Abstract Zobrist Hashing: An Efficient Work Distribution Method for Parallel Best-First Search

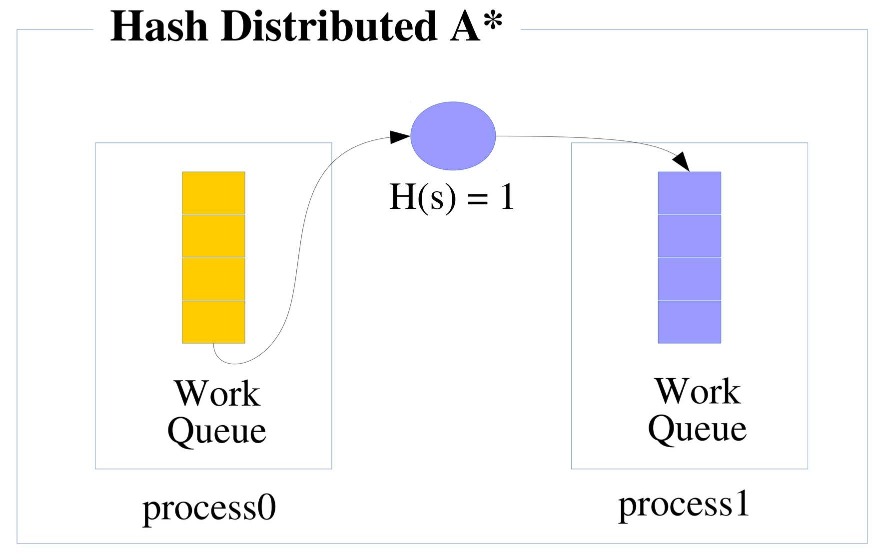
Yuu Jinnai and Alex Fukunaga (The University of Tokyo)

1. Hash Distributed A* (HDA*)

(Kishimoto et al. 2013)

Hash Distributed A* (HDA*) is a parallel best-first graph search (A*) which distrbutes nodes according to a hash function which assigns each state to a unique process.

As HDA* relies on the hash function for load balancing, **the choice of hash function is crucial to its performance!**



Summary

<u>We propose</u>: **Abstract Zobrist hashing**, a **hybrid** of two strategies, which incorporates their strengths and alleviating their limitations.

<u>Main contribution</u>: We showed that Abstract Zobrist hashing outperforms previous methods in sliding-tile puzzle, MSA, and domain-independent classic planning.

	load balance	communication overhead
Zobrist hashing (Zobrist 1970; Kishimoto et al 2013)	○	×
Abstraction (Zhou and Hansen 2007; Burns et al 2010)	×	○
Abstract Zobrist hashing (New!)	○	○

2. Previous Work Distribution Methods

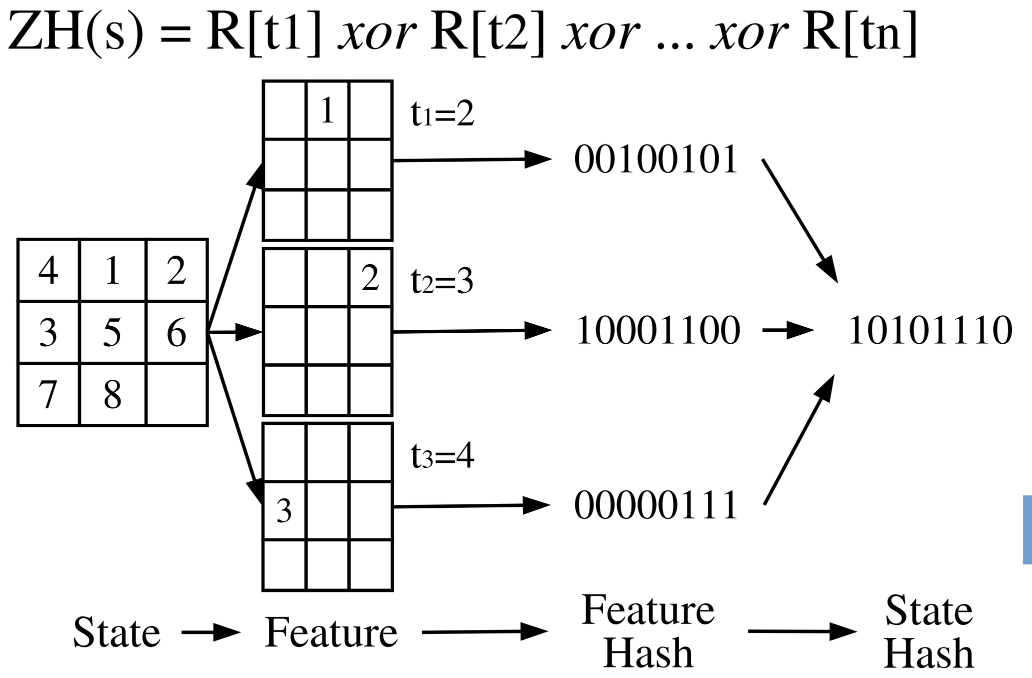
2.1. Zobrist hashing (ZHDA*)

(Zobrist 1970; Kishimoto et al. 2013)

<u>Goal</u>: Distribute nodes uniformly among process

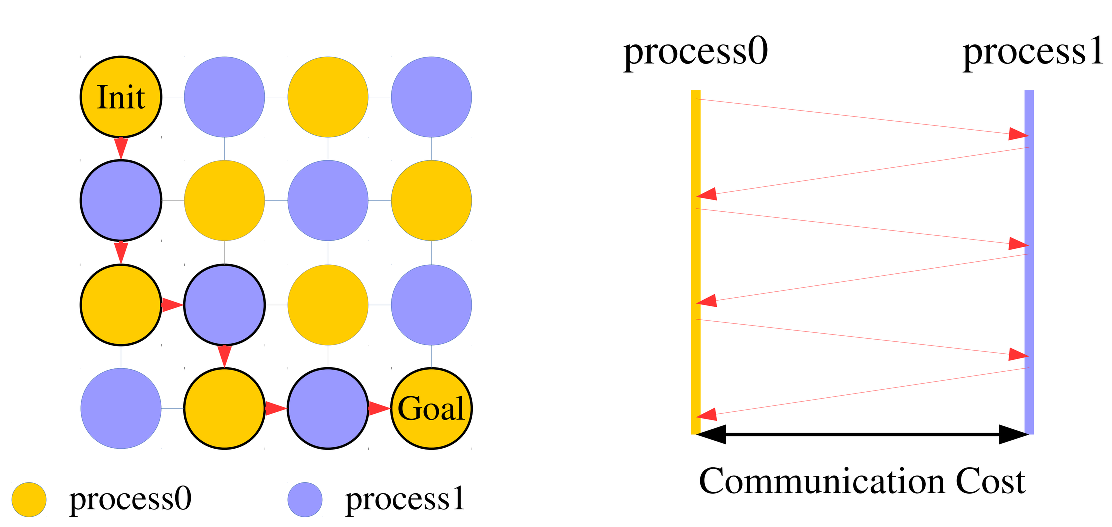
<u>Method</u>: *xor* the hash value of each feature

ZH(s) = R[t1] *xor* R[t2] *xor* ... *xor* R[tn]



<u>Strength</u>: good load balance

<u>Limitation</u>: high communication overhead



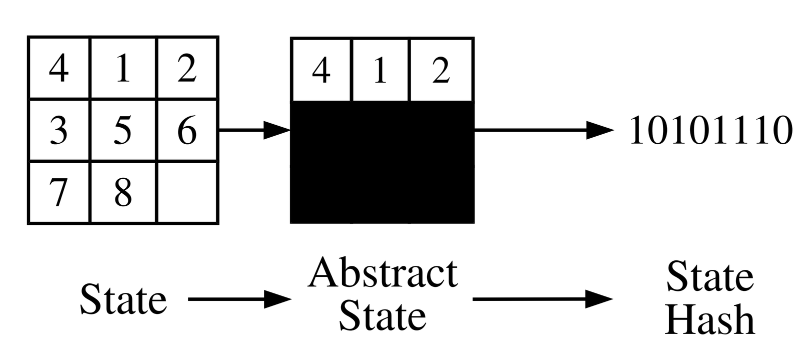
2.2. Abstraction (AHDA*)

(Zhou and Hansen 2007; Burns et al. 2010)

<u>Goal</u>: Assign neighbor nodes to the same process

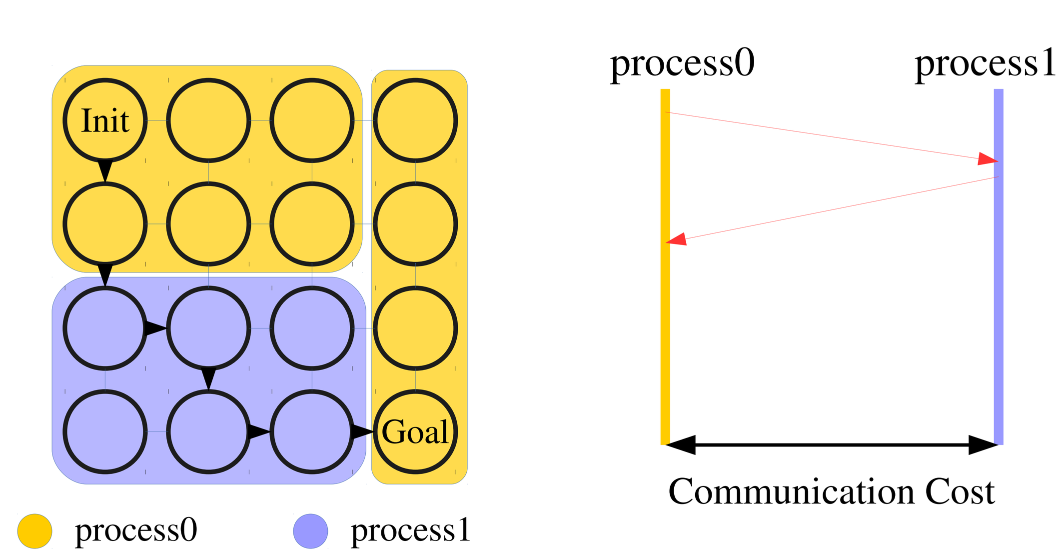
<u>Method</u>: Project nodes in the state space into abstract states, and abstract states are assigned to processors using a modulus operator.

AH(s) = R[A(s)]



<u>Strength</u>: low communication overhead

<u>Limitation</u>: worse load balance



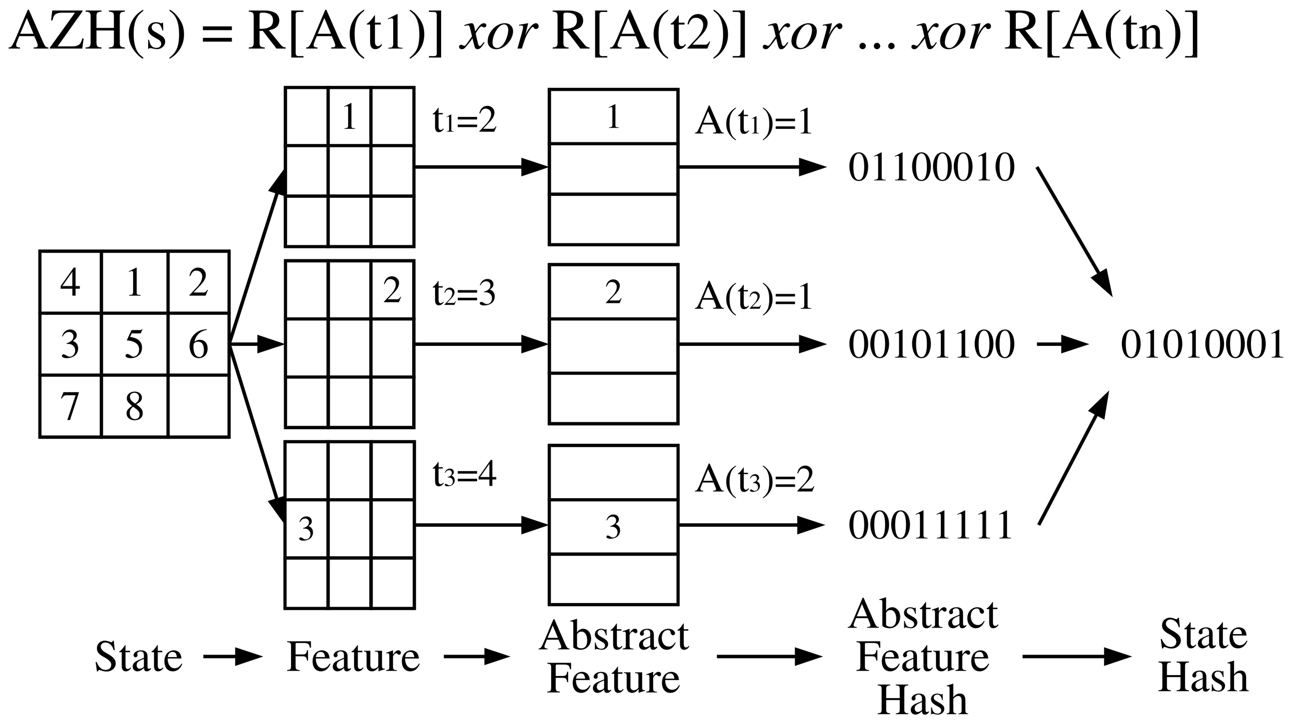
3. Abstract Zobrist hashing (AZHDA*)

new!

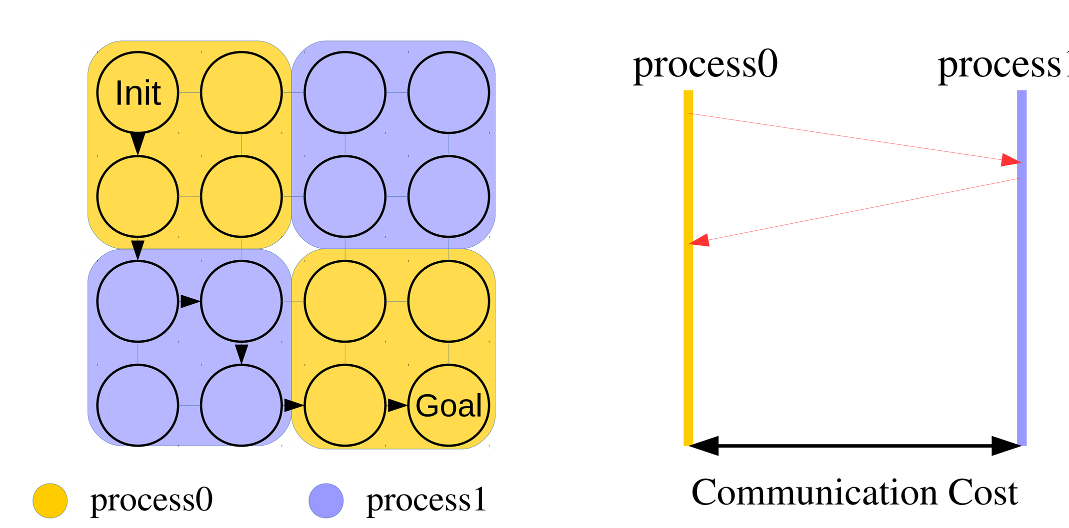
<u>Goal</u>: Distributes nodes uniformly while assigning neighbor nodes to the same process

<u>Method</u>: Project features into abstract features and *xor* the hash value of each abstract feature

AZH(s) = R[A(t1)] *xor* R[A(t2)] *xor* ... *xor* R[A(tn)]



AZH Simultaneously address good load balance *and* communication overhead!



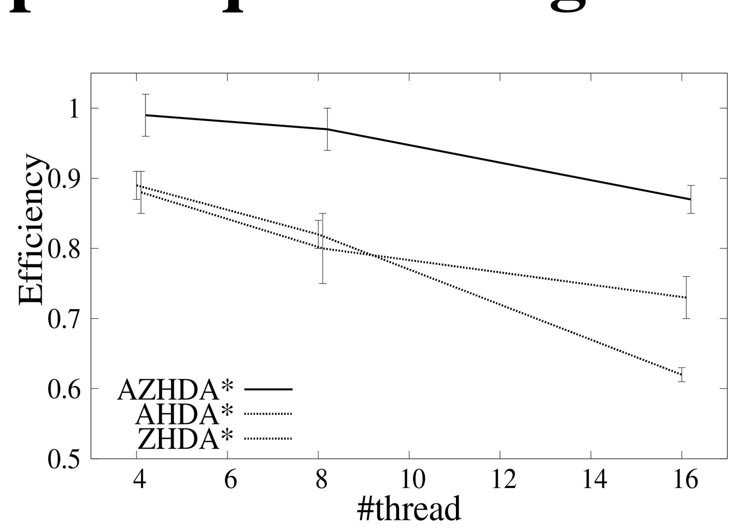
4. Results on Sliding-tile and Multiple Sequence Alignment

AZHDA* outperformed previous methods on 15-puzzle, 24-puzzle, and MSA, successfully mitigating communication overhead and achieving good load balance, while ZHDA* and AHDA* only succeeded in achieving either low CO or good load balancing.

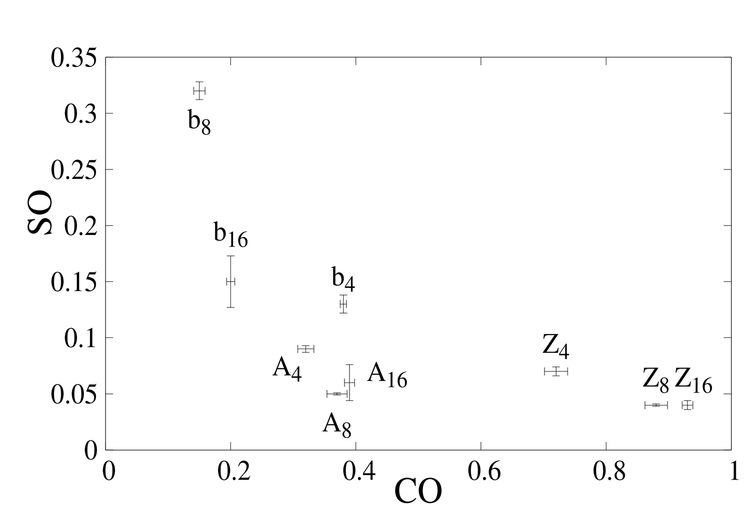
Communication overhead (CO) := (#nodes sent to other processes) / (#nodes generated)

Search overhead (SO) := (#nodes expanded in parallel) / (#nodes expanded in sequential A*)

Search overhead is caused by poor load balance.

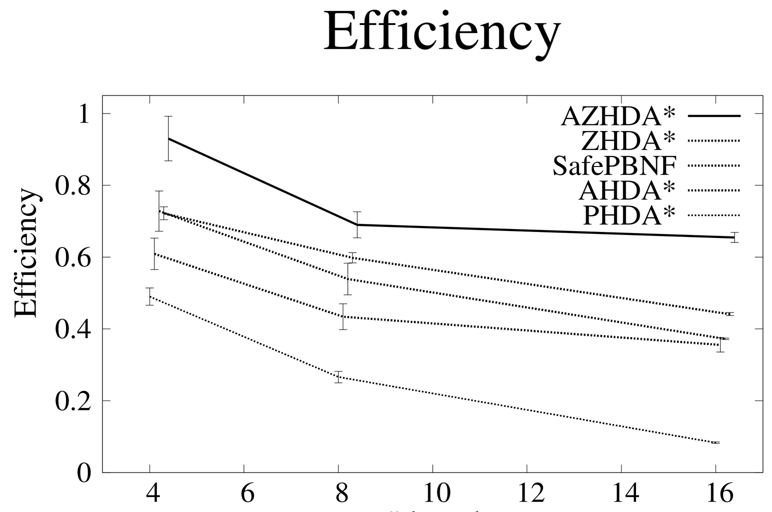


24 Puzzle Walltime Efficiency

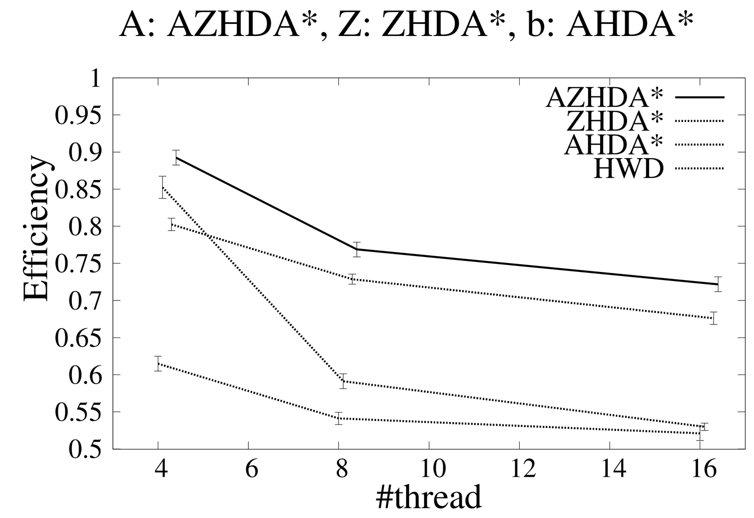


24 Puzzle CO vs SO

A: AZHDA*, Z: ZHDA*, b: AHDA*



15 Puzzle Walltime Efficiency



MSA Walltime Efficiency

5. Results on Domain-Independent Planning

AZHDA* achieved modest improvements over previous methods with **automatically generated abstract features**.

#thread=8	A*		AZHDA*			ZHDA*			AHDA*		
	Time (sec)	Expd	Eff	CO	SO	Eff	CO	SO	Eff	CO	SO
Airport9	156.59	4902509	0.799	0.480	-0.005	0.808	0.560	-0.007	0.720	0.240	0.024
Barman1-1	30.32	4719078	0.641	0.770	0.133	0.707	0.960	-0.002	0.613	0.640	0.144
Blocks10-2	234.27	39226226	0.988	0.540	0.009	0.932	0.890	0.011	1.013	0.400	0.041
Elevators11-26	142.66	5329127	0.654	0.380	0.007	0.601	0.850	-0.001	0.648	0.070	0.114
Floortile1-1	254.74	30523550	0.768	0.850	-0.032	0.806	0.900	-0.046	0.905	0.300	-0.018
Freecell5	141.74	9142007	0.774	0.650	0.000	0.751	0.890	-0.002	0.375	0.080	0.100
Gripper7	77.23	10101217	0.766	0.720	0.001	0.743	0.880	0.004	0.849	0.340	0.003
Logistics98-32	25.44	1837543	0.758	0.460	-0.058	0.628	0.880	0.000	0.808	0.310	-0.009
PipesNoTk14	231.50	34862961	0.835	0.820	-0.002	0.883	0.860	-0.000	1.016	0.400	-0.102
Sokoban11-13	129.66	10048931	0.893	0.630	-0.001	0.874	0.820	-0.001	0.288	0.150	0.172
Transport08-14	85.89	9830158	0.831	0.310	-0.041	0.694	0.870	0.001	0.695	0.770	0.024
Visitall11-7Half	10.13	1952096	0.873	0.880	-0.044	0.867	0.890	-0.041	0.804	0.880	-0.039
Zenotravel8	26.52	1330916	0.784	0.570	-0.045	0.661	0.890	-0.003	0.749	0.700	-0.006
Average	118.97	12600486	**0.797**	0.620	-0.006	0.766	0.857	-0.007	0.729	0.406	0.034
Total	1646.74	163806319	Time (sec)		**282.29**	Time (sec)		298.92	Time (sec)		341.73