КЫРГЫЗ-ТҮРК
МАНАС УНИВЕРСИТЕТИ

Atatürk
University

Stochastic Parallel Machine Scheduling under Processing Time Uncertainties

Selçuk Gören

Kyrgyz-Turkish Manas University

TURK2025

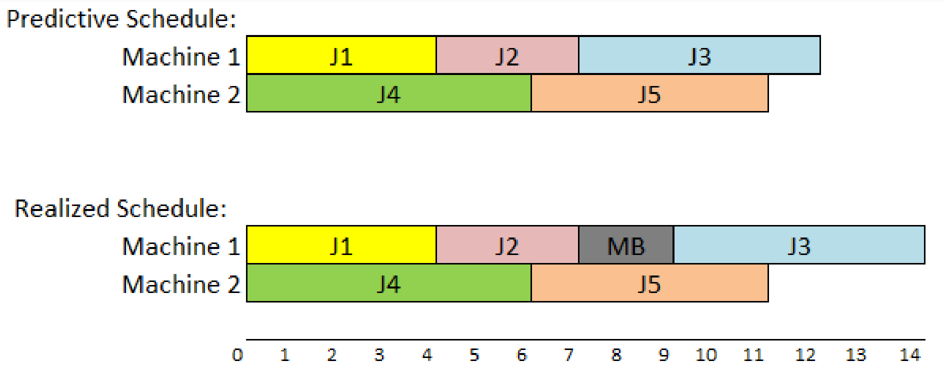
Presentation Plan

- Introduction
- Problem Definition
- Proposed Method (main idea)
- Algorithms
- Results and Conclusion

TURK2025

Introduction

- Scheduling: assigning resources (machines) to tasks (jobs)
- Objective functions: makespan, tardiness, completion time, earliness
- Literature is rich in deterministic problems
  - Processing times
  - Release dates
  - Due dates



Problem Definition

- $Pm||C_{\max}$
- m identical parallel machines
- The objective is to minimize makespan

$$\min z$$

$$s.t.$$

$$z \geq \sum_{j=1}^{n} p_j x_{ij} \qquad \forall i$$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad \forall j$$

$$x_{ij} \in \{0,1\} \qquad \forall i, j$$

Problem Definition

- $\mathbb{E}[f(X)] \neq f(\mathbb{E}[X])$
- Scenario-based stochastic version
- Deterministic equivalent

$$\min \sum_{s=1}^{3^n} z_s \, \alpha_s$$

$$z_s \geq \sum_{j=1}^{n} p_{js} x_{ij} \qquad \forall i, s$$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad \forall j$$

$$x_{ij} \in \{0,1\} \qquad \forall i, j$$

Dual Decomposition: Nonanticipativity

TURK2025

- Create copies of here-and-now first-stage variables x_{ij} as x_{ijs}

$$\min \sum_{s=1}^{3^n} z_s\, a_s$$

$$s.t. \qquad z_s \geq \sum_{j=1}^{n} p_{js} x_{ijs} \qquad \forall i, s$$

$$\sum_{i=1}^{m} x_{ijs} = 1 \qquad \forall j, s$$

$$\boxed{\left(\sum_{s=2}^{3^n} a_s\right) x_{ij1} = \sum_{s=2}^{3^n} a_s x_{ijs} \qquad \forall i, j}$$

$$x_{ijs} \in \{0,1\} \qquad \forall i, j, s$$

Dual Decomposition: Lagrangian Relaxation

TURK2025

- Dualize nonanticipativity constraints to get the Lagrangian relaxation

$$Z_{LR} = \min \sum_{s=1}^{3^n} z_s\, a_s + \sum_{j=1}^{n} \sum_{i=1}^{m} \lambda_{ij} \left(\sum_{s=2}^{3^n} a_s \left(x_{ij1} - x_{ijs} \right) \right)$$

$$s.t. \qquad z_s \geq \sum_{j=1}^{n} p_{js} x_{ijs} \qquad \forall i, s$$

$$\sum_{i=1}^{m} x_{ijs} = 1 \qquad \forall j, s$$

$$x_{ijs} \in \{0,1\} \qquad \forall i, j, s$$

Dual Decomposition: Voilà

- Now each scenario has its own sub problem:

Sub Problem with $s = 1$:

$$Z_{LR_1} = \min a_1 z_1 + \sum_{j=1}^{n} \sum_{i=1}^{m} \lambda_{ij} (1 - a_1) x_{ij1}$$

$$s.t. \quad z_1 \geq \sum_{j=1}^{n} p_{j1} x_{ij1} \qquad \forall i$$

$$\sum_{i=1}^{m} x_{ij1} = 1 \qquad \forall j$$

$$x_{ij1} \in \{0,1\} \qquad \forall i, j$$

Dual Decomposition: Voilà

- Now each scenario has its own sub problem:

Sub Problem with fixed $s \neq 1$:

$$Z_{LR_s} \min a_s z_s - \sum_{j=1}^{n} \sum_{i=1}^{m} \lambda_{ij} \; a_s x_{ijs}$$

$s.t.$

$$z_s \geq \sum_{j=1}^{n} p_{js} x_{ijs} \qquad \forall i$$

$$\sum_{i=1}^{m} x_{ijs} = 1 \qquad \forall j$$

$$x_{ijs} \in \{0,1\} \qquad \forall i, j$$

Dual Decomposition: Combining Solutions

- We can solve each sub problem separately and then

$$Z_{LR(\lambda)} = \sum_{s=1}^{3^n} Z_{LR_s}$$

- We have a small problem: Any set of λ_{ij}'s provides just a lower bound!

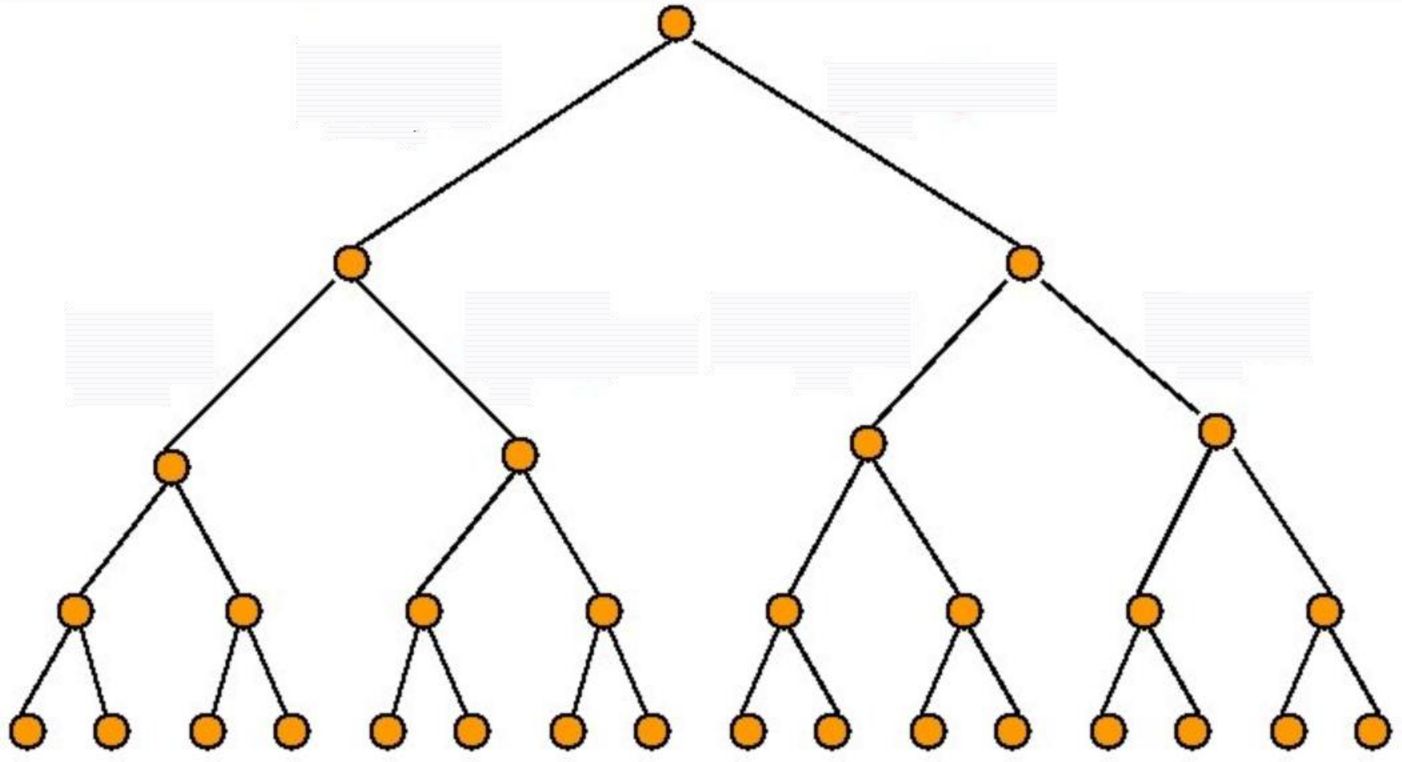
- OK, then let's find the best λ_{ij}'s.

TURK2025

Subgradient Search Algorithm

- Initialization: $\lambda_{ij} = \lambda_{ij}{}^{0}$
- Iteration k: $\lambda_{ij} \leftarrow \lambda_{ij}{}^{k}$
- Decompose and solve $Z_{LR}\left(\lambda_{ij}{}^{k}\right)$ with solution $x\left(\lambda_{ij}{}^{k}\right)$ and the objective function value $z(\lambda_{ij}{}^{k})$
- $\lambda_{ij}{}^{k+1} = \max\{ \left(\lambda_{ij}{}^{k} - \mu_k \cdot \left(\sum_{s=2}^{3^n} a_s (x_{ij1} - x_{ijs})\right)\right), 0\}$
- $k \leftarrow k+1$
- We don't need to have strong duality! This is only a heuristic.

Branch-and-Bound Algorithm



Branch-and-Bound Algorithm

- Step 1: Set solution of the stochastic problem $\underline{z} = +\infty$ and denote L as the node set in the search tree
- Step 2: Terminate if there is no node in the node set L. The incumbent solution $\hat{x}$ that yields objective value $\underline{z}$ is optimal.
- Step 3: Select a node P from L, solve the z_{LD} of P ($z_{LD}(P)$) by subgradient search and delete it from the problem set. Go to Step 2 if P is infeasible.

Branch-and-Bound Algorithm

- Step 4: If $\underline{z} \leq z_{LD}(P)$ go to Step 2. Else,

- If the nonanticipativity holds (i.e. solution is feasible), update $\underline{z}$ by $\underline{z} = z_{LD}(P)$ and delete all the problems P' with $z_{LD}(P') \geq \underline{z}$, return back to Step 2.

- If the nonanticipativity does not hold, compute average $\overline{x^i}$, if $0 \leq \overline{x^i} \leq 0.5$, set x^i as 0, and 1 otherwise. Check feasibility. If feasible, then compute objective function z of this new problem and update $\underline{z}$ by $\underline{z} = \min\{\underline{z}, z\}$ and delete all the problems with higher Lagrangian dual values than $\underline{z}$, Continue with Step 5.

- Step 5. To branch, compute the average $\overline{x^i}$ values, select the most non-integer $\overline{x^i}$ (the closest one to 0.5) and add two new problems with new constraints $x^i = 0$ and $x^i = 1$. Continue with Step 2.

Computational Tests

Expected Processing Time (EPT)	**U[1,100]**
Processing Time Variation	Low: {EPT * 0.8 ,EPT, EPT *1.2} Medium : {EPT * 0.5 ,EPT, EPT *1.5} High : {EPT * 0.2 ,EPT, EPT *1.8}
Total number of jobs (n)	10
Machine number (m)	3,5

TURK2025

Computational Tests

			Direct Model		Dual Decomposition		
n	m	Variation	GAP	Incumbent Solution	GAP	Incumbent Solution	Lower Bound
10	3	Medium	5%	216,67	14%	216,461	186
10	3	Medium	15%	213,64	14%	207,652	177,7
10	3	Medium	8%	175,74	15%	175,303	149,4
10	3	High	5%	236,28	21%	235,561	186,7
10	3	High	75%	528,36	21%	226,603	178,6
10	3	High	25%	198,09	21%	191,65	150,5
10	5	Low	79%	552,36	10%	133,314	119,7
10	5	Low	6%	122,92	6%	122,092	114,5
10	5	Low	79%	445,36	7%	102,948	96,01
10	5	Medium	93%	552,36	19%	148,529	120,4
10	5	Medium	5%	141,18	15%	140,765	120,2
10	5	Medium	47%	187,49	17%	119,074	99,33
10	5	High	75%	552,36	25%	170,767	127,8
10	5	High	35%	220,30	17%	160,734	133,3
10	5	High	75%	445,36	18%	135,897	111

Future Research Directions

- Number of scenarios increases exponentially with n
  - A clever way to sample among all possible scenarios
  - How to handle continuous processing time distributions?
- Maybe we don't need the tightest lower bound at each node in the B&B tree
  - Only invoke subgradient search when necessarry
- How to handle large problems with many machines and many jobs?
  - Heuristics

Thank you!