A General Theory of Additive State Space Abstractions

Journal of Artificial Intelligence Research 32 (2008)

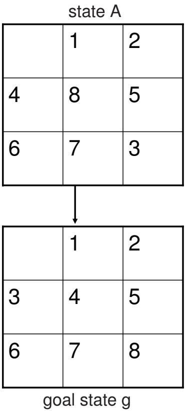
F. Yang, J. Culberson, R. Holte, U. Zahavi, A. Felner

David Adametz

Overview

- Example: 8-puzzle
  - What is an abstraction?
  - Heuristic of an abstraction
- Formal definition
- Example: (N, K)-TopSpin
  - Abstractions
  - Cost splitting
  - Benchmarks
- Conclusion

Example: 8-puzzle



state A

	1	2
4	8	5
6	7	3

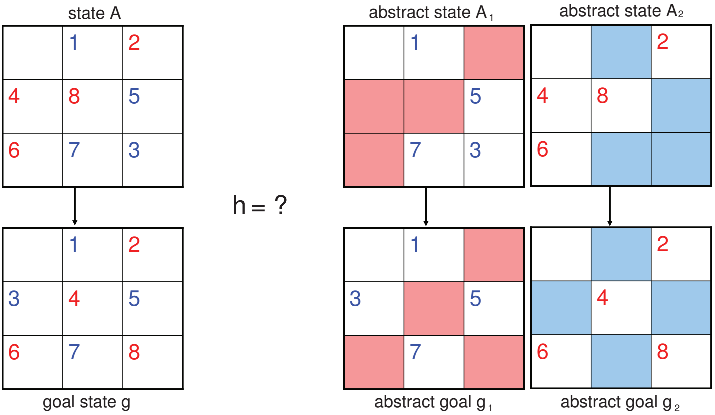
	1	2
3	4	5
6	7	8

goal state g

h = ?

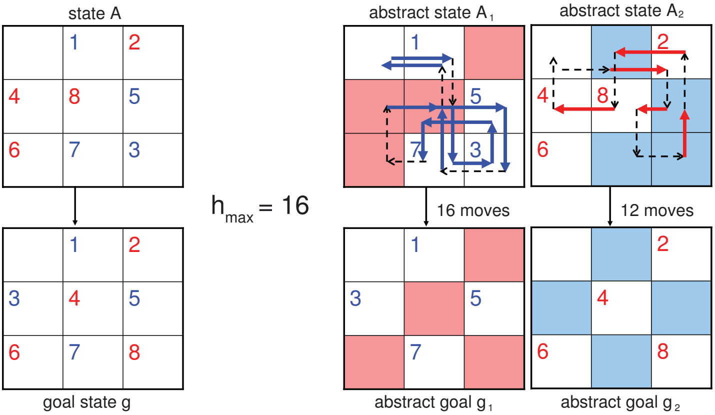
- Elegant way to define heuristic.

Example: 8-puzzle – What is an abstraction?



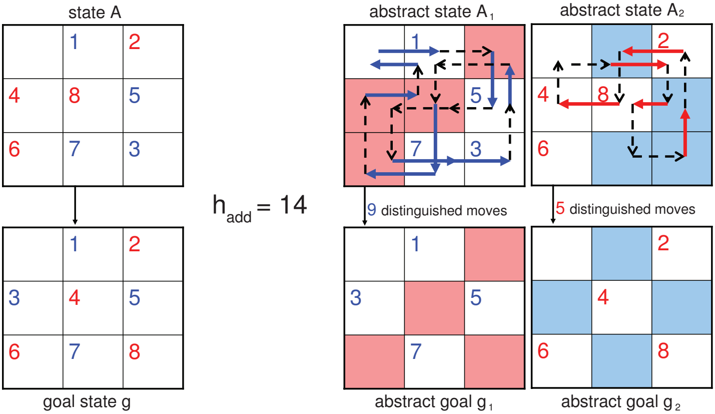
- State variables are divided into **mutually exclusive sets**.
- Each abstraction **defines a heuristic**.
- **No hierarchy** of abstractions.

Example: 8-puzzle – Maximum heuristic $h_{\max}$



- Every move has cost 1.

Example: 8-puzzle – Additive heuristic h_{add}



state A

	1	2
4	8	5
6	7	3

goal state g

	1	2
3	4	5
6	7	8

$h_{add} = 14$

abstract state A_1

	1	
		5
	7	3

9 distinguished moves

abstract goal g_1

	1	
3		5
	7	

abstract state A_2

		2
4	8	
6		

5 distinguished moves

abstract goal g_2

		2
	4	
6		8

- moving a **distinguished** tile ⇒ cost 1
- moving a **don't-care** tile ⇒ cost 0
- Why? Don't count a move twice.

Example: 8-puzzle – More formally

Original space:

- $C(u, v)$ is the cost of moving from state u to state v.

Abstract space i, $i = 1 \dots k$:

- Every abstraction has state $u_i := \psi_i(u)$ and $v_i := \psi_i(v)$.
- $C_i(u_i, v_i)$ is called **primal cost**

$$C_i(u_i, v_i) = \begin{cases} 1, & \text{moving a distinguished tile} \\ 0, & \text{moving a don't-care tile} \end{cases}$$

- $R_i(u_i, v_i)$ is called **residual cost**

$$R_i(u_i, v_i) = \begin{cases} 0, & \text{moving a distinguished tile} \\ 1, & \text{moving a don't-care tile} \end{cases}$$

- $C(u, v) = C_i(u_i, v_i) + R_i(u_i, v_i)$

Formal definition I

How can we find the **joint heuristic** of all abstractions?

- Compute the **maximum**

$$h_i(t,g) = \min_{\pi_i} \left(C_i(t_i,g_i) + R_i(t_i,g_i) \right)$$

$$h_{\max}(t,g) = \max_i \left(h_i(t,g) \right)$$

- Compute the **sum**, but **avoid overlaps**, i.e. $R_i(t_i,g_i)$!

$$C_i^*(t_i,g_i) = \min_{\pi_i} \left(C_i(t_i,g_i) \right)$$

$$h_{\text{add}}(t,g) = \sum_{i=1}^{k} C_i^*(t_i,g_i)$$

Formal definition II

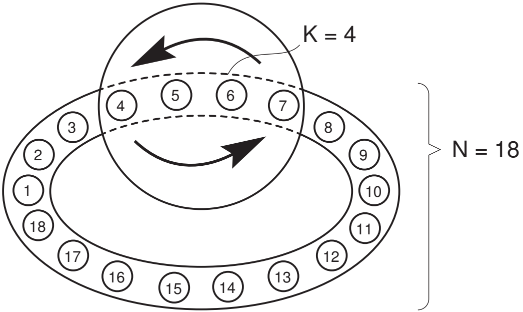
$h_i(t, g)$ is **admissable** and **consistent** if...

- $\forall (u, v) \in \Pi, \quad (u_i, v_i) \in \Pi_i$
- $\forall \pi \in \Pi, \quad C(\pi) \geq C_i(\pi_i) + R_i(\pi_i)$

$h_{\text{add}}(t, g)$ is **admissable** and **consistent** if...

- $\forall \pi \in \Pi, \quad C(\pi) \geq \sum_{i=1}^{k} C_i(\pi_i)$

Example: (N, K)-TopSpin



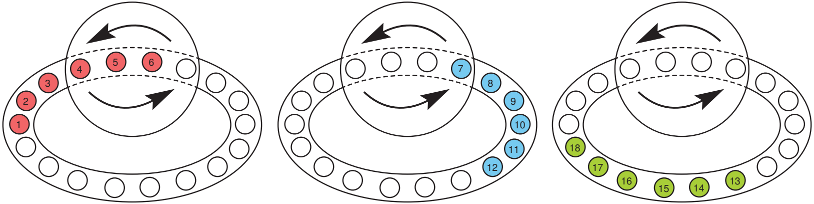
Goal:

- all N tiles on the track are ordered

Possible actions:

- move all N tiles around the track
- reverse all K tiles in the circle

Example: (N, K)-TopSpin – Abstractions and PDB



Usually:

- 1 pattern database (PDB) **per abstraction**

Here:

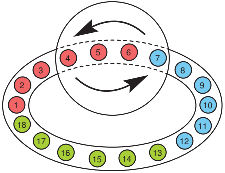
- use the **symmetry** of the problem: re-label tiles from $1 \ldots 6$
- if all abstractions have the same tile size, **1 PDB** is sufficient

Cost splitting

	1	2
4	8	5
6	7	3

Previously: 8-puzzle

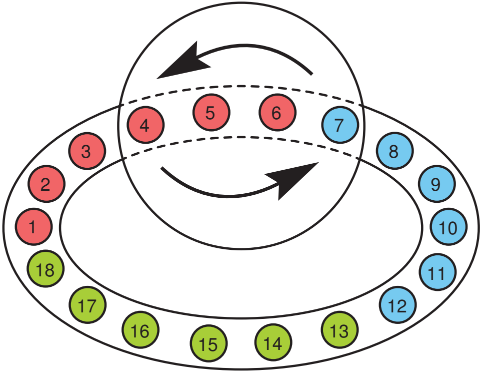
- The cost of moving tile "1" is **fully attributed** to abstraction 1, in which the tile is distinguished.
- Abstraction 2 **receives cost** $C_2 = 0$ (*don't-care move*).



Now: $(18, 4)$-TopSpin

- A **single action** affects distinguished states from **multiple abstractions**.
- **Which abstraction** receives the cost?

Example: (N, K)-TopSpin – Cost splitting



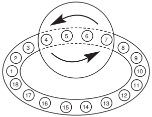
Cost in original space:

$$C(\pi) = 1$$

Primal cost:

$$C_1(\pi_1) = \frac{3}{4}$$
$$C_2(\pi_2) = \frac{1}{4}$$
$$C_3(\pi_3) = 0$$

Positive Results: $(N, 4)$-TopSpin



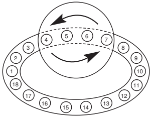
N	Abs	Average Solution Length	h_{max}		h_{add} based on cost-splitting		Nodes Ratio
			Nodes	Time	Nodes	Time	
12	6-6	9.138	14,821	0.05	53,460	0.16	3.60
12	4-4-4	9.138	269,974	1.10	346,446	1.33	1.28
12	3-3-3-3	9.138	1,762,262	8.16	1,388,183	6.44	**0.78**
16	8-8	14.040	1,361,042	3.42	2,137,740	4.74	1.57
16	4-4-4-4	14.040	4,494,414,929	13,575.00	251,946,069	851.00	**0.056**
18	9-9	17.000	38,646,344	165.42	21,285,298	91.76	**0.55**
18	6-6-6	17.000	18,438,031,512	108,155.00	879,249,695	4,713.00	**0.04**

[1]

- more abstractions ⇒ h_{add} is better than h_{max}

[1] AMD Athlon 2.4 GHz, 1 GB memory

Negative Results: $(12, K)$-TopSpin



K	Abs	h_{max}		h_{add} based on cost-splitting		Nodes Ratio
		Nodes	Time	Nodes	Time	
3	3-3-3-3	486,515	2.206	207,479	0.952	**0.42**
4	3-3-3-3	1,762,262	8.164	1,388,183	6.437	**0.78**
5	3-3-3-3	8,978	0.043	20,096	0.095	2.23
6	3-3-3-3	193,335,181	901.000	2,459,204,715	11,457.000	12.72

- K small $\Rightarrow$ h_{add} is better than h_{max}
- $N = 12$, $K = 5$ is ideal
- $N = 12$, $K = 6$ is difficult

Conclusion

- Formal conditions for $h_{\max}$ and h_{add} to be **consistent** and **admissable**.
- Every abstraction defines 1 heuristic (1 **PDB**).
- Every action in abstract space has two costs: **primal** (C_i) and **residual** (R_i).
- **Cost splitting** often necessary for h_{add} (e.g. in TopSpin)
- h_{add} is not always superior to $h_{\max}$.
- h_{add} can **substantially reduce nodes** and search time, but **may also fail**.
- Open question: **How to choose abstractions?**
  - Which states?
  - How many states?
  - How many abstractions?

Thanks for your attention!

Questions?