Predicting Macro-Learning Performance Using Structural Regularity

Anton Gustafsson,
Jendrik Seipp,
Elliot Gestrin

Motivation

- Macro-learning can dramatically improve planning
- But can also hurt performance
- No reliable way to predict when it helps
- Goal: predict macro-learning usefulness before applying it

Core Idea: Structural Regularity

Hypothesis:

Domains with recurring structural patterns are more likely to benefit from macro-learning.

Metric:

Measure repetition in plans by counting repeating action tuples across instances.

What about parameters/objects?

There are different object identities between instances.

Therefore, we cannot compare them directly.

But ignoring parameters makes the metric too coarse.

Parameter Flow

We track how parameters flow between actions rather than object identities.

Example:

action1(a,b,c)

action2(c,b,a)

has the same parameter flow as

action1(d,e,f)

action2(f,e,d)

Metric Pipeline

Plans from a Domain

↓

Repeating Action Tuples

↓

Parameter Flows

↓

Similarity Scores

↓

Structural Regularity Score

↓

Predict Macro-Learning Impact

Example:

action1(a,b,c)

action2(c,b,a)

has the same parameter flow as

action1(d,e,f)

action2(f,e,d)

Dataset

- 10 macro-learning studies
- 47 planning domains
- 84 labeled domain-study pairs
- Labels: Negative / Neutral / Positive
- 1373 solved planning tasks

Experimental Setup

- Compute a structural regularity score for each domain
- Learn two threshold to classify the training domains as:
  - -1: Negative macro-learning impact
  - 0: Neutral impact
  - 1: Positive impact
- Evaluate the learned thresholds on unseen test domains

Training Results

Structural Regularity confusion matrix:

True Label	Pred. -1	Pred. 0	Pred. 1
-1	3	7	1
0	0	12	3
1	0	18	16

Compared with theoretical optimum:

True Label	Pred. -1	Pred. 0	Pred. 1
-1	7	3	1
0	0	12	3
1	0	3	31

Testing Results

Structural Regularity confusion matrix:

True Label	Pred. -1	Pred. 0	Pred. 1
-1	0	2	1
0	1	3	1
1	1	5	5

Compared with theoretical optimum:

True Label	Pred. -1	Pred. 0	Pred. 1
-1	3	0	0
0	0	4	1
1	0	0	11

Discussion

Planner-specific factors remain important.

Key Observation:

Short recurring patterns were most informative.

Future Work

- Only measure short recurring patterns
- Per-macro-learning-method analysis
- Macro learners inspired by parameter flow

Conclusions

- Introduced structural regularity
- Compiled a macro-learning dataset
- First step toward predicting macro usefulness
- Generalization remains limited
- Results robust across multiple train/test splits

Definition 1. Let $\langle a_1, \ldots, a_n \rangle$ be an action tuple, i.e., a contiguous subsequence of a plan. For each action a_i, let $\mathit{par}(a_i) = \langle \mathit{par}(a_i)_1, \ldots, \mathit{par}(a_i)_{k_i} \rangle$ denote its parameters.

The *parameter flow* of the tuple is defined as $T_{a_1,\ldots,a_n} = (\sigma, F)$, where $\sigma = \langle \mathit{name}(a_1), \ldots, \mathit{name}(a_n) \rangle$ is the sequence of action names, and

$$F = \left\{ \langle i, j \rangle \in \{1, \ldots, k_1\} \times \{1, \ldots, k_n\} \mid \mathit{par}(a_1)_i = \mathit{par}(a_n)_j \right\}$$

is the set of indices for all shared parameters between the first and last actions of the tuple.

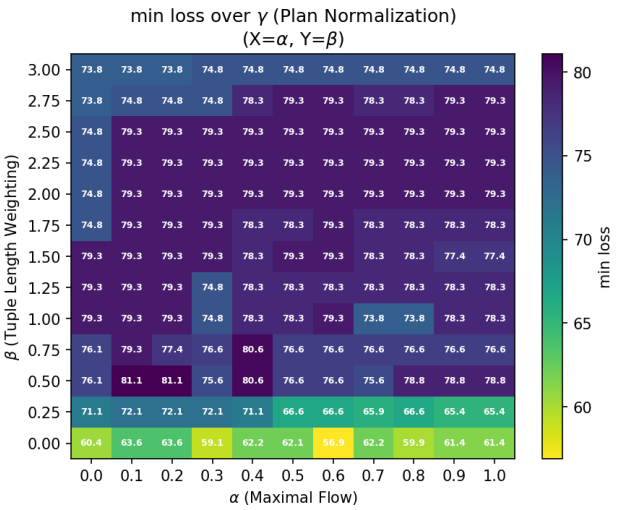
Definition 2. Let $T_{a_1,\ldots,a_n} = \langle \sigma, F \rangle$ and $T_{b_1,\ldots,b_m} = \langle \sigma', F' \rangle$ be two parameter flows. Let $\alpha \in [0,1]$ be a tunable parameter controlling the fraction of parameter correspondences that must match for the similarity score to attain 1. We define their *similarity score* as

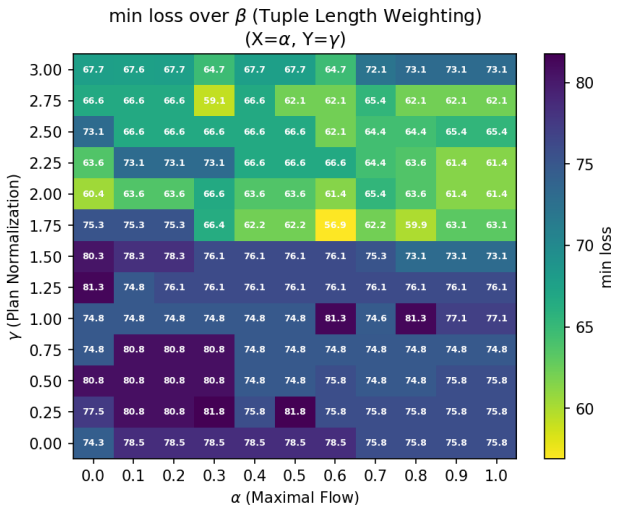
$$S_\alpha\left(T_{a_1,\ldots,a_n}, T_{b_1,\ldots,b_m}\right) =$$

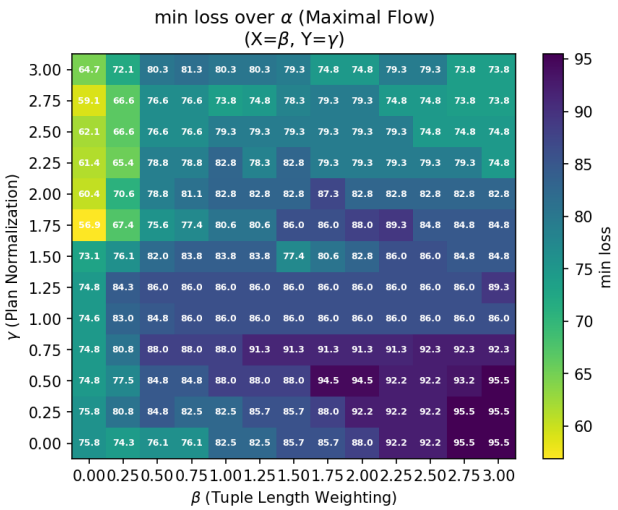
$$\begin{cases} \min\left(1,\ \frac{1+|F\cap F'|}{1+\left\lceil \alpha \cdot \min\left(|par(a_1)|,\ |par(a_n)|\right)\right\rceil}\right) & \text{if } \sigma = \sigma', \\ 0 & \text{otherwise.} \end{cases}$$

Definition 3. Let $\beta \geq 0$ be a tunable parameter controlling the importance of longer action tuples. We define the *problem-level structural regularity* of a plan π as

$$\mathit{structreg}_{\alpha,\beta}(\pi) = \sum_{n=2}^{|\pi|} \sum_{1 \leq i < j \leq |\pi| - n + 1} n^{\beta} \cdot S_{\alpha}\big(T^{\pi}(i,n),\, T^{\pi}(j,n)\big).$$







After selecting the parameter configuration with the lowest total loss ($\alpha = 0.6$, $\beta = 0.0$, and $\gamma = 1.75$), thresholds are re-estimated using the full training set and are used for final evaluation ($t_1 = 146.77$, $t_2 = 105182.16$). Further parameter selection results are provided in Appendix B.

Domain	Track	Source	Label	#Solved
Blocksworld	ipc-2000-typed	Armano, Cherchi, and Vargiu (2004)	1	35
Elevator	hoffmann-miconic-simple-adl	Armano, Cherchi, and Vargiu (2004)	−1	150
Satellite	ipc-2004-deterministic-strips	Botea et al. (2005)	1	36
Airport	ipc-2004-deterministic-adl	Botea et al. (2005)	0	34
Pipesworld No-Tankage	ipc-2004-deterministic	Botea et al. (2005)	1	43
Pipesworld Tankage	ipc-2004-deterministic	Botea et al. (2005)	0	43
PSR	ipc-2004-deterministic-middle-compiled	Botea et al. (2005)	0	50
Philosophers	ipc-2004-deterministic-derived-adl	Botea et al. (2005)	1	48
Optical Telegraph	ipc-2004-deterministic-derived-adl	Botea et al. (2005)	1	4
Driverlog	ipc-2002-deterministic-untyped	Coles and Smith (2007)	−1	20
Freecell	ipc-2000	Coles and Smith (2007)	−1	79
Satellite	ipc-2004-deterministic-strips	Coles and Smith (2007)	0	36
Depots	ipc-2002-deterministic-untyped	Coles and Smith (2007)	1	20
Airport	ipc-2004-deterministic-adl	Coles and Smith (2007)	0	34
Philosophers	ipc-2004-deterministic-derived-adl	Coles and Smith (2007)	1	48
Pipesworld Tankage	ipc-2004-deterministic	Coles and Smith (2007)	1	43
Pipesworld No-Tankage	ipc-2004-deterministic	Coles and Smith (2007)	0	43
Blocksworld	ipc-2000-typed	Chrpa (2010)	1	35
Depots	ipc-2002-deterministic-untyped	Chrpa (2010)	1	20
Zenotravel	ipc-2002-deterministic-untyped	Chrpa (2010)	0	20
Rovers	ipc-2002-deterministic-typed	Chrpa (2010)	0	40
Gripper	ipc-1998	Chrpa (2010)	1	20
Satellite	ipc-2004-deterministic-strips	Chrpa (2010)	−1	36
Freecell	ipc-2000	Chrpa (2010)	−1	79
Sokoban	ipc-2008-deterministic-sat-strips	Chrpa (2010)	−1	29
Gold Miner	ipc-2008-learning	Chrpa (2010)	1	30
Pipesworld No-Tankage	ipc-2004-deterministic	Chrpa (2010)	−1	43
Pipesworld Tankage	ipc-2004-deterministic	Chrpa (2010)	−1	43
N-puzzle	ipc-2008-learning	Chrpa (2010)	−1	30
Blocksworld	ipc-2011-learning	Dulac et al. (2013)	1	35
Ferry	ipc-2011-learning	Dulac et al. (2013)	1	30
Satellite	ipc-2011-learning	Dulac et al. (2013)	1	28
Gripper	ipc-2011-learning	Dulac et al. (2013)	1	17
Grid	ipc-2011-learning	Dulac et al. (2013)	1	30
Depots	ipc-2011-learning	Dulac et al. (2013)	1	0
Mprime	ipc-2011-learning	Dulac et al. (2013)	1	30
Parking	ipc-2011-learning	Dulac et al. (2013)	1	30
Barman	ipc-2011-learning	Dulac et al. (2013)	1	15
Barman	ipc-2011-learning	Chrpa, Vallati, and McCluskey (2014)	0	15
Depots	ipc-2011-learning	Chrpa, Vallati, and McCluskey (2014)	1	0
Gripper	ipc-2011-learning	Chrpa, Vallati, and McCluskey (2014)	1	20
TPP	ipc-2011-learning	Chrpa, Vallati, and McCluskey (2014)	1	30
Parking	ipc-2011-learning	Chrpa, Vallati, and McCluskey (2014)	0	20

For each domain, we solve a set of planning instances using the satisficing *LAMA* planner (Richter and Westphal 2010). We convert the resulting plans into makespan maximal form using *Mimir* (Ståhlberg 2023) when possible.[1] This re-orders the actions such that objects are more likely to repeat in subsequent actions, emulating the behavior of macro actions. Out of 1579 tasks, 1373 were solved and 1271 were made makespan maximal, each with 8 GiB memory limits and 30 minutes time out on an Intel Xeon Gold 6130 CPU. The structural regularity calculations used a memory limit of 90 GB and no time limit, though computations were quick. Of the 84 domain-track data points, 79 were used in our experiments. The remaining domains were excluded because no instances were successfully solved.

To account for both this imbalance and the ordinal nature of the labels, we define a weighted penalty function. Let N_i denote the number of instances with label i. We define the class-dependent weight function $w : \{-1, 0, 1\} \rightarrow \mathbb{R}_{>0}$ as

$$w(y) = \frac{\max(N_{-1}, N_0, N_1)}{N_y}.$$

Additionally, for the predicted label $\hat{y}$ and the true label y, we define the penalty function as

$$\textit{penalty}(\hat{y}, y) = w(y) \cdot |\hat{y} - y|$$

Label Ambiguity

- Different macro-learning studies may disagree on the same domain

- Solution: Each domain-study pair is treated as a separate observation