PAL: Pointer Network-based Schedulability Analysis

Jaeyoung Lee

Yonsei University

January 16, 2026

Part I

THE PROBLEM

Research Motivation and Problem Definition

The Problem

- Solving the priority assignment for global fixed-priority scheduling.
- Critical Challenge : Finding schedulability when state-of-the-art (SOTA) heuristic assignments on m processors fail to find a solution.

Why it's Difficult

1. **Absence of Optimality**: No known optimal priority assignment exists for multiprocessor platforms.
   $\Rightarrow$ Schedulable assignments may exist but cannot be found by current rules.
2. **Computational Complexity**: It is intractable to test all $n!$ possible priority assignments as the number of tasks n increases.

Research Goal

- Develop a Machine Learning framework to infer a schedulable priority assignment for a task set with n tasks on an m processor platform.

Core Research Questions and Design Goals

Q1: Architecture and Sample Structure

- **Input**: Sequence of task parameters.
- **Output**: Sequence of task indexes (where position determines priority).
- Neural Architecture : **Pointer Network**.

Q2: Optimal Training Sample Selection

- **Goal**: Use the single "best" training sample to infer priority assignments for other task sets.
- **Definition of "Best"**: The assignment with the smallest system hazard.
- **System Hazard**: Quantifies the safety margin and sufficiency of schedulability for a given assignment.

Q3: Scaling for Large n

- **Strategy**: Utilize pseudo-premier priority assignments (assignments where the hazard is lower than any known SOTA heuristic).
- **Mechanism**: Leverage **inductive properties** to generate high-quality pseudo-premier candidates for large n by scaling up from small-n samples.

KEY FRAMEWORK CAPABILITIES

1. Systematic Sample Generation

- Implements a high-quality data generation process by leveraging two types of inductive properties :

2. Model Training

- Trains the Pointer Network using a mix of mathematically proven Premier samples and generated Pseudo-premier samples.

3. Large-Scale Inference

- Infers a pseudo-premier priority assignment for complex task sets with large n (e.g., $n = 15$).

Part II

Background and Goal

System Model and Basic Definitions

System Assumptions

- **Task Model**: Implicit-deadline sporadic task model, where each task τ_i is characterized by (T_i, C_i).
- **Platform**: Multiprocessor platform consisting of m identical processors.

Key Definitions

- Schedulability Test for gFP : Finding sufficient conditions to determine if a task set is schedulable under a *given* priority assignment.
- Schedulable Priority Assignment : Finding *at least one* priority assignment among $n!$ possibilities that ensures the task set is schedulable on m processors.

  $\Rightarrow$ Finding such an assignment becomes significantly more difficult as the number of tasks n increases.

Existing Approaches and the Trade-off

Approach 1: Audsley's OPA

- Uses the Optimal Priority Assignment (OPA) algorithm.
- **Requirement**: Requires a specific "order-independent" test.
- **Limitation**: These tests typically offer lower performance (higher pessimism).

Approach 2: Heuristic Priority Assignments

- Uses high-performance "order-dependent" tests. (OPA incompatible)
- **Limitation**: Applied to fixed rules (like DM/RM), which may miss schedulable assignments.

The Trade-off: There is no dominance between these two; performance differs on each task set. PAL aims to discover priority assignments that heuristics miss while utilizing high-performance, OPA-incompatible tests .

Problem Definition

Problem 1: The Research Objective
Given a task set τ, for which the existing heuristic priority assignments cannot be deemed schedulable by the target schedulability test on m processors, Find at least one priority assignment for τ, which is deemed schedulable by the test on m processors.

TARGET SCHEDULABILITY TEST: RTA-LC

Lemma 1 (RTA-LC): A task set τ is schedulable by gFP with assignment $\mathcal{P}$ if $R_k \leq T_k$ for all $\tau_k \in \tau$. R_k is found via Eq. (1) using the upper bound from Eq. (4).

$$R_k = C_k + \left\lfloor \frac{\sum_{\tau_i \in H_k} I_{k \leftarrow i}(R_k)}{m} \right\rfloor \tag{1}$$

Equation (4): Interference Upper Bound

$$\sum_{\tau_i \in H_k} I_{k \leftarrow i}(I) \leq \sum_{\tau_i \in H_k} [W_i'(I)]^{I - C_k + 1} + \sum_{\tau_i \in H_k | \text{largest } m-1} \left([W_i(I)]^{I - C_k + 1} - [W_i'(I)]^{I - C_k + 1} \right)$$

Logic and Constraints

- **Notation**: $[A]^B$ denotes $\min\{A, B\}$.
- **Critical Instance**: Occurs when $m - 1$ carry-in jobs interfere with τ_k.
- **Calculation**: Interference is capped by the slack $I - C_k + 1$. The bound is the sum of all higher-priority workloads plus the $m - 1$ largest carry-in penalties.

Part III

PAL: Priority Assignment Learning Framework

Sample Structure Design

PAL utilizes supervised learning to map task parameters to priority orders using input-output pairs.

Sample Structure Definition

- **Input**: A sequence of task parameters (T_i, C_i).
- **Output**: A sequence of task indexes where the position in the sequence represents the priority level.

Example

- **Tasks**: $\tau_1(5, 1)$, $\tau_2(3, 2)$, $\tau_3(10, 5)$.
- **Desired Priority**: τ_2 (Highest) $\rightarrow \tau_1 \rightarrow \tau_3$ (Lowest).
- **Representation**:

$$\text{Input: } [[5, 1], [3, 2], [10, 5]] \rightarrow \text{ Output: } [2, 1, 3]$$

WHY POINTER NETWORKS?

The framework requires an architecture capable of interpreting output elements as an ordering of input elements.

Requirements

- Output must embody the relational meaning of task orders.
- Uses an Attention-based encoder-decoder architecture.

Key Advantages of Pointer Networks

1. **Direct Mapping**: Maps input parameters to output indexes directly.
2. **Distinct Ordering**: Enables output elements to maintain a distinct, non-repeating order.
3. **Size Flexibility**: Capable of inferring outputs for task sets of different sizes than those seen during training.

The Need for Sample Regulation

If a task set has s possible schedulable assignments, should we use all of them for training?

The "Bad Sample" Problem

- Not all schedulable assignments provide good guidelines for generalization.
- Example: Task set $\tau_1(5,2), \tau_2(5,2), \tau_3(10,1)$ on 2 processors.
  - Assignments $[1,2,3]$ and $[3,2,1]$ both work.
- Modified set (heavier): $\tau_1(5,4), \tau_2(5,4), \tau_3(10,2)$.
  - Assignment $[1,2,3]$ works, but $[3,2,1]$ **fails**.

Conclusion: We must identify and use only the "best" possible sample to train a robust model.

DEFINING THE "PREMIER" ASSIGNMENT

To identify the best training sample, PAL introduces the concept of System Hazard .

System Hazard (Θ)

- Quantifies how "sufficiently" a task set is schedulable:

$$\Theta(\tau, \mathcal{P}, m) = \max_{\tau_i \in \tau} \frac{R_i}{T_i}$$

- $\Theta \leq 1.0$ implies the set is schedulable on m processors.
- **Robustness**: Smaller hazard makes it easier to preserve schedulability if task parameters change or new tasks are added.

Definition 1: Premier Assignment

- A priority assignment is Premier if it yields the smallest possible system hazard (≤ 1.0) among all $n!$ assignments.

Solving Problem 1 via Problem 2

The ultimate goal of discovering assignments that heuristics miss can be achieved by targeting the most robust assignment possible.

Problem 1 (Research Goal)

- Given a task set where SOTA heuristics fail, find at least one schedulable priority assignment.

Problem 2 (PAL's Target)

- Given a task set on m processors, find the Premier priority assignment.

Logic: By training PAL to find the absolute best assignment (Problem 2), it gains the capability to find feasible solutions in the difficult "gap" where heuristics fail (Problem 1).

Part IV

Derivation of Inductive Properties

Challenges for Large n

The Scalability Gap

1. It is computationally intractable to generate Premier samples by exhaustively testing all priority assignments for large n.
2. It is impossible to determine with certainty whether a specific assignment is truly "Premier" without testing all $n!$ combinations.

Observation 1: Complexity Analysis
to calculate the system hazard of all priority assignments by Lemma 1

- **Exhaustive Search**: $O(n! \cdot n^2 \cdot \max T_i)$
- **Heuristic Calculation**: $O(n^2 \cdot \max T_i)$

Insight: Since we aim to solve cases where heuristics fail, we do not need to calculate every assignment. We only need an assignment that performs better than known heuristics.

PSEUDO-PREMIER PRIORITY ASSIGNMENT

Definition 2: Pseudo-premier
A priority assignment $\mathcal{P}$ for a task τ on an m-processor platform is said to be "pseudo-premier" , if $\mathcal{P}$ yields a system hazard $\Theta(\tau, \mathcal{P}, m)$, which is no larger than 1.0 and smaller than all the system hazards associated with the existing heuristic priority assignments for τ on an m-processor platform.

Problem 3: The Refined Goal

- Given a task set τ on m processors, find a pseudo-premier priority assignment for τ on m processors.
- This allows PAL to be trained on high-quality samples without requiring $n!$ searches for every data point.

Inductive Property 1 (I1) — Lemma 4

Why lemma 4 and 5
We can generate premier task set with different number of processor and number of task. This can be mathematically derived by using the idea from Lemma 1.

Lemma 4
The following inductive property makes it possible to generate a pseudo-premier sample with $(m = x, n = y + 1)$ from a pseudo-premier sample with $(m = x, n = y)$.

Inductive property 1 (I1)
Suppose we know the premier priority assignment $\mathcal{P}$ for $\tau = \{\tau_i\}_{1 \le i \le y}$ on x processors. We construct τ^+ by adding a new task τ_{y+1} to τ, and $\mathcal{P}^+$ by adding τ_{y+1} as the lowest priority to $\mathcal{P}$. If we set C_{y+1} and T_{y+1} such that $\frac{R_{y+1}}{T_{y+1}} \le \Theta(\tau, \mathcal{P}, x)$, then $\mathcal{P}^+$ is the premier priority assignment for τ^+ on x processors.

Why?
As lowest priority task on gFP does not change other tasks R_i If the newly added $\frac{R_{y+1}}{T_{y+1}}$ is smaller or equal to $\Theta(\tau, \mathcal{P}, x)$ it is still premier

Inductive Property 2 (I2) — Lemma 5

Lemma 5
The following inductive property makes it possible to generate the premier sample with $(m = x + 1, n = y + 1)$ from the premier sample with $(m = x, n = y)$.

Inductive property 2 (I2)
Suppose that we know the premier priority assignment for $\tau = \{\tau_i\}_{1 \le i \le y}$ on x processors. We construct τ^+ by adding a new task τ_{y+1}, and $\mathcal{P}^+$ by adding τ_{y+1} as the highest priority to $\mathcal{P}$. If we set $C_{y+1} \ge \max_{1 \le i \le y}(T_i - C_i + 1)$ and $T_{y+1} = \frac{C_{y+1}}{\Theta(\tau, \mathcal{P}, x)}$, then $\mathcal{P}^+$ is the premier priority assignment for τ^+ on $(x + 1)$ processors.

Why?
When added with highest priority task with max execution time to interfere other task with additionally new processor. It can be interpreted as adding new processor and run the newly added task on it. This means overall no change were made on the system.

$(R_k - C_k) \cdot x \le \text{Prev Sum} < (R_k - C_k + 1) \cdot x$

$(R_k - C_k) \cdot (x + 1) + 1 \le \text{New Sum} < (R_k - C_k + 1) \cdot (x + 1)$

$$R_k - C_k = \left\lfloor \frac{\text{Prev Sum}}{x} \right\rfloor$$

$$\left\lfloor \frac{\text{New Sum}}{x+1} \right\rfloor = R_k - C_k$$

Problem
Make highly biased task set since it only adds highest and lowest priority task.

Inductive Property 3 (I3) and 3' (I3') — Observation 2

Why Observation 2 and 3
These are Empirical properties that we found from observing PAL. These properties will be used to further enhance the inference performance in PAL.

Observation 2
The following inductive property makes it possible to generate (pseudo)premier samples with large n from (pseudo)premier samples with small n, which was observed in our experiments.

Inductive Property 3 (I3)
If PAL is trained with a sufficiently large number of (pseudo)premier samples with $(m = x, n \leq y)$, it is possible to infer a (pseudo)premier priority assignment for a task set with $(m = x, n = y + \alpha)$, with a low probability, where $\alpha \geq 1$.

Inductive Property 3' (I3')
If we add a small number of (pseudo)premier samples with $(m = x, n = y + \alpha)$ to the training set for I3, the probability of inferring a (pseudo)premier priority assignment for a task set with $(m = x, n = y + \alpha)$ increases dramatically.

Inductive Property 4 (I4) and 4' (I4') — Observation 3

Observation 3
The following inductive property makes it possible to generate (pseudo)premier samples with large m from (pseudo)premier samples with small m, which was observed in our experiments.

Inductive Property 4 (I4)
If PAL is trained with a sufficiently large number of (pseudo)premier samples with $(m = x, n \leq y)$, it is possible to infer a (pseudo)premier priority assignment for a task set with $(m = x + \beta, n = y)$, with a low probability, where $\beta \geq 1$.

Inductive Property 4' (I4')
If we add a small number of (pseudo)premier samples with $(m = x + \beta, n = y)$ to the training set for I4, the probability of inferring a (pseudo)premier priority assignment for a task set with $(m = x + \beta, n = y)$ increases dramatically.

Part V

Implementation Details

Neural Architecture and Training Settings

Model Configuration

- **Architecture**: Pointer Network.
- **Cells**: Encoder and Decoder built with LSTM cells .
- **Hidden Units**: 512 units.

Hyperparameters and Optimization

- **Learning Rate**: 0.001.
- **Batch Size**: 512.
- **Optimizer**: Adam optimizer.
- **Loss Function**: Sparse categorical cross-entropy.
- **Dataset Size**: 500,000 trained samples.

Task Set Generation Process

The Challenges

- **Issue 1**: How to generate unbiased pseudo-premier samples.
- **Issue 2**: How to make the process tractable for large n.

Task Set Filtering Conditions

1. **Infeasibility of Heuristics**: The task set should not be deemed schedulable by Lemma 1 when using standard heuristic priority assignments.
2. **Feasibility**: The task set must be schedulable by gFP with at least one priority assignment (verified by the necessary C-RTA test).
3. **Strategic Focus**: Discard sets schedulable by OPA with DA-LC to focus on cases where heuristics fail.

Task Set Generation Methodology

Utilization Distributions

- Uses 10 different distributions for U_i:
  - Binomial distribution (constants: 0.1, 0.3, 0.5, 0.7, 0.9).
  - Exponential distribution (constants: 0.1, 0.3, 0.5, 0.7, 0.9).

Generation Steps for (m, n)

1. Randomly select one of the 10 distributions.
2. Generate period $T_i \in [10, 1000]$ using a log-uniform distribution .
3. Generate U_i from the chosen distribution and calculate C_i.
4. Apply filtering conditions (Condition 1, 2, and OPA check).

Securing Large-Scale Unbiased Samples

Target Pairs (m, n)

- $(m = 2, 5 \leq n \leq 15)$.
- $(m = 4, 5 \leq n \leq 20)$.
- $(m = 6, 7 \leq n \leq 25)$.

The Hybrid Implementation Approach

- **Method 1 (Direct Derivation)**: Directly derive target-size pseudo-premier sample from an existing smaller size pseudo-premier sample

  $\Rightarrow$ Fast but biased (high/low utilization).
- **Method 2 (PAL Inference)**: Train PAL with smaller-size samples only, and infer priority assignment on the task set generated for large n. Only use the sample if it's assignment is pseudo-premier.

  $\Rightarrow$ Unbiased but slow acceptance rates at large n.

Solution: Use Method 1 to create "biased seeds" to boost PAL's performance, then use Method 2 to generate mass quantities of unbiased samples.

Step-by-Step Systematic Implementation

Example for 2 Processors ($n^* > 8$)

1. **Base ($n = 5$ to 8)**: Generate samples via exhaustive testing .
2. **Initial Augmentation**: Apply augmentation with $\rho = 5$.
3. **Scaling ($n > 8$)**: Use Method 1 to create 100 biased seeds for n using 100 unbiased samples from $n - 1$.
4. **Mass Production**: Generate $100,000/(n - 4)$ pseudo-premier samples using Method 2.
5. **Final Augmentation**: Apply augmentation with $\rho = 5$ to the new set.

Mechanism of Data Augmentation

How Augmentation is Performed

- Shuffles the order of tasks and their corresponding priorities for a given verified sample.
- **Efficiency**: Reduces the time required to generate high volumes of training data.

Why it Works for PAL

- While the underlying scheduling logic is identical, changing the input sequence forces PAL to interpret the sample differently.
- This exposes the Attention mechanism to different input/output orderings, leading to a more robust model.

Part VI

Experimental Evaluation

Evaluation Methodology

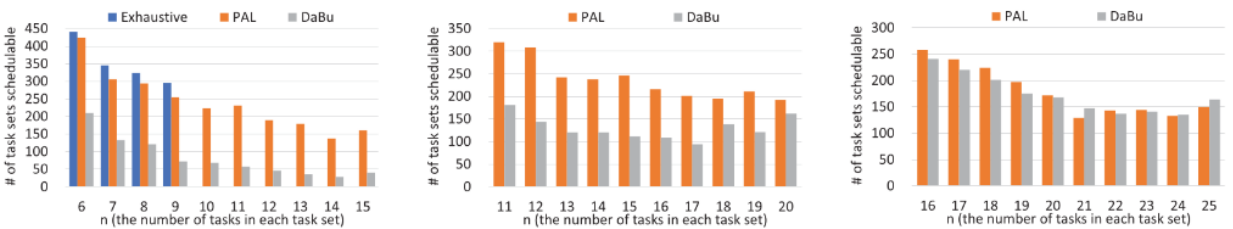
Test Dataset

- Generated 1,000 task sets for each configuration:
  - $(m = 2, 6 \leq n \leq 15)$
  - $(m = 4, 11 \leq n \leq 20)$
  - $(m = 6, 16 \leq n \leq 25)$
- **Filtering**: All sets are unschedulable by standard heuristics (Lemma 1) and OPA with DA-LC.

Comparison Targets

1. **PAL**: The proposed Pointer Network framework.
2. **DaBu**: A backtracking technique using Lemma 1.
3. **ZLL**: Heuristic priority assignment using a powerful (but slower) schedulability test.

Effectiveness in Solving Problem 1



(a) $m = 2$ with $6 \leq n \leq 15$ (b) $m = 4$ with $11 \leq n \leq 20$ (c) $m = 6$ with $16 \leq n \leq 25$

Fig. 1. The number of task sets whose schedulable priority assignment subject to Lemma 1 is identified by Exhaustive, PAL and DaBu

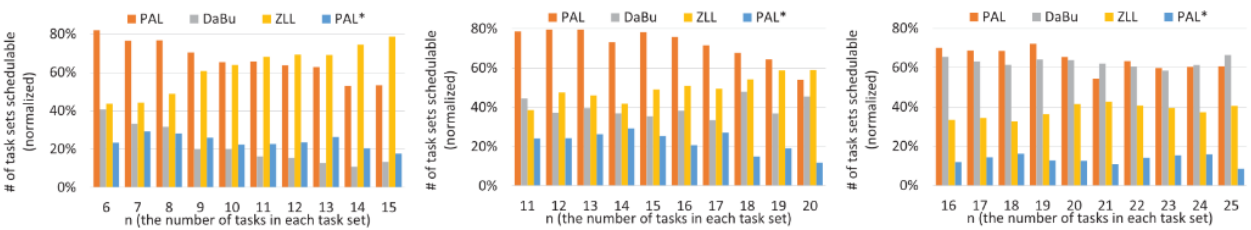
Success Ratios

- PAL identifies nearly all schedulable priority assignments for $n \leq 9$, with trends expected to hold for larger n.
- PAL dominates in finding unique assignments that other approaches (DaBu, ZLL) miss.

Comparative Analysis

- **PAL vs. DaBu**: PAL outperforms DaBu on small processor counts ($m = 2$).
- **Scaling to Large** m: DaBu tends to outperform PAL as processor counts and task counts increase.

Coverage of gFP-Schedulable Task Sets



(a) $m = 2$ with $6 \leq n \leq 15$ (b) $m = 4$ with $11 \leq n \leq 20$ (c) $m = 6$ with $16 \leq n \leq 25$

Fig. 2. The number of task sets schedulable by PAL, DaBu, ZLL and PAL*, normalized by the number of task sets schedulable by at least one of them

Unique Discovery

- PAL successfully covers a significant portion of the schedulability "gap" that was previously deemed unschedulable by existing studies.
- It consistently generates unique priority assignments not suggested by DaBu or ZLL.

Impact of the Schedulability Test

- **ZLL Performance**: Outperforms PAL on small m with very large n.
- **Reason**: ZLL utilizes a more powerful (computationally expensive) test than Lemma 1, allowing it to find schedulability even with weaker priority assignments.

Computational Complexity and Timing

Time Breakdown

1. **Sample Generation**: The most time-consuming phase.
2. **Model Training**: Requires significant one-time overhead.
3. **Inference**: Extremely fast, taking only a few seconds per task set.

Efficiency Insights

- Phases 1 and 2 only need to be executed once.
- Parallel execution of sample generation and training can further reduce implementation time.

Limitations and Potential Improvements

Limitations of the Current Framework

- **Incremental Construction**: PAL requires the incremental construction of pseudo-premier samples to scale effectively.
- **Scaling Complexity**: As the number of tasks n increases, it becomes progressively more difficult for the model to accurately infer a schedulable priority assignment.

Possible Improvements and Future Directions

- **Extended Task Models**: Adapting the framework for constrained-deadline task models , which would involve managing an increased number of input parameters per task (T_i, C_i, D_i).
- **Alternative Optimization Criteria**: Investigating other criteria beyond the minimum system hazard to define the "best" priority assignment for training.
- **Hybrid Pipelines**: Exploring a combined approach that integrates PAL's global pattern recognition with localized backtracking search techniques like DaBu.