Accelerated Vector Pruning for Optimal POMDP Solvers Supplementary Material

Erwin Walraven and **Matthijs T. J. Spaan**
Delft University of Technology
Mekelweg 4, 2628 CD
Delft, The Netherlands

This supplement contains the proof of Lemma 1, details on the experimental setup, additional experimental results, an illustration of our algorithm and information about the source code that is publicly available.

Proof of Lemma 1

This section contains the proof of Lemma 1, which was omitted from the paper.

Lemma 1. *Each corner of the feasible region of the standard LP (U, w) corresponds to a corner belief of value function U.*

Proof. The concave surface defining the LP solutions can be defined as $w \cdot b - \max_{u \in U} u \cdot b$, where b is a belief. The expression $\max_{u \in U} u \cdot b$ corresponds to the convex surface of value function U. By definition, the slope of the value function U changes at the corner beliefs. Since w is a single vector, it follows that the concave surface $w \cdot b - \max_{u \in U} u \cdot b$ also changes slope at the corner beliefs. □

Details about experimental setup

Table 1 shows the parameters of the POMDP domains involved in the experiments. Most of the domain descriptions can be obtained from `www.pomdp.org`. The last column shows the number of LPs that was used in the first experiment in the paper. Some domains could be solved with fewer than 30000 LPs. For RockSmp4x4 we only consider the first 1000 LPs due to memory limits. The domain 4x3CO is not included in the experiments since it represents a POMDP with full observability.

In the vector pruning experiment we use a value function of the 4x5x2 domain to generate 1000 separate value functions V_q for which $|V_q| = q$. The procedure to generate these value functions works as follows. During incremental pruning we store a value function V immediately after the final pruning step in a dynamic programming stage. A value function V_q contains the first q vectors from V. Since $\texttt{prune}(V) = V$ it holds that each value function V_q does not contain dominated vectors. This property is important because it means that a pruning algorithm needs to solve q LPs to prune V_q.

Name	$\|S\|$	$\|A\|$	$\|O\|$	#LPs experiment 1
1D	4	2	2	50
4x3	11	4	6	30000
4x4	16	4	2	2331
4x5x2	39	4	4	30000
AircraftID	12	6	5	30000
Cheese	11	4	7	1514
Hallway	60	5	21	30000
Hallway2	92	5	17	30000
Network	7	4	2	30000
Partpaint	4	4	2	7534
RockS4x4	257	9	2	1000
Shuttle	8	3	5	30000
Tiger-grid	36	5	17	30000

Table 1: Parameters of the POMDP domains

Experimental results for other LP solvers

Figures 1 and 3 contain the results of the vector pruning experiments with Gurobi and lpsolve. The differences in speedup and running time can be explained by observing that different APIs are used to interact with the LP solvers, and internally the LP solvers may use different algorithms and heuristics. However, since our method outperforms other methods in all cases, we can conclude that our method works well, regardless of the LP solver that is used.

Tables 2 and 3 contain the results of the LP experiment for Gurobi and lpsolve. The conclusions that can be derived are identical to the conclusions discussed in the main paper.

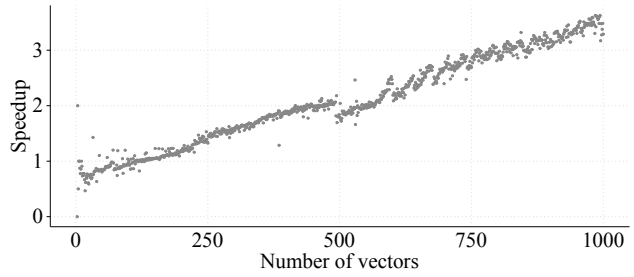
Figures 2 and 4 contain the results of the incremental pruning experiments for Gurobi and lpsolve. The figures show that our variant of incremental pruning consistently outperforms other methods involved in the comparison.

Domain	Std (s)	Dec (s)	Speedup	Constr. (%)
4x5x2	77.79	18.53	4.20	11.9 ± 15.9
AircraftID	51.12	12.82	3.99	8.7 ± 14.6
Hallway2	73.14	21.94	3.33	18.0 ± 27.0
Tiger-grid	67.66	20.72	3.26	12.6 ± 18.3
4x3	38.97	16.29	2.39	17.8 ± 19.4
Shuttle	34.40	16.25	2.12	18.4 ± 20.2
RockS4x4	0.97	0.49	1.99	37.7 ± 23.4
Hallway	26.77	15.95	1.68	26.8 ± 27.7
Network	11.94	10.30	1.16	26.7 ± 19.8
Partpaint	1.96	1.83	1.07	38.7 ± 31.8
1D	0.01	0.01	1.00	84.0 ± 21.8
Cheese	0.21	0.21	1.00	87.8 ± 22.1
4x4	0.43	0.55	0.78	76.0 ± 22.5

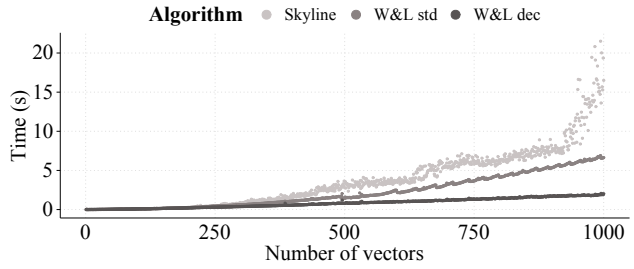
Table 2: Results LP experiment: Gurobi

Domain	Std (s)	Dec (s)	Speedup	Constr. (%)
4x5x2	88.95	20.83	4.27	12.4 ± 16.8
AircraftID	37.70	12.66	2.98	8.8 ± 14.7
Hallway2	80.47	27.37	2.94	18.2 ± 27.5
4x3	42.65	14.79	2.88	17.8 ± 19.4
Tiger-grid	60.02	23.72	2.53	12.8 ± 18.5
Shuttle	39.82	16.87	2.36	18.5 ± 20.3
Hallway	24.19	17.76	1.36	26.7 ± 27.9
Network	10.57	8.50	1.24	26.6 ± 19.8
RockS4x4	1.08	0.92	1.17	36.8 ± 23.1
Cheese	0.13	0.14	0.96	87.8 ± 22.1
1D	0.00	0.00	0.75	84.0 ± 21.8
4x4	0.30	0.41	0.72	75.3 ± 23.0
Partpaint	1.20	1.69	0.71	38.8 ± 31.8

Table 3: Results LP experiment: lpsolve

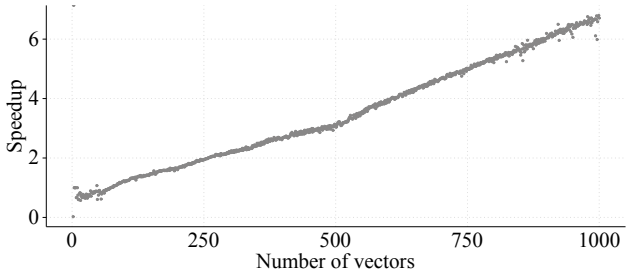


(a) Vector pruning speedup

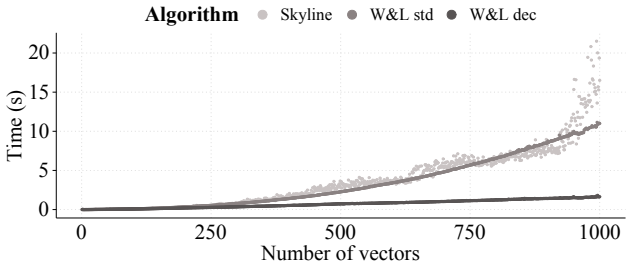


(b) Pruning method comparison

Figure 1: Pruning experiments with Gurobi



(a) Vector pruning speedup



(b) Pruning method comparison

Figure 3: Pruning experiments with lpsolve

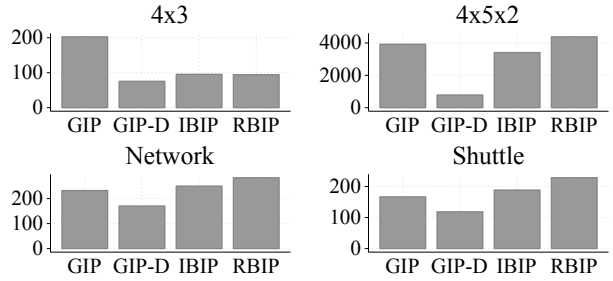


Figure 2: Incremental pruning experiment: Gurobi

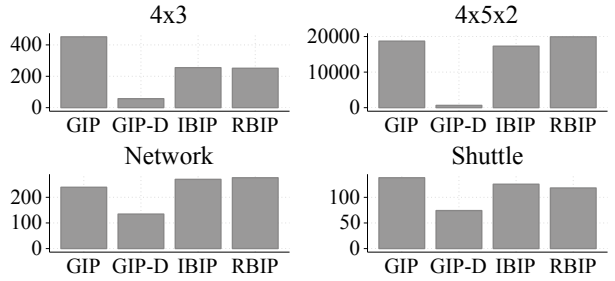


Figure 4: Incremental pruning experiment: lpsolve

Number of constraints used by the algorithm

In the paper we observed that the relatively large standard deviation in the first experiment is caused by small LPs, in which a large fraction of the constraints is used. Figure 5 shows the number of constraints (i.e., the percentage) as a function of the total number of constraints for several domains. The graphs confirm that in small LPs a relatively large number of constraints is added, which in turn affects the standard deviation.

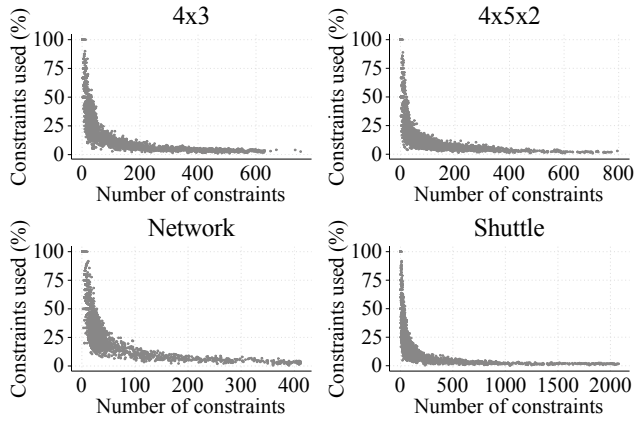


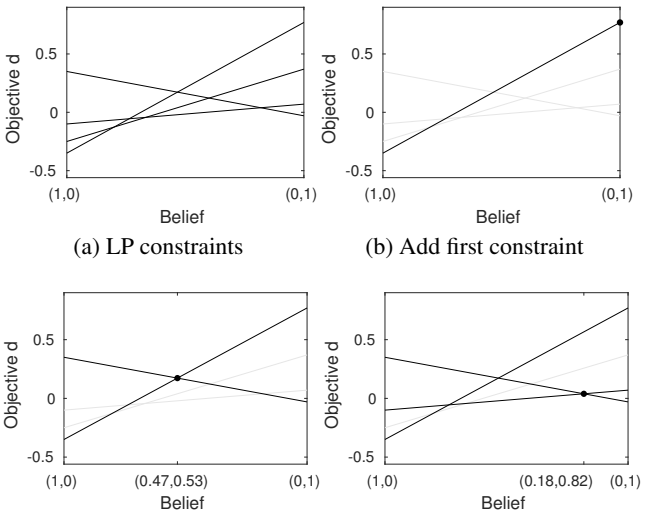
Figure 5: Number of constraints used by the decomposed LP

Open-source software: SolvePOMDP

An implementation of incremental pruning combined with the decomposed LP can be found in SolvePOMDP, an open-source java program for solving Partially Observable Markov Decision Processes. The program includes an exact value iteration algorithm for POMDPs, as well as an approximate algorithm based on point-based value iteration. More information about SolvePOMDP can be found on `www.erwinwalraven.nl/solvepomdp`.

Illustration of the decomposed LP algorithm

We consider the linear program shown in Figure 6a and we show how Algorithm 3 finds the optimal solution. We assume that the algorithm selects belief $b' = (1, 0)$ on line 7. In the first iteration it adds the constraint with the lowest objective value in b', which is shown in Figure 6b. After solving the master LP the belief point $b' = (0, 1)$ is found, which is represented by the dot. The same procedure is used to select a new constraint, after which the master LP is solved again to obtain belief $b' = (0.47, 0.53)$, shown in Figure 6c. In the third iteration belief $b' = (0.18, 0.82)$ is found, which is shown in Figure 6d. This belief b' corresponds to the optimal solution. One of the constraints has not been added during the execution of the algorithm.



(a) LP constraints (b) Add first constraint

(c) Add second constraint (d) Add third constraint

Figure 6: Example execution of the decomposed LP