e11\_0 e4\_0" intLanes=":n5\_0\_0 :n5\_9\_0 :n5\_10\_0 :n5\_3\_0 :n5\_4\_0 :n5\_5\_0 :n5\_6\_0 :n5\_7\_0 :n5\_11\_0" shape="2.00,183.06 7.54,179.86 6.88,177.94 7.04,177.27 7.54,176.78 8.38,176.50 9.54,176.40 9.54,170.00 7.54,169.62 7.04,169.13 6.88,168.46 7.04,167.59 7.54,166.54 2.00,163.34 0.42,166.62 -0.53,169.91 -0.85,173.20 -0.53,176.49 0.42,179.78">

<request index="0" response="000000000" foes="100010000" cont="0"/>

<request index="1" response="011000000" foes="011110000" cont="1"/>

<request index="2" response="010001000" foes="010001000" cont="1"/>

<request index="3" response="010000000" foes="010000100" cont="0"/>

<request index="4" response="010000011" foes="110000011" cont="0"/>

<request index="5" response="001000010" foes="001000010" cont="0"/>

<request index="6" response="000000000" foes="000100010" cont="0"/>

<request index="7" response="000000000" foes="000011110" cont="0"/>

<request index="8" response="000010001" foes="000010001" cont="1"/>

</junction>

<junction id="n6" type="priority" x="100.00" y="346.40" incLanes="e6\_rev\_0 e12\_0 e5\_0" intLanes=":n6\_0\_0 :n6\_9\_0 :n6\_10\_0 :n6\_3\_0 :n6\_4\_0 :n6\_5\_0 :n6\_6\_0 :n6\_7\_0 :n6\_11\_0" shape="109.54,349.60 109.54,343.20 107.54,342.82 107.04,342.33 106.88,341.66 107.04,340.79 107.54,339.74 102.00,336.54 100.67,338.08 100.00,338.27 99.33,338.08 98.67,337.50 98.00,336.54 92.46,339.74 94.51,342.75 96.89,345.22 99.58,347.13 102.58,348.50 105.90,349.33">

<request index="0" response="000000000" foes="100010000" cont="0"/>

<request index="1" response="011000000" foes="011110000" cont="1"/>

<request index="2" response="010001000" foes="010001000" cont="1"/>

<request index="3" response="010000000" foes="010000100" cont="0"/>

<request index="4" response="010000011" foes="110000011" cont="0"/>

<request index="5" response="001000010" foes="001000010" cont="0"/>

<request index="6" response="000000000" foes="000100010" cont="0"/>

<request index="7" response="000000000" foes="000011110" cont="0"/>

<request index="8" response="000010001" foes="000010001" cont="1"/>

</junction>

<junction id="n\_center" type="traffic\_light" x="200.00" y="173.20" incLanes="e7\_rev\_0 e8\_rev\_0 e9\_rev\_0 e10\_rev\_0 e11\_rev\_0 e12\_rev\_0" intLanes=":n\_center\_0\_0 :n\_center\_1\_0 :n\_center\_2\_0 :n\_center\_36\_0 :n\_center\_37\_0 :n\_center\_38\_0 :n\_center\_6\_0 :n\_center\_7\_0 :n\_center\_8\_0 :n\_center\_39\_0 :n\_center\_40\_0 :n\_center\_41\_0 :n\_center\_12\_0 :n\_center\_13\_0 :n\_center\_14\_0 :n\_center\_42\_0 :n\_center\_43\_0 :n\_center\_44\_0 :n\_center\_18\_0 :n\_center\_19\_0 :n\_center\_20\_0 :n\_center\_45\_0 :n\_center\_46\_0 :n\_center\_47\_0 :n\_center\_24\_0 :n\_center\_25\_0 :n\_center\_26\_0 :n\_center\_48\_0 :n\_center\_49\_0 :n\_center\_50\_0 :n\_center\_30\_0 :n\_center\_31\_0 :n\_center\_32\_0 :n\_center\_51\_0 :n\_center\_52\_0 :n\_center\_53\_0" shape="210.00,196.92 215.54,193.72 213.04,188.43 212.21,184.10 213.04,180.73 215.54,178.32 219.71,176.88 225.54,176.40 225.54,170.00 219.71,169.52 215.54,168.08 213.04,165.67 212.21,162.30 213.04,157.97 215.54,152.68 210.00,149.48 206.98,153.49 203.58,155.89 200.00,156.70 196.42,155.89 193.02,153.49 190.00,149.48 184.46,152.68 186.96,157.97 187.79,162.30 186.96,165.67 184.46,168.08 180.29,169.52 174.46,170.00 174.46,176.40 180.29,176.88 184.46,178.32 186.96,180.73 187.79,184.10 186.96,188.43 184.46,193.72 190.00,196.92 193.02,192.91 196.42,190.51 200.00,189.70 203.58,190.51 206.98,192.91" radius="20.00">

<request index="0" response="000000010000000000000100000010000000" foes="100000010000001000000100000010000000" cont="0"/>

<request index="1" response="011111010000000000001100000110000000" foes="011111110000011000001100000110000000" cont="0"/>

<request index="2" response="011100001111000000011000001110000000" foes="011110001111111000011100001110000000" cont="0"/>

<request index="3" response="011100001110000111011100011110000000" foes="011100001110000111111100011110000000" cont="1"/>

<request index="4" response="011000001100000110000011011110000000" foes="011000001100000110000011111110000000" cont="1"/>

<request index="5" response="010000001000000100000010000001000000" foes="010000001000000100000010000001000000" cont="1"/>

<request index="6" response="010000000000000000000000000000000000" foes="010000001000000100000010000000100000" cont="0"/>

<request index="7" response="010000000000001000000000000000011000" foes="110000011000001100000110000000011111" cont="0"/>

<request index="8" response="001000000000011000001000000000011000" foes="001111111000011100001110000000011110" cont="0"/>

<request index="9" response="001000000111011000011000000000011000" foes="001110000111111100011110000000011100" cont="1"/>

<request index="10" response="001000000110000000011000000000011000" foes="001100000110000011111110000000011000" cont="1"/>

<request index="11" response="001000000100000010000001000000010000" foes="001000000100000010000001000000010000" cont="1"/>

<request index="12" response="000000000100000000000000000000010000" foes="001000000100000010000000100000010000" cont="0"/>

<request index="13" response="000000001100000100000000011111010000" foes="011000001100000110000000011111110000" cont="0"/>

<request index="14" response="000000011100001000000000011110001100" foes="111000011100001110000000011110001111" cont="0"/>

<request index="15" response="000111011100011100000000011100001110" foes="000111111100011110000000011100001110" cont="1"/>

<request index="16" response="000110000011011100000000011000001100" foes="000110000011111110000000011000001100" cont="1"/>

<request index="17" response="000100000010000001000000010000001000" foes="000100000010000001000000010000001000" cont="1"/>

<request index="18" response="000100000010000000000000010000000000" foes="000100000010000000100000010000001000" cont="0"/>

<request index="19" response="001100000110000000011111010000000000" foes="001100000110000000011111110000011000" cont="0"/>

<request index="20" response="011000001110000000011100001111000000" foes="011100001110000000011110001111111000" cont="0"/>

<request index="21" response="011100011110000000011100001110000111" foes="111100011110000000011100001110000111" cont="1"/>

<request index="22" response="000011011110000000011000001100000110" foes="000011111110000000011000001100000110" cont="1"/>

<request index="23" response="000010000001000000010000001000000100" foes="000010000001000000010000001000000100" cont="1"/>

<request index="24" response="000000000000000000010000000000000000" foes="000010000000100000010000001000000100" cont="0"/>

<request index="25" response="000000000000011000010000000000001000" foes="000110000000011111110000011000001100" cont="0"/>

<request index="26" response="001000000000011000001000000000011000" foes="001110000000011110001111111000011100" cont="0"/>

<request index="27" response="011000000000011000001000000111011000" foes="011110000000011100001110000111111100" cont="1"/>

<request index="28" response="011000000000011000001000000110000000" foes="111110000000011000001100000110000011" cont="1"/>

<request index="29" response="000001000000010000001000000100000010" foes="000001000000010000001000000100000010" cont="1"/>

<request index="30" response="000000000000010000000000000100000000" foes="000000100000010000001000000100000010" cont="0"/>

<request index="31" response="000000011111010000000000001100000100" foes="000000011111110000011000001100000110" cont="0"/>

<request index="32" response="000000011110001100000000011100001000" foes="000000011110001111111000011100001110" cont="0"/>

<request index="33" response="000000011100001110000111011100011100" foes="000000011100001110000111111100011110" cont="1"/>

<request index="34" response="000000011000001100000110000011011100" foes="000000011000001100000110000011111110" cont="1"/>

<request index="35" response="000000010000001000000100000010000001" foes="000000010000001000000100000010000001" cont="1"/>

</junction>

<junction id=":n1\_9\_0" type="internal" x="301.09" y="343.80" incLanes=":n1\_1\_0 e6\_0" intLanes=":n1\_4\_0 :n1\_5\_0 :n1\_6\_0 :n1\_7\_0"/>

<junction id=":n1\_10\_0" type="internal" x="304.86" y="339.58" incLanes=":n1\_2\_0 e6\_0 e7\_0" intLanes=":n1\_3\_0 :n1\_7\_0"/>

<junction id=":n1\_11\_0" type="internal" x="291.56" y="345.53" incLanes=":n1\_8\_0 e1\_rev\_0 e7\_0" intLanes=":n1\_0\_0 :n1\_4\_0"/>

<junction id=":n2\_9\_0" type="internal" x="398.29" y="170.96" incLanes=":n2\_1\_0 e1\_0" intLanes=":n2\_4\_0 :n2\_5\_0 :n2\_6\_0 :n2\_7\_0"/>

<junction id=":n2\_10\_0" type="internal" x="396.52" y="165.58" incLanes=":n2\_2\_0 e1\_0 e8\_0" intLanes=":n2\_3\_0 :n2\_7\_0"/>

<junction id=":n2\_11\_0" type="internal" x="395.03" y="180.08" incLanes=":n2\_8\_0 e2\_rev\_0 e8\_0" intLanes=":n2\_0\_0 :n2\_4\_0"/>

<junction id=":n3\_9\_0" type="internal" x="303.47" y="7.74" incLanes=":n3\_2\_0 e3\_rev\_0 e9\_0" intLanes=":n3\_3\_0 :n3\_7\_0"/>

<junction id=":n3\_10\_0" type="internal" x="297.20" y="0.36" incLanes=":n3\_4\_0 e2\_0" intLanes=":n3\_0\_0 :n3\_1\_0 :n3\_7\_0 :n3\_8\_0"/>

<junction id=":n3\_11\_0" type="internal" x="291.66" y="-0.80" incLanes=":n3\_5\_0 e2\_0 e9\_0" intLanes=":n3\_1\_0 :n3\_6\_0"/>

<junction id=":n4\_9\_0" type="internal" x="108.44" y="0.87" incLanes=":n4\_5\_0 e10\_0 e4\_rev\_0" intLanes=":n4\_1\_0 :n4\_6\_0"/>

<junction id=":n4\_10\_0" type="internal" x="98.91" y="2.60" incLanes=":n4\_7\_0 e3\_0" intLanes=":n4\_1\_0 :n4\_2\_0 :n4\_3\_0 :n4\_4\_0"/>

<junction id=":n4\_11\_0" type="internal" x="95.14" y="6.82" incLanes=":n4\_8\_0 e10\_0 e3\_0" intLanes=":n4\_0\_0 :n4\_4\_0"/>

<junction id=":n5\_9\_0" type="internal" x="1.71" y="175.44" incLanes=":n5\_1\_0 e4\_0" intLanes=":n5\_4\_0 :n5\_5\_0 :n5\_6\_0 :n5\_7\_0"/>

<junction id=":n5\_10\_0" type="internal" x="3.48" y="180.82" incLanes=":n5\_2\_0 e11\_0 e4\_0" intLanes=":n5\_3\_0 :n5\_7\_0"/>

<junction id=":n5\_11\_0" type="internal" x="4.97" y="166.32" incLanes=":n5\_8\_0 e11\_0 e5\_rev\_0" intLanes=":n5\_0\_0 :n5\_4\_0"/>

<junction id=":n6\_9\_0" type="internal" x="102.80" y="346.04" incLanes=":n6\_1\_0 e5\_0" intLanes=":n6\_4\_0 :n6\_5\_0 :n6\_6\_0 :n6\_7\_0"/>

<junction id=":n6\_10\_0" type="internal" x="108.34" y="347.20" incLanes=":n6\_2\_0 e12\_0 e5\_0" intLanes=":n6\_3\_0 :n6\_7\_0"/>

<junction id=":n6\_11\_0" type="internal" x="96.53" y="338.66" incLanes=":n6\_8\_0 e12\_0 e6\_rev\_0" intLanes=":n6\_0\_0 :n6\_4\_0"/>

<junction id=":n\_center\_36\_0" type="internal" x="208.94" y="190.47" incLanes=":n\_center\_3\_0 e10\_rev\_0" intLanes=":n\_center\_7\_0 :n\_center\_8\_0 :n\_center\_9\_0 :n\_center\_10\_0 :n\_center\_14\_0 :n\_center\_15\_0 :n\_center\_16\_0 :n\_center\_17\_0 :n\_center\_18\_0 :n\_center\_19\_0 :n\_center\_20\_0 :n\_center\_25\_0 :n\_center\_26\_0 :n\_center\_27\_0 :n\_center\_32\_0 :n\_center\_33\_0 :n\_center\_34\_0"/>

<junction id=":n\_center\_37\_0" type="internal" x="210.15" y="192.54" incLanes=":n\_center\_4\_0 e10\_rev\_0" intLanes=":n\_center\_7\_0 :n\_center\_8\_0 :n\_center\_9\_0 :n\_center\_10\_0 :n\_center\_11\_0 :n\_center\_12\_0 :n\_center\_13\_0 :n\_center\_19\_0 :n\_center\_20\_0 :n\_center\_26\_0 :n\_center\_27\_0 :n\_center\_33\_0 :n\_center\_34\_0"/>

<junction id=":n\_center\_38\_0" type="internal" x="211.48" y="194.68" incLanes=":n\_center\_5\_0 e10\_rev\_0 e11\_rev\_0 e12\_rev\_0 e8\_rev\_0 e9\_rev\_0" intLanes=":n\_center\_6\_0 :n\_center\_13\_0 :n\_center\_20\_0 :n\_center\_27\_0 :n\_center\_34\_0"/>

<junction id=":n\_center\_39\_0" type="internal" x="219.43" y="174.09" incLanes=":n\_center\_9\_0 e11\_rev\_0" intLanes=":n\_center\_2\_0 :n\_center\_3\_0 :n\_center\_4\_0 :n\_center\_13\_0 :n\_center\_14\_0 :n\_center\_15\_0 :n\_center\_16\_0 :n\_center\_20\_0 :n\_center\_21\_0 :n\_center\_22\_0 :n\_center\_23\_0 :n\_center\_24\_0 :n\_center\_25\_0 :n\_center\_26\_0 :n\_center\_31\_0 :n\_center\_32\_0 :n\_center\_33\_0"/>

<junction id=":n\_center\_40\_0" type="internal" x="221.82" y="174.08" incLanes=":n\_center\_10\_0 e11\_rev\_0" intLanes=":n\_center\_3\_0 :n\_center\_4\_0 :n\_center\_13\_0 :n\_center\_14\_0 :n\_center\_15\_0 :n\_center\_16\_0 :n\_center\_17\_0 :n\_center\_18\_0 :n\_center\_19\_0 :n\_center\_25\_0 :n\_center\_26\_0 :n\_center\_32\_0 :n\_center\_33\_0"/>

<junction id=":n\_center\_41\_0" type="internal" x="224.34" y="174.00" incLanes=":n\_center\_11\_0 e10\_rev\_0 e11\_rev\_0 e12\_rev\_0 e7\_rev\_0 e9\_rev\_0" intLanes=":n\_center\_4\_0 :n\_center\_12\_0 :n\_center\_19\_0 :n\_center\_26\_0 :n\_center\_33\_0"/>

<junction id=":n\_center\_42\_0" type="internal" x="210.49" y="156.82" incLanes=":n\_center\_15\_0 e12\_rev\_0" intLanes=":n\_center\_1\_0 :n\_center\_2\_0 :n\_center\_3\_0 :n\_center\_8\_0 :n\_center\_9\_0 :n\_center\_10\_0 :n\_center\_19\_0 :n\_center\_20\_0 :n\_center\_21\_0 :n\_center\_22\_0 :n\_center\_26\_0 :n\_center\_27\_0 :n\_center\_28\_0 :n\_center\_29\_0 :n\_center\_30\_0 :n\_center\_31\_0 :n\_center\_32\_0"/>

<junction id=":n\_center\_43\_0" type="internal" x="211.68" y="154.74" incLanes=":n\_center\_16\_0 e12\_rev\_0" intLanes=":n\_center\_2\_0 :n\_center\_3\_0 :n\_center\_9\_0 :n\_center\_10\_0 :n\_center\_19\_0 :n\_center\_20\_0 :n\_center\_21\_0 :n\_center\_22\_0 :n\_center\_23\_0 :n\_center\_24\_0 :n\_center\_25\_0 :n\_center\_31\_0 :n\_center\_32\_0"/>

<junction id=":n\_center\_44\_0" type="internal" x="212.86" y="152.52" incLanes=":n\_center\_17\_0 e10\_rev\_0 e11\_rev\_0 e12\_rev\_0 e7\_rev\_0 e8\_rev\_0" intLanes=":n\_center\_3\_0 :n\_center\_10\_0 :n\_center\_18\_0 :n\_center\_25\_0 :n\_center\_32\_0"/>

<junction id=":n\_center\_45\_0" type="internal" x="191.06" y="155.93" incLanes=":n\_center\_21\_0 e7\_rev\_0" intLanes=":n\_center\_0\_0 :n\_center\_1\_0 :n\_center\_2\_0 :n\_center\_7\_0 :n\_center\_8\_0 :n\_center\_9\_0 :n\_center\_14\_0 :n\_center\_15\_0 :n\_center\_16\_0 :n\_center\_25\_0 :n\_center\_26\_0 :n\_center\_27\_0 :n\_center\_28\_0 :n\_center\_32\_0 :n\_center\_33\_0 :n\_center\_34\_0 :n\_center\_35\_0"/>

<junction id=":n\_center\_46\_0" type="internal" x="189.85" y="153.86" incLanes=":n\_center\_22\_0 e7\_rev\_0" intLanes=":n\_center\_1\_0 :n\_center\_2\_0 :n\_center\_8\_0 :n\_center\_9\_0 :n\_center\_15\_0 :n\_center\_16\_0 :n\_center\_25\_0 :n\_center\_26\_0 :n\_center\_27\_0 :n\_center\_28\_0 :n\_center\_29\_0 :n\_center\_30\_0 :n\_center\_31\_0"/>

<junction id=":n\_center\_47\_0" type="internal" x="188.52" y="151.72" incLanes=":n\_center\_23\_0 e11\_rev\_0 e12\_rev\_0 e7\_rev\_0 e8\_rev\_0 e9\_rev\_0" intLanes=":n\_center\_2\_0 :n\_center\_9\_0 :n\_center\_16\_0 :n\_center\_24\_0 :n\_center\_31\_0"/>

<junction id=":n\_center\_48\_0" type="internal" x="180.57" y="172.31" incLanes=":n\_center\_27\_0 e8\_rev\_0" intLanes=":n\_center\_2\_0 :n\_center\_3\_0 :n\_center\_4\_0 :n\_center\_5\_0 :n\_center\_6\_0 :n\_center\_7\_0 :n\_center\_8\_0 :n\_center\_13\_0 :n\_center\_14\_0 :n\_center\_15\_0 :n\_center\_20\_0 :n\_center\_21\_0 :n\_center\_22\_0 :n\_center\_31\_0 :n\_center\_32\_0 :n\_center\_33\_0 :n\_center\_34\_0"/>

<junction id=":n\_center\_49\_0" type="internal" x="178.18" y="172.32" incLanes=":n\_center\_28\_0 e8\_rev\_0" intLanes=":n\_center\_0\_0 :n\_center\_1\_0 :n\_center\_7\_0 :n\_center\_8\_0 :n\_center\_14\_0 :n\_center\_15\_0 :n\_center\_21\_0 :n\_center\_22\_0 :n\_center\_31\_0 :n\_center\_32\_0 :n\_center\_33\_0 :n\_center\_34\_0 :n\_center\_35\_0"/>

<junction id=":n\_center\_50\_0" type="internal" x="175.66" y="172.40" incLanes=":n\_center\_29\_0 e10\_rev\_0 e12\_rev\_0 e7\_rev\_0 e8\_rev\_0 e9\_rev\_0" intLanes=":n\_center\_1\_0 :n\_center\_8\_0 :n\_center\_15\_0 :n\_center\_22\_0 :n\_center\_30\_0"/>

<junction id=":n\_center\_51\_0" type="internal" x="189.51" y="189.58" incLanes=":n\_center\_33\_0 e9\_rev\_0" intLanes=":n\_center\_1\_0 :n\_center\_2\_0 :n\_center\_3\_0 :n\_center\_4\_0 :n\_center\_8\_0 :n\_center\_9\_0 :n\_center\_10\_0 :n\_center\_11\_0 :n\_center\_12\_0 :n\_center\_13\_0 :n\_center\_14\_0 :n\_center\_19\_0 :n\_center\_20\_0 :n\_center\_21\_0 :n\_center\_26\_0 :n\_center\_27\_0 :n\_center\_28\_0"/>

<junction id=":n\_center\_52\_0" type="internal" x="188.32" y="191.66" incLanes=":n\_center\_34\_0 e9\_rev\_0" intLanes=":n\_center\_1\_0 :n\_center\_2\_0 :n\_center\_3\_0 :n\_center\_4\_0 :n\_center\_5\_0 :n\_center\_6\_0 :n\_center\_7\_0 :n\_center\_13\_0 :n\_center\_14\_0 :n\_center\_20\_0 :n\_center\_21\_0 :n\_center\_27\_0 :n\_center\_28\_0"/>

<junction id=":n\_center\_53\_0" type="internal" x="187.14" y="193.88" incLanes=":n\_center\_35\_0 e10\_rev\_0 e11\_rev\_0 e7\_rev\_0 e8\_rev\_0 e9\_rev\_0" intLanes=":n\_center\_0\_0 :n\_center\_7\_0 :n\_center\_14\_0 :n\_center\_21\_0 :n\_center\_28\_0"/>

<connection from="e1" to="e8\_rev" fromLane="0" toLane="0" via=":n2\_6\_0" dir="r" state="M"/>

<connection from="e1" to="e2" fromLane="0" toLane="0" via=":n2\_7\_0" dir="R" state="M"/>

<connection from="e1" to="e1\_rev" fromLane="0" toLane="0" via=":n2\_8\_0" dir="t" state="m"/>

<connection from="e10" to="e4" fromLane="0" toLane="0" via=":n4\_0\_0" dir="r" state="m"/>

<connection from="e10" to="e3\_rev" fromLane="0" toLane="0" via=":n4\_1\_0" dir="l" state="m"/>

<connection from="e10" to="e10\_rev" fromLane="0" toLane="0" via=":n4\_2\_0" dir="t" state="m"/>

<connection from="e10\_rev" to="e9" fromLane="0" toLane="0" via=":n\_center\_18\_0" tl="n\_center" linkIndex="18" dir="r" state="o"/>

<connection from="e10\_rev" to="e8" fromLane="0" toLane="0" via=":n\_center\_19\_0" tl="n\_center" linkIndex="19" dir="R" state="o"/>

<connection from="e10\_rev" to="e7" fromLane="0" toLane="0" via=":n\_center\_20\_0" tl="n\_center" linkIndex="20" dir="s" state="o"/>

<connection from="e10\_rev" to="e12" fromLane="0" toLane="0" via=":n\_center\_21\_0" tl="n\_center" linkIndex="21" dir="L" state="o"/>

<connection from="e10\_rev" to="e11" fromLane="0" toLane="0" via=":n\_center\_22\_0" tl="n\_center" linkIndex="22" dir="l" state="o"/>

<connection from="e10\_rev" to="e10" fromLane="0" toLane="0" via=":n\_center\_23\_0" tl="n\_center" linkIndex="23" dir="t" state="o"/>

<connection from="e11" to="e5" fromLane="0" toLane="0" via=":n5\_3\_0" dir="r" state="m"/>

<connection from="e11" to="e4\_rev" fromLane="0" toLane="0" via=":n5\_4\_0" dir="l" state="m"/>

<connection from="e11" to="e11\_rev" fromLane="0" toLane="0" via=":n5\_5\_0" dir="t" state="m"/>

<connection from="e11\_rev" to="e10" fromLane="0" toLane="0" via=":n\_center\_24\_0" tl="n\_center" linkIndex="24" dir="r" state="O"/>

<connection from="e11\_rev" to="e9" fromLane="0" toLane="0" via=":n\_center\_25\_0" tl="n\_center" linkIndex="25" dir="R" state="O"/>

<connection from="e11\_rev" to="e8" fromLane="0" toLane="0" via=":n\_center\_26\_0" tl="n\_center" linkIndex="26" dir="s" state="O"/>

<connection from="e11\_rev" to="e7" fromLane="0" toLane="0" via=":n\_center\_27\_0" tl="n\_center" linkIndex="27" dir="L" state="o"/>

<connection from="e11\_rev" to="e12" fromLane="0" toLane="0" via=":n\_center\_28\_0" tl="n\_center" linkIndex="28" dir="l" state="o"/>

<connection from="e11\_rev" to="e11" fromLane="0" toLane="0" via=":n\_center\_29\_0" tl="n\_center" linkIndex="29" dir="t" state="o"/>

<connection from="e12" to="e6" fromLane="0" toLane="0" via=":n6\_3\_0" dir="r" state="m"/>

<connection from="e12" to="e5\_rev" fromLane="0" toLane="0" via=":n6\_4\_0" dir="l" state="m"/>

<connection from="e12" to="e12\_rev" fromLane="0" toLane="0" via=":n6\_5\_0" dir="t" state="m"/>

<connection from="e12\_rev" to="e11" fromLane="0" toLane="0" via=":n\_center\_30\_0" tl="n\_center" linkIndex="30" dir="r" state="o"/>

<connection from="e12\_rev" to="e10" fromLane="0" toLane="0" via=":n\_center\_31\_0" tl="n\_center" linkIndex="31" dir="R" state="o"/>

<connection from="e12\_rev" to="e9" fromLane="0" toLane="0" via=":n\_center\_32\_0" tl="n\_center" linkIndex="32" dir="s" state="o"/>

<connection from="e12\_rev" to="e8" fromLane="0" toLane="0" via=":n\_center\_33\_0" tl="n\_center" linkIndex="33" dir="L" state="o"/>

<connection from="e12\_rev" to="e7" fromLane="0" toLane="0" via=":n\_center\_34\_0" tl="n\_center" linkIndex="34" dir="l" state="o"/>

<connection from="e12\_rev" to="e12" fromLane="0" toLane="0" via=":n\_center\_35\_0" tl="n\_center" linkIndex="35" dir="t" state="o"/>

<connection from="e1\_rev" to="e6\_rev" fromLane="0" toLane="0" via=":n1\_0\_0" dir="L" state="M"/>

<connection from="e1\_rev" to="e7\_rev" fromLane="0" toLane="0" via=":n1\_1\_0" dir="l" state="m"/>

<connection from="e1\_rev" to="e1" fromLane="0" toLane="0" via=":n1\_2\_0" dir="t" state="m"/>

<connection from="e2" to="e9\_rev" fromLane="0" toLane="0" via=":n3\_0\_0" dir="r" state="M"/>

<connection from="e2" to="e3" fromLane="0" toLane="0" via=":n3\_1\_0" dir="R" state="M"/>

<connection from="e2" to="e2\_rev" fromLane="0" toLane="0" via=":n3\_2\_0" dir="t" state="m"/>

<connection from="e2\_rev" to="e1\_rev" fromLane="0" toLane="0" via=":n2\_0\_0" dir="L" state="M"/>

<connection from="e2\_rev" to="e8\_rev" fromLane="0" toLane="0" via=":n2\_1\_0" dir="l" state="m"/>

<connection from="e2\_rev" to="e2" fromLane="0" toLane="0" via=":n2\_2\_0" dir="t" state="m"/>

<connection from="e3" to="e10\_rev" fromLane="0" toLane="0" via=":n4\_3\_0" dir="r" state="M"/>

<connection from="e3" to="e4" fromLane="0" toLane="0" via=":n4\_4\_0" dir="R" state="M"/>

<connection from="e3" to="e3\_rev" fromLane="0" toLane="0" via=":n4\_5\_0" dir="t" state="m"/>

<connection from="e3\_rev" to="e2\_rev" fromLane="0" toLane="0" via=":n3\_3\_0" dir="L" state="M"/>

<connection from="e3\_rev" to="e9\_rev" fromLane="0" toLane="0" via=":n3\_4\_0" dir="l" state="m"/>

<connection from="e3\_rev" to="e3" fromLane="0" toLane="0" via=":n3\_5\_0" dir="t" state="m"/>

<connection from="e4" to="e11\_rev" fromLane="0" toLane="0" via=":n5\_6\_0" dir="r" state="M"/>

<connection from="e4" to="e5" fromLane="0" toLane="0" via=":n5\_7\_0" dir="R" state="M"/>

<connection from="e4" to="e4\_rev" fromLane="0" toLane="0" via=":n5\_8\_0" dir="t" state="m"/>

<connection from="e4\_rev" to="e3\_rev" fromLane="0" toLane="0" via=":n4\_6\_0" dir="L" state="M"/>

<connection from="e4\_rev" to="e10\_rev" fromLane="0" toLane="0" via=":n4\_7\_0" dir="l" state="m"/>

<connection from="e4\_rev" to="e4" fromLane="0" toLane="0" via=":n4\_8\_0" dir="t" state="m"/>

<connection from="e5" to="e12\_rev" fromLane="0" toLane="0" via=":n6\_6\_0" dir="r" state="M"/>

<connection from="e5" to="e6" fromLane="0" toLane="0" via=":n6\_7\_0" dir="R" state="M"/>

<connection from="e5" to="e5\_rev" fromLane="0" toLane="0" via=":n6\_8\_0" dir="t" state="m"/>

<connection from="e5\_rev" to="e4\_rev" fromLane="0" toLane="0" via=":n5\_0\_0" dir="L" state="M"/>

<connection from="e5\_rev" to="e11\_rev" fromLane="0" toLane="0" via=":n5\_1\_0" dir="l" state="m"/>

<connection from="e5\_rev" to="e5" fromLane="0" toLane="0" via=":n5\_2\_0" dir="t" state="m"/>

<connection from="e6" to="e7\_rev" fromLane="0" toLane="0" via=":n1\_6\_0" dir="r" state="M"/>

<connection from="e6" to="e1" fromLane="0" toLane="0" via=":n1\_7\_0" dir="R" state="M"/>

<connection from="e6" to="e6\_rev" fromLane="0" toLane="0" via=":n1\_8\_0" dir="t" state="m"/>

<connection from="e6\_rev" to="e5\_rev" fromLane="0" toLane="0" via=":n6\_0\_0" dir="L" state="M"/>

<connection from="e6\_rev" to="e12\_rev" fromLane="0" toLane="0" via=":n6\_1\_0" dir="l" state="m"/>

<connection from="e6\_rev" to="e6" fromLane="0" toLane="0" via=":n6\_2\_0" dir="t" state="m"/>

<connection from="e7" to="e1" fromLane="0" toLane="0" via=":n1\_3\_0" dir="r" state="m"/>

<connection from="e7" to="e6\_rev" fromLane="0" toLane="0" via=":n1\_4\_0" dir="l" state="m"/>

<connection from="e7" to="e7\_rev" fromLane="0" toLane="0" via=":n1\_5\_0" dir="t" state="m"/>

<connection from="e7\_rev" to="e12" fromLane="0" toLane="0" via=":n\_center\_0\_0" tl="n\_center" linkIndex="0" dir="r" state="o"/>

<connection from="e7\_rev" to="e11" fromLane="0" toLane="0" via=":n\_center\_1\_0" tl="n\_center" linkIndex="1" dir="R" state="o"/>

<connection from="e7\_rev" to="e10" fromLane="0" toLane="0" via=":n\_center\_2\_0" tl="n\_center" linkIndex="2" dir="s" state="o"/>

<connection from="e7\_rev" to="e9" fromLane="0" toLane="0" via=":n\_center\_3\_0" tl="n\_center" linkIndex="3" dir="L" state="o"/>

<connection from="e7\_rev" to="e8" fromLane="0" toLane="0" via=":n\_center\_4\_0" tl="n\_center" linkIndex="4" dir="l" state="o"/>

<connection from="e7\_rev" to="e7" fromLane="0" toLane="0" via=":n\_center\_5\_0" tl="n\_center" linkIndex="5" dir="t" state="o"/>

<connection from="e8" to="e2" fromLane="0" toLane="0" via=":n2\_3\_0" dir="r" state="m"/>

<connection from="e8" to="e1\_rev" fromLane="0" toLane="0" via=":n2\_4\_0" dir="l" state="m"/>

<connection from="e8" to="e8\_rev" fromLane="0" toLane="0" via=":n2\_5\_0" dir="t" state="m"/>

<connection from="e8\_rev" to="e7" fromLane="0" toLane="0" via=":n\_center\_6\_0" tl="n\_center" linkIndex="6" dir="r" state="O"/>

<connection from="e8\_rev" to="e12" fromLane="0" toLane="0" via=":n\_center\_7\_0" tl="n\_center" linkIndex="7" dir="R" state="O"/>

<connection from="e8\_rev" to="e11" fromLane="0" toLane="0" via=":n\_center\_8\_0" tl="n\_center" linkIndex="8" dir="s" state="O"/>

<connection from="e8\_rev" to="e10" fromLane="0" toLane="0" via=":n\_center\_9\_0" tl="n\_center" linkIndex="9" dir="L" state="o"/>

<connection from="e8\_rev" to="e9" fromLane="0" toLane="0" via=":n\_center\_10\_0" tl="n\_center" linkIndex="10" dir="l" state="o"/>

<connection from="e8\_rev" to="e8" fromLane="0" toLane="0" via=":n\_center\_11\_0" tl="n\_center" linkIndex="11" dir="t" state="o"/>

<connection from="e9" to="e3" fromLane="0" toLane="0" via=":n3\_6\_0" dir="r" state="m"/>

<connection from="e9" to="e2\_rev" fromLane="0" toLane="0" via=":n3\_7\_0" dir="l" state="m"/>

<connection from="e9" to="e9\_rev" fromLane="0" toLane="0" via=":n3\_8\_0" dir="t" state="m"/>

<connection from="e9\_rev" to="e8" fromLane="0" toLane="0" via=":n\_center\_12\_0" tl="n\_center" linkIndex="12" dir="r" state="o"/>

<connection from="e9\_rev" to="e7" fromLane="0" toLane="0" via=":n\_center\_13\_0" tl="n\_center" linkIndex="13" dir="R" state="o"/>

<connection from="e9\_rev" to="e12" fromLane="0" toLane="0" via=":n\_center\_14\_0" tl="n\_center" linkIndex="14" dir="s" state="o"/>

<connection from="e9\_rev" to="e11" fromLane="0" toLane="0" via=":n\_center\_15\_0" tl="n\_center" linkIndex="15" dir="L" state="o"/>

<connection from="e9\_rev" to="e10" fromLane="0" toLane="0" via=":n\_center\_16\_0" tl="n\_center" linkIndex="16" dir="l" state="o"/>

<connection from="e9\_rev" to="e9" fromLane="0" toLane="0" via=":n\_center\_17\_0" tl="n\_center" linkIndex="17" dir="t" state="o"/>

<connection from=":n1\_0" to="e6\_rev" fromLane="0" toLane="0" dir="L" state="M"/>

<connection from=":n1\_1" to="e7\_rev" fromLane="0" toLane="0" via=":n1\_9\_0" dir="l" state="m"/>

<connection from=":n1\_9" to="e7\_rev" fromLane="0" toLane="0" dir="l" state="M"/>

<connection from=":n1\_2" to="e1" fromLane="0" toLane="0" via=":n1\_10\_0" dir="t" state="m"/>

<connection from=":n1\_10" to="e1" fromLane="0" toLane="0" dir="t" state="M"/>

<connection from=":n1\_3" to="e1" fromLane="0" toLane="0" dir="r" state="M"/>

<connection from=":n1\_4" to="e6\_rev" fromLane="0" toLane="0" dir="l" state="M"/>

<connection from=":n1\_5" to="e7\_rev" fromLane="0" toLane="0" dir="t" state="M"/>

<connection from=":n1\_6" to="e7\_rev" fromLane="0" toLane="0" dir="r" state="M"/>

<connection from=":n1\_7" to="e1" fromLane="0" toLane="0" dir="R" state="M"/>

<connection from=":n1\_8" to="e6\_rev" fromLane="0" toLane="0" via=":n1\_11\_0" dir="t" state="m"/>

<connection from=":n1\_11" to="e6\_rev" fromLane="0" toLane="0" dir="t" state="M"/>

<connection from=":n2\_0" to="e1\_rev" fromLane="0" toLane="0" dir="L" state="M"/>

<connection from=":n2\_1" to="e8\_rev" fromLane="0" toLane="0" via=":n2\_9\_0" dir="l" state="m"/>

<connection from=":n2\_9" to="e8\_rev" fromLane="0" toLane="0" dir="l" state="M"/>

<connection from=":n2\_2" to="e2" fromLane="0" toLane="0" via=":n2\_10\_0" dir="t" state="m"/>

<connection from=":n2\_10" to="e2" fromLane="0" toLane="0" dir="t" state="M"/>

<connection from=":n2\_3" to="e2" fromLane="0" toLane="0" dir="r" state="M"/>

<connection from=":n2\_4" to="e1\_rev" fromLane="0" toLane="0" dir="l" state="M"/>

<connection from=":n2\_5" to="e8\_rev" fromLane="0" toLane="0" dir="t" state="M"/>

<connection from=":n2\_6" to="e8\_rev" fromLane="0" toLane="0" dir="r" state="M"/>

<connection from=":n2\_7" to="e2" fromLane="0" toLane="0" dir="R" state="M"/>

<connection from=":n2\_8" to="e1\_rev" fromLane="0" toLane="0" via=":n2\_11\_0" dir="t" state="m"/>

<connection from=":n2\_11" to="e1\_rev" fromLane="0" toLane="0" dir="t" state="M"/>

<connection from=":n3\_0" to="e9\_rev" fromLane="0" toLane="0" dir="r" state="M"/>

<connection from=":n3\_1" to="e3" fromLane="0" toLane="0" dir="R" state="M"/>

<connection from=":n3\_2" to="e2\_rev" fromLane="0" toLane="0" via=":n3\_9\_0" dir="t" state="m"/>

<connection from=":n3\_9" to="e2\_rev" fromLane="0" toLane="0" dir="t" state="M"/>

<connection from=":n3\_3" to="e2\_rev" fromLane="0" toLane="0" dir="L" state="M"/>

<connection from=":n3\_4" to="e9\_rev" fromLane="0" toLane="0" via=":n3\_10\_0" dir="l" state="m"/>

<connection from=":n3\_10" to="e9\_rev" fromLane="0" toLane="0" dir="l" state="M"/>

<connection from=":n3\_5" to="e3" fromLane="0" toLane="0" via=":n3\_11\_0" dir="t" state="m"/>

<connection from=":n3\_11" to="e3" fromLane="0" toLane="0" dir="t" state="M"/>

<connection from=":n3\_6" to="e3" fromLane="0" toLane="0" dir="r" state="M"/>

<connection from=":n3\_7" to="e2\_rev" fromLane="0" toLane="0" dir="l" state="M"/>

<connection from=":n3\_8" to="e9\_rev" fromLane="0" toLane="0" dir="t" state="M"/>

<connection from=":n4\_0" to="e4" fromLane="0" toLane="0" dir="r" state="M"/>

<connection from=":n4\_1" to="e3\_rev" fromLane="0" toLane="0" dir="l" state="M"/>

<connection from=":n4\_2" to="e10\_rev" fromLane="0" toLane="0" dir="t" state="M"/>

<connection from=":n4\_3" to="e10\_rev" fromLane="0" toLane="0" dir="r" state="M"/>

<connection from=":n4\_4" to="e4" fromLane="0" toLane="0" dir="R" state="M"/>

<connection from=":n4\_5" to="e3\_rev" fromLane="0" toLane="0" via=":n4\_9\_0" dir="t" state="m"/>

<connection from=":n4\_9" to="e3\_rev" fromLane="0" toLane="0" dir="t" state="M"/>

<connection from=":n4\_6" to="e3\_rev" fromLane="0" toLane="0" dir="L" state="M"/>

<connection from=":n4\_7" to="e10\_rev" fromLane="0" toLane="0" via=":n4\_10\_0" dir="l" state="m"/>

<connection from=":n4\_10" to="e10\_rev" fromLane="0" toLane="0" dir="l" state="M"/>

<connection from=":n4\_8" to="e4" fromLane="0" toLane="0" via=":n4\_11\_0" dir="t" state="m"/>

<connection from=":n4\_11" to="e4" fromLane="0" toLane="0" dir="t" state="M"/>

<connection from=":n5\_0" to="e4\_rev" fromLane="0" toLane="0" dir="L" state="M"/>

<connection from=":n5\_1" to="e11\_rev" fromLane="0" toLane="0" via=":n5\_9\_0" dir="l" state="m"/>

<connection from=":n5\_9" to="e11\_rev" fromLane="0" toLane="0" dir="l" state="M"/>

<connection from=":n5\_2" to="e5" fromLane="0" toLane="0" via=":n5\_10\_0" dir="t" state="m"/>

<connection from=":n5\_10" to="e5" fromLane="0" toLane="0" dir="t" state="M"/>

<connection from=":n5\_3" to="e5" fromLane="0" toLane="0" dir="r" state="M"/>

<connection from=":n5\_4" to="e4\_rev" fromLane="0" toLane="0" dir="l" state="M"/>

<connection from=":n5\_5" to="e11\_rev" fromLane="0" toLane="0" dir="t" state="M"/>

<connection from=":n5\_6" to="e11\_rev" fromLane="0" toLane="0" dir="r" state="M"/>

<connection from=":n5\_7" to="e5" fromLane="0" toLane="0" dir="R" state="M"/>

<connection from=":n5\_8" to="e4\_rev" fromLane="0" toLane="0" via=":n5\_11\_0" dir="t" state="m"/>

<connection from=":n5\_11" to="e4\_rev" fromLane="0" toLane="0" dir="t" state="M"/>

<connection from=":n6\_0" to="e5\_rev" fromLane="0" toLane="0" dir="L" state="M"/>

<connection from=":n6\_1" to="e12\_rev" fromLane="0" toLane="0" via=":n6\_9\_0" dir="l" state="m"/>

<connection from=":n6\_9" to="e12\_rev" fromLane="0" toLane="0" dir="l" state="M"/>

<connection from=":n6\_2" to="e6" fromLane="0" toLane="0" via=":n6\_10\_0" dir="t" state="m"/>

<connection from=":n6\_10" to="e6" fromLane="0" toLane="0" dir="t" state="M"/>

<connection from=":n6\_3" to="e6" fromLane="0" toLane="0" dir="r" state="M"/>

<connection from=":n6\_4" to="e5\_rev" fromLane="0" toLane="0" dir="l" state="M"/>

<connection from=":n6\_5" to="e12\_rev" fromLane="0" toLane="0" dir="t" state="M"/>

<connection from=":n6\_6" to="e12\_rev" fromLane="0" toLane="0" dir="r" state="M"/>

<connection from=":n6\_7" to="e6" fromLane="0" toLane="0" dir="R" state="M"/>

<connection from=":n6\_8" to="e5\_rev" fromLane="0" toLane="0" via=":n6\_11\_0" dir="t" state="m"/>

<connection from=":n6\_11" to="e5\_rev" fromLane="0" toLane="0" dir="t" state="M"/>

<connection from=":n\_center\_0" to="e12" fromLane="0" toLane="0" dir="r" state="M"/>

<connection from=":n\_center\_1" to="e11" fromLane="0" toLane="0" dir="R" state="M"/>

<connection from=":n\_center\_2" to="e10" fromLane="0" toLane="0" dir="s" state="M"/>

<connection from=":n\_center\_3" to="e9" fromLane="0" toLane="0" via=":n\_center\_36\_0" dir="L" state="m"/>

<connection from=":n\_center\_36" to="e9" fromLane="0" toLane="0" dir="L" state="M"/>

<connection from=":n\_center\_4" to="e8" fromLane="0" toLane="0" via=":n\_center\_37\_0" dir="l" state="m"/>

<connection from=":n\_center\_37" to="e8" fromLane="0" toLane="0" dir="l" state="M"/>

<connection from=":n\_center\_5" to="e7" fromLane="0" toLane="0" via=":n\_center\_38\_0" dir="t" state="m"/>

<connection from=":n\_center\_38" to="e7" fromLane="0" toLane="0" dir="t" state="M"/>

<connection from=":n\_center\_6" to="e7" fromLane="0" toLane="0" dir="r" state="M"/>

<connection from=":n\_center\_7" to="e12" fromLane="0" toLane="0" dir="R" state="M"/>

<connection from=":n\_center\_8" to="e11" fromLane="0" toLane="0" dir="s" state="M"/>

<connection from=":n\_center\_9" to="e10" fromLane="0" toLane="0" via=":n\_center\_39\_0" dir="L" state="m"/>

<connection from=":n\_center\_39" to="e10" fromLane="0" toLane="0" dir="L" state="M"/>

<connection from=":n\_center\_10" to="e9" fromLane="0" toLane="0" via=":n\_center\_40\_0" dir="l" state="m"/>

<connection from=":n\_center\_40" to="e9" fromLane="0" toLane="0" dir="l" state="M"/>

<connection from=":n\_center\_11" to="e8" fromLane="0" toLane="0" via=":n\_center\_41\_0" dir="t" state="m"/>

<connection from=":n\_center\_41" to="e8" fromLane="0" toLane="0" dir="t" state="M"/>

<connection from=":n\_center\_12" to="e8" fromLane="0" toLane="0" dir="r" state="M"/>

<connection from=":n\_center\_13" to="e7" fromLane="0" toLane="0" dir="R" state="M"/>

<connection from=":n\_center\_14" to="e12" fromLane="0" toLane="0" dir="s" state="M"/>

<connection from=":n\_center\_15" to="e11" fromLane="0" toLane="0" via=":n\_center\_42\_0" dir="L" state="m"/>

<connection from=":n\_center\_42" to="e11" fromLane="0" toLane="0" dir="L" state="M"/>

<connection from=":n\_center\_16" to="e10" fromLane="0" toLane="0" via=":n\_center\_43\_0" dir="l" state="m"/>

<connection from=":n\_center\_43" to="e10" fromLane="0" toLane="0" dir="l" state="M"/>

<connection from=":n\_center\_17" to="e9" fromLane="0" toLane="0" via=":n\_center\_44\_0" dir="t" state="m"/>

<connection from=":n\_center\_44" to="e9" fromLane="0" toLane="0" dir="t" state="M"/>

<connection from=":n\_center\_18" to="e9" fromLane="0" toLane="0" dir="r" state="M"/>

<connection from=":n\_center\_19" to="e8" fromLane="0" toLane="0" dir="R" state="M"/>

<connection from=":n\_center\_20" to="e7" fromLane="0" toLane="0" dir="s" state="M"/>

<connection from=":n\_center\_21" to="e12" fromLane="0" toLane="0" via=":n\_center\_45\_0" dir="L" state="m"/>

<connection from=":n\_center\_45" to="e12" fromLane="0" toLane="0" dir="L" state="M"/>

<connection from=":n\_center\_22" to="e11" fromLane="0" toLane="0" via=":n\_center\_46\_0" dir="l" state="m"/>

<connection from=":n\_center\_46" to="e11" fromLane="0" toLane="0" dir="l" state="M"/>

<connection from=":n\_center\_23" to="e10" fromLane="0" toLane="0" via=":n\_center\_47\_0" dir="t" state="m"/>

<connection from=":n\_center\_47" to="e10" fromLane="0" toLane="0" dir="t" state="M"/>

<connection from=":n\_center\_24" to="e10" fromLane="0" toLane="0" dir="r" state="M"/>

<connection from=":n\_center\_25" to="e9" fromLane="0" toLane="0" dir="R" state="M"/>

<connection from=":n\_center\_26" to="e8" fromLane="0" toLane="0" dir="s" state="M"/>

<connection from=":n\_center\_27" to="e7" fromLane="0" toLane="0" via=":n\_center\_48\_0" dir="L" state="m"/>

<connection from=":n\_center\_48" to="e7" fromLane="0" toLane="0" dir="L" state="M"/>

<connection from=":n\_center\_28" to="e12" fromLane="0" toLane="0" via=":n\_center\_49\_0" dir="l" state="m"/>

<connection from=":n\_center\_49" to="e12" fromLane="0" toLane="0" dir="l" state="M"/>

<connection from=":n\_center\_29" to="e11" fromLane="0" toLane="0" via=":n\_center\_50\_0" dir="t" state="m"/>

<connection from=":n\_center\_50" to="e11" fromLane="0" toLane="0" dir="t" state="M"/>

<connection from=":n\_center\_30" to="e11" fromLane="0" toLane="0" dir="r" state="M"/>

<connection from=":n\_center\_31" to="e10" fromLane="0" toLane="0" dir="R" state="M"/>

<connection from=":n\_center\_32" to="e9" fromLane="0" toLane="0" dir="s" state="M"/>

<connection from=":n\_center\_33" to="e8" fromLane="0" toLane="0" via=":n\_center\_51\_0" dir="L" state="m"/>

<connection from=":n\_center\_51" to="e8" fromLane="0" toLane="0" dir="L" state="M"/>

<connection from=":n\_center\_34" to="e7" fromLane="0" toLane="0" via=":n\_center\_52\_0" dir="l" state="m"/>

<connection from=":n\_center\_52" to="e7" fromLane="0" toLane="0" dir="l" state="M"/>

<connection from=":n\_center\_35" to="e12" fromLane="0" toLane="0" via=":n\_center\_53\_0" dir="t" state="m"/>

<connection from=":n\_center\_53" to="e12" fromLane="0" toLane="0" dir="t" state="M"/>

</net>

 python traci\_runner.py

Adding dynamic patient POIs...

--- Running Dispatch Cycle at T=1.0s ---

Pending Patients: 8, Available Ambulances: 8

--- Executing Optimal Assignments ---

  Assigned: amb\_1 -> P02 (medium) -> H-01 (City\_General)

C:\Users\SAHANA MT\Desktop\fri\traci\_runner.py:344: UserWarning: Call to deprecated function getAllProgramLogics, use getCompleteRedYellowGreenDefinition instead.

  logic = traci.trafficlight.getCompleteRedYellowGreenDefinition(controlled\_tls\_id)

  Assigned: amb\_2 -> P07 (high) -> H-02 (Green\_Heart)

  Assigned: amb\_3 -> P08 (low) -> H-01 (City\_General)

  Assigned: amb\_4 -> P05 (medium) -> H-01 (City\_General)

  Assigned: amb\_5 -> P01 (medium) -> H-01 (City\_General)

  Assigned: amb\_6 -> P03 (low) -> H-04 (Tumakuru\_Trauma)

  Assigned: amb\_7 -> P06 (high) -> H-02 (Green\_Heart)

  Assigned: amb\_8 -> P04 (high) -> H-04 (Tumakuru\_Trauma)

--- Dispatch Cycle End: 0 patients pending, 0 ambulances available ---

INFO: amb\_1 reached patient P02 at T=6.0s. Stopping for pickup.

INFO: amb\_4 reached patient P05 at T=11.0s. Stopping for pickup.

INFO: amb\_1 finished pickup for P02. Proceeding to hospital.

INFO: amb\_3 reached patient P08 at T=17.0s. Stopping for pickup.

INFO: amb\_4 finished pickup for P05. Proceeding to hospital.

INFO: amb\_3 finished pickup for P08. Proceeding to hospital.

INFO: amb\_6 reached patient P03 at T=33.0s. Stopping for pickup.

INFO: amb\_6 finished pickup for P03. Proceeding to hospital.

INFO: amb\_8 reached patient P04 at T=48.0s. Stopping for pickup.

INFO: amb\_8 finished pickup for P04. Proceeding to hospital.

MISSION COMPLETE: amb\_3 delivered patient P08 at T=59.0s.

MISSION COMPLETE: amb\_4 delivered patient P05 at T=66.0s.

MISSION COMPLETE: amb\_6 delivered patient P03 at T=151.0s.

INFO: amb\_7 reached patient P06 at T=307.0s. Stopping for pickup.

INFO: amb\_2 reached patient P07 at T=308.0s. Stopping for pickup.

INFO: amb\_7 finished pickup for P06. Proceeding to hospital.

INFO: amb\_2 finished pickup for P07. Proceeding to hospital.

INFO: amb\_5 reached patient P01 at T=324.0s. Stopping for pickup.

INFO: amb\_5 finished pickup for P01. Proceeding to hospital.

MISSION COMPLETE: amb\_2 delivered patient P07 at T=348.0s.

MISSION COMPLETE: amb\_5 delivered patient P01 at T=364.0s.

MISSION COMPLETE: amb\_7 delivered patient P06 at T=386.0s.

MISSION COMPLETE: amb\_1 delivered patient P02 at T=644.0s.

MISSION COMPLETE: amb\_8 delivered patient P04 at T=666.0s.

All missions complete. Starting 10s cooldown before shutdown.

Cooldown finished. Ending simulation.

Simulation ended at time: 676.00

PS C:\Users\SAHANA MT\Desktop\fri>

**Cons (Challenges & Areas for Improvement 👎)**

* **Persistent Congestion:** Despite actuated lights, the simulation still shows **severe traffic delays**, especially around the central intersection. Waiting times often dominate the total trip time for ambulances, significantly impacting response speed. The actuated logic helps but doesn't solve the core bottleneck.
* **Basic Preemption:** The current traffic light preemption for ambulances is rudimentary. It forces a green phase but isn't guaranteed to clear traffic effectively or coordinate across multiple lights for a true "green wave."
* **Static Patient Load:** Patients are generated only at the beginning. A more realistic simulation would introduce new emergency calls dynamically over tim

import os

import sys

import traci

import heapq

import random

import xml.etree.ElementTree as ET

from collections import Counter

from scipy.optimize import linear\_sum\_assignment

import numpy as np

import math # Needed for distance calculation

# --- GA CONFIGURATION (Unchanged) ---

GA\_CONFIG = {

    "population\_size": 50, "generations": 5, "mutation\_rate": 0.3,

    "crossover\_rate": 0.8, "tournament\_size": 3, "elitism\_size": 2

}

# --- SIMULATION DATA (Unchanged) ---

EDGE\_POOL = [

    "e1", "e2", "e3", "e4", "e5", "e6", "e7", "e8", "e9", "e10", "e11", "e12",

    "e1\_rev", "e2\_rev", "e3\_rev", "e4\_rev", "e5\_rev", "e6\_rev",

    "e7\_rev", "e8\_rev", "e9\_rev", "e10\_rev", "e11\_rev", "e12\_rev"

]

PATIENT\_PROFILES = [

    {"name": "Ravi Kumar", "condition": "Cardiac Arrest", "keywords": ["Cardiology"], "severity": "high"},

    {"name": "Sita Devi", "condition": "Multiple Fractures", "keywords": ["Trauma Care", "Orthopedics"], "severity": "high"},

    {"name": "Amit Patel", "condition": "Head Injury", "keywords": ["Trauma Care"], "severity": "high"},

    {"name": "Vikram Rathod", "condition": "Chest Pain", "keywords": ["Cardiology", "Emergency Care"], "severity": "high"},

    {"name": "Arjun Singh", "condition": "Severe Lacerations", "keywords": ["Emergency Care"], "severity": "medium"},

    {"name": "Priya Sharma", "condition": "Breathing Difficulty", "keywords": ["General Medicine"], "severity": "medium"},

    {"name": "Anjali Gupta", "condition": "Minor Burn", "keywords": ["General Medicine"], "severity": "low"},

    {"name": "Meera Iyer", "condition": "Sprained Ankle", "keywords": ["Orthopedics"], "severity": "low"},

    {"name": "Suresh Verma", "condition": "Allergic Reaction", "keywords": ["Emergency Care"], "severity": "medium"},

    {"name": "Deepa Mehta", "condition": "Abdominal Pain", "keywords": ["General Medicine"], "severity": "medium"},

]

def generate\_random\_patients(num\_patients):

    patients = {}

    if num\_patients > len(PATIENT\_PROFILES) or num\_patients > len(EDGE\_POOL):

        raise ValueError("Cannot generate more unique patients than available profiles/edges.")

    chosen\_profiles = random.sample(PATIENT\_PROFILES, num\_patients)

    chosen\_edges = random.sample(EDGE\_POOL, num\_patients) if num\_patients <= len(EDGE\_POOL) else random.choices(EDGE\_POOL, k=num\_patients)

    for i in range(num\_patients):

        patient\_id = f"P{i+1:02}"; profile = chosen\_profiles[i]

        patients[patient\_id] = {

            "name": profile["name"], "condition": profile["condition"], "keywords": profile["keywords"],

            "start\_edge": chosen\_edges[i], "severity": profile["severity"], "status": "pending"

        }

    return patients

ALL\_PATIENTS = generate\_random\_patients(8)

HOSPITALS\_TEMPLATE = {

    "H-01": {"name": "City\_General", "specialties": ["General Medicine", "Emergency Care"], "initial\_beds": 12, "dest\_edge": "e6"},

    "H-02": {"name": "Green\_Heart", "specialties": ["Cardiology"], "initial\_beds": 5, "dest\_edge": "e2"},

    "H-04": {"name": "Tumakuru\_Trauma", "specialties": ["Trauma Care", "Orthopedics"], "initial\_beds": 3, "dest\_edge": "e4"}

}

HOSPITALS = {h\_id: data.copy() for h\_id, data in HOSPITALS\_TEMPLATE.items()}

for h\_id in HOSPITALS: HOSPITALS[h\_id]["available\_beds"] = HOSPITALS[h\_id]["initial\_beds"]

SEVERITY\_WEIGHTS = { "high": 1.5, "medium": 1.0, "low": 0.5 }

AMBULANCES = ["amb\_1", "amb\_2", "amb\_3", "amb\_4", "amb\_5", "amb\_6", "amb\_7", "amb\_8"]

# --- Dijkstra Pathfinding Class (Unchanged) ---

class DijkstraForSUMO:

    def \_\_init\_\_(self, net\_file):

        self.net\_file = net\_file; self.graph, self.edge\_to\_junctions, self.junction\_pair\_to\_edge = {}, {}, {}

        self.\_build\_graph()

    def \_build\_graph(self):

        try:

            tree = ET.parse(self.net\_file); root = tree.getroot(); junctions\_with\_edges = set()

            for edge in root.findall('edge'):

                if edge.get('function') != 'internal':

                    edge\_id = edge.get('id'); from\_node, to\_node = edge.get('from'), edge.get('to')

                    if from\_node and to\_node:

                        junctions\_with\_edges.add(from\_node); junctions\_with\_edges.add(to\_node)

                        if from\_node not in self.graph: self.graph[from\_node] = {}

                        self.graph[from\_node][to\_node] = 1

                        self.edge\_to\_junctions[edge\_id] = (from\_node, to\_node); self.junction\_pair\_to\_edge[(from\_node, to\_node)] = edge\_id

            for node in junctions\_with\_edges:

                 if node not in self.graph: self.graph[node] = {}

        except FileNotFoundError: raise ValueError(f"Net file '{self.net\_file}' not found.")

        except ET.ParseError as e: raise ValueError(f"Failed to parse net file '{self.net\_file}': {e}")

    def find\_shortest\_path\_time(self, start\_edge, end\_edge):

        if not start\_edge or not end\_edge: return float('inf')

        if start\_edge not in self.edge\_to\_junctions or end\_edge not in self.edge\_to\_junctions: return float('inf')

        \_, time = self.find\_dynamic\_shortest\_path(start\_edge, end\_edge); return time

    def find\_dynamic\_shortest\_path(self, start\_edge, end\_edge):

        if start\_edge == end\_edge:

            try: return [start\_edge], max(1.0, traci.edge.getTraveltime(start\_edge))

            except traci.TraCIException: return [start\_edge], 1.0

        start\_node\_tuple = self.edge\_to\_junctions.get(start\_edge); end\_node\_tuple = self.edge\_to\_junctions.get(end\_edge)

        if not start\_node\_tuple or not end\_node\_tuple: return None, float('inf')

        start\_junction, end\_junction = start\_node\_tuple[1], end\_node\_tuple[0]

        if start\_junction not in self.graph: return None, float('inf')

        distances = {node: float('inf') for node in self.graph};

        if start\_junction in distances: distances[start\_junction] = 0

        else: return None, float('inf')

        previous\_nodes = {node: None for node in self.graph}; pq = [(0, start\_junction)]

        path\_found = False

        while pq:

            dist, current\_node = heapq.heappop(pq)

            if dist > distances.get(current\_node, float('inf')): continue

            if current\_node == end\_junction: path\_found = True; break

            if current\_node in self.graph:

                for neighbor in self.graph[current\_node]:

                    edge\_id = self.junction\_pair\_to\_edge.get((current\_node, neighbor))

                    if edge\_id and neighbor in distances:

                        try: weight = max(1.0, traci.edge.getTraveltime(edge\_id))

                        except traci.TraCIException: weight = 1000.0

                        distance = dist + weight

                        if distance < distances[neighbor]:

                            distances[neighbor] = distance; previous\_nodes[neighbor] = current\_node; heapq.heappush(pq, (distance, neighbor))

        if not path\_found or distances.get(end\_junction, float('inf')) == float('inf'): return None, float('inf')

        path\_nodes = []; current = end\_junction

        while current is not None: path\_nodes.insert(0, current); current = previous\_nodes.get(current)

        if not path\_nodes or path\_nodes[0] != start\_junction: return None, float('inf')

        path\_edges = [self.junction\_pair\_to\_edge.get((path\_nodes[i], path\_nodes[i+1])) for i in range(len(path\_nodes) - 1)]

        valid\_junction\_edges = [edge for edge in path\_edges if edge]

        full\_edge\_path = [start\_edge] + valid\_junction\_edges

        if full\_edge\_path[-1] != end\_edge: full\_edge\_path.append(end\_edge)

        try: first\_edge\_time = max(1.0, traci.edge.getTraveltime(start\_edge))

        except traci.TraCIException: first\_edge\_time = 1.0

        calculated\_path\_time = distances.get(end\_junction, float('inf'))

        total\_time = float('inf') if calculated\_path\_time == float('inf') else first\_edge\_time + calculated\_path\_time

        return full\_edge\_path, total\_time

# --- Event-Driven GA Functions (Unchanged) ---

def create\_chromosome\_event(current\_patients):

    hospital\_ids = list(HOSPITALS.keys())

    return [random.choice(hospital\_ids) for \_ in current\_patients]

def calculate\_fitness\_event(chromosome, router, current\_patient\_ids, current\_ambulance\_ids):

    specialty\_penalty, bed\_penalty = 0, 0; num\_current\_patients = len(current\_patient\_ids)

    hospital\_assignments = Counter(chromosome)

    for i, patient\_id in enumerate(current\_patient\_ids):

        hospital\_id = chromosome[i]

        if patient\_id not in ALL\_PATIENTS or hospital\_id not in HOSPITALS: continue

        if not any(spec in HOSPITALS[hospital\_id]["specialties"] for spec in ALL\_PATIENTS[patient\_id]["keywords"]): specialty\_penalty += 5000

    for h\_id, count in hospital\_assignments.items():

        if h\_id in HOSPITALS and count > HOSPITALS[h\_id]["available\_beds"]: bed\_penalty += 2000 \* (count - HOSPITALS[h\_id]["available\_beds"])

    current\_vehicle\_ids = traci.vehicle.getIDList(); valid\_ambulance\_ids = [amb\_id for amb\_id in current\_ambulance\_ids if amb\_id in current\_vehicle\_ids]

    num\_ambulances = len(valid\_ambulance\_ids); amb\_edges = []; final\_valid\_amb\_ids = []

    if num\_ambulances == 0: return float('inf')

    for amb\_id in valid\_ambulance\_ids:

        try: edge = traci.vehicle.getRoadID(amb\_id); amb\_edges.append(edge); final\_valid\_amb\_ids.append(amb\_id)

        except traci.TraCIException: pass

    valid\_ambulance\_ids = final\_valid\_amb\_ids; num\_ambulances = len(valid\_ambulance\_ids)

    if num\_ambulances == 0: return float('inf')

    num\_assignments = min(num\_current\_patients, num\_ambulances); assignable\_patient\_ids = current\_patient\_ids[:num\_assignments]

    cost\_matrix = np.full((num\_ambulances, num\_assignments), float('inf'))

    for i in range(num\_ambulances):

        for j, patient\_id in enumerate(assignable\_patient\_ids):

            hospital\_id = chromosome[j]; patient\_info = ALL\_PATIENTS.get(patient\_id); hospital\_info = HOSPITALS.get(hospital\_id)

            if not patient\_info or not hospital\_info: continue

            time\_to\_patient = router.find\_shortest\_path\_time(amb\_edges[i], patient\_info["start\_edge"])

            time\_to\_hospital = router.find\_shortest\_path\_time(patient\_info["start\_edge"], hospital\_info["dest\_edge"])

            if time\_to\_patient != float('inf') and time\_to\_hospital != float('inf'):

                mission\_time = time\_to\_patient + time\_to\_hospital

                weighted\_time = mission\_time \* SEVERITY\_WEIGHTS.get(patient\_info.get("severity", "medium"), 1.0); cost\_matrix[i, j] = weighted\_time

    try:

        # Check if matrix is all infinities before calling assignment

        if np.all(cost\_matrix == float('inf')):

             total\_weighted\_time = float('inf')

        else:

             amb\_indices, assigned\_pat\_indices\_in\_matrix = linear\_sum\_assignment(cost\_matrix)

             # Sum the costs of the assigned pairs, ignore infinite costs if assignment failed for some

             valid\_costs = cost\_matrix[amb\_indices, assigned\_pat\_indices\_in\_matrix]

             total\_weighted\_time = valid\_costs[valid\_costs != float('inf')].sum()

             if np.all(valid\_costs == float('inf')):

                 total\_weighted\_time = float('inf')

    except ValueError as e:

         return float('inf')

    return total\_weighted\_time + specialty\_penalty + bed\_penalty

def selection(population, fitnesses):

    pop\_fit\_pairs = list(zip(population, fitnesses)); k = min(GA\_CONFIG["tournament\_size"], len(pop\_fit\_pairs))

    if not pop\_fit\_pairs or k == 0: return population[0] if population else []

    tournament = random.sample(pop\_fit\_pairs, k); return min(tournament, key=lambda x: x[1])[0]

def crossover(parent1, parent2):

    if random.random() > GA\_CONFIG["crossover\_rate"] or len(parent1) <= 1: return list(parent1), list(parent2)

    point = random.randint(1, len(parent1) - 1); p1 = list(parent1); p2 = list(parent2); return p1[:point] + p2[point:], p2[:point] + p1[point:]

def mutate(chromosome):

    mutated\_chromosome = list(chromosome)

    if random.random() < GA\_CONFIG["mutation\_rate"] and mutated\_chromosome:

        idx = random.randrange(len(mutated\_chromosome)); mutated\_chromosome[idx] = random.choice(list(HOSPITALS.keys()))

    return mutated\_chromosome

# --- Mission Management & Dispatch Logic ---

mission\_log\_counter = 1

def generate\_dispatch\_report\_event(dispatch\_details):

    global mission\_log\_counter

    try:

        with open("dispatch\_summary.txt", "a") as f:

            if mission\_log\_counter == 1:

                f.write("========================================\n")

                f.write("   Event-Driven Dispatch Summary\n")

                f.write("========================================\n\n")

            mission = dispatch\_details; patient = ALL\_PATIENTS.get(mission['patient\_id']); hospital = HOSPITALS.get(mission['hospital\_id'])

            if not patient or not hospital: return

            f.write(f"--- Mission {mission\_log\_counter} (Dispatched at T={mission['dispatch\_time']:.1f}s) ---\n")

            f.write(f"  Ambulance: {mission['ambulance\_id']}\n\n")

            f.write(f"  Patient Details:\n    - ID:         {mission['patient\_id']}\n    - Name:       {patient['name']}\n    - Condition:  {patient['condition']} (Severity: {patient['severity']})\n    - Location:   Edge '{patient['start\_edge']}'\n\n")

            f.write(f"  Assigned Hospital:\n    - ID:         {mission['hospital\_id']}\n    - Name:       {hospital['name']}\n    - Specialties:{', '.join(hospital['specialties'])}\n    - Beds (Current Avail. Before): {mission['beds\_at\_dispatch']} -> {mission['beds\_at\_dispatch'] - 1}\n")

            f.write(f"    - Destination:Edge '{hospital['dest\_edge']}'\n\n----------------------------------------\n\n")

            mission\_log\_counter += 1

    except Exception as e: print(f"Error writing report: {e}")

# --- NEW: Buffer Zone / Priority Constants and State ---

PRIORITY\_THRESHOLD\_ETA = 10.0 # Seconds before arrival to activate priority

PRIORITY\_VISUAL\_COLOR = (255, 165, 0, 150) # Orange, semi-transparent

prioritized\_vehicles = {} # {amb\_id: {'tls\_id': tls\_id, 'visual\_id': poly\_id}}

# --- Traffic Light Control Logic ---

TLS\_IDS = ["n\_center", "n1", "n2", "n3", "n4", "n5", "n6"]

MIN\_GREEN\_TIME = 5; MAX\_GREEN\_TIME = 40

YELLOW\_TIME = 3; ALL\_RED\_TIME = 2

# !! CRITICAL: VERIFY THESE MAPPINGS AGAINST YOUR hexagon.net.xml !!

PHASE\_MAP = {

    "n\_center": { 0: ["e7\_rev\_0", "e10\_rev\_0"], 3: ["e8\_rev\_0", "e11\_rev\_0"], 6: ["e9\_rev\_0", "e12\_rev\_0"] },

    "n1": { 0: ["e6\_0", "e1\_rev\_0"], 2: ["e7\_0"] }, # Green phases 0 and 2

    "n2": { 0: ["e1\_0", "e2\_rev\_0"], 2: ["e8\_0"] }, # Green phases 0 and 2

    "n3": { 0: ["e2\_0", "e3\_rev\_0"], 2: ["e9\_0"] }, # Green phases 0 and 2

    "n4": { 0: ["e3\_0", "e4\_rev\_0"], 2: ["e10\_0"] },# Green phases 0 and 2

    "n5": { 0: ["e4\_0", "e5\_rev\_0"], 2: ["e11\_0"] },# Green phases 0 and 2

    "n6": { 0: ["e5\_0", "e6\_rev\_0"], 2: ["e12\_0"] } # Green phases 0 and 2

}

YELLOW\_PHASE\_INDICES = {

    "n\_center": { 0: 1, 3: 4, 6: 7 },

    "n1": { 0: 1, 2: 3 }, # Green 0 -> Yellow 1, Green 2 -> Yellow 3

    "n2": { 0: 1, 2: 3 },

    "n3": { 0: 1, 2: 3 },

    "n4": { 0: 1, 2: 3 },

    "n5": { 0: 1, 2: 3 },

    "n6": { 0: 1, 2: 3 }

}

ALL\_RED\_PHASE\_INDICES = { # Index of an all-red phase (or None if missing)

    "n\_center": 2,

    "n1": None, # Assuming no dedicated all-red phase from XML

    "n2": None,

    "n3": None,

    "n4": None,

    "n5": None,

    "n6": None

}

DETECTOR\_MAP = {

    "e7\_rev\_0": "det\_e7\_rev\_0", "e8\_rev\_0": "det\_e8\_rev\_0", "e9\_rev\_0": "det\_e9\_rev\_0",

    "e10\_rev\_0": "det\_e10\_rev\_0", "e11\_rev\_0": "det\_e11\_rev\_0", "e12\_rev\_0": "det\_e12\_rev\_0",

    "e1\_rev\_0": "det\_e1\_rev\_0\_n1", "e7\_0": "det\_e7\_0\_n1", "e6\_0": "det\_e6\_0\_n1",

    "e2\_rev\_0": "det\_e2\_rev\_0\_n2", "e8\_0": "det\_e8\_0\_n2", "e1\_0": "det\_e1\_0\_n2",

    "e2\_0": "det\_e2\_0\_n3", "e3\_rev\_0": "det\_e3\_rev\_0\_n3", "e9\_0": "det\_e9\_0\_n3",

    "e10\_0": "det\_e10\_0\_n4", "e3\_0": "det\_e3\_0\_n4", "e4\_rev\_0": "det\_e4\_rev\_0\_n4",

    "e4\_0": "det\_e4\_0\_n5", "e5\_rev\_0": "det\_e5\_rev\_0\_n5", "e11\_0": "det\_e11\_0\_n5",

    "e5\_0": "det\_e5\_0\_n6", "e6\_rev\_0": "det\_e6\_rev\_0\_n6", "e12\_0": "det\_e12\_0\_n6",

}

tls\_current\_phase\_indices = {}; tls\_phase\_start\_times = {}; tls\_states = {}

def get\_phase\_demand(tls\_id, phase\_index):

    demand = 0; lanes = PHASE\_MAP.get(tls\_id, {}).get(phase\_index, [])

    for lane\_id in lanes:

        det\_id = DETECTOR\_MAP.get(lane\_id)

        if det\_id:

            try: demand += traci.lanearea.getLastStepHaltingNumber(det\_id)

            except traci.TraCIException: pass

    return demand

def switch\_to\_phase(tls\_id, new\_phase\_index, current\_time):

    global tls\_current\_phase\_indices, tls\_phase\_start\_times, tls\_states

    try:

        # Check if new\_phase\_index is valid

        program\_length = len(traci.trafficlight.getCompleteRedYellowGreenDefinition(tls\_id)[0].phases)

        if new\_phase\_index >= program\_length:

             print(f"Warn: Invalid phase {new\_phase\_index} for {tls\_id}. Max={program\_length-1}. Ignoring."); return

        traci.trafficlight.setPhase(tls\_id, new\_phase\_index)

        tls\_current\_phase\_indices[tls\_id] = new\_phase\_index

        tls\_phase\_start\_times[tls\_id] = current\_time

        # Determine state

        if new\_phase\_index in YELLOW\_PHASE\_INDICES.get(tls\_id, {}).values(): tls\_states[tls\_id] = "YELLOW"

        elif new\_phase\_index == ALL\_RED\_PHASE\_INDICES.get(tls\_id): tls\_states[tls\_id] = "RED\_CLEAR"

        else: tls\_states[tls\_id] = "GREEN"

    except (traci.TraCIException, KeyError, IndexError) as e: print(f"Error switching TLS {tls\_id}: {e}")

# --- NEW: Function to activate visual + priority ---

def activate\_priority(amb\_id, tls\_id, current\_time):

    global prioritized\_vehicles

    if amb\_id in prioritized\_vehicles: return # Already active

    print(f"PRIORITY: Activating for {amb\_id} at {tls\_id} (T={current\_time:.1f}s). Using SpeedMode 31.")

    poly\_id = None

    # Set Speed Mode 31

    try:

        traci.vehicle.setSpeedMode(amb\_id, 31)

    except traci.TraCIException as e:

        print(f"Warn: Failed to set speed mode for {amb\_id}: {e}")

        return # Cannot activate priority if speed mode fails

    # Add visual cue

    try:

        shape = traci.junction.getShape(tls\_id)

        if shape:

            center\_x = sum(p[0] for p in shape) / len(shape); center\_y = sum(p[1] for p in shape) / len(shape)

            expanded\_shape = [(p[0] + (p[0]-center\_x)\*0.15, p[1] + (p[1]-center\_y)\*0.15) for p in shape]

            poly\_id = f"priority\_visual\_{tls\_id}\_{amb\_id}"

            traci.polygon.add(poly\_id, expanded\_shape, PRIORITY\_VISUAL\_COLOR, layer=15, fill=True)

    except traci.TraCIException as e: print(f"Warn: Viz add failed for {tls\_id}: {e}")

    prioritized\_vehicles[amb\_id] = {'tls\_id': tls\_id, 'visual\_id': poly\_id}

# --- NEW: Function to deactivate visual + priority ---

def deactivate\_priority(amb\_id):

    global prioritized\_vehicles

    if amb\_id not in prioritized\_vehicles: return

    state\_info = prioritized\_vehicles[amb\_id]

    print(f"PRIORITY: Deactivating for {amb\_id} at {state\_info['tls\_id']}.")

    # Reset Speed Mode

    try: traci.vehicle.setSpeedMode(amb\_id, 0) # Reset to default (obeys lights)

    except traci.TraCIException: pass # Ignore if vehicle gone

    # Remove visual cue

    if state\_info.get('visual\_id'):

        try: traci.polygon.remove(state\_info['visual\_id'])

        except traci.TraCIException: pass

    del prioritized\_vehicles[amb\_id]

# --- REVISED: Check proximity and manage priority ---

def check\_ambulance\_proximity\_and\_manage\_priority(active\_missions, current\_time):

    global prioritized\_vehicles

    active\_amb\_ids = list(active\_missions.keys())

    # Check currently prioritized vehicles to see if they've cleared

    for amb\_id in list(prioritized\_vehicles.keys()):

        if amb\_id not in active\_amb\_ids: # Ambulance finished mission entirely

             deactivate\_priority(amb\_id); continue

        try:

             tls\_id = prioritized\_vehicles[amb\_id]['tls\_id']

             amb\_edge = traci.vehicle.getRoadID(amb\_id)

             # Check if ambulance is past the intersection

             incoming\_edges = traci.junction.getIncomingEdges(tls\_id)

             if amb\_edge not in incoming\_edges:

                  # Robust check: Route index past junction edges

                  amb\_route = traci.vehicle.getRoute(amb\_id); current\_route\_index = traci.vehicle.getRouteIndex(amb\_id)

                  # Internal edges start with ':'

                  junction\_edges = incoming\_edges # Removed the function call that caused the error

                  junction\_edge\_indices = [i for i, edge in enumerate(amb\_route) if edge in junction\_edges]

                  # If no junction edges found in route OR index is past the last one

                  if not junction\_edge\_indices or current\_route\_index > max(junction\_edge\_indices):

                       deactivate\_priority(amb\_id)

        except traci.TraCIException: deactivate\_priority(amb\_id); continue # Error, deactivate

    # Check active ambulances for potential new priority requests

    for amb\_id, mission in active\_missions.items():

        if mission["stage"] == "done" or amb\_id in prioritized\_vehicles: continue # Skip done/already prioritized

        try:

            route = traci.vehicle.getRoute(amb\_id)

            if not route or len(route) < 1: continue # Needs at least current edge

            current\_edge = route[0]

            junction\_id = traci.edge.getToJunction(current\_edge) # Junction at end of current edge

            if junction\_id in TLS\_IDS: # Check if it's a controlled light

                tls\_id = junction\_id

                # Calculate distance and ETA

                current\_lane = traci.vehicle.getLaneID(amb\_id)

                dist\_to\_junction = traci.lane.getLength(current\_lane) - traci.vehicle.getLanePosition(amb\_id)

                speed = max(1.0, traci.vehicle.getSpeed(amb\_id))

                eta = dist\_to\_junction / speed

                if eta < PRIORITY\_THRESHOLD\_ETA:

                    # Ambulance is close, activate priority

                    activate\_priority(amb\_id, tls\_id, current\_time)

        except traci.TraCIException: continue # Ignore errors

# --- Adaptive Traffic Light Logic (Checks for priority activation) ---

def run\_adaptive\_tls\_logic(tls\_id, current\_time, prioritized\_vehicles\_state):

    global tls\_current\_phase\_indices, tls\_phase\_start\_times, tls\_states

    # --- Skip if an ambulance has priority at this light ---

    if tls\_id in [state['tls\_id'] for state in prioritized\_vehicles\_state.values()]:

        # print(f"TLS {tls\_id}: Skipping adaptive logic due to active priority.") # Optional debug

        return

    # Initialize state if not present (Unchanged)

    if tls\_id not in tls\_states:

        try:

             tls\_current\_phase\_indices[tls\_id] = traci.trafficlight.getPhase(tls\_id); tls\_phase\_start\_times[tls\_id] = current\_time

             if tls\_current\_phase\_indices[tls\_id] in YELLOW\_PHASE\_INDICES.get(tls\_id, {}).values(): tls\_states[tls\_id] = "YELLOW"

             elif tls\_current\_phase\_indices[tls\_id] == ALL\_RED\_PHASE\_INDICES.get(tls\_id): tls\_states[tls\_id] = "RED\_CLEAR"

             else: tls\_states[tls\_id] = "GREEN"

        except traci.TraCIException: print(f"Warn: Cannot init TLS {tls\_id}"); return

    # Get current state (Unchanged)

    current\_phase\_index = tls\_current\_phase\_indices.get(tls\_id, 0); phase\_start\_time = tls\_phase\_start\_times.get(tls\_id, current\_time)

    current\_state = tls\_states.get(tls\_id, "GREEN"); phase\_elapsed\_time = current\_time - phase\_start\_time

    # --- Actuated Logic (Unchanged, handles GREEN, YELLOW, RED\_CLEAR) ---

    if current\_state == "YELLOW":

        if phase\_elapsed\_time >= YELLOW\_TIME:

            all\_red\_phase = ALL\_RED\_PHASE\_INDICES.get(tls\_id)

            # --- CORRECTED INDENTATION ---

            if all\_red\_phase is not None:

                switch\_to\_phase(tls\_id, all\_red\_phase, current\_time)

            else: # No all-red? Decide next green immediately

                max\_demand = -1; best\_next\_phase = -1

                for green\_phase\_idx in PHASE\_MAP.get(tls\_id, {}).keys():

                    demand = get\_phase\_demand(tls\_id, green\_phase\_idx)

                    if demand > max\_demand: max\_demand = demand; best\_next\_phase = green\_phase\_idx

                if best\_next\_phase != -1:

                    switch\_to\_phase(tls\_id, best\_next\_phase, current\_time)

                elif PHASE\_MAP.get(tls\_id): # Default to first green phase if no demand

                    switch\_to\_phase(tls\_id, list(PHASE\_MAP[tls\_id].keys())[0], current\_time)

        return # Explicit return after handling YELLOW logic

    elif current\_state == "RED\_CLEAR":

        if phase\_elapsed\_time >= ALL\_RED\_TIME:

            max\_demand = -1; best\_next\_phase = -1

            for green\_phase\_idx in PHASE\_MAP.get(tls\_id, {}).keys():

                demand = get\_phase\_demand(tls\_id, green\_phase\_idx)

                if demand > max\_demand: max\_demand = demand; best\_next\_phase = green\_phase\_idx

            if best\_next\_phase != -1:

                switch\_to\_phase(tls\_id, best\_next\_phase, current\_time)

            # --- CORRECTED INDENTATION ---

            elif PHASE\_MAP.get(tls\_id): # Default to first green phase

                switch\_to\_phase(tls\_id, list(PHASE\_MAP[tls\_id].keys())[0], current\_time)

        return # Explicit return after handling RED\_CLEAR logic

    elif current\_state == "GREEN":

        if phase\_elapsed\_time >= MIN\_GREEN\_TIME:

            current\_phase\_demand = get\_phase\_demand(tls\_id, current\_phase\_index); conflicting\_demand = 0

            for other\_phase\_idx in PHASE\_MAP.get(tls\_id, {}).keys():

                 if other\_phase\_idx != current\_phase\_index: conflicting\_demand = max(conflicting\_demand, get\_phase\_demand(tls\_id, other\_phase\_idx))

            switch\_decision = False

            if phase\_elapsed\_time >= MAX\_GREEN\_TIME: switch\_decision = True

            elif current\_phase\_demand == 0 and conflicting\_demand > 0: switch\_decision = True

            elif conflicting\_demand > current\_phase\_demand: switch\_decision = True

            if switch\_decision:

                yellow\_phase = YELLOW\_PHASE\_INDICES.get(tls\_id, {}).get(current\_phase\_index)

                if yellow\_phase is not None: switch\_to\_phase(tls\_id, yellow\_phase, current\_time)

                else:

                     all\_red\_phase = ALL\_RED\_PHASE\_INDICES.get(tls\_id);

                     if all\_red\_phase is not None: switch\_to\_phase(tls\_id, all\_red\_phase, current\_time)

                     else: # No yellow, no all\_red -> decide next green directly

                          max\_demand = -1; best\_next\_phase = -1

                          for green\_phase\_idx in PHASE\_MAP.get(tls\_id, {}).keys():

                               if green\_phase\_idx != current\_phase\_index:

                                    demand = get\_phase\_demand(tls\_id, green\_phase\_idx)

                                    if demand > max\_demand: max\_demand = demand; best\_next\_phase = green\_phase\_idx

                          if best\_next\_phase != -1: switch\_to\_phase(tls\_id, best\_next\_phase, current\_time)

                          elif PHASE\_MAP.get(tls\_id): switch\_to\_phase(tls\_id, list(PHASE\_MAP[tls\_id].keys())[0], current\_time)

# --- update\_active\_missions (Manual Stop Version - Unchanged) ---

def update\_active\_missions(active\_missions, available\_ambulances, router):

    finished\_mission\_ambulance\_ids = []; current\_time = traci.simulation.getTime(); ambulance\_became\_free = False

    for amb\_id, mission in list(active\_missions.items()):

        current\_edge = None

        try:

            if amb\_id not in traci.vehicle.getIDList():

                if mission["stage"] != "done":

                    print(f"MISSION COMPLETE: {amb\_id} finished (veh removed) for {mission['patient\_id']} at T={current\_time:.1f}s.")

                    mission["stage"] = "done"

                    if mission.get("stage") in ["to\_hospital", "pickup"] and mission.get("hospital\_id") in HOSPITALS:

                         HOSPITALS[mission["hospital\_id"]]["available\_beds"] = min(HOSPITALS[mission["hospital\_id"]]["available\_beds"] + 1, HOSPITALS[mission["hospital\_id"]]["initial\_beds"])

                    finished\_mission\_ambulance\_ids.append(amb\_id)

                continue

            current\_edge = traci.vehicle.getRoadID(amb\_id)

            if mission["stage"] == "to\_patient":

                if current\_edge == mission["patient\_edge"]:

                    print(f"INFO: {amb\_id} reached patient {mission['patient\_id']} at T={current\_time:.1f}s. Stopping for pickup.")

                    traci.vehicle.setSpeed(amb\_id, 0); mission["stage"] = "pickup"; mission["pickup\_start\_time"] = current\_time

                elif current\_time % 5 == 0: traci.vehicle.changeTarget(amb\_id, mission["patient\_edge"])

            elif mission["stage"] == "pickup":

                pickup\_duration = 10.0

                if current\_time >= mission["pickup\_start\_time"] + pickup\_duration:

                    print(f"INFO: {amb\_id} finished pickup for {mission['patient\_id']}. Proceeding.")

                    mission["stage"] = "to\_hospital"; traci.vehicle.setSpeed(amb\_id, -1)

                    traci.vehicle.setSpeedMode(amb\_id, 31) # Ensure still ignoring lights if needed

                    traci.vehicle.changeTarget(amb\_id, mission["hospital\_edge"])

                else: traci.vehicle.setSpeed(amb\_id, 0)

            elif mission["stage"] == "to\_hospital":

                if current\_edge == mission["hospital\_edge"]:

                    print(f"MISSION COMPLETE: {amb\_id} delivered patient {mission['patient\_id']} at T={current\_time:.1f}s.")

                    mission["stage"] = "done"

                    if mission.get("hospital\_id") in HOSPITALS:

                         HOSPITALS[mission["hospital\_id"]]["available\_beds"] = min(HOSPITALS[mission["hospital\_id"]]["available\_beds"] + 1, HOSPITALS[mission["hospital\_id"]]["initial\_beds"])

                    finished\_mission\_ambulance\_ids.append(amb\_id)

                elif current\_time % 5 == 0: traci.vehicle.changeTarget(amb\_id, mission["hospital\_edge"])

        except traci.TraCIException as e:

            print(f"TraCI Error updating {amb\_id}: {e}. Marking done.")

            if mission["stage"] != "done":

                mission["stage"] = "done"

                if mission.get("stage") in ["to\_hospital", "pickup"] and mission.get("hospital\_id") in HOSPITALS:

                     HOSPITALS[mission["hospital\_id"]]["available\_beds"] = min(HOSPITALS[mission["hospital\_id"]]["available\_beds"] + 1, HOSPITALS[mission["hospital\_id"]]["initial\_beds"])

                finished\_mission\_ambulance\_ids.append(amb\_id)

    for amb\_id in finished\_mission\_ambulance\_ids:

        if amb\_id in active\_missions: del active\_missions[amb\_id]

        if amb\_id not in available\_ambulances: available\_ambulances.append(amb\_id); ambulance\_became\_free = True

        deactivate\_priority(amb\_id) # Deactivate priority when mission finishes

    return ambulance\_became\_free

# --- run\_dispatch\_cycle (Unchanged) ---

def run\_dispatch\_cycle(pending\_patient\_ids, available\_ambulance\_ids, active\_missions, router):

    if not pending\_patient\_ids or not available\_ambulance\_ids: return

    current\_time = traci.simulation.getTime()

    print(f"\n--- Running Dispatch Cycle at T={current\_time:.1f}s ---"); print(f"Pending: {len(pending\_patient\_ids)}, Available Amb: {len(available\_ambulance\_ids)}")

    current\_vehicle\_ids = traci.vehicle.getIDList(); valid\_available\_amb\_ids = [a for a in available\_ambulance\_ids if a in current\_vehicle\_ids]

    if not valid\_available\_amb\_ids: print("No valid ambulances in sim."); return

    num\_current\_patients = len(pending\_patient\_ids); num\_available\_ambulances = len(valid\_available\_amb\_ids)

    population = [create\_chromosome\_event(pending\_patient\_ids) for \_ in range(GA\_CONFIG["population\_size"])]

    for gen in range(GA\_CONFIG["generations"]):

        fitnesses = [calculate\_fitness\_event(chrom, router, pending\_patient\_ids, valid\_available\_amb\_ids) for chrom in population]

        valid\_pop\_fitness = [(p, f) for p, f in zip(population, fitnesses) if f != float('inf')]

        if not valid\_pop\_fitness: continue

        pop\_with\_fitness = sorted(valid\_pop\_fitness, key=lambda x: x[1])

        # print(f"  Dispatch Gen {gen+1}, Best Fitness: {pop\_with\_fitness[0][1]:.2f}")

        new\_population = [chrom for chrom, fit in pop\_with\_fitness[:GA\_CONFIG["elitism\_size"]]]

        valid\_population = [p for p, f in pop\_with\_fitness]; valid\_fitnesses = [f for p, f in pop\_with\_fitness]

        num\_to\_generate = GA\_CONFIG["population\_size"] - GA\_CONFIG["elitism\_size"];

        if num\_to\_generate % 2 != 0: num\_to\_generate -= 1

        for \_ in range(num\_to\_generate // 2):

            k = min(GA\_CONFIG["tournament\_size"], len(valid\_population))

            parent1 = selection(valid\_population, valid\_fitnesses) if k > 0 else random.choice(valid\_population) if valid\_population else []

            parent2 = selection(valid\_population, valid\_fitnesses) if k > 0 else random.choice(valid\_population) if valid\_population else []

            if not parent1 or not parent2: continue

            child1, child2 = crossover(parent1, parent2); new\_population.extend([mutate(child1), mutate(child2)])

        if (GA\_CONFIG["population\_size"] - GA\_CONFIG["elitism\_size"]) % 2 != 0 and new\_population: new\_population.append(mutate(new\_population[0]))

        population = new\_population[:GA\_CONFIG["population\_size"]]

    final\_fitnesses = [calculate\_fitness\_event(chrom, router, pending\_patient\_ids, valid\_available\_amb\_ids) for chrom in population]

    valid\_final\_pop\_fitness = [(p, f) for p, f in zip(population, final\_fitnesses) if f != float('inf')]

    if not valid\_final\_pop\_fitness: print("Dispatch Cycle Error: No valid final plan."); return

    best\_plan\_chromosome = min(valid\_final\_pop\_fitness, key=lambda x: x[1])[0]

    num\_assignments = min(num\_current\_patients, num\_available\_ambulances); severity\_map = {"high": 0, "medium": 1, "low": 2}

    sorted\_pending\_patient\_ids = sorted(pending\_patient\_ids, key=lambda p\_id: severity\_map.get(ALL\_PATIENTS.get(p\_id, {}).get('severity'), 99))

    assignable\_patient\_ids = sorted\_pending\_patient\_ids[:num\_assignments]; patient\_indices\_in\_chromosome = {p\_id: pending\_patient\_ids.index(p\_id) for p\_id in assignable\_patient\_ids}

    amb\_edges = [traci.vehicle.getRoadID(amb\_id) for amb\_id in valid\_available\_amb\_ids]; cost\_matrix = np.full((num\_available\_ambulances, num\_assignments), float('inf'))

    for i in range(num\_available\_ambulances):

        for j, patient\_id in enumerate(assignable\_patient\_ids):

            chromosome\_idx = patient\_indices\_in\_chromosome[patient\_id]; hospital\_id = best\_plan\_chromosome[chromosome\_idx]

            patient\_info = ALL\_PATIENTS.get(patient\_id); hospital\_info = HOSPITALS.get(hospital\_id);

            if not patient\_info or not hospital\_info: continue

            time\_to\_patient = router.find\_shortest\_path\_time(amb\_edges[i], patient\_info["start\_edge"])

            time\_to\_hospital = router.find\_shortest\_path\_time(patient\_info["start\_edge"], hospital\_info["dest\_edge"])

            if time\_to\_patient != float('inf') and time\_to\_hospital != float('inf'):

                mission\_time = time\_to\_patient + time\_to\_hospital; weighted\_time = mission\_time \* SEVERITY\_WEIGHTS.get(patient\_info.get("severity", "medium"), 1.0)

                cost\_matrix[i, j] = weighted\_time

    try:

        if np.all(cost\_matrix == float('inf')): amb\_indices, assigned\_pat\_indices\_in\_matrix = [],[]

        else: amb\_indices, assigned\_pat\_indices\_in\_matrix = linear\_sum\_assignment(cost\_matrix)

    except ValueError: print("Dispatch Cycle Error: Final assignment failed."); return

    assigned\_ambulance\_ids\_this\_cycle = set(); dispatched\_patient\_ids\_this\_cycle = set()

    print("--- Executing Optimal Assignments ---")

    for amb\_idx, matrix\_pat\_idx in zip(amb\_indices, assigned\_pat\_indices\_in\_matrix):

        if cost\_matrix[amb\_idx, matrix\_pat\_idx] == float('inf'): continue

        amb\_id = valid\_available\_amb\_ids[amb\_idx]; patient\_id = assignable\_patient\_ids[matrix\_pat\_idx]

        chromosome\_idx = patient\_indices\_in\_chromosome[patient\_id]; hospital\_id = best\_plan\_chromosome[chromosome\_idx]

        if amb\_id in assigned\_ambulance\_ids\_this\_cycle or patient\_id not in pending\_patient\_ids: continue

        if HOSPITALS[hospital\_id]["available\_beds"] <= 0: print(f"Skipping assignment: No beds at {hospital\_id} for {patient\_id}"); continue

        patient\_edge = ALL\_PATIENTS[patient\_id]["start\_edge"]; hospital\_edge = HOSPITALS[hospital\_id]["dest\_edge"]

        beds\_before = HOSPITALS[hospital\_id]["available\_beds"]; dispatch\_detail = {'patient\_id': patient\_id, 'ambulance\_id': amb\_id, 'hospital\_id': hospital\_id,'beds\_at\_dispatch': beds\_before, 'dispatch\_time': current\_time}

        try:

            traci.vehicle.changeTarget(amb\_id, patient\_edge)

            active\_missions[amb\_id] = {"stage": "to\_patient", "patient\_id": patient\_id, "patient\_edge": patient\_edge, "hospital\_edge": hospital\_edge, "hospital\_id": hospital\_id}

            HOSPITALS[hospital\_id]["available\_beds"] -= 1; ALL\_PATIENTS[patient\_id]["status"] = "assigned"

            assigned\_ambulance\_ids\_this\_cycle.add(amb\_id); dispatched\_patient\_ids\_this\_cycle.add(patient\_id); generate\_dispatch\_report\_event(dispatch\_detail)

            print(f"  Assigned: {amb\_id} -> {patient\_id} ({ALL\_PATIENTS[patient\_id]['severity']}) -> {hospital\_id} ({HOSPITALS[hospital\_id]['name']})")

        except traci.TraCIException as e:

             print(f"Error dispatching {amb\_id} -> {patient\_id}: {e}. Reverting."); HOSPITALS[hospital\_id]["available\_beds"] += 1; ALL\_PATIENTS[patient\_id]["status"] = "pending"

             if amb\_id in active\_missions and active\_missions[amb\_id]["patient\_id"] == patient\_id: del active\_missions[amb\_id]

    available\_ambulance\_ids[:] = [amb for amb in available\_ambulance\_ids if amb not in assigned\_ambulance\_ids\_this\_cycle]

    pending\_patient\_ids[:] = [pat for pat in pending\_patient\_ids if pat not in dispatched\_patient\_ids\_this\_cycle]

    print(f"--- Dispatch Cycle End: {len(pending\_patient\_ids)} pending, {len(available\_ambulance\_ids)} available ---")

# --- Main Simulation Entry Point ---

def run\_simulation():

    sumo\_cmd = [os.path.join(os.environ.get("SUMO\_HOME", "."), "bin", "sumo-gui"),

                "-c", "hexagon.sumocfg",

                "--tripinfo-output", "tripinfo\_results.xml",

                "--start", # Start paused

                "--quit-on-end"

               ]

    traci.start(sumo\_cmd)

    # Clear/Create summary file & Reset counter

    with open("dispatch\_summary.txt", "w") as f: f.write("")

    global mission\_log\_counter; mission\_log\_counter = 1

    # Add dynamic POIs

    print("Adding dynamic patient POIs...")

    for patient\_id, info in ALL\_PATIENTS.items():

        try:

            edge\_id = info['start\_edge']; lane\_id = edge\_id + "\_0"; length = traci.lane.getLength(lane\_id)

            x, y = traci.simulation.convert2D(edge\_id, length / 2)

            traci.poi.add(patient\_id, x, y, color=(255,0,0,255), poiType="patient", layer=10)

        except traci.TraCIException as e: print(f"Warn: POI add failed {patient\_id}: {e}")

    try: router = DijkstraForSUMO('hexagon.net.xml')

    except Exception as e: print(f"CRIT ERR Dijkstra: {e}"); traci.close(); sys.exit(1)

    # Event-Driven State Variables

    pending\_patients = list(ALL\_PATIENTS.keys())

    available\_ambulances = list(AMBULANCES)

    active\_missions = {}

    needs\_dispatch\_run = True

    # Reset hospital beds

    for h\_id in HOSPITALS: HOSPITALS[h\_id]["available\_beds"] = HOSPITALS[h\_id]["initial\_beds"]

    # Initialize TLS states

    global tls\_current\_phase\_indices, tls\_phase\_start\_times, tls\_states

    tls\_current\_phase\_indices = {}; tls\_phase\_start\_times = {}; tls\_states = {}

    # Initialize prioritized vehicles state

    global prioritized\_vehicles; prioritized\_vehicles = {}

    # --- Main Event Loop ---

    step = 0; max\_steps = 1000; cooldown\_start = -1

    while step < max\_steps:

         try:

              traci.simulationStep()

              current\_time = traci.simulation.getTime()

              # --- MAIN LOOP ORDER ---

              # 1. Check ambulance proximity & manage priority state (activate/deactivate visuals, setSpeedMode)

              check\_ambulance\_proximity\_and\_manage\_priority(active\_missions, current\_time)

              # 2. Run Actuated Traffic Light Logic for ALL lights (will skip if priority active)

              for tls\_id in TLS\_IDS:

                   # Pass the \*correct\* priority state dictionary

                   run\_adaptive\_tls\_logic(tls\_id, current\_time, prioritized\_vehicles)

              # 3. Update active ambulance missions (movement, pickup timer, delivery check)

              ambulance\_became\_free = update\_active\_missions(active\_missions, available\_ambulances, router)

              # 4. Check for new patients (placeholder)

              new\_patient\_added = False

              # 5. Trigger dispatch cycle if needed

              if (needs\_dispatch\_run or ambulance\_became\_free or new\_patient\_added) and pending\_patients and available\_ambulances:

                  current\_vehicle\_ids = traci.vehicle.getIDList()

                  valid\_available\_amb\_ids = [a for a in available\_ambulances if a in current\_vehicle\_ids]

                  if valid\_available\_amb\_ids:

                      run\_dispatch\_cycle(pending\_patients, valid\_available\_amb\_ids, active\_missions, router)

                      needs\_dispatch\_run = False

              # 6. Check end condition with cooldown

              if not pending\_patients and not active\_missions and step > 10:

                   if cooldown\_start == -1: print(f"All missions done T={current\_time:.1f}s. Cooldown."); cooldown\_start = current\_time

                   if current\_time >= cooldown\_start + 10: print("Cooldown finished. Ending."); break

              else: cooldown\_start = -1

              step += 1

         except traci.FatalTraCIError: print("TraCI connection lost."); break

         except Exception as e: print(f"!! Runtime Error step {step}: {e}"); break

    print(f"\nSimulation ended at time: {traci.simulation.getTime():.2f}")

    traci.close()

if \_\_name\_\_ == "\_\_main\_\_":

    if "SUMO\_HOME" not in os.environ: sys.exit("Declare SUMO\_HOME.")

    run\_simulation()

traci\_runner.python file

