GlovoJS: A BDI-based Solution for Autonomous Parcel Delivery

Edoardo Meneghini & Lorenzo Orsingher

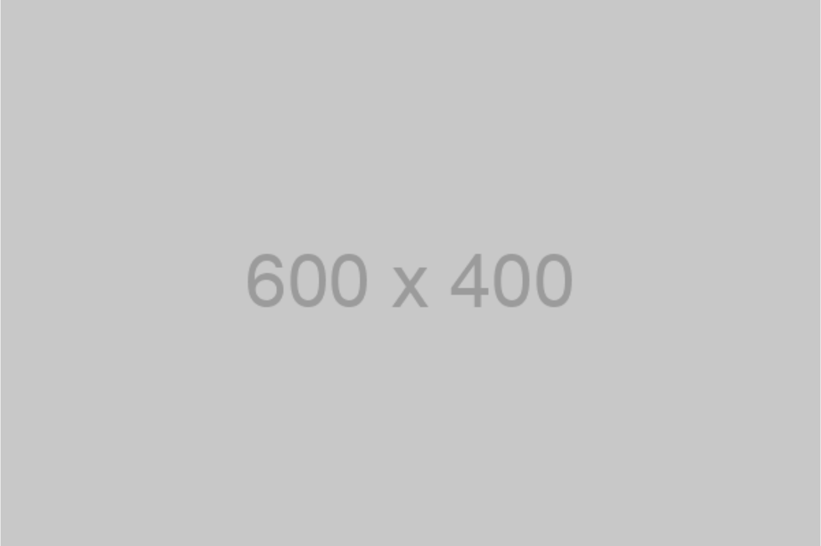
University of Trento, a.y. 2023/2024

Outline

1. Introduction to Deliveroo.js
2. System Architecture
3. Algorithms and Implementation
4. Performance Optimization
5. Evaluation and Results
6. Challenges and Solutions
7. Future Improvements
8. Conclusions

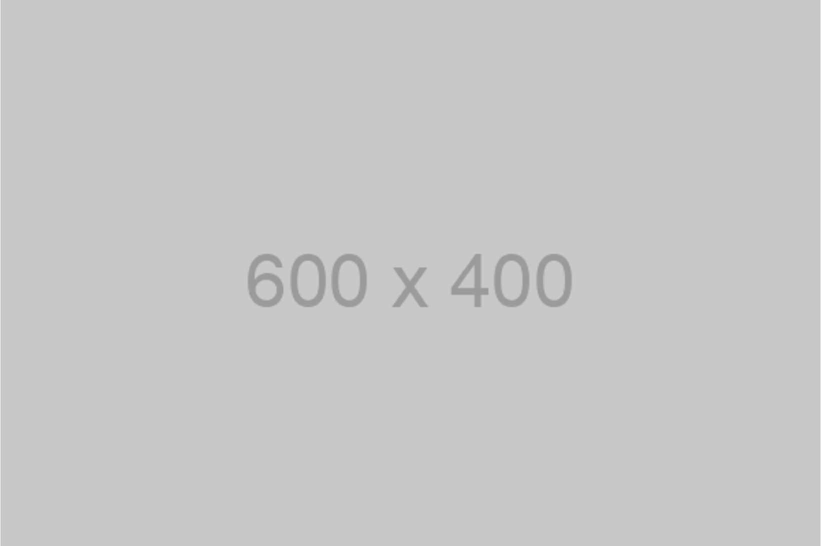
Introduction to Deliveroo.js

- 2D grid-based parcel delivery game
- Agents compete to achieve highest score
- Key features:
    - Dynamic parcel spawning
    - Decaying parcel values
    - Multiple agents
    - Delivery zones



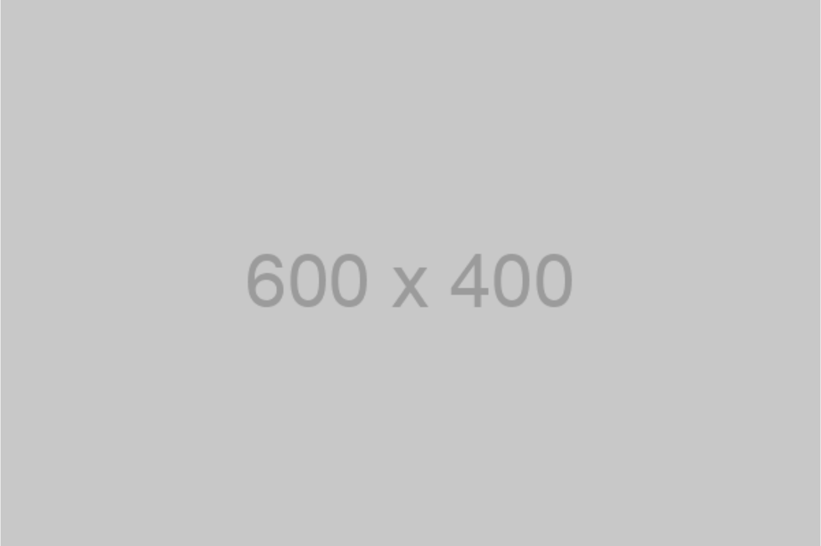
System Architecture

- Modular design for scalability and maintainability
- Key components:
  - Agent
  - Rider
  - Brain (Genetic Algorithm)
  - Field
  - Dashboard



BDI Architecture

- Beliefs: Agent's knowledge of the environment
- Desires: Goals (maximize score)
- Intentions: Generated plans



PDDL Integration

- Core component for pathfinding and planning
- Dynamic generation of PDDL domain and problem descriptions
- Parallel solving for multiple pathfinding problems
- Alternative to traditional pathfinding algorithms

Algorithms: Path Planning

- Breadth-First Search (BFS)
  - Optimal for unweighted graphs
  - Simple implementation
  - Memory efficient
  - Adaptable to dynamic obstacles

Algorithms: Decision Making

- Genetic Algorithm for plan optimization
- Key features:
  - Graph representation of environment
  - Multi-crossover operation
  - Adaptive planning
  - Scalability
  - Long-term optimization

Genetic Algorithm: Key Steps

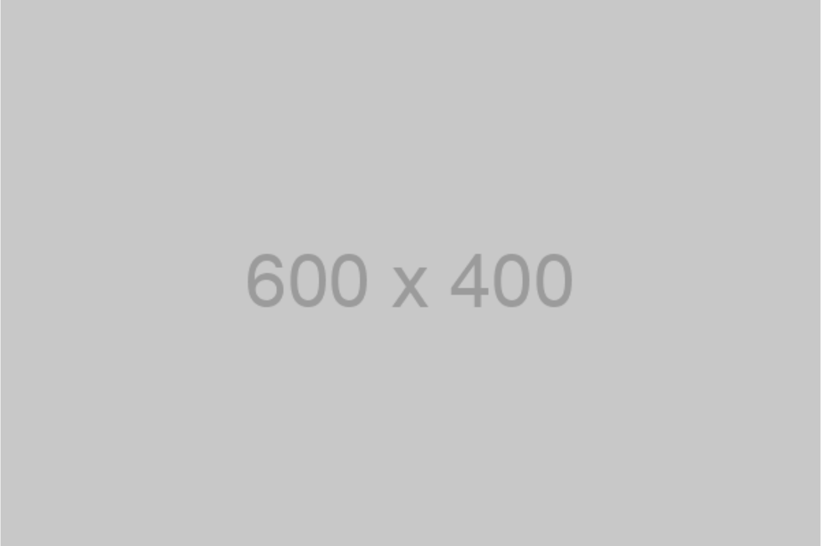
1. Initialize population
2. Evaluate fitness
3. Select parents
4. Perform crossover and mutation
5. Create new generation
6. Repeat until termination condition

Multi-Agent Coordination

- Centralized approach using external "Brain"
- Global state consideration for optimal planning
- Efficient use of PDDL solver
- Linear complexity increase with number of agents
- Constraint: No two plans can contain the same parcel

Performance Optimization: Parallelization

- Bundling multiple pathfinding requests
- Challenges:
    - One unreachable goal invalidates entire problem
    - Exponential complexity growth with goals
- Solution: Chunking large problems



Performance Optimization: Caching

- Storing precomputed paths
- Cache key: concatenation of locations and blocked tiles
- Bidirectional path storage
- Significant performance improvement, especially with PDDL
- "Boost" mode for precomputing paths

Evaluation and Results

- Performance metrics:
  - Dashboard for real-time visualization
  - Analytics for resource usage
  - Enhanced logs for debugging
- Benchmarking scenarios:
  - Single-agent
  - Multi-agent

Benchmarking Results

- Single-agent scenario:
  - BFS agent outperformed PDDL agents
  - BOOSTPDDL showed improvements over ALLPDDL
- Multi-agent scenario:
  - BFS agent still performed best
  - PDDL agents struggled in larger, sparser maps

Challenges and Solutions

1. Local Planner Overhead
    - Solution: Bundling multiple planning queries
2. PDDL Goal Specification Limitations
    - Solution: Fallback mechanism for unachievable goals
3. Large Response Payload Processing
    - Partial offset of gains from reduced call frequency
4. Performance Limitations of Local Solver
    - Remained a bottleneck for real-time decision-making

Future Improvements

1. Solver efficiency
    - Reduce overhead in PDDL solver calls
2. Robust e2e communication protocols
    - Implement end-to-end encryption for agent communication
3. Game-specific strategies
    - Develop advanced tactics for multi-agent coordination

Conclusions

- Successful combination of genetic algorithms, BDI architecture, and PDDL
- Effective multi-agent coordination and path planning
- Areas for improvement:
    - Computational efficiency in dynamic environments
    - Advanced planning strategies
    - Integration of PDDL-based problem solving

Thank You!

Questions?