Solving the Curse of Dimensionality

Graph-Based Computation for Activity-Based Travel Demand Models

RESEARCH OBJECTIVE

Solving the curse of dimensionality of DP nature on DDCM activity-based travel demand model with **graph-based computation approach**.

Press Space for next page →

Part I: Theoretical Foundation

Understanding the Problem & Our Approach

The DDCM Framework

Dynamic Discrete Choice Models model sequential decision-making over time.

THE CORE EQUATION (BELLMAN)

$$V(s) = \ln \sum_{a} \exp\left(u(s,a) + V(s')\right)$$

- $V(s)$: Value of being in state s
- $u(s,a)$: Immediate utility of action a
- **LogSumExp**: "Soft Max" from Gumbel-distributed errors

The Curse of Dimensionality

A state is a **multi-dimensional tuple**:

Dimension	Size	Example
Location	1,240	zones
Mode	8	car, bus, walk...
Activity	10	work, shop, leisure...
Time	96	15-min steps
History	varies	mandatory progress

Why It's Computationally Intractable

MEMORY

Storing the utility array for a realistic city requires **Terabytes of RAM**

TIME

Västberg (2020): 4-10s per agent → Full city: **1,000+ days**

THE DENSE MATRIX TRAP

Traditional Recursive Logit requires matrix inversion:

$$Z = (I - M)^{-1} b$$

Even if M is sparse, the inverse is **always dense** → $O(N^3)$ complexity

Our Objective

THE PROBLEM

- ❌ Intractable for large cities
- ❌ Requires massive clusters
- ❌ Slow iteration cycles

TARGET OUTCOME

- ✓ Orders of magnitude faster
- ✓ Graph-based computation
- ✓ Consumer GPU hardware

The Key Insight

Reachability & Condition-Based Pruning

Inspiration: Reachability Analysis

From **Control Theory** and **Robotics** (Doshi et al., 2022)

We don't calculate paths for *every* point. We calculate **Reachable Tubes**:

- **BRT**: States that *can* reach the target
- **FRT**: States reachable from start

> "Don't solve for the world. Solve for the Tube."

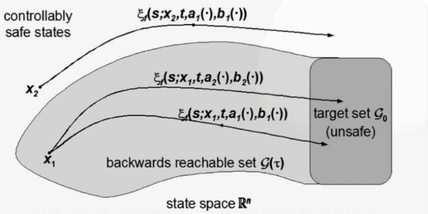


Fig. 1: Target set and backward reachable set. Several trajectories are shown starting at the same time t but from different states x and subject to different input signals $a(\cdot)$ and $b(\cdot)$. Input signal $a(\cdot)$ is chosen to drive the trajectory away from the target set, while input signal $b(\cdot)$ is chosen to drive the trajectory toward the target. Figure taken from [37].

Mapping to DDCM

Apply **Conditions** to prune the state space:

HARD CONSTRAINTS

- **Time**: Can't teleport
- **Geography**: Won't walk 50km
- **Logic**: Can't drive without car

THE RESULT

`S_reachable « S_naive`

~99.99%

State Reduction

Graph Theory Foundations

The Paradigm Shift

From Matrices to Graphs

OLD: MATRIX INVERSION

$Z = (I - M)^{-1} b$

- Store full transition matrix
- Inversion is $O(N^3)$
- Dense output

NEW: GRAPH TRAVERSAL

$G = (V, E)$

- Store only reachable states
- BFS is $O(V + E)$
- Stays sparse

Level-Synchronous BFS

Process graph **layer by layer** where layers = time steps

The "Wave" Algorithm:

Gather all states at time t

Deduplicate identical states (merge paths)

Expand to generate next states

Scatter to future time buckets

WHY IT WORKS

- Maximizes GPU parallelism (SIMD)
- Prevents exponential explosion via deduplication
- Natural fit for time-dependent problems

CSR: Compressed Sparse Row

Memory-efficient graph storage

Format	1.5M States, 326M Edges
Dense Matrix	8.7 TB ❌
Edge List (COO)	2.6 GB
CSR	**1.3 GB ✓**

HOW CSR WORKS

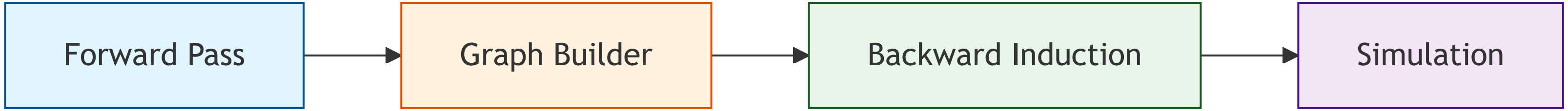
Store only non-zero entries using 3 arrays:

- **Row Pointers**: Where each row starts
- **Column Indices**: Destinations

Part II: Implementation

The 4-Phase Pipeline

Overall Architecture



Phase	Time	What It Does
Forward Pass	~30s	Discover reachable states
Graph Build	~29s	Construct CSR graph
Backward Induction	~2s/home	Compute value function
Simulation	~0.4s	Generate trajectories

State Representation

State Tensor: (N, 8) integers on GPU

Col	Name	Range
0	Time	0-1440
1	Zone	1-144
2	Activity	0-9
3	Duration	0-31
4	Mode	0-7
5-6	Vehicle In Use	0/1
7	History	0-15

Backward Induction

Solving the Bellman Equation

$$V(s) = \ln \sum_a \exp(u(s,a) + V(s'))$$

KEY IMPLEMENTATION: SCATTER LOGSUMEXP

- Aggregate Q-values by source state
- Uses "max trick" for numerical stability
- Processes **326M edges** in one GPU operation

Performance: ~2 seconds per home zone

Optimization 1: Universal Graph

24.7× Speedup

The Waste in Per-Home Graphs

OLD APPROACH

```
144 homes × (Forward 30s + Graph 29s + BI 2s) = 8,784s
```

THE PROBLEM

Each graph has ~1.5M states, but they **heavily overlap**!

Universal Graph Solution

KEY INSIGHT (BISIMULATION)

Once you **leave home**, your options depend on **current state**, not origin.

Time	1 Home	144 Homes
0:00	1	144
10:00	12,000	12,200
14:00	24,000	24,400

NEW APPROACH

```
Forward Pass (144 origins): 30s  ← ONCE
Graph Build: 29s                 ← ONCE
BI per home: 144 × 2s = 288s
TOTAL: 357s → 24.7× faster!
```

Optimization 2: Basis Function

9.3× BI Speedup

The Theory

LINEAR VALUE FUNCTION APPROXIMATION

$$\hat{V}(s,h) = V_{ref}(s) + \beta \times \Delta TT$$

where ΔTT = travel time difference to home

WHY IT WORKS

- V functions differ mainly by **terminal location**
- Difference propagates **linearly** through Bellman
- Train β on 8 zones → predict all 144

Result: R^2 = 0.98, **9.3× faster**

Optimization 3: RMDP

20,736× Reduction

The Scaling Problem

HOME × WORK × SCHOOL COMBINATIONS

Scenario	Graphs Needed	Time
144 homes × 144 works	20,736	**13+ days**
Universal Graph	144	~3 hours

We need something better...

RMDP: Relational MDP

CORE IDEA (BOUTILIER ET AL., 2001)

Use **abstract roles** instead of concrete zones:

ZoneID	Type	Description
`HOME`	Abstract	Agent’s home
`WORK`	Abstract	Agent’s work
`ZONE_1..144`	Concrete	Fixed zones

The decision structure is identical for all agents!

RMDP in Action

ZONEBINDING: RESOLVE AT RUNTIME

```
State: "I'm at WORK at 5 PM"

Alice (WORK = Zone 50) → resolves to Zone 50
Bob (WORK = Zone 73)   → resolves to Zone 73
```

One graph works for ALL (home, work) combinations

RMDP Performance

Metric	Before	After	Improvement
Graphs needed	20,736	**1**	**20,736×**
Build time	13+ days	~47s	~24,000×
Memory	O(N graphs)	O(1)	Massive

Results Summary

Full Pipeline Comparison

144 HOME ZONES × 1 MANDATORY SEQUENCE

Phase	Original	+ Universal	+ Basis Func
Forward	144×30s	**30s**	**30s**
Graph	144×29s	**29s**	**29s**
BI	288s	288s	**20s**
TOTAL	**8,784s**	**347s**	**79s**
Speedup	1×	25×	**111×**

RMDP: The Ultimate Scaling

144 × 144 HOME × WORK COMBINATIONS

Approach	Graphs	Time
Original	20,736	**13+ days**
RMDP	**1**	**~47s**

Research Contributions

Integration Gap: Discrete Choice + Reachability Analysis

Bisimulation Gap: Universal Graph via State Aggregation

Basis Function Gap: Travel-time VFA for spatial transfer

RMDP Gap: Role abstraction in activity-based modeling

Summary

24.7×

Universal Graph

9.3×

Basis Function

20,736×

RMDP

Combined: ~100× - 20,000× faster

From **13 days** → **47 seconds**

Future Direction: Generalized Conditions

Beyond optimization → **Generalization**

THE VISION: PLUG-AND-PLAY CONSTRAINTS

Make conditions modular so new constraints can be added without rewriting the algorithm

EXAMPLE EXTENSIONS

- Joint activities (with household)
- Social coordination constraints
- Multi-agent interactions

IDEAS BEING EXPLORED

- **LLM as Input**: Natural language → constraints
- **Inverse-RL**: Learn constraints from observed data

Thank You

Solving the Curse of Dimensionality in DDCM

Graph-Based Computation | PyTorch & CUDA | CSR Graphs | Level-Sync BFS