PAGFN: Priority-Aware Attention Meets Generative Flow Networks for Global Fixed-Priority Assignment

Jaeyoung Lee

Yonsei University

February 13, 2026

Part I

PRELIMINARY

PRELIMINARY: THE CHALLENGE OF $n!$

Global Fixed Priority Assignment Problem

- **Objective**: Assign a unique priority to each task in a set τ to ensure schedulability on a multi-processor (m) platform.
- **Combinatorial Explosion**: The number of all possible priority assignments grows factorially with task count ($n!$).
- **The Problem**: Searching this space for a "feasible" assignment is computationally expensive, yet crucial for hard real-time systems.

Roadmap of Solutions

- **Classic Algorithms**: Heuristics $\rightarrow$ OPA $\rightarrow$ OPA-BK.
- **Neural Networks**: PAL $\rightarrow$ PANDA $\rightarrow$ RLRD $\rightarrow$ PAGFN.

Part II

Classic Algorithms

Classic Algorithms: Simple Heuristics

Rule-based Priority Ordering

- **Logic**: Use a simple, deterministic rule to assign priorities based on task parameters (C_i, D_i, T_i).
- **DMPO (Deadline Monotonic)**:
  - *Priority* $\propto D_i$.
  - Higher priority for tasks with shorter relative deadlines.
  - Optimal for single-processors but sub-optimal in multi-processor environments due to complex inter-task interference.
- **D-CMPO (Deadline minus Computation)**:
  - *Priority* $\propto (D_i - C_i)$.
  - Higher priority for tasks with shorter slack.
- **DkC**: *Priority* $\propto (D_i - k \cdot C_i)$.

The Gap: These rules cannot capture the intricate interference patterns of multi-processors, often failing even when a solution exists.

OPA: Audsley's Algorithm

Reducing Complexity to $O(n^2)$

- **Core Mechanism (Bottom-up Approach)**:
  1. **Start at the Bottom**: Set priority level k to the lowest.
  2. **Candidate Search**: Pick an unassigned task and test if it is schedulable at priority k.
     - Crucial Assumption : Assume all other unassigned tasks have a **higher** priority than the candidate.
  3. **Assignment**: If it passes, assign priority k to that task and move to $k - 1$.
  4. **Termination**: Success if all tasks are assigned; Failure if no unassigned task can "survive" at level k.

Significance: Guarantees an optimal result relative to the specific schedulability test (DA-LC) used.

Three Necessary Conditions for OPA

The Mathematical Foundation of Audsley's Logic

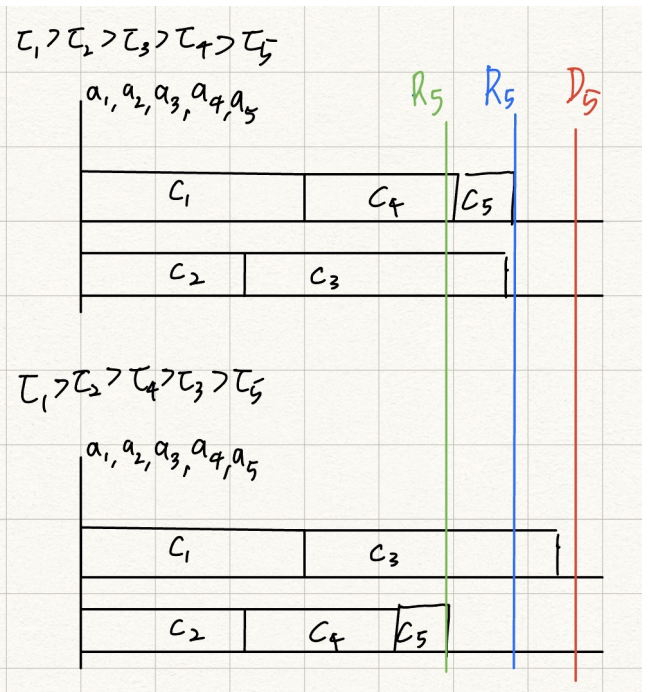
1. **Independence of Higher Priority Order**: Schedulability of task τ_k at priority k must depend only on the **set** of higher-priority tasks, not their relative ordering.
   - *Note*: RTA-LC violates this, making it directly incompatible with standard OPA.
2. **Independence of Lower Priority Tasks**: Schedulability of τ_k must be entirely independent of tasks assigned to **lower priorities** (lp(k)).
3. **Priority Monotonicity**: Moving a task to a higher priority level should not make it *unschedulable* if it was previously schedulable at a lower level.

DA-LC vs. RTA-LC

The Trade-off between Precision and OPA-Compatibility

- **DA-LC (Density Analysis)**:
  - Uses **Relative Deadline** (D_i) as an upper bound for interference.
  - **Advantage**: OPA-compatible.
  - **Disadvantage**: Highly pessimistic; fails on many valid task sets.
- **RTA-LC (Response Time Analysis)**:
  - Calculates the actual response time by accounting for carry-in and specific interference.
  - **Advantage**: High precision (Low pessimism).
  - **Disadvantage**: OPA-incompatible because interference varies based on the relative order of higher-priority tasks.

Relative Deadline vs Response Time



OPA-BK: Combining Precision with Optimal Search

Motivation: To use high-performance tests (RTA-LC) within an OPA-like framework.

- **The Process**:
  1. **Greedy Phase**: Try to assign priorities using the simpler D-RTA-LC test.
  2. **Backtracking Phase**: If no task passes, select a task satisfying a "Necessary Condition" (C-RTA) and assign it tentatively.
  3. **Validation**: Once complete, verify with the exact RTA-LC .
  4. **Recovery**: If verification fails, **backtrack** to the last tentative assignment and try another candidate.

Limitation: Worst-case complexity remains $O(n!/m!)$; requires time-outs for practical use.

Part III

ML Approach

ML Approach 1: PAL

Priority Assignment Learning through Inductive Properties

- **Sample Generation**: Uses mathematical inductive properties to generate high-quality data.
  - **Premier Samples**: Proven optimal assignments.
  - **Pseudo-premier Samples**: Assignments where System Hazard $\Theta \leq 1.0$ and performance is better than existing heuristics.
- **System Hazard Definition**:

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

- **Architecture**: Uses a Pointer Network to map task sets to priority sequences.
- **Focus**: Prioritizes data quality and training speed over complex embedding to find schedulable solutions.

MDP Formulation as Stepwise Decision Making

Stepwise Decision Process

- GFPA is treated as a sequential process where priorities are assigned one by one.
- **Sequential Assignment**: A task is selected and assigned the highest remaining priority level.

MDP Components

- **State-Based Decisions**: Every decision is made based on the current state (assigned vs. unassigned tasks).
- **Goal**: Formulate GFPA as a Markov Decision Process (MDP).
- **Objective**: Maximize the final reward (ensuring total schedulability) by optimizing individual assignment steps.

PANDA: Architecture and Input

Input Feature Engineering

- Uses augmented feature vectors instead of raw parameters (C_i, T_i, D_i).
- **Features**: Logarithmic features ($\log(1 + C_i)$), Utilization ratios (C_i/T_i, C_i/D_i), and Slack time ($T_i - C_i$, $T_i - D_i$).

Encoder-Decoder Structure

- **Encoder (Self-Attention)**: Multi-Head Self-Attention layers extract "relational features" to understand inter-task dependencies.
- **Decoder**: Attention-based recurrent network maintaining the partition of assigned (O^t) and unassigned (L^t) task sets.

PANDA: MDP AND TRAINING STRATEGY

MDP Formulation

- **State (s^t)**: Defined as (O^t, L^t).
- **Action (a^t)**: Selects one task from L^t for the highest remaining priority.

Addressing Reward Sparsity

- **Algorithm**: Uses REINFORCE to maximize expected reward.
- **Solution 1 (Progressive RTA-Reward)**: Intermediate rewards calculated via RTA at every step.
- **Solution 2 (DM-Based Guided Learning)**: Uses imitation learning based on the DM heuristic to accelerate early training.

PANDA: Deployment and Limitations

Inference Strategy

- ***k*-Sampling**: Generates k multiple assignments and checks all against schedulability tests.
- **Utilization-Based Ensemble**: Specialized models for specific utilization ranges (e.g., $[0, 0.9]$ vs. $(0.9, 1.9]$).

The Critical Limitation

- **Context Encoding**: $e_t^{(O)}$ represents assigned tasks as a set, disregarding the internal relative priority ordering.
- Identical embeddings for different sequences (e.g., $\{1, 2, 3\}$ vs $\{3, 2, 1\}$) limit hierarchical capture.

RLRD: Bridging the Latency Gap

Motivation: The Latency Problem

- Autoregressive models (PANDA) have $O(n^2)$ complexity, causing high inference latency.
- RLRD aims for low-latency, millisecond-level decision-making.

Architecture: Teacher-Student Distillation

- **Teacher**: High-performance PANDA model (Sequential RL).
- **Student**: Lightweight, non-iterative network.
  - Outputs a scalar **score** for all tasks in parallel ($O(n)$).
  - Final assignment determined by sorting these scores.

RLRD: Training and Inference

Training Mechanism: DiffRank

- **Challenge**: Sorting is non-differentiable (zero gradients).
- **Solution**: Differentiable Approximated Ranking (DiffRank) relaxes discrete sorting into a continuous function.

Inference: Efficient Sequence Sampling

- **Gumbel-Top-k**: Adds Gumbel noise to predicted scores and re-sorts.
- **Efficiency**: Generates k candidates with log-linear complexity, avoiding quadratic costs of iterative sampling.

m	N	Util	OPA		RL		RL-S		RD		RD-G	
			Ratio	Time	Ratio	Time	Ratio	Time	Ratio	Time	Ratio	Time
4	32	3.0	78.1%	0.3531s	87.5%	0.0616s	89.4%	0.0697s	86.5%	0.0145s	90.1%	0.0263s
		3.1	63.5%	0.3592s	74.8%	0.0599s	77.6%	0.0840s	73.8%	0.0139s	78.9%	0.0390s
		3.2	44.9%	0.3487s	56.9%	0.0623s	60.1%	0.0991s	56.0%	0.0140s	61.2%	0.0509s
		3.3	26.6%	0.3528s	35.7%	0.0621s	38.5%	0.9123s	35.8%	0.0131s	39.2%	0.0620s
6	48	4.4	84.2%	0.4701s	92.5%	0.1021s	94.3%	0.1153s	91.76%	0.0298s	94.3%	0.0406s
		4.6	61.9%	0.4600s	78.4%	0.1057s	78.4%	0.1508s	74.6%	0.0308s	79.3%	0.0728s
		4.8	33.2%	0.4287s	46.5%	0.1082s	50.4%	0.1967s	45.4%	0.0290s	50.8%	0.1123s
		5.0	11.5%	0.3888s	15.7%	0.1010s	18.1%	0.2474s	18.0%	0.0256s	20.2%	0.1615s
8	64	5.7	92.9%	0.6686s	97.8%	0.1437s	98.6%	0.1596s	97.5%	0.0502s	98.5%	0.0537s
		6.0	72.9%	0.6460s	86.5%	0.1490s	89.9%	0.2043s	85.0%	0.0364s	88.7%	0.0907s
		6.3	37.6%	0.5798s	53.5%	0.1559s	57.5%	0.2800s	52.5%	0.0509s	57.7%	0.1695s
		6.6	10.4%	0.4806s	15.1%	0.1488s	17.7%	0.4093s	17.0%	0.0390s	19.6%	0.2715s

Part IV

Current SOTA Approach: PAGFN

PAGFN: Core Motivation

Goal: Solving GFPS on Multiprocessors

- Specifically targets cases where traditional heuristic algorithms and standard Reinforcement Learning (RL) fail.

The Problem with Previous Models

- **Context Limitation**: Existing models encode assigned tasks as an unordered "set," failing to distinguish the relative hierarchy among tasks already assigned.
- **Mode Collapse**: Standard RL (REINFORCE) tends to converge to a single deterministic local optimum.
- **Reward Sparsity**: Valid solutions are sparse in priority assignment; RL struggles without heavy, potentially biased reward shaping.

Architecture: Priority-Aware Encoder-Decoder

Input Feature Engineering

- Augments input with Logarithmic Features ($\log(1 + C_i)$), Ratio Features (C_i/T_i), and Difference Features ($T_i - C_i$) to grasp task hardness.

Priority-Aware Module

- Explicitly encodes priority index using Sine-Cosine Position Encoding.
- **Assigned Tasks**: Summed with their specific priority embedding.
- **Unassigned Tasks**: Summed with a learnable parameter v_{null}.

Cross-Attention Decoder

- Uses the "Priority-Aware" vector as the **Query** and task features as the **Key**/**Value**.
- Allows selecting the next task specifically based on the current priority level.

Comparison: PANDA vs. PAGFN Architecture

Structural Evolution

- Both use an Encoder-Decoder structure.
- **PANDA**: Uses self-attention between assigned and non-assigned sets.
- **PAGFN**: Uses cross-attention with the explicit priority level.

Training Strategy: Generative Flow Networks

GFlowNet Algorithm

- Objective: Learn a policy where the probability of generating a sequence is proportional to its reward: $P(x) \propto R(x)$.
- Encourages the exploration of all valid solutions rather than converging on one.

Trajectory Balance (TB) Loss

- Enforces consistency across the entire generation path.
- Ensures the model learns how early actions contribute to final schedulability.
- Does not use reward shaping (like progressive RTA-Reward), avoiding potential local optima bias.

Two-Stage Training Strategy

1. **Expert-Guided Pretraining**
   - Pre-trains by imitating the Deadline Monotonic (DM) heuristic.
   - Establishes a "good enough" starting policy for initial exploration.
2. **Reward-Prioritized Replay**
   - Saves high-reward trajectories in a buffer and replays them.
   - Prevents "forgetting" rare valid solutions discovered during exploration.

Inference Optimization and Setup

State-Invariant Strategy

- Exploits multiprocessor scheduling property: on m processors, the top m highest-priority tasks are mathematically equivalent.
- PAGFN encodes the first m tasks with the same priority index.
- Drastically reduces the search space for the neural network.

Experimental Environment

- **Hardware**
  - Intel Xeon Silver 4210R CPU @10core/20thread @3.2GHZ
  - NVIDIA GeForce RTX 3090 GPU
- **Dataset**: Both implicit and constrained deadline task sets (differentiating it from previous methods).

Experimental Results: Implicit Deadlines

TABLE III: Comparison for implicit deadline task sets with varying (m, n, u).

m	n	u	Schedulability Ratio (SR)							
			DM	D-CM	DkC	OPA	PANDA	RLRD	PAGFN	OPA-bk
4	32	3.0	71.0	79.9	81.6	78.5	87.1	86.5	87.8	96.0
		3.1	57.5	68.9	71.6	63.9	74.6	73.8	76.1	87.4
		3.2	39.2	49.6	52.9	44.2	57.0	55.9	57.8	78.0
		3.3	23.1	30.0	33.2	27.3	35.6	35.8	37.3	57.6
6	48	4.4	61.2	74.6	81.8	84.4	90.5	89.8	92.7	98.5
		4.6	38.7	51.6	59.5	61.6	70.9	68.2	74.7	92.5
		4.8	17.3	26.3	33.3	33.2	40.1	39.3	45.6	73.6
		5.0	4.8	9.2	12.1	11.4	14.0	15.1	17.1	36.7
8	64	5.8	48.9	66.4	77.6	87.5	94.8	94.6	95.5	99.7
		6.1	25.9	40.9	50.9	60.6	76.4	76.6	78.7	95.9
		6.4	7.1	15.1	21.6	28.4	38.5	38.7	40.5	72.4
		6.7	1.5	3.1	4.6	6.4	8.3	8.0	9.1	25.7

Experimental Results: Constrained Deadlines

TABLE IV: Comparison for constrained deadline task sets with varying (m, n, u).

m	n	u	Schedulability Ratio (SR)							
			DM	D-CM	DkC	OPA	PANDA	RLRD	PAGFN	OPA-bk
4	32	2.0	44.1	76.3	81.8	83.8	81.1	79.1	85.3	88.5
		2.1	39.3	70.6	76.8	78.9	78.6	76.2	80.1	84.3
		2.2	32.3	62.6	69.3	70.9	71.2	70.1	73.2	78.9
		2.3	26.7	54.6	61.6	62.8	63.3	62.7	65.9	71.2
6	48	3.0	26.7	54.6	61.6	62.8	63.3	62.7	88.6	94.2
		3.2	22.1	59.4	78.2	80.6	80.7	79.5	81.9	89.0
		3.4	14.8	45.1	65.6	68.0	67.8	65.8	70.2	78.4
		3.6	8.5	30.7	49.3	51.3	51.9	50.7	53.7	68.0
8	64	4.0	19.4	63.2	90.3	92.3	89.1	88.7	93.3	95.5
		4.3	12.1	47.4	80.3	82.6	78.9	78.4	84.0	90.9
		4.6	6.2	31.1	63.6	66.6	63.8	63.0	68.7	78.8
		4.9	2.5	16.7	41.5	45.2	43.2	43.3	45.9	60.1

Experimental Results: Runtime Comparison

TABLE VI: Runtime Comparison

Model	Implicit deadline (m, n, u)			Constrained deadline (m, n, u)		
	(4, 32, 3.2)	(6,48, 5.0)	(8, 64, 6.0)	(4, 32, 2.2)	(6, 48, 3.2)	(8, 64, 4.8)
DMPO	0.0079	0.0118	0.0221	0.0033	0.0050	0.0068
D-CMPO	0.0085	0.0125	0.0243	0.0046	0.0079	0.0115
DkC	0.0087	0.0134	0.0270	0.0058	0.0121	0.0206
OPA	0.3528	0.4198	0.6440	0.2671	0.3756	0.5392
PANDA	0.0618	0.1106	0.1590	0.0575	0.1043	0.1480
RLRD	0.0142	0.0293	0.0367	0.0117	0.0266	0.0328
PAGFN	0.1013	0.1801	0.3129	0.1078	0.2115	0.2817
OPA-bk	5.6969	20.4322	36.5127	1.3223	5.4198	12.0788

Efficiency Comparison

TABLE VII: Schedulability Ratio of OPA-bk with Different Timeouts *vs* PAGFN

Model	Implicit deadline (m, n, u)	Constrained deadline (m, n, u)
	(8, 64, 6.0)	(8, 64, 4.8)
1s Timeout	57.2	48.7
0.5s Timeout	20.5	5.9
PAGFN	76.9	51.7

Part V

Discussion & Suggestions

5 pages excluded