End-to-End PPO-Based Reinforcement Learning Framework for Scalable 0/1 Knapsack Problem Solving

From Data Generation to Large-Scale Generalization

Gang Lin
Student ID: 2874886

University of Birmingham

19/08/2025

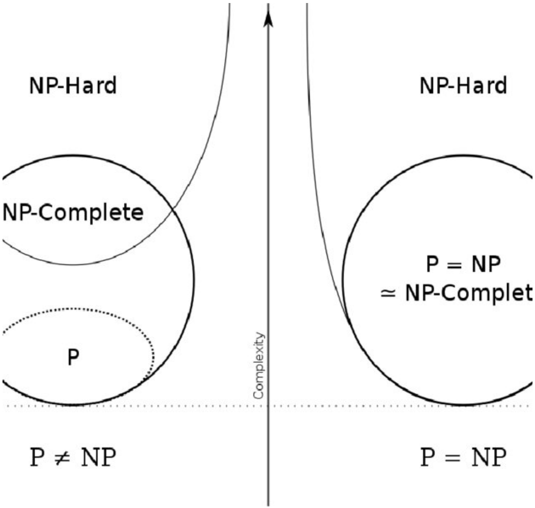


Figure: Landscapes of computational complexity.

Knapsack Problem Description:

- Given n items and a knapsack with capacity W.
- Each item i has a weight w_i and a value v_i.

Objective:

- Maximize the total value of selected items, subject to the total weight not exceeding W.
- Each item must either be taken (1) or left (0).

Mathematical Formulation:

$$\text{maximize} \quad \sum_{i=1}^{n} v_i x_i$$

$$\text{subject to} \quad \sum_{i=1}^{n} w_i x_i \leq W$$

$$x_i \in \{0,1\}, \quad \forall i$$

Related Work: A Comparative Overview

Work (Author, Year)	Architecture	Algorithm / Approach	Scalability / Generalization	Problem Domain
Foundational Pointer Network & RL Models (Often Lack Generalization)				
Vinyals et al. (2015)	Pointer Network	Supervised Learning (Constructive)	Fixed-scale (Train and test on same small sizes)	TSP, Convex Hull
Bello et al. (2017)	Pointer Network	RL (REINFORCE) (Constructive)	Fixed-scale (e.g., trained on N=50, tested on N=50)	TSP, **Knapsack**
GNN-based and Hybrid Models (Often Generalize Better)				
Dai et al. (2017)	GNN (structure2vec)	RL (DQN) (Constructive)	Generalizes to unseen & larger scale graphs due to graph-based nature	MVC, MAXCUT, TSP
Cappart et al. (2021)	DRL + CP (Hybrid)	DRL learns a heuristic for a Constraint Programming (CP) solver (Improvement)	Generalizes well to new, unseen instances of various sizes	TSPTW, **Knapsack**
Advanced Transformer-based Models (Mixed Generalization)				
Kool et al. (2019)	Transformer	RL (REINFORCE) (Constructive)	Generalizes to larger scales (e.g., train N=50, test N=100), but performance may degrade	TSP, CVRP
Yildiz (2022)	Transformer/Attn	RL (DQN) (Constructive)	Fixed-scale (Performance degrades significantly on different sizes)	**Knapsack**
Que et al. (2023)	Transformer	RL (PPO) (Constructive)	Fixed-scale (Trained and tested on same N)	3D Packing
Zhang et al. (2025)	Dueling DQN	RL (Dueling DQN) with state modification (Constructive)	Fixed-scale (Trained and tested on specific small sizes)	**0/1 Knapsack**
My Work	**Custom Arch. + PPO**	**RL (PPO) (Constructive)**	**Generalizes to larger scales (Train on N, Test on >N with 70% acc.)**	**0/1 Knapsack**

My Contribution: My work addresses a key limitation of many prior models by building a **generalizable framework**. Unlike fixed-scale approaches, our model is trained to solve knapsack problems of varying sizes, including those larger than seen during training, and is supported by a powerful, integrated platform for research.

Scalability Limits of Traditional & Commercial Solvers

1. Space Complexity

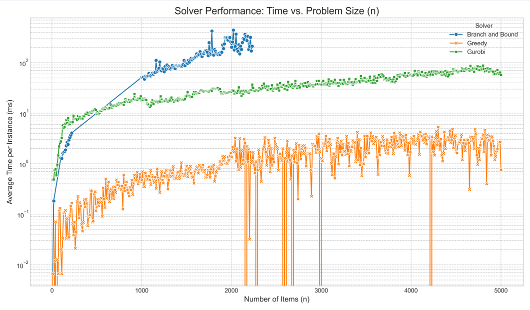


Figure: Performance degradation due to memory constraints.

- Suffer from the "curse of dimensionality".
- Leads to a **memory explosion**, making them infeasible for large-scale problems.

2. Time Complexity

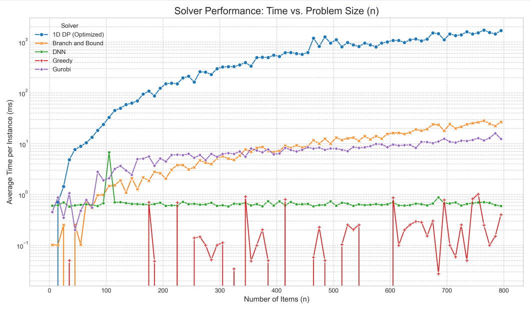


Figure: Performance comparison of various solvers.

- Runtime of Commercial Solver like Gurobi still exhibits **exponential growth**, becoming a bottleneck for very large problems.

Key RL Components for 0/1 KP

- **State (s_t):** The set of available items and the current remaining knapsack capacity.
- **Action (a_t):** The selection of one item from the available set that fits the capacity.
- **Policy ($\pi_\theta(a|s)$):** A neural network that maps the current state to a probability distribution over valid actions (items to select).
- **Reward (R_{t+1}):** The value (v_i) of the selected item.
- **Episode (τ):** A sequence of item selections, ending when no more items can be legally packed.

1. Bellman Expectation Equation (Policy-based)

Calculates the value function $\mathbf{v}^\pi$ for a **given policy** π.

$$\mathbf{v}^\pi = \mathbf{r}^\pi + \gamma \mathbf{P}^\pi \mathbf{v}^\pi$$

This is the foundation for the **Critic** in Actor-Critic methods like PPO, which evaluates the current policy.

2. Bellman Optimality Equation (Value-based)

Defines the optimal value function $\mathbf{v}^*$ by finding the best action at each state.

$$\mathbf{v}^* = \max_a \left(\mathbf{r}(a) + \gamma \mathbf{P}(a)\mathbf{v}^*\right)$$

This is the target for value-based methods like Q-Learning, which directly learn the optimal policy.

REINFORCE: Algorithm and Architecture

Training Algorithm

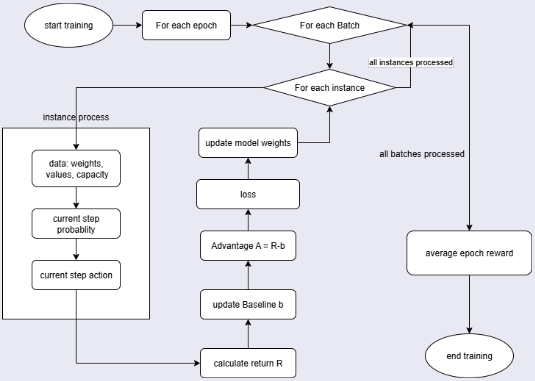


Figure: The REINFORCE training loop with an EMA baseline.

- The policy is updated based on the total return of the episode.
- A baseline is used to reduce gradient variance.

Model Architecture

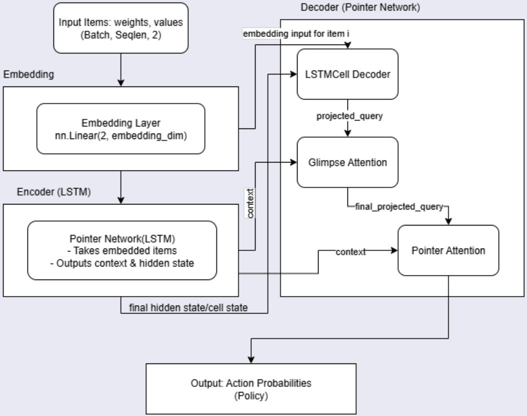


Figure: Pointer Network-based architecture for sequential item selection.

- One Actor and no Critic.

PPO: Algorithm and Architecture

Training Algorithm

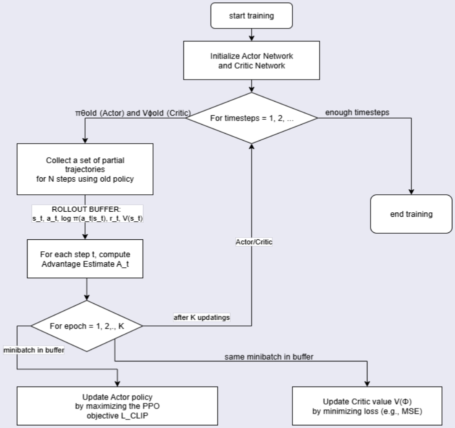


Figure: The PPO training loop using an Actor-Critic framework.

- Multiple optimization on the same minibatch.

Model Architecture

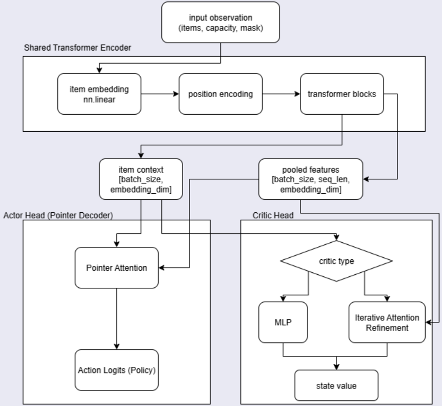


Figure: The model has two heads: one for the policy (Actor) and one for the value (Critic).

- Actor and Critic share the same encoder.

Dataset Generation and Preprocessing

Dataset Specification

We generated three distinct datasets for training, validation, and testing to ensure a robust evaluation of the model's generalization capabilities.

Parameter	**Training Set**	**Validation Set**	**Test Set**
Item Count Range (n)	5 to 50	5 to 50	**5 to 200**
Step Size	5	5	5
Instances per Size	100	30	50
Total Instances	**1,000**	**300**	**1,950**

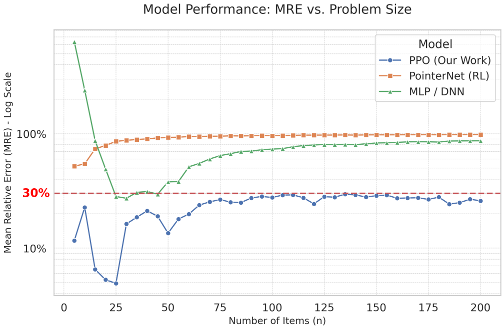
Item Properties

- Weights (w_i) and values (v_i) are integers sampled uniformly from $U[1, 100]$.
- There is no correlation between an item's weight and its value.
- All inputs are normalized before being fed to the model.

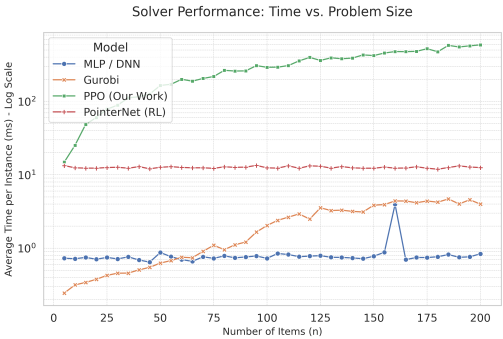
Problem Instance Constraints

- The knapsack capacity (C) is set relative to the total weight of all items ($\sum w_i$).
- The ratio $\frac{\sum w_i}{C}$ is randomly sampled from $U[0.1, 0.9]$.

Results: Accuracy and Inference Time



(a) Mean Relative Error (MRE) vs. Problem Size.



(b) Inference Time vs. Problem Size.

Key Findings: Accuracy

- Our PPO model maintains a low Mean Relative Error (MRE), demonstrating high solution quality and strong generalization.
- Pointer Network shows a higher error rate.
- The pure MLP model fails to generalize effectively.

Key Findings: Inference Time

- PPO's inference time is practical for large instances.
- Pointer Network is faster but less accurate.
- MLP is the fastest but provides poor solutions.

Conclusion

Performance Summary

- **PPO vs. Pointer Network (Accuracy):**
  - **Algorithmic Superiority:** PPO's Actor-Critic (TD) method provides low-variance updates.
  - **Architectural Advantage:** The **Transformer** encoder captures the global, combinatorial nature of the problem more effectively than a sequential **LSTM**.
  - **Framework Robustness:** Leveraging **Stable Baselines 3** provides key stabilizations like adaptive observation normalization ('VecNormalize').
- **PPO vs. Pointer Network (Speed):**
  - **Core Architecture:** **Transformer** is more computationally intensive than **LSTM**.
  - **Model Components:** Extra Critic Network requires extra computation.
  - **Evaluation Method:** Stable_baseline3 cannot support batch evaluation.

Effective Training Techniques

The success of the framework relies on several key techniques:

- **Input Normalization:**
  Normalizing item attributes (w_i, v_i) and the knapsack capacity (C) is crucial.
- **Observation & Reward Normalization:**
  Using 'VecNormalize' for both observations and rewards stabilizes the learning process significantly.
- **Heuristic Preprocessing:**
  Sorting items by value-density (v_i/w_i) before feeding them to the model provides a strong inductive bias and improves performance.

Future Work & Open Questions

Architectural Exploration

- **The "Simple Critic" Anomaly:**
  A simple MLP Critic achieved higher accuracy (70%) than a more complex attention-based head (60%). Future work should investigate if this is due to optimization challenges or a regularization effect.

- **Global State Representation (‘[CLS]‘ Token):**
  Initial experiments with a ‘[CLS]‘ token for global state representation surprisingly decreased performance. This warrants further investigation.

- **Hyperparameter Tuning:**
  While a 3-layer MLP Critic works well, its optimal width and the interplay with network depth remain open questions for further tuning.

Problem Formulation & Reward Shaping

- **Explore Alternative Formulation:** Our model uses a **"Decision" formulation** (select one from all remaining items). An alternative **"Selection" formulation** (decide 'take' or 'skip' for items sequentially) could be investigated.

- **Advanced Reward Shaping:** For the current "Decision" model, an initial attempt at adding a final shaping reward (to encourage a fuller knapsack) decreased accuracy. Further research into more advanced shaping techniques (e.g., potential-based rewards) is needed.