Appendix

Scheduling — ILP formulation

Symbol	Meaning
Sets and indices	
$G = \{g_1, \ldots, g_N\}$	Set of task groups (size N)
$\mathcal{M} = \{m_1, \ldots, m_K\}$	Set of machines (size K)
$\kappa(m)$	Type of machine m
$\mu_{g,j}$	Required type for task (g, j)
$j,\ n_g$	Task index in group g; number of tasks in g
Parameters	
$p_{g,j} > 0$	Processing time of (g, j)
$d_{g,1} = 0,\ d_{g,j} \geq 0$	Fixed gap between consecutive tasks
$t_g^\star \geq 0$	Preferred start of group g
$\overline{T}$	Planning horizon (for big-M)
$c_g \geq 0$	Coefficient for *LinearPenalty*
Decision variables	
t_g	Start time of group g
$s_{g,j}$	Start time of task (g, j)
$x_{g,j,m} \in \{0, 1\}$	Assign (g, j) to machine m
$y_{(g,j),(g',j'),m} \in \{0, 1\}$	Order of two tasks on the same m
$e_g^+, e_g^- \geq 0$	Deviation s.t. $t_g - t_g^\star = e_g^+ - e_g^-$
$z_g^L, z_g^U \geq 0$	Hinges for *LinearWithRangePenalty*
Auxiliary / complexity	
M	Big-M constant with $M \geq \overline{T}$
$J_g := n_g$	Tasks in group g
$N_{\text{task}} = \sum_g J_g,\ N_{\text{event}} = 2\, N_{\text{task}}$	Total tasks; number of events
$c_{\max} = \max_\kappa c_\kappa$	Max machines among types
$B_g,\ S_k$	Offset trials for g; scheduled tasks before trial k

Sets and indices. Task groups $G = \{g_1, \ldots, g_N\}$. Machines $\mathcal{M} = \{m_1, \ldots, m_K\}$, each with a type $\kappa(m)$. For $g \in G$, tasks are indexed by $j = 1, \ldots, n_g$ and require type $\mu_{g,j}$.

Parameters. Processing times $p_{g,j} > 0$; fixed inter-task gaps $d_{g,1} = 0$, $d_{g,j} \geq 0$ $(j \geq 2)$; preferred start $t_g^\star \geq 0$. Define cumulative offsets

$$o_{g,1} = 0,$$
$$o_{g,j} = \sum_{\ell=1}^{j-1} (p_{g,\ell} + d_{g,\ell+1}) \ (j \geq 2).$$

Planning horizon $\overline{T}$ (for big-M). For penalties: *LinearPenalty* coefficient $c_g \geq 0$; *LinearWithRangePenalty* uses $\texttt{lower}_g$, $\texttt{upper}_g$ and slopes $\texttt{lower\_coefficient}_g$, $\texttt{upper\_coefficient}_g$, defined relative to $t_g^\star$ as in the paper.

Decision variables. Group starts $t_g \geq 0$; task starts $s_{g,j}$. Assignment binaries $x_{g,j,m} \in \{0, 1\}$; ordering binaries $y_{(g,j),(g',j'),m} \in \{0, 1\}$. For *LinearPenalty*: $e_g^+, e_g^- \geq 0$ with $t_g - t_g^\star = e_g^+ - e_g^-$. For *LinearWithRangePenalty*: hinges $z_g^L, z_g^U \geq 0$. (Optional) Allowed-window selection $w_{g,h} \in \{0, 1\}$.

Objective. Choose terms group-wise according to the penalty type. *LinearPenalty*:

$$\min \sum_{g \in G} c_g \, (e_g^+ + e_g^-).$$

LinearWithRangePenalty (let $\Delta_g := t_g - t_g^\star$):

$$\min \sum_{g \in G} \Big(\texttt{lower\_coefficient}\, z_g^L + \texttt{upper\_coefficient}\, z_g^U \Big).$$

NonePenalty contributes zero.

Constraints. *Task-start definition (matches the recursive form in the paper):*

$$\begin{aligned} s_{g,1} &= t_g, \\ s_{g,j} - s_{g,j-1} &= p_{g,j-1} + d_{g,j} \quad (j = 2, \dots, n_g). \end{aligned}$$

Type-feasible unique assignment:

$$x_{g,j,m} = 0 \;\text{ if }\; \kappa(m) \neq \mu_{g,j}, \qquad \sum_{m \in \mathcal{M}:\, \kappa(m) = \mu_{g,j}} x_{g,j,m} = 1, \quad \forall g, j.$$

Non-overlap on each machine (big-M): for any m and distinct $(g,j) \neq (g',j')$,

$$\begin{aligned} s_{g,j} + p_{g,j} &\leq s_{g',j'} + M\Big(1 - y_{(g,j),(g',j'),m}\Big) + M\big(2 - x_{g,j,m} - x_{g',j',m}\big)\,, \\ s_{g',j'} + p_{g',j'} &\leq s_{g,j} + M\, y_{(g,j),(g',j'),m} + M\big(2 - x_{g,j,m} - x_{g',j',m}\big)\,, \end{aligned}$$

with $M \geq \overline{T}$ chosen safely.

Within-group precedence: the order is implicit from the start-definition constraints above (no extra constraint needed).

Across-group precedence (optional, induced by the DFA): if $(u,v) \in \mathcal{P}$ requires that v starts after u finishes,

$$t_v \;\geq\; s_{u,n_u} + p_{u,n_u} \quad .$$

LinearPenalty linearisation:

$$t_g - t_g^\star = e_g^+ - e_g^-, \qquad e_g^+, e_g^- \geq 0.$$

LinearWithRange hinges: with $\Delta_g = t_g - t_g^\star$,

$$z_g^L \geq \texttt{lower} - \Delta_g, \quad z_g^L \geq 0, \qquad z_g^U \geq \Delta_g - \texttt{upper}, \quad z_g^U \geq 0.$$

(Optional) hard blackout windows. If allowed windows for g are precomputed as $\{[A_{g,h}, B_{g,h}]\}_h$ over the horizon, add binaries $w_{g,h} \in \{0,1\}$ with

$$\sum_h w_{g,h} = 1, \qquad t_g \geq \sum_h A_{g,h} w_{g,h}, \qquad t_g \leq \sum_h B_{g,h} w_{g,h}.$$

Greedy scheduling — complexity

Implementation note. Let $N_{\text{task}} = \sum_{g \in G} n_g$ denote the total number of tasks and $N_{\text{event}} = 2\, N_{\text{task}}$ the number of start/finish events. In our implementation, the conflict check rescans scheduled events at each offset trial; thus the greedy initialiser runs in $O(N_{\text{task}}^2)$ in the worst case. After start times are fixed, first-fit machine assignment over the time-ordered event list is $O(N_{\text{event}})$.

Computational complexity of the greedy scheduler (as implemented)

Let $J_g = n_g$ be the number of tasks in group g, and let $N_{\text{task}} = \sum_{g \in G} J_g$ denote the total number of tasks (hence the number of start/finish events is $N_{\text{event}} = 2\,N_{\text{task}}$). For machine types, write $c_{\max} = \max_\kappa c_\kappa$ for the largest number of parallel machines of any type.

Placement loop. Groups are processed in (is_not_started, $t_g^\star$) order. For each group g the code tries integer offsets $\Delta \in \{0, 1, -1, 2, -2, \dots\}$ until no conflict is found, setting

$$\begin{aligned} t_g &= \max(t_{\text{ref}},\, t_g^\star + \Delta), \\ s_{g,1} &= t_g, \\ s_{g,j} - s_{g,j-1} &= p_{g,j-1} + d_{g,j} \quad (j \geq 2). \end{aligned}$$

Let B_g be the number of offset trials used for g, and let S_k be the number of tasks already scheduled before the k-th trial (over all groups). In each trial the code (i) recomputes the group's internal start times in $O(J_g)$ and (ii) checks conflicts by re-scanning all previously scheduled tasks, which costs $O(S_k)$. Hence the k-th trial is $O(J_g + S_k)$.

Time for placement. Summing over groups and their trials,

$$\begin{aligned} \sum_{g \in G} \sum_{k=1}^{B_g} O(J_g + S_k) &= O\Big(\sum_g B_g J_g\Big) \;+\; O\Big(\sum_k S_k\Big) \\ &= O(N_{\text{task}}^2) \;\; \text{the worst case.} \end{aligned}$$

The quadratic term arises because each trial re-scans the set of already scheduled tasks, and these sets grow along the loop (i.e. a double sum over prefixes of $\{J_g\}$).

Machine assignment. After times are fixed, the code builds and sorts the list of $N_{\text{event}} = 2\,N_{\text{task}}$ events in $O(N_{\text{event}} \log N_{\text{event}}) = O(N_{\text{task}} \log N_{\text{task}})$ time and performs a first–fit sweep. For starts/finishes of type κ, it linearly scans the c_κ machines of that type; writing $N_{\text{event},\kappa}$ for the number of events of type κ,

$$\text{first–fit time} \;=\; O\Big(\sum_\kappa N_{\text{event},\kappa}\, c_\kappa\Big) \;\leq\; O(N_{\text{event}}\, c_{\max}) \;=\; O(N_{\text{task}}\, c_{\max}).$$

$$\boxed{\begin{aligned} \text{time} &= O(N_{\text{task}}^2) + O(N_{\text{task}} \log N_{\text{task}}) + O(N_{\text{task}}\, c_{\max}) \\ &= O(N_{\text{task}}^2 + N_{\text{task}}\, c_{\max}) \end{aligned}}$$

with the $O(N_{\text{task}}^2)$ placement dominating in the worst case when $c_{\max} \leq N_{\text{task}}$. The space usage is

$$\boxed{\text{space} \;=\; O(N_{\text{task}} + |\mathcal{M}|)}.$$