Core Framework: Thrun-based Representation Learning for Online Heuristic Selection in the Knapsack Problem

1 The Central Idea

1.1 Thrun's Representation Learning (1996)

Original concept: Learn a transformation function $g^* : \mathbb{R}^d \to \mathbb{R}^{d'}$ that maps input features to a new space where:

Instances with same label are close together, instances with different labels are far apart

Energy function (as described in Chen & Liu, 2018):

$$E(g) = \sum_{(x_i, y_i)} \left[\sum_{\substack{(x_j, y_j) \\ y_j = y_i}} \|g(x_i) - g(x_j)\| - \sum_{\substack{(x_k, y_k) \\ y_k \neq y_i}} \|g(x_i) - g(x_k)\| \right] \tag{1}$$

Minimizing $E(g)$ creates a geometrically structured space where simple classifiers (KNN, Shepard's method) can generalize effectively. This approach enables lifelong learning by creating representations that facilitate knowledge transfer across related tasks.

1.2 Adaptation to Heuristic Selection

Application:

- **Input space**: Instance features $\phi(I) \in \mathbb{R}^3$ describing Knapsack instances
- **Labels**: Optimal heuristic $y = \omega(I) \in \{0, 1, 2, 3\}$
- **Goal**: Learn g_θ such that instances where MAXPW is optimal cluster together, instances where MAXP is optimal cluster elsewhere, etc.

Key insight: If successful, we can predict the best heuristic for a new instance by:

1. Transform: $\mathbf{e}_{\text{new}} = g_\theta(\phi(I_{\text{new}}))$
2. Find nearest neighbors in embedding space
3. Select heuristic most common among neighbors (KNN) or weighted by distance (Shepard)

2 Problem Characterization

2.1 Knapsack Instance Space

Definition: Each instance $I = (n, \mathbf{p}, \mathbf{w}, c)$ where:

$$
\begin{aligned}
n &: \text{number of items} \\
\mathbf{p} = (p_1, \ldots, p_n) &: \text{profit vector} \\
\mathbf{w} = (w_1, \ldots, w_n) &: \text{weight vector} \\
c &: \text{knapsack capacity}
\end{aligned}
$$

Objective: Find $\mathbf{x} \in \{0, 1\}^n$ maximizing $\sum_i p_i x_i$ subject to $\sum_i w_i x_i \leq c$.

2.2 Heuristic Set

Four heuristics $\mathcal{H} = \{h_0, h_1, h_2, h_3\}$:

h_0**: DEFAULT (DEF)**

- **Description**: Packs items in their original order without any reordering
- **Rationale**: Serves as baseline; performance depends on initial instance ordering
- **Computational complexity**: $\mathcal{O}(n)$

h_1**: MINIMUM WEIGHT (MINW)**

- **Description**: Sorts items by weight in ascending order and packs lightest items first
- **Rationale**: Maximizes number of items packed; effective when capacity is tight and item values are similar
- **Sorting criterion**: $w_1 \leq w_2 \leq \cdots \leq w_n$
- **Computational complexity**: $\mathcal{O}(n \log n)$

h_2**: MAXIMUM PROFIT (MAXP)**

- **Description**: Sorts items by profit in descending order and packs most valuable items first
- **Rationale**: Greedy approach prioritizing absolute value; effective when capacity is sufficient and profit variation is high
- **Sorting criterion**: $p_1 \geq p_2 \geq \cdots \geq p_n$
- **Computational complexity**: $\mathcal{O}(n \log n)$

h_3**: MAXIMUM PROFIT-TO-WEIGHT RATIO (MAXPW)**

- **Description**: Sorts items by profit-to-weight ratio in descending order
- **Rationale**: Optimizes value per unit weight; approximates fractional knapsack solution
- **Sorting criterion**: $\frac{p_1}{w_1} \geq \frac{p_2}{w_2} \geq \cdots \geq \frac{p_n}{w_n}$
- **Computational complexity**: $\mathcal{O}(n \log n)$

2.3 Oracle Function

Definition: For any instance I:

$$\omega(I) = \arg\max_{h \in \mathcal{H}} \text{profit}(h(I)) \tag{2}$$

This defines the **ground truth** label for supervised learning.

2.4 Feature Extraction

Base features (Zárate-Aranda & Ortiz-Bayliss, 2025):

$$\phi_{\text{base}}(I) = \begin{pmatrix} P(I) \\ W(I) \\ C(I) \end{pmatrix} \tag{3}$$

where:

$$P(I) = \frac{\bar{p}}{\max_i p_i} = \frac{\frac{1}{n}\sum_{i=1}^{n} p_i}{\max_{i \in \{1,\ldots,n\}} p_i} \quad \in [0,1] \tag{4}$$

$$W(I) = \frac{\bar{w}}{\max_i w_i} = \frac{\frac{1}{n}\sum_{i=1}^{n} w_i}{\max_{i \in \{1,\ldots,n\}} w_i} \quad \in [0,1] \tag{5}$$

$$C(I) = \frac{\rho(\mathbf{p}, \mathbf{w}) + 1}{2} \quad \in [0,1] \tag{6}$$

with $\rho(\mathbf{p}, \mathbf{w})$ denoting **Pearson correlation coefficient**:

$$\rho(\mathbf{p}, \mathbf{w}) = \frac{\sum_{i=1}^{n}(p_i - \bar{p})(w_i - \bar{w})}{\sqrt{\sum_{i=1}^{n}(p_i - \bar{p})^2}\sqrt{\sum_{i=1}^{n}(w_i - \bar{w})^2}} \tag{7}$$

Assumption on correlation structure: We assume that the Plata-González generator (Plata-González et al., 2019) produces instances where the relationship between profit and weight is **approximately linear**. This justifies the use of Pearson correlation, which captures linear associations.

Critical consideration: These three features are **scale-invariant by design** (normalized by instance-specific statistics). This raises the fundamental question:

Are P, W, C sufficient to determine optimal heuristic when (n, range, c) vary?

3 Transformation Network Architecture

3.1 Neural Network Parameterization

Multi-layer perceptron:

$$g_\theta : \mathbb{R}^3 \to \mathbb{R}^{d'} \tag{8}$$

where d' is the embedding dimension (to be determined empirically). The network comprises:

$$g_\theta(\mathbf{x}) = f_L \circ \sigma \circ f_{L-1} \circ \sigma \circ \cdots \circ \sigma \circ f_1(\mathbf{x}) \tag{9}$$

where:

- $f_\ell : \mathbb{R}^{d_{\ell-1}} \to \mathbb{R}^{d_\ell}$ are affine transformations
- σ denotes activation function (e.g., ReLU)
- Layer dimensions $\{d_0, d_1, \ldots, d_L\}$ and architecture details to be determined through hyperparameter tuning

3.2 Contrastive Loss Function

Training objective:

$$\mathcal{L}_{\text{contrast}}(\theta; \mathcal{B}) = \frac{1}{|\mathcal{B}|^2} \sum_{i,j \in \mathcal{B}} \ell_{\text{pair}}(\mathbf{e}_i, \mathbf{e}_j, y_i, y_j) \tag{10}$$

where $\mathbf{e}_i = g_\theta(\phi(I_i))$ and:

$$\ell_{\text{pair}}(\mathbf{e}_i, \mathbf{e}_j, y_i, y_j) = \begin{cases} \|\mathbf{e}_i - \mathbf{e}_j\|^2 & \text{if } y_i = y_j \quad \text{(pull together)} \\ \max(0, \gamma - \|\mathbf{e}_i - \mathbf{e}_j\|)^2 & \text{if } y_i \neq y_j \quad \text{(push apart)} \end{cases} \tag{11}$$

Margin parameter: $\gamma > 0$ enforces minimum separation between heuristic classes (typically $\gamma = 1.0$).

3.3 Classification in Embedding Space

3.3.1 K-Nearest Neighbors

Given new instance I_{new}:

$$\hat{h}_{\text{KNN}}(I_{\text{new}}) = \text{mode}\left(\{y_j : j \in \mathcal{N}_k(g_\theta(\phi(I_{\text{new}})))\}\right) \tag{12}$$

where $\mathcal{N}_k(\mathbf{e})$ denotes the indices of k nearest neighbors to embedding $\mathbf{e}$ in memory buffer $\mathcal{M}$.

3.3.2 Shepard's Method

Shepard's method provides weighted voting based on inverse distance:

$$\hat{h}_{\text{Shepard}}(I_{\text{new}}) = \arg\max_{h \in \mathcal{H}} \sum_{\substack{(I_j, y_j) \in \mathcal{M} \\ y_j = h}} w_j(I_{\text{new}}) \tag{13}$$

where Shepard weights are defined as:

$$w_j(I_{\text{new}}) = \frac{1}{\|g_\theta(\phi(I_{\text{new}})) - g_\theta(\phi(I_j))\|^p} \tag{14}$$

with power parameter $p \geq 1$ (typically $p = 2$).

Property: Shepard's method is an **exact interpolant**: if $I_{\text{new}} = I_j$ for some $I_j \in \mathcal{M}$, then $\hat{h}_{\text{Shepard}}(I_{\text{new}}) = y_j$.

4 Online Continual Learning Framework

4.1 Non-stationary Stream

Setting: Instances arrive sequentially:

$$\mathcal{S} = (I_1, I_2, I_3, \ldots) \tag{15}$$

where $I_t \sim \mathcal{P}_t$ and distribution $\mathcal{P}_t$ changes over time through variations in:

- **Instance size**
- **Profit range**
- **Correlation structure**
- **Capacity**

Note on difficulty considerations: For this work, we focus on the Zárate-Aranda and Plata-González parameter ranges but acknowledge that generalization to Pisinger's harder instances remains an open question requiring future validation.

Challenge: Must adapt to new distributions while retaining performance on old ones (avoid catastrophic forgetting).

4.2 Curriculum Learning Strategy for Concept Drift

Motivation: The order in which instances are presented significantly impacts the model's ability to adapt to distribution shifts while avoiding catastrophic forgetting. We employ a **curriculum learning** approach that structures the data stream to facilitate progressive adaptation.

4.2.1 Drift Types

We consider three types of concept drift:

1. Abrupt Drift (Block Presentation)

$$\mathcal{S}_{\text{abrupt}} = \underbrace{I_1^A, \ldots, I_{100}^A}_{\text{Class A}}, \underbrace{I_1^B, \ldots, I_{100}^B}_{\text{Class B}}, \underbrace{I_1^C, \ldots, I_{100}^C}_{\text{Class C}}, \ldots \tag{16}$$

Characteristics:

- Instances grouped by (n, c, range) configuration
- Sharp distribution boundaries
- Tests model's ability to handle sudden changes
- Worst-case scenario for catastrophic forgetting

2. Gradual Drift (Smooth Transition)

$$\mathcal{S}_{\text{gradual}} = I_1^A, I_2^A, I_1^B, I_3^A, I_2^B, I_4^A, I_3^B, I_1^C, \ldots \tag{17}$$

Implementation: Linear interpolation between parameter configurations over $T_{\text{transition}}$ instances:

$$n(t) = n_{\text{old}} + \frac{t}{T_{\text{transition}}}(n_{\text{new}} - n_{\text{old}}) \tag{18}$$

$$\text{range}(t) = \text{range}_{\text{old}} + \frac{t}{T_{\text{transition}}}(\text{range}_{\text{new}} - \text{range}_{\text{old}}) \tag{19}$$

Characteristics:

- Instances from adjacent distributions interleaved
- Progressive parameter shifts
- More realistic scenario for real-world deployment

3. Recurrent Drift (Cyclic Pattern)

$$\mathcal{S}_{\text{recurrent}} = \mathcal{C}_1 \to \mathcal{C}_2 \to \mathcal{C}_3 \to \mathcal{C}_1 \to \mathcal{C}_2 \to \cdots \tag{20}$$

Characteristics:

- Distributions reappear periodically
- Tests knowledge retention over long horizons
- Evaluates memory buffer effectiveness

4.2.2 Adaptive Retraining Schedule

The retraining frequency T adapts based on detected drift magnitude:

$$T(t) = \begin{cases} T_{\min} & \text{if } d_{\text{drift}}(t) > \theta_{\text{high}} \quad \text{(frequent retraining)} \\ T_{\text{nominal}} & \text{if } \theta_{\text{low}} \leq d_{\text{drift}}(t) \leq \theta_{\text{high}} \\ T_{\max} & \text{if } d_{\text{drift}}(t) < \theta_{\text{low}} \quad \text{(stable period)} \end{cases} \tag{21}$$

where drift magnitude is estimated via:

$$d_{\text{drift}}(t) = \frac{1}{W} \sum_{i=t-W}^{t} \nVdash\{\hat{h}(I_i) \neq \omega(I_i)\} \tag{22}$$

Parameters:

- W: sliding window size for error estimation
- $\theta_{\text{low}}, \theta_{\text{high}}$: drift detection thresholds
- $T_{\min} = 50$, $T_{\text{nominal}} = 100$, $T_{\max} = 200$

4.2.3 Memory Buffer Strategy Under Drift

When distribution changes, prioritize:

1. Temporal Diversity

$$\text{priority}_{\text{time}}(I_j) = \frac{t_{\text{current}} - t_j}{t_{\text{current}}} \tag{23}$$

Keep instances from different time periods to preserve historical knowledge.

2. Class Balance

$$\text{priority}_{\text{class}}(I_j) = \frac{1}{|\{I_k \in \mathcal{M}_t : \omega(I_k) = \omega(I_j)\}|} \tag{24}$$

Ensure all four heuristics remain represented.

3. Embedding Diversity

$$\text{priority}_{\text{diversity}}(I_j) = \min_{I_k \in \mathcal{M}_t \setminus \{I_j\}} \|g_\theta(\phi(I_j)) - g_\theta(\phi(I_k))\| \tag{25}$$

Retain instances that cover different regions of embedding space.

Combined priority:

$$\text{priority}(I_j) = \alpha \cdot \text{priority}_{\text{time}}(I_j) + \beta \cdot \text{priority}_{\text{class}}(I_j) + \gamma \cdot \text{priority}_{\text{diversity}}(I_j) \tag{26}$$

with $\alpha + \beta + \gamma = 1$.

Strategy	Drift Type	Adaptation Speed	Forgetting Risk
Abrupt	Block	Slow	High
Gradual	Smooth	Medium	Medium
Recurrent	Cyclic	Fast (if seen before)	Low

Table 1: Curriculum strategies and their characteristics

4.2.4 Evaluation Protocol

Compare three curriculum strategies:

Metrics:

- **Forward Transfer (FWT)**: Performance on new distribution after training on old
- **Backward Transfer (BWT)**: Performance on old distribution after training on new
- **Average Accuracy (AA)**: Overall performance across all seen distributions

$$\mathrm{FWT} = \frac{1}{K-1}\sum_{i=2}^{K}[\mathrm{acc}(D_i) - \mathrm{acc}_{\mathrm{baseline}}(D_i)] \tag{27}$$

$$\mathrm{BWT} = \frac{1}{K-1}\sum_{i=1}^{K-1}[\mathrm{acc}_{\mathrm{final}}(D_i) - \mathrm{acc}(D_i)] \tag{28}$$

$$\mathrm{AA} = \frac{1}{K}\sum_{i=1}^{K}\mathrm{acc}_{\mathrm{final}}(D_i) \tag{29}$$

where:

- D_i: i-th distribution in curriculum
- K: total number of distributions
- $\mathrm{acc}(D_i)$: accuracy on D_i immediately after training on it
- $\mathrm{acc}_{\mathrm{final}}(D_i)$: accuracy on D_i after training on all subsequent distributions
- $\mathrm{acc}_{\mathrm{baseline}}(D_i)$: accuracy on D_i with no prior training

4.3 Experience Replay Buffer

Memory: Maintain bounded buffer $\mathcal{M}_t$ with:

$$|\mathcal{M}_t| \leq M_{\max} \tag{30}$$

Content: $\mathcal{M}_t = \{(I_1, \omega(I_1), t_1, \mathbf{e}_1), (I_2, \omega(I_2), t_2, \mathbf{e}_2), \ldots\}$

Each entry stores:

- Instance I_j and its optimal heuristic $\omega(I_j)$
- Timestamp t_j for temporal priority
- Embedding $\mathbf{e}_j = g_\theta(\phi(I_j))$ for diversity computation

Insertion policy: When $|\mathcal{M}_t| = M_{\max}$, replace instance with **lowest combined priority** (as defined in Section 4.2.4).

4.4 Periodic Retraining

Schedule: Every T new instances:

1. Sample mini-batch $\mathcal{B}$ from $\mathcal{M}_t$
2. Compute $\mathcal{L}_{\text{contrast}}(\theta; \mathcal{B})$
3. Update: $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$
4. Repeat for E epochs
5. Update embeddings for all instances in $\mathcal{M}_t$
6. Refit classifier (KNN or Shepard)

Rationale: Periodic retraining on diverse buffer prevents forgetting while adapting to recent data.

5 Assumption to Validate

5.1 Existence of Discriminable Embedding Space

Statement: There exists a transformation $g_\theta : \mathbb{R}^3 \to \mathbb{R}^{d'}$ that creates an embedding space where the four heuristic classes are geometrically separable **despite** variations in (n, range, c).

Formalization: For the learned embedding space $\mathcal{E} = \{g_\theta(\phi(I)) : I \in \mathcal{I}\}$, we require:

$$\min_{h \neq h'} \left\{ \inf_{\substack{I \in \mathcal{C}_h \\ I' \in \mathcal{C}_{h'}}} \|g_\theta(\phi(I)) - g_\theta(\phi(I'))\| \right\} \geq 2\delta \tag{31}$$

where $\delta = \max_{h \in \mathcal{H}} \sup_{I, I' \in \mathcal{C}_h} \|g_\theta(\phi(I)) - g_\theta(\phi(I'))\|$ is the maximum intra-class diameter and $\mathcal{C}_h = \{I \in \mathcal{I} : \omega(I) = h\}$.

The neural transformation g_θ can create a representation where:

- Instances with same optimal heuristic form tight clusters (small δ)
- Different heuristic classes are well-separated (large inter-class distance)
- This structure persists across different values of n, range, c, and correlation structures

Validation metric: Silhouette score

$$s = \frac{1}{N} \sum_{i=1}^{N} \frac{b_i - a_i}{\max(a_i, b_i)} \geq 0.4 \tag{32}$$

where:

- a_i = average distance to instances with same ω in embedding space
- b_i = average distance to nearest different-class instances in embedding space

Even if features (P, W, C) are sufficient, we need to verify that a learnable transformation exists that exploits this sufficiency to create a useful embedding space. The existence of such a space is not guaranteed a priori.

References

- Chen, Z., & Liu, B. (2018). *Lifelong machine learning* (2nd ed.). Morgan & Claypool Publishers. https://doi.org/10.2200/S00832ED1V01Y201802AIM037
- Pisinger, D. (2005). Where are the hard knapsack problems? *Computers & Operations Research, 32*(9), 2271–2284. https://doi.org/10.1016/j.cor.2004.03.002
- Plata-González, L. F., Amaya, I., Ortiz-Bayliss, J. C., Conant-Pablos, S. E., Terashima-Marín, H., Coello Coello, C. A. (2019). Evolutionary-based tailoring of synthetic instances for the Knapsack problem. Soft Computing, 23(23), 12711-12728. https://doi.org/10.1007/s00500-019-03822-w
- Sim, K., Hart, E., & Paechter, B. (2015). A lifelong learning hyper-heuristic method for bin packing. *Evolutionary Computation, 23*(1), 37–67. https://doi.org/10.1162/EVCO_a_00121
- Zárate-Aranda, J. E., & Ortiz-Bayliss, J. C. (2025). Machine-learning-based hyper-heuristics for solving the Knapsack Problem. *Pattern Recognition Letters, 196*, 338–343. https://doi.org/10.1016/j.patrec.2024.11.021