JAMACT

UN MODELLO DI INDUZIONE DINAMICA DEL PHANTOM TRAFFIC JAM PER AGEVOLARE IL TRANSITO DEI VEICOLI DI TRASPOSTO PUBBLICO

CANDIDATO

Biagio Scotto di Covella

RELATORE

Prof. Walter Balzano

Presentazione Tesi di Laurea Triennale in Informatica, 12/07/2024

INDICE

1. Introduzione

2. Phantom Traffic Jam

3. Modello Jamact

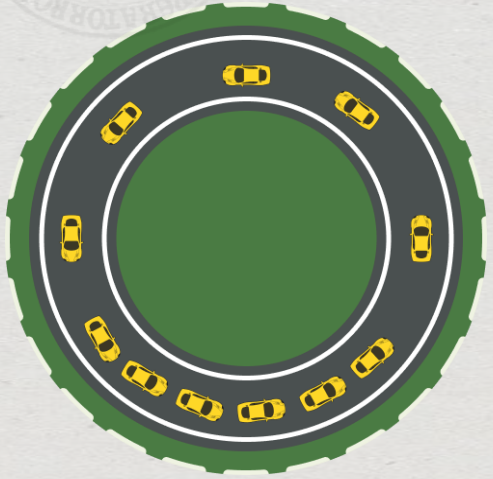
4. Caso d'uso: Autobus

*Dati tratti dal "20° Rapporto sulla mobilità degli italiani" di Isfort

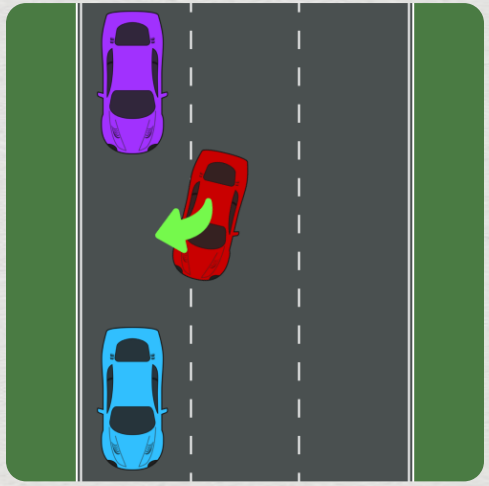
*Dati tratti dal "20° Rapporto sulla mobilità degli italiani" di Isfort

IL TRAFFICO FANTASMA

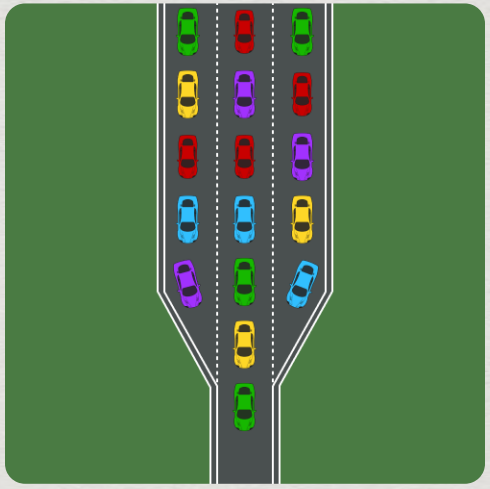
- Formazione: **frenate improvvise** e **cambi di corsia** che causano **onde di rallentamento** tra i veicoli, creando ingorghi senza cause visibili.
- Jamitons: **onde di congestione** che si propagano lungo una fila di veicoli, simili alle onde sonore, causate da **variazioni nella velocità** dei conducenti.



The Wide Moving Jam



The Butterfly Effect



The Bottleneck

CAMBIO DI PROSPETTIVA

PROBLEMA → SOLUZIONE

Il traffico fantasma può essere indotto per agevolare il transito di veicoli prioritari

MODELLO JAMACT

CONTESTO

- Smart Cities
- Veicoli connessi e autonomi
- Internet of things

ELEMENTI FONDANTI

- Veicoli Prioritari
- Punti Critici
- Controllo della Velocità
- Adeguamento Dinamico

OBIETTIVI

- Agevolare transito veicoli prioritari
- Gestire efficacemente il flusso di traffico

MODELLO JAMACT

STATIC PREVISIONING

1. Inserimento destinazione veicolo prioritario
2. Individuazione dei punti critici
3. Calcolo dei punti di gestione di Jamact

DYNAMIC ADAPTION

1. Controllo avanzamento veicolo prioritario
2. Attivazione sistema e rallentamento
3. Ricalcolo intensità del jamiton

IMPLEMENTAZIONE

```
def critical_point(graph, graph_undirected,
   start_point, end_point):
    start_node = ox.distance.nearest_nodes(graph,
       X=start_point[1], Y=start_point[0])
    end_node = ox.distance.nearest_nodes(graph, X=
       end_point[1], Y=end_point[0])

    path_nodes = nx.shortest_path(graph,
       start_node, end_node, 'length')
    internal_path_nodes = path_nodes[1:-1]

    critical_points = [node for node in
       internal_path_nodes if 'street_count' in
       graph_undirected.nodes[node] and
       graph_undirected.nodes[node]['street_count
       '] > 2]
    return path_nodes, critical_points
```

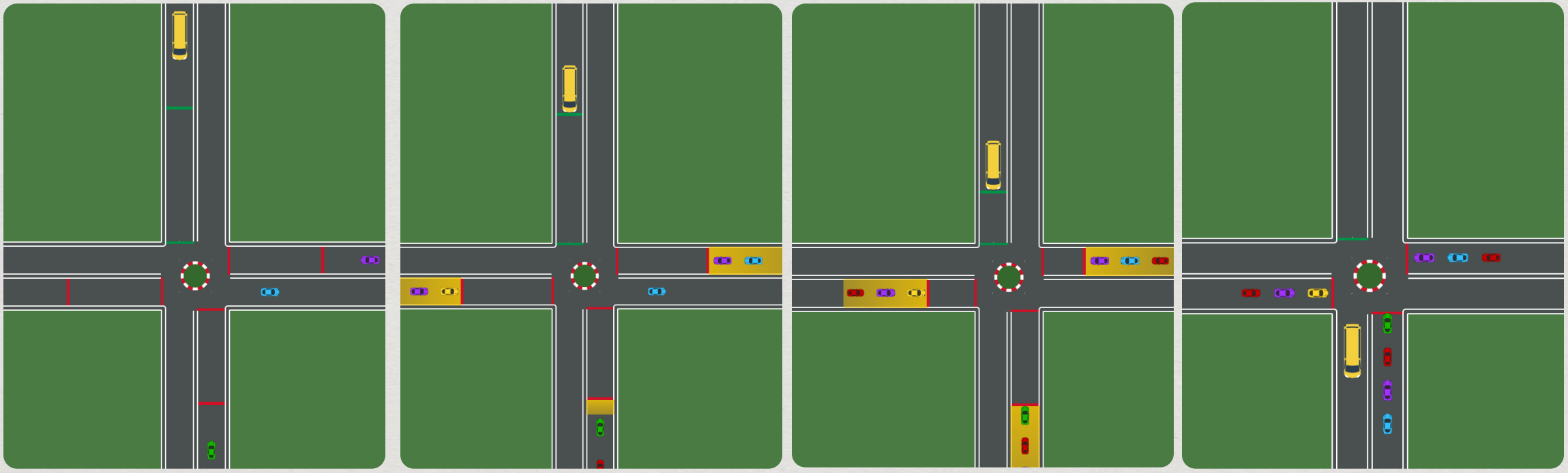
```
def generateVehicle(number_vehicle):

    new_vehicle = PriorityVehicle(index,
       vehicle_type, start_number, end_number,
       coordinates_priority, stop_lines_priority,
       activation_point, speed_priority)
    vehicles_queue = deque()

    while number_vehicle > count_vehicle:
        count_vehicle += 1
        vehicle_type = random.randint(0, 4)
        start_number = random.choice([0, 1, 2, 3])
        end_number = random.choice([0, 1, 2, 3])
        coordinates = generateCoordinate()
        stop_lines = next_critical_point(
           coordinates)
        starting_point = next_starting_point(
           stop_lines)
        speed = random.randint(0, 70) / 3.6
        index = len(vehicles_queue)

        new_vehicle = Vehicle(index, vehicle_type,
            start_number, end_number, coordinates,
            stop_lines, starting_point, speed)
        vehicles_queue.append(new_vehicle)

    return vehicles_queue
```

IMPLEMENTAZIONE

```
def moveAllVehicles(vehicles_queue, jam):

    def vehicle_thread(vehicle):
        while not stop_event.is_set():
            if jam == 1:
                vehicle.moveJam(vehicles_queue)
            else:
                vehicle.moveNoJam(vehicles_queue)

    threads = []
    for vehicle in vehicles:
        thread = threading.Thread(target=
           vehicle_thread, args=(vehicle,))
        threads.append(thread)
        thread.start()

    for thread in threads:
        thread.join()
```

```
def moveJam(vehicle)
    if vehicle.cross_starting_point and
       priorityVehicle.cross_activation_point:
        update_speed = updateSpeed()
        if update_speed <= vehicle.speed:
            vehicle.speed = update_speed

    vehicle.advances()

    if vehicle.arrived_stop:
        print("Veicolo arrivato all'incrocio")
```

SIMULAZIONI



AVANZAMENTO VEICOLI | ATTIVAZIONE JAMACT | AVANZAMENTO JAMACT | DISATTIVAZIONE JAMACT

RISULTATI

4,54%

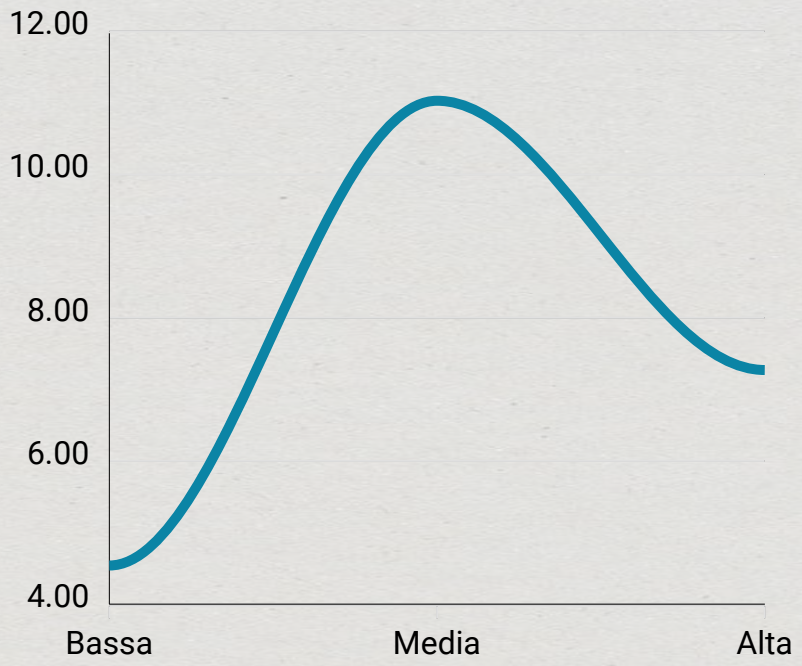
Impatto sui veicoli prioritari con bassa densità di traffico

11,03%

Impatto sui veicoli prioritari con media densità di traffico

7,27%

Impatto sui veicoli prioritari con alta densità di traffico



COME LE SAFETY CAR

AUTO FORMULA 1

VEICOLI NON PRIORITARI

SAFETY CAR

JAMITON

JAMACT

VEICOLI DI EMERGENZA

TRASPORTO PUBBLICO

PERCHÈ GLI AUTOBUS?

3,1%	**8000**	**1**	**7,0%**
Contributo emissioni autobus*	Persone trasportate in un'ora da un autobus*	Autobus	Utilizzo mezzi pubblici tra i cittadini occupati
68,7%	**~900**	**20**	**73,7%**
Contributo emissioni automobili*	Persone trasportate in un'ora da un'automobile*	Automobili	Utilizzo mezzi privati tra i cittadini occupati

*Dati tratti dal "20° Rapporto sulla mobilità degli italiani" di Isfort

JAMACT PER AUTOBUS

STATIC PREVISIONING

1. Calcolo dei tempi di percorrenza
2. Schematizzazione orari finestre di passaggio per un punto critico

→ Fase statica del modello di base

DYNAMIC ADAPTION

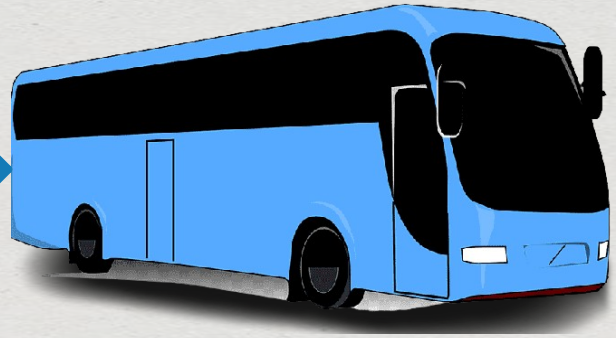
1. Controllo avanzamento autobus e aggiornamento orari
2. Rallentamento graduale veicoli non prioritari

→ Fase dinamica del modello di base

COMMESSO VIAGGIATORE

Gli autobus devono visitare tutte le fermate prefissate in un ordine specifico, minimizzando i tempi di attraversamento dei punti critici

1. Minimizzare il Tempo di Percorrenza degli Autobus
2. Ridurre l'Interferenza dei Veicoli Non Prioritari
3. Ottimizzare il Flusso del Traffico



IMPLEMENTAZIONE

ALGORITMO GREEDY

```
def greedyInitialSolution(critical_points,
   current_time):
    initial_solution = {}

    for point in critical_points:
        arrival_time = calculateArrivalTime(point,
            current_time)
        departure_time = calculateDepartureTime(
           point, arrival_time)
        initial_solution[point['id']] = (
           arrival_time, departure_time)
        current_time = departure_time

    return initial_solution
```

PROGRAMMAZIONE DINAMICA

```
def dynamicProgrammingRefinement(vehicle, point,
   initial_solution, current_time):
    priority_arrival_time = initial_solution[point
       ['id']][0]
    priority_departure_time = initial_solution[
       point['id']][1]
    arrival_time = calculateArrivalTime(point,
       current_time)
    departure_time = calculateDepartureTime(point,
        arrival_time)

    if checkTime(priority_arrival_time,
       priority_departure_time, arrival_time,
       departure_time):
        arrival_time = max(priority_departure_time
           , arrival_time)
        departure_time = calculateDepartureTime(
           point, arrival_time)
        vehicle.speed = adjust_speed(vehicle,
           arrival_time, departure_time)

    return (arrival_time, departure_time)
```

SIMULAZIONI ESTENSIONE JAMACT

Numero corse	1
Validità	Ord.
Bacoli - Via Fusaro	6:00
Bacoli - Sella Di Baia	6:03
Bacoli - Piazza De Gaspari	6:05
Bacoli - Via Risorgimento	6:12
Bacoli - Centro	6:15
Bacoli - Via Lungolago	6:17
Miseno	|
Bacoli - Via Miliscola (lidi)	6:19
Bacoli - Rotonda Miliscola	6:20
Bacoli - Via Mercato di Sabato	6:23
Torregaveta - Cumana	6:25

Numero corse	26
Validità	Ord.
Bacoli - Via Fusaro	14:20
Bacoli - Sella Di Baia	14:24
Bacoli - Piazza De Gaspari	14:27
Bacoli - Via Risorgimento	14:34
Bacoli - Centro	14:37
Bacoli - Via Lungolago	14:39
Miseno	|
Bacoli - Via Miliscola (lidi)	14:39
Bacoli - Rotonda Miliscola	14:40
Bacoli - Via Mercato di Sabato	14:47
Torregaveta - Cumana	14:50

RISULTATI ESTENSIONE JAMACT

24 minuti
TARGET

	Jamat Non Attivo	Jamact Attivo	Diminuzione Tempo
Bassa Densità di Traffico	24 minuti e 27 secondi	24 minuti e 4 secondi	23 secondi
Media Densità di Traffico	25 minuti e 24 secondi	24 minuti e 48 secondi	36 secondi
Alta Densità di Traffico	25 minuti e 38 secondi	25 minuti e 26 secondi	12 secondi

RISULTATI ESTENSIONE JAMACT

1,59%

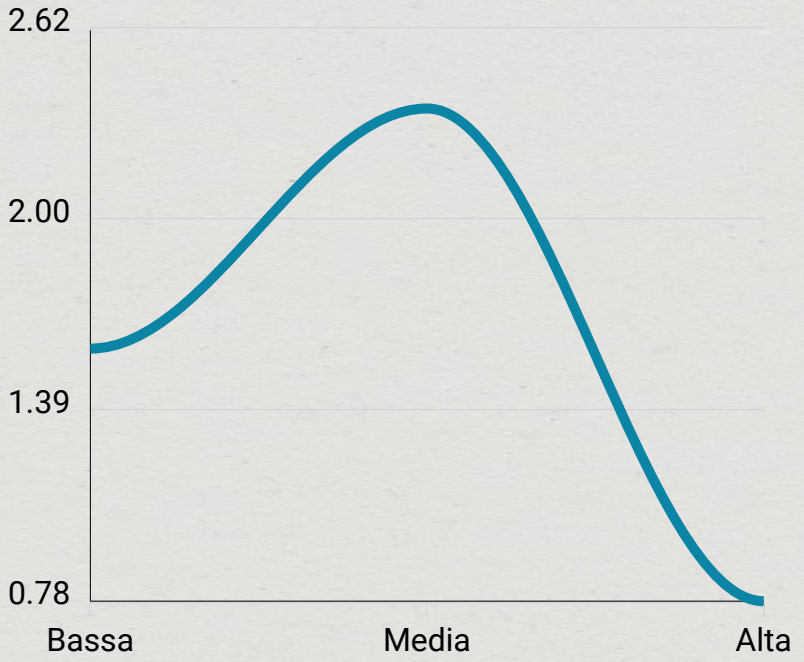
Impatto sui veicoli prioritari con bassa densità di traffico

2,36%

Impatto sui veicoli prioritari con media densità di traffico

0,78%

Impatto sui veicoli prioritari con alta densità di traffico



Grazie

CANDIDATO

Biagio Scotto di Covella

RELATORE

Prof. Walter Balzano

Presentazione Tesi di Laurea Triennale in Informatica, 12/07/2024