A Comparison of Sampling Strategies for Learning Policies

Farid Musayev[1] Dominik Drexler[1] Daniel Gnad[1,2] Jendrik Seipp[1]

[1]Linköping University [2]Heidelberg University

Machine Reasoning Lab

LINKÖPING UNIVERSITY

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

WASP WALLENBERG AI, AUTONOMOUS SYSTEMS AND SOFTWARE PROGRAM

Motivation

Motivation

- **Learning policies**[1] for classical planning:
  - Classical machine learning
  - State spaces grow exponentially with the number of objects.
  - We cannot use all available transitions for learning — we *sample*.

[1] Musayev et al., "Combining Heuristics and Transition Classifiers in Classical Planning".

Motivation

- **Learning policies**[1] for classical planning:
  - ▷ Classical machine learning
  - ▷ State spaces grow exponentially with the number of objects.
  - ▷ We cannot use all available transitions for learning — we *sample*.

- **Question:** How should we sample training data from large state spaces?
  - ▷ Which transitions are informative for learning?

[1] Musayev et al., "Combining Heuristics and Transition Classifiers in Classical Planning".

Motivation

- **Learning policies**[1] for classical planning:
  - Classical machine learning
  - State spaces grow exponentially with the number of objects.
  - We cannot use all available transitions for learning — we *sample*.

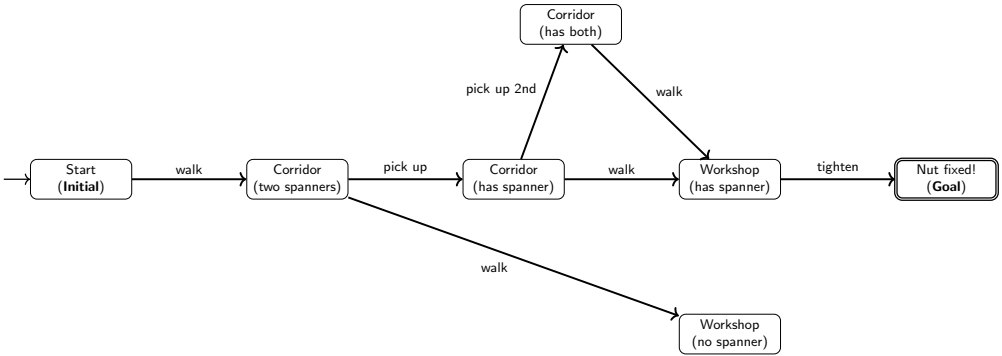
- **Question:** How should we sample training data from large state spaces?
  - Which transitions are informative for learning?

- **Contribution:** Systematic study of four criteria and transitions from optimal plans.
  - Goal-distance stratification
  - Symmetry detection
  - Good-transition ratio
  - Cost of error

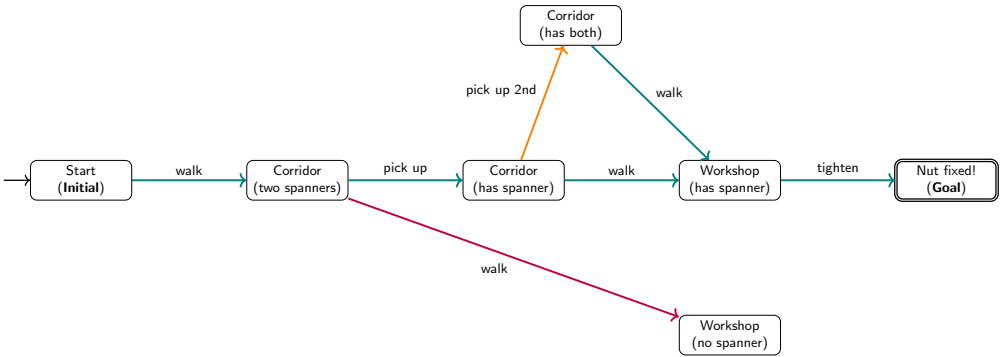
[1] Musayev et al., "Combining Heuristics and Transition Classifiers in Classical Planning".

Learning Policies

Policies for Classical Planning



Policies for Classical Planning



Good: decreases goal distance. Bad: increases or maintains goal distance. Uns: leads to a dead-end.

Learning Pipeline

Learning Pipeline

Planning with Policies

- Policies are run in a **lookahead search** within Fast Downward[2].
- **Algorithm:**
  - Follow the policy from the initial state.
  - Store all successors along the trajectory in an open list.
  - If the policy fails — select a state from the open list, sorted by predicted unsolvability (for domains with dead-ends) then by decreasing g-value (depth-first).
- **Limits:** 30-minute time limit, 8 GiB memory per test instance.

[2] Helmert, "The Fast Downward Planning System".

Sampling Strategies

Sampling Criteria

- **Goal-distance stratification** — divide the state space into strata by $h^*(s)$.
  - ▷ Avoids under/over-sampling critical goal-distance regions.

[3] Drexler et al., "Symmetries and Expressive Requirements for Learning General Policies".

Sampling Criteria

- **Goal-distance stratification** — divide the state space into strata by $h^*(s)$.
  - ▷ Avoids under/over-sampling critical goal-distance regions.
- **Symmetry detection** — identify and exclude symmetric[3] transitions.
  - ▷ Reduces redundancy; improves data diversity.

[3] Drexler et al., "Symmetries and Expressive Requirements for Learning General Policies".

Sampling Criteria

- **Goal-distance stratification** — divide the state space into strata by $h^*(s)$.
  - ▷ Avoids under/over-sampling critical goal-distance regions.
- **Symmetry detection** — identify and exclude symmetric[3] transitions.
  - ▷ Reduces redundancy; improves data diversity.
- **Good-transition ratio** ρ_+ — fraction of outgoing transitions that are good in a state.
  - ▷ Prioritize states with *low* ρ_+: harder decisions, higher learning value.

[3] Drexler et al., "Symmetries and Expressive Requirements for Learning General Policies".

Sampling Criteria

- **Goal-distance stratification** — divide the state space into strata by $h^*(s)$.
  - ▷ Avoids under/over-sampling critical goal-distance regions.
- **Symmetry detection** — identify and exclude symmetric[3] transitions.
  - ▷ Reduces redundancy; improves data diversity.
- **Good-transition ratio** ρ_+ — fraction of outgoing transitions that are good in a state.
  - ▷ Prioritize states with *low* ρ_+: harder decisions, higher learning value.
- **Cost of error** C_{err} — extra effort incurred by making a wrong decision in a state.
  - ▷ Prioritize states where mistakes are most costly.

[3] Drexler et al., "Symmetries and Expressive Requirements for Learning General Policies".

Sampling Strategies

- **Baseline — Random:** $\sigma(T)$ — sample transitions uniformly at random.

Sampling Strategies

- **Baseline — Random:** $\sigma(T)$ — sample transitions uniformly at random.
- **Transition-based sampling:**
  - $\sigma(T_g)$ — goal-distance-stratified transition sampling
  - $\sigma(\widetilde{T}_g)$ — also detect and exclude symmetric transitions

Sampling Strategies

- **Baseline — Random:** $\sigma(T)$ — sample transitions uniformly at random.
- **Transition-based sampling:**
  - $\sigma(T_{\mathrm{g}})$ — goal-distance-stratified transition sampling
  - $\sigma(\widetilde{T}_{\mathrm{g}})$ — also detect and exclude symmetric transitions
- **State-based sampling** (all use goal-distance + symmetry detection):
  - $\sigma(\widetilde{S}_{\mathrm{g}}, \rho_{+})$ — prioritize states with low good-transition ratio ρ_{+}
  - $\sigma(\widetilde{S}_{\mathrm{g}}, C_{\mathrm{err}})$ — prioritize states with high cost of error C_{err}

Sampling Strategies

- **Baseline — Random:** $\sigma(T)$ — sample transitions uniformly at random.
- **Transition-based sampling:**
  - $\sigma(T_{\mathrm{g}})$ — goal-distance-stratified transition sampling
  - $\sigma(\widetilde{T}_{\mathrm{g}})$ — also detect and exclude symmetric transitions
- **State-based sampling** (all use goal-distance + symmetry detection):
  - $\sigma(\widetilde{S}_{\mathrm{g}}, \rho_{+})$ — prioritize states with low good-transition ratio ρ_{+}
  - $\sigma(\widetilde{S}_{\mathrm{g}}, C_{\mathrm{err}})$ — prioritize states with high cost of error C_{err}
- **Optimal plans** (with symmetry detection):
  - $\sigma(\widetilde{T}_{*})$ — transitions from optimal plans only

Is Random Sampling Enough?

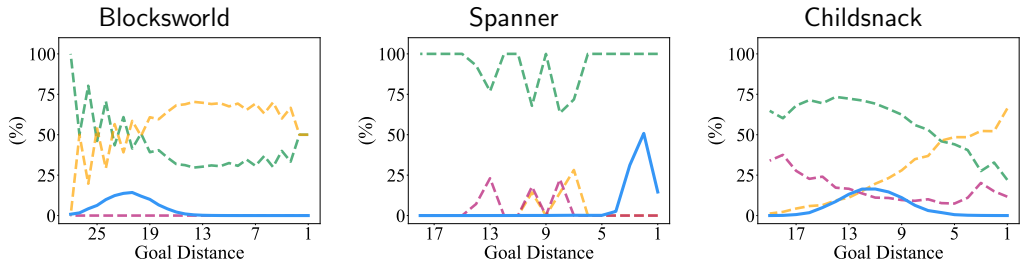
Is Random Sampling Enough?

Domain	$\sigma(T)$	$\sigma(T_g)$
Blocksworld (90)	20±9	**51±16**
Childsnack (66)	**61±3**	43±19
Ferry (76)	74±1	**75±1**
Floortile (90)	**37±12**	31±1
Miconic (90)	**90±0**	**90±0**
Rovers (76)	**50±5**	**50±6**
Satellite (90)	**55±3**	52±3
Sokoban (86)	**22±1**	**22±2**
Spanner (90)	48±15	**64±0**
Transport (66)	**66±0**	**66±0**
Sum (820)	523±49	**544±48**

Number of solved tasks per domain and sampling strategy on the IPC 2023 Learning Track instances.

Not always — the picture is more nuanced.

Is Random Sampling Enough?



Blue (solid): share of transitions per distance. Dashed: proportions of good / bad / unsolvable transitions per distance.

- Blocksworld: transitions cluster early — random misses critical regions near the goal.
- Spanner: random undersamples the critical choice point (pick up spanner or move).
- Childsnack: stratification alone can *degrade* performance — cost of error criterion helps here.

Does Symmetry Detection Help?

Does Symmetry Detection Help?

- **Data efficiency:**
  - ▷ Spanner: 100 000 → 7 000 transitions
  - ▷ Childsnack: 100 000 → 27 000 transitions
- **Generalization:**
  - ▷ Childsnack and Floortile benefit most.
- Other domains (few symmetries): little to no change.

Domain	$\sigma(T_g)$	$\sigma(\widetilde{T}_g)$
Blocksworld (90)	**51±16**	47±13
Childsnack (66)	43±19	**55±12**
Ferry (76)	**75±1**	**75±0**
Floortile (90)	31±1	**44±2**
Miconic (90)	**90±0**	**90±0**
Rovers (76)	50±6	**51±5**
Satellite (90)	52±3	**55±4**
Sokoban (86)	22±2	**23±1**
Spanner (90)	**64±0**	63±0
Transport (66)	**66±0**	**66±0**
Sum (820)	544±48	**569±37**

Number of solved tasks per domain and sampling strategy on the IPC 2023 Learning Track instances.

Do Sampling Criteria Matter?

Do Sampling Criteria Matter?

	Baseline	Sampling Criteria		Optimal Plans
Domain	$\sigma(T)$	$\sigma(\widetilde{S}_{\mathrm{g}}, \rho_{+})$	$\sigma(\widetilde{S}_{\mathrm{g}}, C_{\mathrm{err}})$	$\sigma(\widetilde{T}_{*})$
Blocksworld (90)	20±9	**69±8**	65±10	68±2
Childsnack (66)	61±3	43±17	**62±1**	32±0
Ferry (76)	74±1	**75±1**	**75±1**	67±0
Floortile (90)	37±12	36±1	36±1	**60±0**
Miconic (90)	**90±0**	**90±0**	**90±0**	**90±0**
Rovers (76)	50±5	50±5	51±5	**52±3**
Satellite (90)	**55±3**	53±5	**55±3**	52±7
Sokoban (86)	22±1	22±1	**25±1**	20±0
Spanner (90)	48±15	**65±0**	**65±0**	64±0
Transport (66)	**66±0**	65±1	65±1	**66±0**
Sum (820)	523±49	568±39	**589±23**	571±12

Number of solved tasks per domain and sampling strategy on the IPC 2023 Learning Track instances.

Yes — but it depends on the domain.

Comparison to State of the Art

Comparison to State of the Art

Domain	LAMA-first[4]	$h^{\mathrm{WLF}5}_{\mathrm{GPR}}$	$\sigma(\widetilde{S}_{\mathrm{g}}, C_{\mathrm{err}})$	$\sigma(\widetilde{T}_*)$
Blocksworld (90)	60	**72**	65	68
Childsnack (66)	35	31	**62**	32
Ferry (76)	68	**76**	75	67
Floortile (90)	11	2	36	**60**
Miconic (90)	**90**	**90**	**90**	**90**
Rovers (76)	**68**	37	51	52
Satellite (90)	**89**	48	55	52
Sokoban (86)	**40**	38	25	20
Spanner (90)	30	**73**	65	64
Transport (66)	**66**	28	65	**66**
Sum (820)	557	495	**589**	571

Policies learned with our strategies outperform LAMA and WL-GOOSE in total coverage and **win 5–6 out of 10 domains.**

[4] Richter and Westphal, "The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks".

[5] Chen, Trevizan, and Thiébaux, "Return to Tradition: Learning Reliable Heuristics with Classical Machine Learning".

Conclusions

Conclusions

- **Goal-distance stratification** helps in domains with skewed distribution.
- **Symmetry detection** improves data efficiency and generalization.
- **Cost of error** is more consistent than good-transition ratio.
- **Optimal plans** suffice for many domains in our benchmark set.

Conclusions

- **Goal-distance stratification** helps in domains with skewed distribution.
- **Symmetry detection** improves data efficiency and generalization.
- **Cost of error** is more consistent than good-transition ratio.
- **Optimal plans** suffice for many domains in our benchmark set.

Work in Progress:

- Learning algorithm — tree optimization and model selection metric.
- Feature analysis — generate more expressive features.

Conclusions

- **Goal-distance stratification** helps in domains with skewed distribution.
- **Symmetry detection** improves data efficiency and generalization.
- **Cost of error** is more consistent than good-transition ratio.
- **Optimal plans** suffice for many domains in our benchmark set.

Work in Progress:

- Learning algorithm — tree optimization and model selection metric.
- Feature analysis — generate more expressive features.

Thank you!

References

References I

- Chen, Dillon Z., Felipe Trevizan, and Sylvie Thiébaux. "Return to Tradition: Learning Reliable Heuristics with Classical Machine Learning". In: *Proceedings of the Thirty-Fourth International Conference on Automated Planning and Scheduling (ICAPS 2024)*. Ed. by Sara Bernardini and Christian Muise. AAAI Press, 2024, pp. 68–76.
- Drexler, Dominik et al. "Symmetries and Expressive Requirements for Learning General Policies". In: *Proceedings of the Twenty-First International Conference on Principles of Knowledge Representation and Reasoning (KR 2024)*. Ed. by Pierre Marquis, Magdalena Ortiz, and Maurice Pagnucco. IJCAI Organization, 2024.
- Helmert, Malte. "The Fast Downward Planning System". In: *Journal of Artificial Intelligence Research* 26 (2006), pp. 191–246.

References II

- Musayev, Farid et al. “Combining Heuristics and Transition Classifiers in Classical Planning”. In: *Proceedings of the 28th European Conference on Artificial Intelligence (ECAI 2025)*. Ed. by Inês Lynce and Aniello Murano. IOS Press, 2025, pp. 4694–4701.
- Richter, Silvia and Matthias Westphal. “The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks”. In: *Journal of Artificial Intelligence Research* 39 (2010), pp. 127–177.