Hamming Width vs. Novelty Width and Combinations

Bachelor Thesis

Valdrin Sheremetaj

University of Basel

July 24, 2025

Motivation

- Why are some classical planning tasks easy while others are hard?
- Width-based complexity measures provide structural insight into planning difficulty.
- Two major width notions:
  - **Hamming Width (HW)**
  - **Novelty Width (NW)**
- Can we combine them for stronger planners?

Research Questions

- How do Hamming Width and Novelty Width compare theoretically and empirically?
- Can combinations (AND/OR) of HW and NW outperform individual measures?
- How do width measures affect:
  - coverage,
  - runtime,
  - width required,
  - and states generated?

Background: SAS+ Formalism

- SAS+ = **State-Action-State** formalism with multi-valued variables
- Planning task $\Pi = \langle V, I, O, \gamma \rangle$
- Goal: find a sequence of operators from initial state I to a goal state satisfying γ
- State transitions defined by operators with pre/postconditions

What is Width in Planning?

- Width captures structural difficulty of planning problems
- Width-k implies: only k relevant facts needed to consider at each step
- Bounded width $\Rightarrow$ tractable search

Background: Hamming Distance

Hamming Distance:

- Measures how many variables differ between two states s_1 and s_2
- Defined as: $d_H(s_1, s_2) := |\{v \in V \mid s_1(v) \neq s_2(v)\}|$
- Captures how "far" a state is from another state

Background: Novelty

Key Idea: A state is *novel* if it introduces a fact combination not seen before.

- A state has **novelty** k if it makes a new k-sized tuple of facts true, and no smaller such tuple is new
- Captures how "new" a state is in respect to all previously generated states
- The lower the novelty width of a state, the more novel it is.

Hamming Width (HW)

- Bounds deviation during planning: states must differ by $\leq k$ variables
- Prunes states that are too far (Hamming distance $> k$) from the current state
- **Persistent HW**: allows search to continue as long as *some* wrong variable is locally improvable
- Ensures polynomial-time planning when HW $\leq k$

Novelty Width (NW)

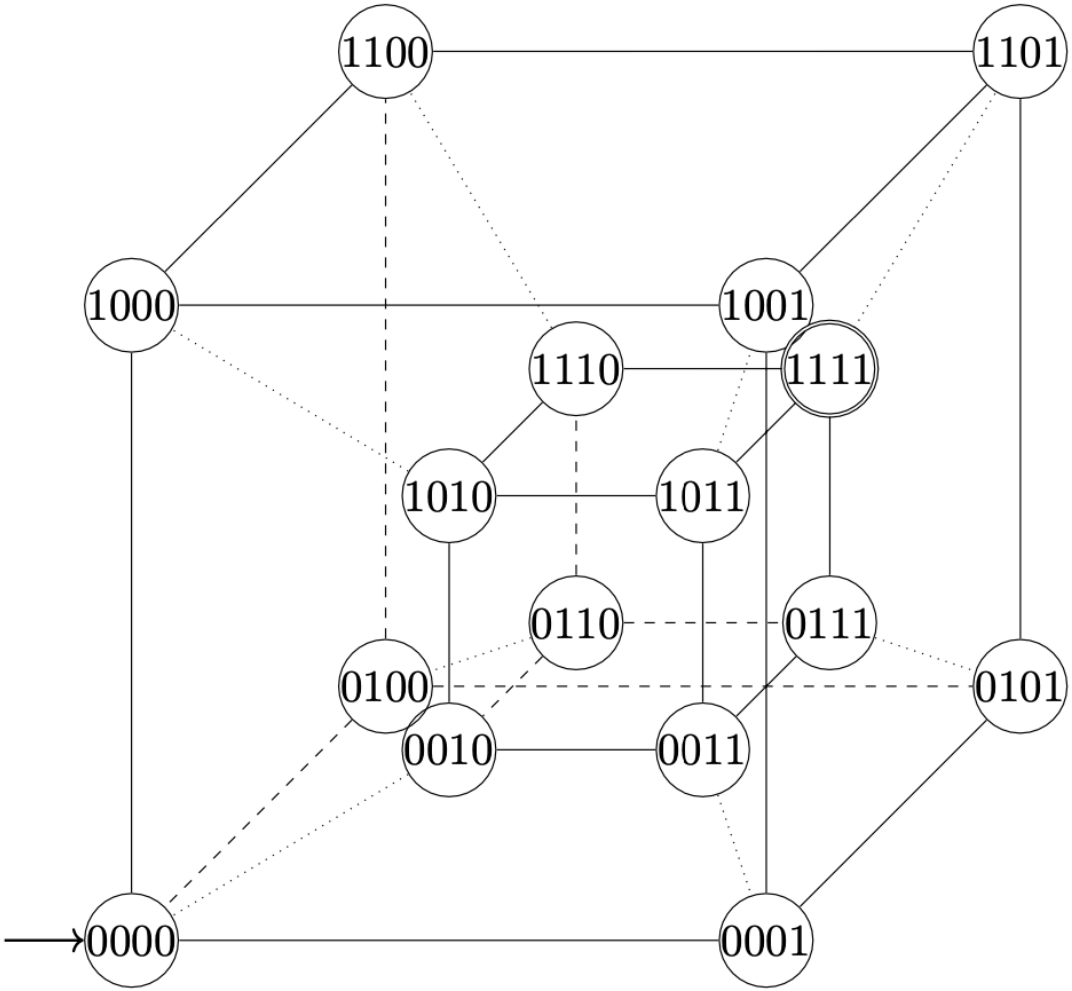
- Based on the novelty of fact tuples (partial assignments) in the state
- Only states with novelty $\leq k$ are expanded
- States that fail to introduce any new tuple of size $\leq k$ are pruned

Running Example: 4-Light Switch Domain

- Four binary switches $x_1, x_2, x_3, x_4 \in \{0, 1\}$
- Initial state: $(0, 0, 0, 0)$
- Goal state: $(1, 1, 1, 1)$
- Each action flips one switch at a time
- Total number of states: $2^4 = 16$

- We'll now visualize the search space these generate.

State Space of 4-Switch Domain



Combining Hamming and Novelty Width

Motivation:

- HW exploits locality
- NW promotes exploration
- Idea: Combine locality and informativeness

Two Variants:

- **AND:** Expand if *both* conditions are met (i.e., Hamming distance $\leq k$ and novelty $\leq k$)
- **OR:** Expand if *either* condition is met

Experimental Setup

- Planner: Fast Downward with Dold's framework
- Benchmarks: blocks, movie, gripper, openstacks, pegsol, visitall
- Hardware: sciCORE Basel, 12 CPU cores, 3900MB/core, 30 min timeout
- Metrics:
  - Coverage
  - Runtime
  - States Generated
  - Width Required

Results: Coverage

Domain	HW	AND	NW	OR
blocks (11)	11	6	7	7
gripper (6)	6	2	2	5
movie (30)	30	30	30	30
openstacks (5)	5	0	0	5
pegsol (2)	2	1	1	2
visitall (4)	4	4	4	4
Total (58)	**58**	**43**	**44**	**53**

- **HW** solves all tasks $\rightarrow$ highest coverage overall
- **NW** covers one more than AND, but still misses some domains $\rightarrow$ both suffer from partial state generation
- **OR** outperforms NW and AND, recovering most of their missed tasks

Results: Runtime (s)

Domain	HW	AND	NW	OR
blocks (6)	0.01	4.11	0.66	0.17
gripper (2)	0.01	1.35	4.58	0.36
movie (30)	0.01	0.04	0.61	0.01
pegsol (1)	0.00	836.43	0.01	0.00
visitall (4)	0.00	0.03	0.01	0.00

- **HW** and **OR** solve all domains in under a second.
- **NW** shows moderate overhead due to cost of generating and evaluating partial states.
- **AND** is significantly slower, see *pegsol*.
- **HW**'s advantage comes from working directly on full states without such overhead.

Results: States Generated

Domain	HW	AND	NW	OR
blocks	343.06	1161.68	1059.17	766.64
gripper	818.37	7375.55	8238.79	4181.88
movie	8050.97	39109.18	39109.18	4157.63
pegsol	21.00	37.00	25.00	14.00
visitall	10.99	105.18	36.28	23.51

- **HW**: few full states
- **NW** and **AND**: many partial states
- **OR**: less overhead than NW/AND
- For HW, only full states are generated and counted, while NW and AND count many partial states

Results: Avg. Width per Domain (k)

Domain	HW	AND	NW	OR
blocks (6)	5.50	7.67	3.33	2.50
gripper (2)	3.00	7.00	6.00	3.00
movie (30)	2.00	6.00	6.00	1.00
pegsol (1)	3.00	6.00	2.00	1.00
visitall (4)	1.00	4.00	1.75	1.00

- **OR** and **HW** use the lowest width across all domains
- **NW** and **AND** require much higher k $\rightarrow$ especially in `movie` and `gripper`

Conclusion

- HW is robust, runtime-efficient and requires low width
- NW is more exploratory but suffers from partial state generation and has a higher width: novelty = stricter criterion
- AND is too restrictive $\rightarrow$ low coverage and performance
- OR balances both strategies well and performs really good