

Reducing Memory Usage in Fast Downward by Working with a Delta Representation of States

Bachelor's Thesis

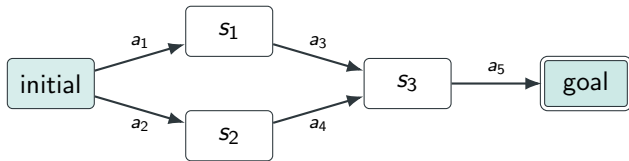
Istref Uka <i.uka@stud.unibas.ch>

Artificial Intelligence Research Group, University of Basel

07.09.2026

Introduction to Fast Downward

- > Given in **planning task**: initial state, possible actions, goal condition
- > **State space**: nodes are states, edges are actions
- > **Graph search** stores encountered states to detect duplicates



Two paths reach the same state s_3
⇒ duplicate detection avoids re-expansion

Existing Full-State Representation

Parent state

0	1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---	---

Successor state

0	1	1	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---	---

Only variable 2 changed.

- > States stored in State Registry
- > Every registered state receives a **StateID**.
- > The **IntPacker** uses only as many bits as each variable domain requires.
- > Several values share one 32-bit PackedStateBin.
- > Every state stores information for **all variables**.

Delta-State Representation

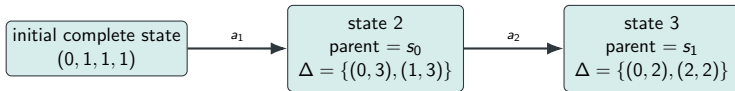
- Store a reference to the predecessor.
- Store only changed **variable-position** / **new-value** pairs.
- Conceptually similar to differential encoding or a sparse difference representation.

$$\Delta(s, s') = \{(v, s'[v]) \mid s[v] \neq s'[v]\}.$$

Potential benefit

If a full state is large but $|\Delta|$ is small, the delta can be much smaller than the complete state.

Reconstructing a Full State



reconstruction direction: state 3 → state 2 → initial

reconstructed state 3: (2, 3, 2, 1)

Concrete Implementation: StateID as the Reference

DeltaStateInfo

```
parent_state : StateID  
effs : unsigned int
```

DeltaPacker

encodes position/value pairs

DeltaStateIDSet

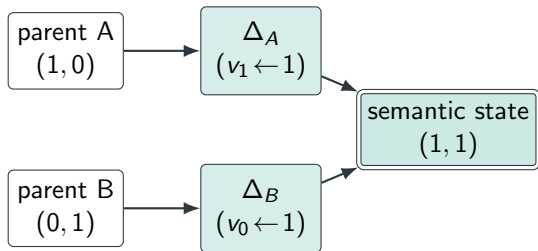
duplicate detection

- Existing **StateID** is reused as the parent reference.
- Minimum delta-state size on the experimental platform: **8 bytes**.
 - 4 bytes parent StateID
 - 4 bytes encoded effects
- A complete packed state can be only **4 bytes**.

Important consequence

Delta-states are **not automatically smaller**. The representation only wins when the full state is sufficiently large relative to the delta.

Duplicate Detection is More Expensive



- Different parents and different deltas can represent the **same semantic state**.
- Raw delta data therefore cannot be used as semantic equality.
- Equality checks may require reconstruction of both states.

Runtime impact

Duplicate detection adds another place where reconstruction can occur repeatedly during search.

Experimental Setup – Run 1

- **Low-change:** exactly 1 changed SAS+ variable per transition.
- **High-change:** Blocks World variant with about 10–11 actual changes per transition.
- Same search settings for original eager and delta-state eager search;
- **Heuristics:**
 - $h_{\max}$ for the low-change problem.
 - Blind heuristic for the high-change problem.
- **Metrics:**
 - Bytes per stored state.
 - Peak memory usage.
 - Search time.

Experimental Setup – Run 2

- › Domains: Blocks, Elevators, Gripper, Logistics, Barman, and Agricola.
- › `lmcut` heuristic for all tasks.
- › Experiments executed on sciCORE.
- › Overall time limit of 20 minutes per run.
- › 173 instances in total:
 - › 77 completed successfully.
 - › 96 timed out.
 - › Outcomes were identical for both implementations.
- › **Metrics:** same as run 1

Memory Results: Controlled Problems

Problem	Bytes per state		Peak memory [KB]	
	Eager	Delta	Eager	Delta
Low-change	16	8	30,528	28,044
High-change	8	20	24,780	28,956

Low-change

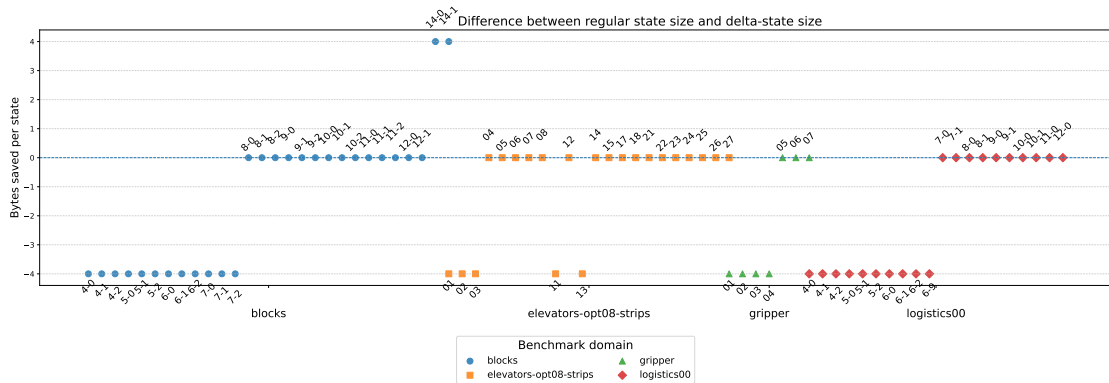
50% less direct state storage, and lower peak memory.

High-change

The delta becomes larger than the complete packed state.

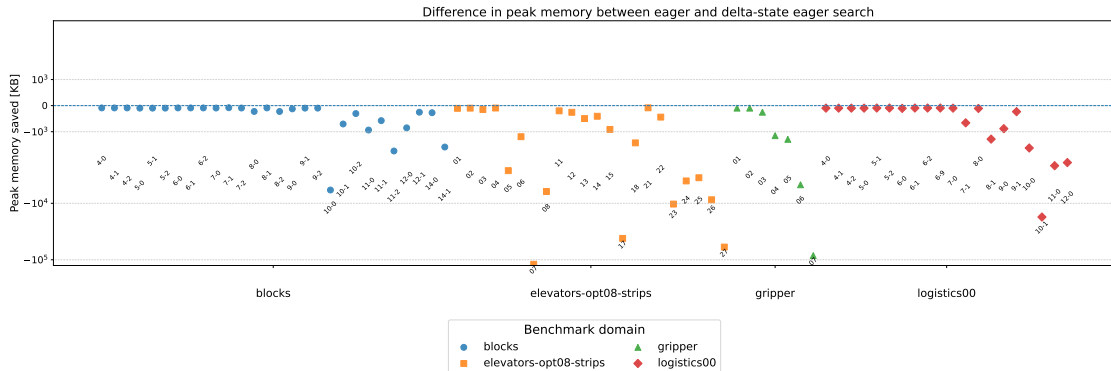
The number of registered semantic states was identical between both implementations.

Benchmark Results: Bytes Saved per State



Reading the plot: positive values mean the delta representation is smaller; negative values mean it is larger.

Benchmark Results: Peak Memory



Main observation: the delta implementation did **not reduce overall peak memory** on any successful benchmark instance.

Runtime and Performance Results

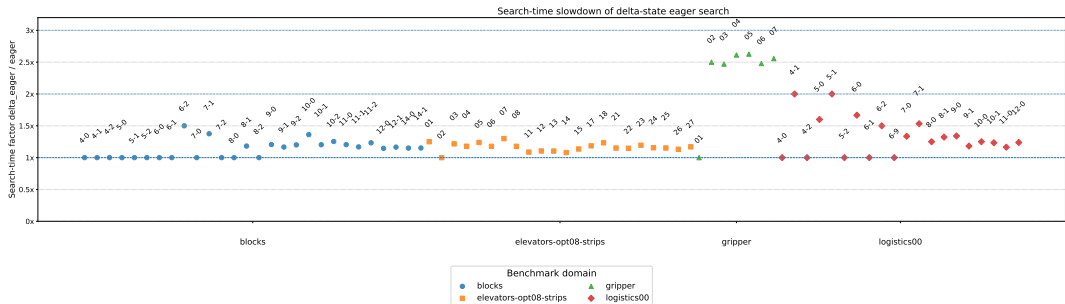
	Eager	Delta
Low-change	3.42 s	252.64 s
High-change	7.28 s	573.49 s

Slowdown

Low-change: $\approx 74\times$
High-change: $\approx 79\times$

Reconstruction is paid repeatedly during heuristic evaluation, goal tests, successor generation and duplicate detection.

Runtime and Performance Results



Conclusion and Outlook

Conclusion

- › Delta-states can reduce **bytes per state** under suitable conditions.
- › The current implementation did not generally reduce **overall peak memory**.
- › Reconstruction introduced a substantial **runtime cost**.

Promising next steps

- › Replace `std::vector` with segmented storage.
- › Separate parent `StateID` from multi-bin effect storage.
- › Introduce complete-state **checkpoints** to limit chain depth.