University of Basel

Heuristics for TopSpin

Marco Dreher <marco.dreher@stud.unibas.ch>

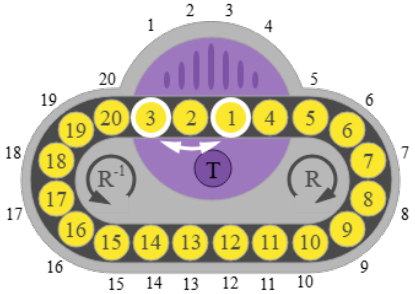
Natural Science Faculty, University of Basel

29.07.2025

Background

TopSpin: The Puzzle

- Circular puzzle with N numbered tokens
- A turnstile of size k reverses k adjacent tokens
- Goal: Sort tokens in ascending order



TopSpin: State Space

State Space Definition

A (N, k)-TopSpin puzzle is defined as a state space:

$$\mathcal{S} = \langle S, A, \text{cost}, T, s_I, S_G \rangle$$

- S: All permutations of $\{1, \ldots, N\}$
- A: Reverse k elements starting at index i, with wraparound
- $\text{cost}(a) = 1$ for all actions a
- $T(s, a)$: Applies reversal action a to state s
- s_I: Any valid permutation (initial state)
- S_G: All rotations of the identity permutation

Heuristics

- Heuristics estimate the cost from a state to the goal.
- Help guiding the search in large state spaces like TopSpin.
- Crucial for algorithms like A*

Definition

A heuristic is a function:

$$h : S \to \mathbb{R}_0^+ \cup \{\infty\}$$

Properties of Heuristics

Perfect Heuristic h^*

Returns the exact cost to reach the goal from any state.

- **Goal-aware:** $h(s) = 0$ for all goal states
- **Safe:** $h^*(s) = \infty$ for all $s \in S$ with $h(s) = \infty$
- **Consistent:** $h(s) \leq cost(a) + h(s')$ for all transitions $s \xrightarrow{a} s'$
- **Admissible:** $h(s) \leq h^*(s)$ (never overestimates true cost)

Search Algorithms

- We solve TopSpin using heuristic search, guided by the evaluation function:

$$f(n) = g(n) + h(n)$$

- $g(n)$: cost from the start to state n
- $h(n)$: heuristic estimate from n to a goal state

A* vs. IDA*

- **A*** uses a priority queue and expands nodes with the lowest f value. It is fast but memory-intensive.
- **IDA*** avoids high memory usage by performing depth-first searches with increasing f-limits. It uses less memory but may revisit states.

Heuristics for TopSpin

Gap Heuristic

- Originally proposed by Helmert for the pancake puzzle
- **Idea**: Count non-consecutive adjacent pairs ("gaps") in the permutation
- Adapted to TopSpin to account for:
  - Arbitrary-position fixed-size reversals (can remove 2 gaps)
  - Circular structure (first-last comparison)
  - Special case: $\{1, N\}$ is not a gap

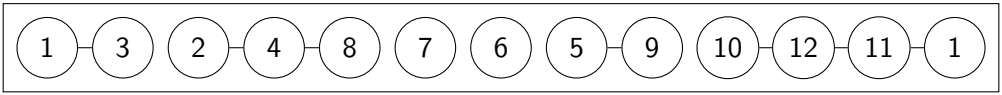
Formal Definition

Let $s = \langle s_1, ..., s_N \rangle$ and $s' = \langle s_1, ..., s_N, s_1 \rangle$, then:

$$h_{\text{gap}}(s) = \left\lceil \frac{\left| \left\{ i \mid |s'_i - s'_{i+1}| > 1 \wedge \{s'_i, s'_{i+1}\} \neq \{1, N\} \right\} \right|}{2} \right\rceil$$

Gap Heuristic: Example and Properties

Example:



- Heuristic value:

$$h_{\text{gap}}(s) = \left\lceil \frac{7}{2} \right\rceil = 4$$

Properties: The gap heuristic fulfills all properties:

- **Consistent:** Each reversal can increase or decrease the heuristic value by at most 1.
- **Goal-aware:** Sorted states have 0 gaps.
- **Admissible:** Implied by goal-awareness and consistency.
- **Safe:** Implied by admissibility.

Distance-Based Heuristic

Idea: Adapt Manhattan distance to TopSpin

- TopSpin is one-dimensional, while Manhattan is for two-dimensional problems
- Let $r \in \{0, \ldots, N-1\}$ represent a rotation. Define the distance under rotation r as:

$$dist(s, r) = \sum_{i=0}^{N-1} \min\left((i - (s_{(i+r) \bmod N} - 1)) \bmod N,\ ((s_{(i+r) \bmod N} - 1) - i) \bmod N\right)$$

- Actions have a greater effect in TopSpin. Consider the maximum effect of any reversal of size k:

$$\Delta max = \sum_{j=0}^{k-1} |2j - (k-1)|$$

Distance-Based Heuristic

Definition of h_{distance}

Let $s = \langle s_1, \ldots, s_N \rangle$ be a TopSpin state, Δmax be the maximum effect any reversal of size k can have and $dist(s, r)$ be the sum of distances of a single rotation. Then:

$$h_{\text{distance}}(s) = \left\lceil \frac{\min\limits_{0 \leq r < N} dist(s, r)}{\Delta max} \right\rceil$$

- **Goal-aware:** At any goal state, the best rotation has distance 0.
- **Consistent:** A single reversal changes the state by at most Δmax.
- **Admissible:** Follows directly from goal-awareness and consistency.
- **Safe:** Since the heuristic is admissible, it is also safe.

Domain Abstraction

Definition (Domain Abstraction)

Let $s = \langle s_1, \ldots, s_N \rangle$ be a TopSpin state. A **domain abstraction** is a tuple $\alpha = \langle \alpha_1, \ldots, \alpha_N \rangle$, where each α_i maps token values to equivalence classes:

$$\alpha_i : \text{values} \to \text{abstract value} \quad \Rightarrow \quad \alpha(s) = \alpha_1(s_1) \times \cdots \times \alpha_N(s_N)$$

Example: Map values ≤ 4 to themselves, others to 5:

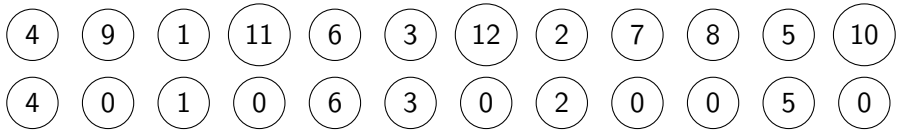
$$s = \langle 8, 5, 6, 4, 2, 1, 7, 3 \rangle \quad \Rightarrow \quad \alpha(s) = \langle 5, 5, 5, 4, 2, 1, 5, 3 \rangle$$

Approach 1: Max Over Multiple Abstractions

Multi-abstraction types:

- **Group**: clusters values (e.g., tokens 1–6 vs. 7–12)
- **Distance**: based on value parity/distance (e.g., even vs. odd)

Example (6-6 Group Abstraction):



Heuristic:

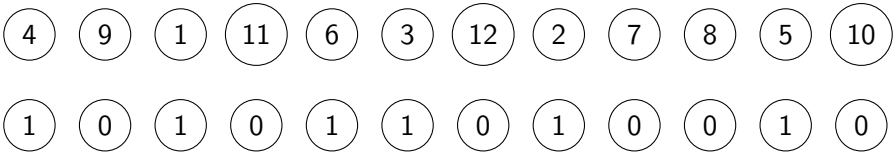
$$h_{abstract}(s, A) = \max_{a \in A} \text{bfs\_solutionLength}(a)$$

Properties: Admissible, Consistent, Goal-aware, Safe.

Approach 2: Single-State Abstraction

Single abstraction: Represent entire state by a single abstract tuple with M groups.

Example (2-GroupS):



Heuristic:

$$h_{abstractS}(s, a_s) = \text{bfs\_solutionLength}(a_s)$$

Properties: Goal-aware, Admissible, Consistent, Safe.

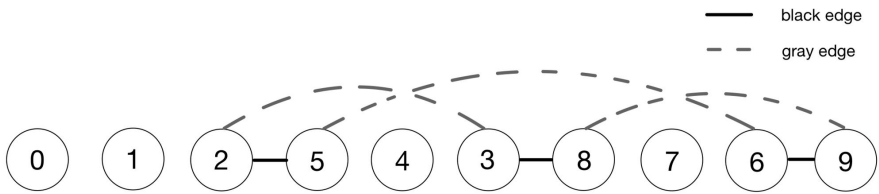
Breakpoint Graph Heuristic

Motivation

- Inspired by sorting signed permutations
- TopSpin is also a permutation problem — idea still applies

Concept

- Extend gap heuristic using a bicolored graph
- Nodes = tokens, edges = relations



Breakpoint Graph Heuristic

Heuristic Definition

Let s be a TopSpin state and s_r the rotation where token 1 is in the first position. Then:

$$h_{\text{breakpoint}}(s) = b(s_r) - c_{\text{approx}}(s_r)$$

- $b(s_r)$: number of breakpoints (black edges)
- $c_{\text{approx}}(s_r)$: approximated maximum number of alternating black/gray cycles

Properties:

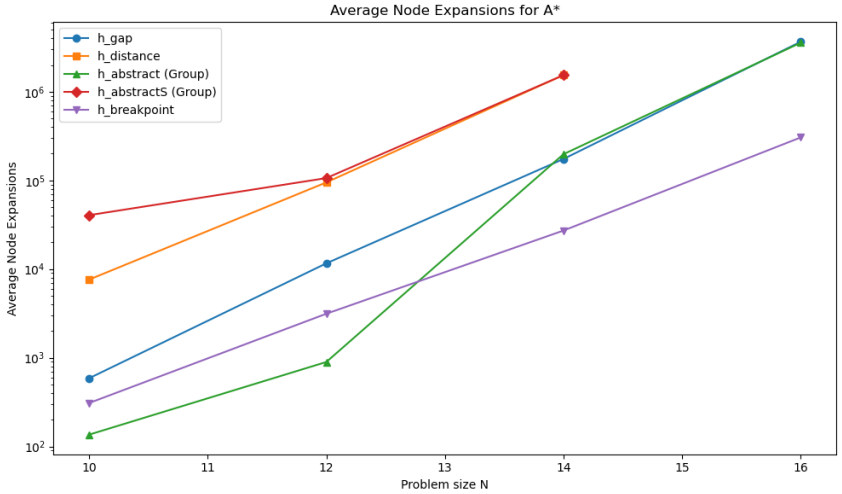
- **Goal-aware:** For any goal state, there are no breakpoints.
- **Consistent:** Not guaranteed, because of the approximation.
- **Admissible:** Not guaranteed, because of the approximation.
- **Safe:** Worst case the heuristic is number of breakpoints.

Experiments

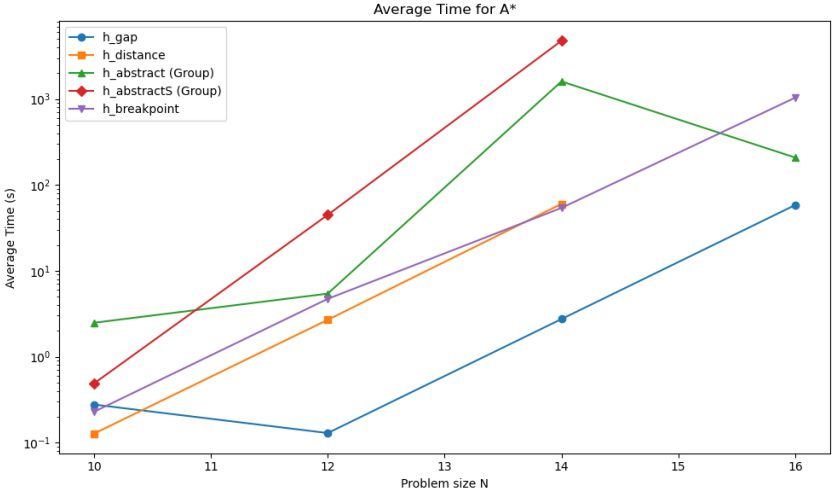
Experimental Setup

- Implemented in C++20
- Evaluated with A* ($N = 10, 12, 14, 16$) and IDA* ($N = 18, 20$)
  - 101 instances for $N = 10, 12, 14$
  - 51 instances for $N = 16$
  - 21 instances for $N = 18, 20$
- Metrics: nodes expanded, runtime, initial h, success rate

Results


Average Node Expansions for A*
h_gap
h_distance
h_abstract (Group)
h_abstractS (Group)
h_breakpoint
Average Node Expansions
10^6
10^5
10^4
10^3
10^2
10
11
12
13
14
15
16
Problem size N

Results


Average Time for A*
h_gap
h_distance
h_abstract (Group)
h_abstractS (Group)
h_breakpoint
Average Time (s)
10^3
10^2
10^1
10^0
10^{-1}
10
11
12
13
14
15
16
Problem size N

Results with IDA*

- $h_{\text{breakpoint}}$ works best for big instances

N	k	**Heuristic**	**Avg Time (s)**	**Med Time (s)**	**Avg Sol**	**Med Sol**	**Avg Nodes (mil)**	**Med Nodes (mil)**	**Success Rate**
18	4	$h_{\text{breakpoint}}$	29399.55	24651.75	16.4	16	103.69	84.68	85.71%
18	4	h_{gap}	26235.48	15083.93	15.5	16	74935.59	40916.77	57.14%
18	4	h_{distance}	15908.27	1111.44	13.4	15	4328.28	282.21	23.81%
20	4	$h_{\text{breakpoint}}$	20272.15	14398.58	16.5	17	53.46	37.84	19%
20	4	h_{gap}	27.869	27.869	14.0	14	66.97	66.97	9.5%

Conclusion

- Breakpoint graph heuristic is promising for bigger instances
- Gap heuristic gives a good balance between speed and nodes expanded
- Domain abstractions useful on small instances $\rightarrow$ PDBs better for bigger instances
- Future work: improvement of the maximum alternating cycle decomposition approximation

Questions?

marco.dreher@stud.unibas.ch