Supervised Dispatching for Identical Parallel Machine Total Tardiness Scheduling

Daniel Zhu[a], Christos T. Maravelias[a,b*]

[a] *Chemical and Biological Engineering, Princeton University, Princeton, NJ 08540, USA*

[b] *Andlinger Center for Energy and the Environment, Princeton University, 86 Olden St., Princeton, NJ 08540, USA*

** Corresponding Author: maravelias@princeton.edu*

Supplemental Material

S1. Input Features and SetDispatch Workflow

Section 3.2 of the main manuscript describes the SetDispatch neural policy at a high level. This section provides the complete input feature set used in the identical-machine experiments and illustrates the policy architecture and computational workflow schematically.

All time-based quantities are normalized by an instance-level time scale $\Lambda = \sum_i p_i \,/m$ before entering the neural network, where $\sum_i p_i$ is the total processing load and m is the machine count. This normalization makes the feature magnitudes comparable across instances of different sizes. Fixed job features are computed once per instance; dynamic job features, rule-based features, machine-state features, and global state features are recomputed at each dispatching state $\mathcal{S}_\ell$ as the schedule evolves. The complete feature set is listed in **Table S1**.

Table S1. Input features used by the SetDispatch policy, grouped by type. $\bar{p}$ is the mean processing time over unscheduled jobs, $C_i = \max(t, r_i) + p_i$ is the completion time of job i if dispatched next, n is the number of jobs, m is the number of machines, j^* is the freed machine, **E** is the eligible set, and **U** is the unscheduled set. Variables r_i, d_i, p_i, t, and q_j are defined in **Section 2** and **3** of the main manuscript. Fixed job features are standardized to zero mean and unit variance within each instance before entering the encoder.

Type	Description	Formula
Fixed job features	Release time, scaled	$r_i \,/\, \Lambda$
	Due time, scaled	$d_i \,/\, \Lambda$
	Processing time, scaled	$p_i \,/\, \Lambda$
	Initial slack s_i [14]	$(d_i - r_i - p_i) \,/\, \Lambda$
	Due-window-to-processing ratio w_i [14]	$\frac{(d_i - r_i)}{p_i}$
	Normalized release rank ρ_i^P [14]	$rank(r_i) \,/\, (n-1)$
	Normalized due rank ε_i^P [14]	$rank(d_i) \,/\, (n-1)$
Dynamic Job Features	Available at t	$1[r_i \le t]$
	Time elapsed since release	$max(0, t - r_i) \,/\, \Lambda$
	Wait remaining until eligible	$max(0, r_i - t) \,/\, \Lambda$
	Slack if dispatched next	$(d_i - C_i) \,/\, \Lambda$
	Tardiness if dispatched next	$max(0, C_i - d_i) \,/\, \Lambda$
	Completion time if dispatched next	$C_i \,/\, \Lambda$
Rule Based Features	Release-gated ATC priority	Eq. (17)
	Modified due date index	$max(d_i, t + p_i) \,/\, \Lambda$
	PRTT index [7]	Eq. (18)
	ATC ordering among eligible jobs	rank of ATC over **E** (high → low)
	MDD ordering among eligible jobs	rank of MDD over **E** (low → high)
	PRTT ordering among eligible jobs	rank of PRTT over **E** (low → high)
	Eligibility mask	$1[i \in \mathbf{E}]$
Machine State Features	Time machine j next becomes free	$q_j \,/\, \Lambda$
	Availability relative to current time	$(q_j - t) \,/\, \Lambda$
	Machine currently idle	$1[q_j \le t]$
	Load ordering across machines	$rank(q_j) \,/\, (m-1)$
	Machine being filled this step	$1[j = j^*]$
Global State Features	Scaled decision time	$t \,/\, \Lambda$
	Fraction of jobs eligible	$\lvert\mathbf{E}\rvert \,/\, n$
	Fraction of jobs unscheduled	$\lvert\mathbf{U}\rvert \,/\, n$
	Fraction of jobs completed	$1 - \lvert\mathbf{U}\rvert \,/\, n$

Figure S1 provides a general schematic of the SetDispatch dispatching workflow, included for clarity.

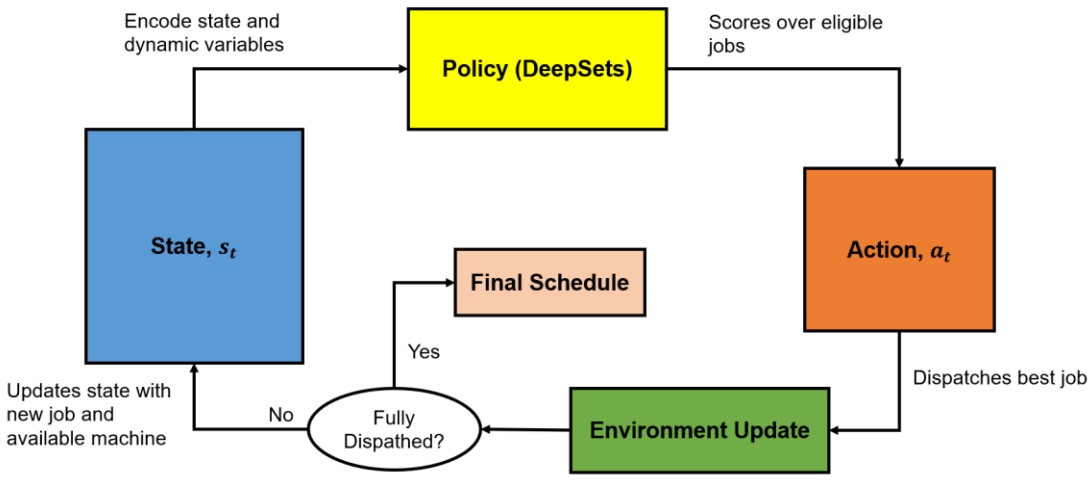

Figure S1. Sequential dispatching loop executed by SetDispatch at inference time.

S2. Full Numerical Gap Results

Table S2 reports the mean normalized tardiness gap γ for all methods across all tested combinations of instance type, machine count, and job size. Each cell is averaged over 100 test instances. The reported metric is

$$\gamma = 100 \cdot \frac{TT(\pi) - TT^*}{TT^*}$$

where $TT(\pi)$ is the total tardiness of the dispatched schedule and TT^* is the MIP optimal value. All reported instances satisfy $TT^* > 0$. MDD is included here for completeness. In the main manuscript, MDD is omitted from **Figure 3** because its gaps on Types B and C are orders of magnitude larger than the other methods and would obscure the scale of the figure. The full MDD results confirm that its global machine-scanning logic performs reasonably on Type A — where it is competitive with and sometimes beats NDPRTT — but degenerates severely under staggered release structures, producing gaps in the thousands of percent on Types B and C.

Table S2. Full mean gap to MIP (%) for all identical-machine test cases. Each cell is the mean over 100 test instances with $TT^* > 0$. Bold entries indicate the best-performing method in each row.

Type	M	N	SetDispatch	ATC	NDPRTT	MDD
A	2	30	**3.73**	17.76	5.99	5.59
		60	**1.51**	9.47	4.76	5.07
		80	**1.63**	6.88	3.99	3.87
		100	**2.03**	6.36	3.78	4.07
		120	**2.15**	5.42	3.76	3.73
	3	30	**6.92**	29.68	10.70	7.36
		60	**2.28**	15.40	6.58	5.50
		80	**2.99**	12.66	5.31	4.56
		100	**3.10**	10.36	5.02	4.44
		120	**3.99**	8.99	4.65	4.47
B	2	30	**29.01**	56.99	33.06	4599.80
		60	**13.59**	45.33	27.09	5624.06
		80	**13.12**	34.68	22.39	5461.93
		100	**16.98**	37.22	22.24	6114.36
		120	19.96	31.69	**18.24**	6399.38
	3	30	**12.29**	28.33	21.19	2602.31
		60	**8.29**	28.34	21.01	4074.24
		80	**8.89**	27.94	18.97	4710.21
		100	**11.05**	27.61	21.23	5376.26
		120	**12.89**	23.88	17.94	5663.54
C	2	30	**3.69**	10.62	5.60	339.70
		60	**0.89**	6.83	4.26	304.93
		80	**1.20**	6.38	5.01	165.57
		100	**1.42**	5.86	5.48	203.52
		120	**1.89**	6.07	5.18	41.68
	3	30	**9.54**	37.88	16.37	1280.39
		60	**3.20**	29.13	13.40	2183.26
		80	**3.88**	26.30	13.02	3810.31
		100	**4.47**	22.05	10.27	2034.52
		120	**5.65**	21.66	9.99	2143.89

S3. Code and Data Availability

The computational results and methodologies presented in this work are reproducible and available for further academic research. Source code for SetDispatch training, dispatch label generation, release-gated heuristic baselines, result reconstruction, and figure generation is available at:

https://github.com/dz5430/SetDispatch-Identical-PMS

The GitHub repository contains the code, summary spreadsheets, and plotting inputs. Large instance CSV files, MIP schedules, dispatch-label pickle files, trained model checkpoints, and decoded output CSVs are stored separately on Google Drive and should be copied into the repository data/ and models/ folders following the manifest provided in the repository.