Single machine scheduling in additive manufacturing with two-dimensional packing constraints

Kan Fang

Tianjin University, China

Zhaofang Mao, Enyuan Fu, Dian Huang, Lin Chen

Global Scheduling Seminar

October 4, 2023
• We consider a single machine scheduling in additive manufacturing with two-dimensional packing constraints (SMSAM-2DP)

• We develop an approximation algorithm and a combinatorial Benders decomposition algorithm (Algorithm-CBD) to solve the problem

• Algorithm CBD performs well
1. Introduction
2. Problem description
3. Approximation algorithm
4. Combinatorial Benders decomposition algorithm (Algorithm CBD)
5. Computational experiments
Additive manufacturing (AM), commonly known as 3D printing, uses 3D digital model files to create objects layer-by-layer.

Advantages of additive manufacturing:
- shorten the product development cycle
- reduce material loss
- create complex geometries without molds

Additive manufacturing market size is expected to rise from USD 16.72 billion in 2022 to reach a value of USD 76.16 billion by 2030, at a compound annual growth rate (CAGR) of 20.8%.

An important part of the fourth industrial revolution (Attaran, 2017).
• Disadvantages of additive manufacturing
  • The slow speed of the process
  • High cost of equipment and materials
  • The need for pre- and post-processing (cleaning, sintering, heat treatment, etc.)

• Some AM technologies allow different parts to be processed simultaneously in the same batch
  • e.g., selective laser melting (SLM), also known as direct metal laser sintering technology (DMLS)

• We focus on the DMLS technology (parts are not allowed to be vertically stacked)
Introduction

- The production process of SLM/DMLS

![Diagram of SLM/DMLS process]

**Figure 1:** From Li et al. (2017)

- **Pre-processing operations** (data preparation, filling of powder materials, adjustment of AM machine, filling up protective atmosphere)
- **Powder layering and laser melting:** generate thin powder layers (typical thickness between 20\(\mu m\) to 60\(\mu m\)), and scan the powder material by a high power laser beam
- **Post-processing operations:** clean machine, replace filters
Introduction

- The production time of a batch is affected by the set of parts allocated to this batch
  - The maximum height of parts that affects the powder layering iterations
  - The total volume of parts that affects the scanning and layer fabrication of parts
  - Machine setup time

- The production time of a batch is a weighted sum of the above three factors (Li et al., 2017; Kucukkoc, 2019; Altekin and Bukchin, 2022)
1 Introduction

2 Problem description

3 Approximation algorithm

4 Combinatorial Benders decomposition algorithm (Algorithm CBD)

5 Computational experiments
SMSAM-2DP problem: Parameters

- Set of parts \( I = \{1, 2, \ldots, n\} \), each part \( i \in I \) has
  - a predetermined orientation
  - length \( \ell_i \)
  - width \( w_i \)
  - height \( h_i \)
  - volume \( v_i \)

- The additive machine has
  - length \( L (\ell_i \leq L) \)
  - width \( W (w_i \leq W) \)
  - height \( H (h_i \leq H) \)
  - scanning time per unit volume \( VT \)
  - recoating time per unit height \( HT \)
  - setup time between any two batches \( ST \)
SMSAM-2DP problem: Objective

• To minimize the makespan
  • The geometry of each part is projected on the XY plane, and the *minimum rectangle limits* is used to place the part in the building chamber
  • A batch is *feasible* if there is no overlap between the rectangular bounding boxes of any two parts
  • Once a batch is started to process parts, it cannot be interrupted until its completion
  • The makespan is equal to the completion time of the last batch in the schedule
Problem description

SMSAM-2DP problem: Decision variables

- **Assignment of parts into batches**
- **Position of parts in each batch**
  - \((x_i, y_i)\): the coordinates of the front-left corner of part \(i\)
  - \(z_b\): 1 if batch \(b\) is opened, 0 otherwise
  - \(u_{ib}\): 1 if part \(i\) is allocated into batch \(b\), 0 otherwise
  - \(\text{left}_{ijb}\): 1 if part \(i\) is located left of part \(j\) in batch \(b\), 0 otherwise
  - \(\text{below}_{ijb}\): 1 if part \(i\) is located behind part \(j\) in batch \(b\), 0 otherwise
  - \(h_b\): height of batch \(b\)
  - \(C_b\): completion time of batch \(b\)
  - \(C_{\text{max}}\): makespan
SMSAM-2DP problem: Constraints

1. Each part $i$ must be allocated to exactly one batch

\[ \sum_{b \in B} u_{ib} = 1 \quad \forall i \in I \]

2. The height of each batch must be greater than the height of each part in this batch

\[ h_i \cdot u_{ib} \leq h_b \quad \forall i \in I, \ b \in B \]

3. Each part cannot be placed outside the machine’s platform in both horizontal (width) or vertical (length) directions

\[ x_i + w_i \leq W + M \cdot (1 - u_{ib}) \quad \forall i \in I, \ b \in B \]

\[ y_i + \ell_i \leq L + M \cdot (1 - u_{ib}) \quad \forall i \in I, \ b \in B \]
SMSAM-2DP problem: Constraints

4. If two parts $i$ and $j$ are allocated into the same batch, they are not allowed to overlap with each other

$$\text{left}_{ijb} + \text{left}_{jib} + \text{below}_{ijb} + \text{below}_{jib} \geq u_{ib} + u_{jb} - 1 \quad \forall i, j \in I, b \in B$$

$$x_i + w_i - M \cdot (2 - u_{ib} - u_{jb}) \leq x_j + M \cdot (1 - \text{left}_{ijb}) \quad \forall i, j \in I, b \in B$$

$$y_i + \ell_i - M \cdot (2 - u_{ib} - u_{jb}) \leq y_j + M \cdot (1 - \text{below}_{ijb}) \quad \forall i, j \in I, b \in B$$

5. Batch $b$ is opened if at least one part is allocated to this batch

$$\sum_{i \in I} u_{ib} \leq M \cdot z_b \quad \forall b \in B$$

$$z_b \leq \sum_{i \in I} u_{ib} \quad \forall b \in B$$
SMSAM-2DP problem: Constraints

6 A batch can be opened only if its previous batch has already been opened

\[ \sum_{i \in I} u_{i(b+1)} \leq M \cdot \sum_{i \in I} u_{ib} \quad \forall b \in B \setminus \{n\} \]

7 Completion time of each batch

\[ C_b \geq C_{b-1} + VT \sum_{i \in I} v_i \cdot u_{ib} + HT \cdot h_b + ST \cdot z_b \quad \forall b \in B \]

8 Calculation of the makespan

\[ C_{\max} \geq C_b \quad \forall b \in B \]
Problem description

- Contribution to the literature: Additive manufacturing scheduling
- nearly 30 papers in 2016-2023

<table>
<thead>
<tr>
<th>Reference</th>
<th>Problem type</th>
<th>Constraint</th>
<th>Objective</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freens et al. (2016)</td>
<td>S</td>
<td>SM</td>
<td>Min. cost function</td>
<td>MILP</td>
</tr>
<tr>
<td>Kucukkoc et al. (2016)</td>
<td>S</td>
<td>RM</td>
<td>Min. production costs</td>
<td>MILP+Heuristic</td>
</tr>
<tr>
<td>Kim et al. (2017)</td>
<td>S</td>
<td>PM/PA</td>
<td>Min. makespan</td>
<td>MILP+GA</td>
</tr>
<tr>
<td>Ransikarbhum et al. (2017)</td>
<td>OAS</td>
<td>RM</td>
<td>Multiobjective</td>
<td>MILP</td>
</tr>
<tr>
<td>Li et al. (2017a)</td>
<td>S</td>
<td>RM</td>
<td>Min. average production costs</td>
<td>MILP+ Heuristic</td>
</tr>
<tr>
<td>Oh et al. (2018c)</td>
<td>NS</td>
<td>R</td>
<td>Min. cycle time</td>
<td>Heuristic</td>
</tr>
<tr>
<td>Dvorak et al. (2018)</td>
<td>NS</td>
<td>RM</td>
<td>Min. makespan, tardiness</td>
<td>CP+Heuristic</td>
</tr>
<tr>
<td>Kucukkoc et al. (2018)</td>
<td>S</td>
<td>RM</td>
<td>Min. maximum lateness</td>
<td>GA</td>
</tr>
<tr>
<td>Fera et al. (2018)</td>
<td>S</td>
<td>SM</td>
<td>Min. lateness/earliness costs</td>
<td>GA</td>
</tr>
<tr>
<td>Chergui et al. (2018)</td>
<td>NS</td>
<td>PM</td>
<td>Min. tardiness</td>
<td>MILP</td>
</tr>
<tr>
<td>Li et al. (2018)</td>
<td>OAS</td>
<td>RM</td>
<td>Max. profit</td>
<td>MILP</td>
</tr>
<tr>
<td>Griffiths et al. (2019)</td>
<td>NS</td>
<td>SM/BO</td>
<td>Min. build costs</td>
<td>ITSP</td>
</tr>
<tr>
<td>Stein et al. (2019)</td>
<td>OAS</td>
<td>RM</td>
<td>Max. revenue</td>
<td>MILP</td>
</tr>
<tr>
<td>Reference</td>
<td>Problem type</td>
<td>Constraint</td>
<td>Objective</td>
<td>Method</td>
</tr>
<tr>
<td>-------------------------</td>
<td>--------------</td>
<td>------------</td>
<td>------------------------------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>Kucukkoc (2019)</td>
<td>S</td>
<td>R</td>
<td>Min. makespan</td>
<td>MILP</td>
</tr>
<tr>
<td>Li et al. (2019b)</td>
<td>OAS</td>
<td>R</td>
<td>Max. profit</td>
<td>MILP</td>
</tr>
<tr>
<td>Wang et al. (2019)</td>
<td>NS</td>
<td>R</td>
<td>Max. nesting rate</td>
<td>Heuristic</td>
</tr>
<tr>
<td>Luzon and Khmelnitsky (2019)</td>
<td>S</td>
<td>SM, F</td>
<td>Min. exp. makespan, flowtime</td>
<td>Queueing theory</td>
</tr>
<tr>
<td>Fera et al. (2020)</td>
<td>S</td>
<td>SM</td>
<td>Min. lateness/earliness costs</td>
<td>TS</td>
</tr>
<tr>
<td>Zhang et al. (2020)</td>
<td>NS</td>
<td>R</td>
<td>Min. makespan</td>
<td>EA</td>
</tr>
<tr>
<td>Kim and Kim (2020)</td>
<td>S</td>
<td>P/PA/SU</td>
<td>Min. makespan</td>
<td>MILP</td>
</tr>
<tr>
<td>Alicastro et al. (2021)</td>
<td>S</td>
<td>SM</td>
<td>Min. makespan</td>
<td>ILS</td>
</tr>
<tr>
<td>Che et al. (2021)</td>
<td>NS</td>
<td>PM/BO</td>
<td>Min. makespan</td>
<td>MILP+SA</td>
</tr>
<tr>
<td>Kapadia et al. (2021)</td>
<td>OAS</td>
<td>PM/BO</td>
<td>Max. profit</td>
<td>GA</td>
</tr>
<tr>
<td>Rohaninejad et al. (2021)</td>
<td>S</td>
<td>R</td>
<td>Min. weighted tardiness</td>
<td>Hybrid GA, LS</td>
</tr>
<tr>
<td>Altekin et al. (2021)</td>
<td>S</td>
<td>R</td>
<td>Multiobjective</td>
<td>MILP+Pareto</td>
</tr>
<tr>
<td>Aloui and Hadj-Hamou (2021)</td>
<td>NS</td>
<td>R</td>
<td>Min. total lateness</td>
<td>MILP+Heuristic</td>
</tr>
<tr>
<td>Kucukkoc et al. (2021)</td>
<td>NS</td>
<td>R</td>
<td>Min. total tardiness</td>
<td>GA</td>
</tr>
<tr>
<td>Zipfel et al. (2021)</td>
<td>NS</td>
<td>PM</td>
<td>Min. total weighted tardiness</td>
<td>ILS</td>
</tr>
<tr>
<td>Altekin and Bukchin (2022)</td>
<td>NS</td>
<td>RM</td>
<td>Min. makespan</td>
<td>MILP</td>
</tr>
<tr>
<td>Lee and Kim (2023)</td>
<td>NS</td>
<td>RM</td>
<td>Min. makespan</td>
<td>MILP+GA, PSO</td>
</tr>
<tr>
<td>Hu et al. (2022)</td>
<td>NS</td>
<td>RM/BO</td>
<td>Min. makespan</td>
<td>MILP+ALNS</td>
</tr>
</tbody>
</table>
Problem description

Problem type:
- S-Scheduling
- NS-Nesting & scheduling
- OAS-Order Acceptance and Scheduling

Objective:
- Min. cost
- Min. makespan
- Min. tardiness/lateness
- Max. profit
- Multiobjective

Constraint:
- SM-Single Machine
- PM-(identical) Parallel Machines
- RM-Unrelated (parallel) machines
- PA-Processing Alternatives
- SU-Set-Ups
- F-Failures
- BO-Build Orientation

Method:
- MILP
- Heuristics: GA, TS, SA, EA, LS...
- CP, Pareto
- Approximation Algorithm
- Exact Algorithm
1 Introduction

2 Problem description

3 Approximation algorithm

4 Combinatorial Benders decomposition algorithm (Algorithm CBD)

5 Computational experiments
• In any optimal schedule, there must be no unforced idleness between any two consecutive batches

• Let $P_b$ be the processing of batch $b$, then the total processing time of all batches $P$ is

\[
P = \sum_{b \in B} P_b = VT \sum_{i \in I} v_i + HT \sum_{b \in B} h_b + ST \sum_{b \in B} z_b
\]

- total scanning time
- total recoating time
- total setup time

• The optimal makespan only depends on the total recoating time and the total setup time
• Suppose \( \sigma^* \) is an optimal schedule, in which the total number of batches opened is \( t \)

• We assume that \( h_1^* \geq h_2^* \geq \cdots \geq h_t^* \), where \( h_k^* \) is the height of batch \( k \) (\( k = 1, \ldots, t \))

• \( C_{\text{max}}(\sigma^*) = VT \sum_{i \in I} v_i + HT \cdot \sum_{k=1}^{t} h_k^* + ST \cdot t \)
Approximation algorithm

- Divide all the parts into three groups

\[ I_1 = \left\{ i : w_i \leq \frac{1}{2} W \quad \& \quad \ell_i \leq \frac{1}{2} L \right\}, \]

\[ I_2 = \left\{ i : w_i > \frac{1}{2} W \right\}, \]

\[ I_3 = \left\{ i : w_i \leq \frac{1}{2} W \quad \& \quad \ell_i > \frac{1}{2} L \right\}. \]

- Let \( n_i \) be the number of parts in group \( I_i \) \((i = 1, 2, 3)\)

- Sort the parts in each group in nonincreasing order of their heights

- Denote \( j^i_k \) as the \( k \)th part in group \( I_i \) \((h_{j^i_1} \geq h_{j^i_2} \geq \cdots \geq h_{j^i_{n_i}})\)
Approximation algorithm: Algorithm GreedyPack

1: Initialize: $\tilde{A} \leftarrow 0, \tilde{L} \leftarrow 0, \tilde{W} \leftarrow 0.$
2: for $i = 1$ to $3$ do
3:    if $I_i \neq \emptyset$ then
4:        Open a new batch so as to pack the parts for each $I_i$. Let $s_i \leftarrow 1.$
5:    end if
6: end for
7: for $k = 1$ to $n_1$ do
8:    $\tilde{A} \leftarrow \tilde{A} + w_{j_k^1} \cdot \ell_{j_k^1}.$
9:    if $\tilde{A} \leq \frac{1}{2} WL$ then
10:       Put part $j_k^1$ into the current batch.
11:    else
12:       Close the current batch. Open a new batch and put part $j_k^1$ into the new batch.
13:       $s_1 \leftarrow s_1 + 1, \tilde{A} \leftarrow w_{j_k^1} \cdot \ell_{j_k^1}.$
14:    end if
15: end for
16: for $k = 1$ to $n_2$ do
17:    $\tilde{L} \leftarrow \tilde{L} + \ell_{j_k^2}.$
18:    if $\tilde{L} \leq L$ then
19:       Put part $j_k^2$ into the current batch.
20:    else
21:       Close the current batch. Open a new batch and put part $j_k^2$ into the new batch.
22:       $s_2 \leftarrow s_2 + 1, \tilde{L} \leftarrow \ell_{j_k^2}.$
23:    end if
24: end for
25: for $k = 1$ to $n_3$ do
26:    $\tilde{W} \leftarrow \tilde{W} + w_{j_k^3}.$
27:    if $\tilde{W} \leq W$ then
28:       Put part $j_k^3$ into the current batch.
29:    else
30:       Close the current batch. Open a new batch and put part $j_k^3$ into the new batch.
31:       $s_3 \leftarrow s_3 + 1, \tilde{W} \leftarrow w_{j_k^3}.$
32:    end if
33: end for
Let $\overline{w} = \max_{i \in \tilde{I}} w_i$, $\overline{\ell} = \max_{i \in \tilde{I}} \ell_i$, $A = \sum_{i \in \tilde{I}} w_i \ell_i$, $x_+ = \max(x, 0)$.

**Theorem (Steinberg 1997)**

If $\overline{w} \leq W$, $\overline{\ell} \leq L$, $2A \leq WL - (2\overline{w} - W)_+ (2\overline{\ell} - L)_+$, then it is possible to pack all the parts in $\tilde{I}$ into the rectangle with width $W$ and length $L$.

- For group $I_1$, since $w_i \leq \frac{1}{2} W$ and $\ell_i \leq \frac{1}{2} L$, the inequalities in Steinberg’s Theorem must hold, and the packing solution for $I_1$ is feasible.
- For groups $I_2$ and $I_3$, it is trivial to see that their packing solutions are feasible.
- Algorithm GreedyPack can provide a feasible packing solution for all the parts.
Approximation algorithm

- \( s_i \): the number of batches opened for each group \( I_i \)
- Denote \( h^i_k \) as the height of the \( k \)th batch in group \( I_i \)

**Lemma**

\[ s_1 \leq 4t \]

**Proof.**

- A new batch can be opened only if \( \tilde{A} + w_i \ell_i > \frac{1}{2} WL \)
- The total area of parts in any two consecutive batches must be at least \( \frac{1}{2} WL \)
- The total area of parts in \( I_1 \) is at least \( s_1/2 \cdot \frac{1}{2} WL \), and is at most \( t \cdot WL \)
- \( \frac{1}{4}s_1 \cdot WL \leq t \cdot WL \Rightarrow s_1 \leq 4t \)
Lemma

For any $k \geq 0$, we have $h_{4k-3}^1 \leq h_k^* \Rightarrow \sum_{k=1}^{s_1} h_k^1 \leq 4 \cdot \sum_{k=1}^{t} h_k^*$

Proof.

• when $k = 1$, obviously true as $h_1^*$ must be the largest height
• We have $h_1^1 \geq \cdots \geq h_{4k-5}^1 \geq h_{4k-4}^1 \geq h_{4k-3}^1 \geq \cdots$
• The total area of the first $4k - 4$ batches must be at least $2(k - 1) \cdot \frac{1}{2} WL = (k - 1) WL = (k - 1) WL$
• The parts in the first $4k - 4$ batches cannot be fully packed into $k - 1$ batches in the optimal schedule $\Rightarrow$ must exist one part $i'$ in the first $4k - 4$ batches that will be packed into a batch between batches $k$ and $t$ in the optimal schedule
• $h_{i'} \geq h_{4k-3}^1 \Rightarrow h_k^* \geq h_{i'} \geq h_{4k-3}^1$
Approximation algorithm

Lemma

\[ s_2 \leq 2t \quad \text{and} \quad s_3 \leq 2t \]

Proof.

- Any of two parts in \( I_2 \) can only be packed together if their total length is not greater than \( L \)
- A new batch needs to be opened only when \( \tilde{L} + \ell_i > L \), where \( \tilde{L} \) is the total length of parts in the current batch
- \( \Rightarrow \) The total area of parts in any two consecutive batches must be at least \( WL \)
- \( \Rightarrow \) The total area of parts in \( I_2 \) is at least \( \frac{s_2}{2} WL \)
- \( \Rightarrow \) \( \frac{s_2}{2} \cdot WL \leq t \cdot WL \Rightarrow s_2 \leq 2t \)
- Similar results hold for group \( I_3 \)
**Approximation algorithm**

**Lemma**

For any $k \geq 0$, we have $h_{2k-1}^2 \leq h_k^*$ and $h_{2k-1}^3 \leq h_k^*$

\[ \Rightarrow \sum_{k=1}^{s_2} h_k^2 \leq 2 \cdot \sum_{k=1}^{t} h_k^*, \text{ and } \sum_{k=1}^{s_3} h_k^3 \leq 2 \cdot \sum_{k=1}^{t} h_k^* \]

**Proof.**

- when $k = 1$, obviously true
- $h_1^2 \geq \cdots \geq h_{2k-3}^2 \geq h_{2k-2}^2 \geq h_{2k-1}^2 \geq \cdots$
- The total area of the first $2k - 2$ batches must be at least $(k - 1) \cdot WL$
- $\Rightarrow$ The parts in the first $2k - 2$ batches cannot be fully packed into $k - 1$ batches in the optimal schedule $\Rightarrow$ must exist one part $i'$ that will be packed into a batch between $k$ and $t$ in the optimal schedule $h_{i'} \geq h_{2k-1}^2 \Rightarrow h_k^* \geq h_{i'} \geq h_{2k-1}^2$
- Similar results hold for group $l_3$
Approximation algorithm

Theorem

The approximation ratio of Algorithm GreedyPack is at most 8

Proof.

- Denote σ as the schedule generated by Algorithm GreedyPack, and \( C_{\text{max}}(\sigma) \) be the corresponding makespan of this schedule

\[
C_{\text{max}}(\sigma) = VT \sum_{i \in I} v_i + HT \cdot \left( \sum_{k=1}^{s_1} h_k^1 + \sum_{k=1}^{s_2} h_k^2 + \sum_{k=1}^{s_3} h_k^3 \right) + ST \cdot \sum_{i=1}^{3} s_i
\]

\[
\leq VT \sum_{i \in I} v_i + HT \cdot 8 \sum_{k=1}^{t} h_k^* + ST \cdot 8t
\]

\[
\leq 8 \cdot \left( VT \sum_{i \in I} v_i + HT \cdot \sum_{k=1}^{t} h_k^* + ST \cdot t \right)
\]

\[
= 8 \cdot C_{\text{max}}(\sigma^*) \quad \text{(quite loose!)}
\]
1 Introduction

2 Problem description

3 Approximation algorithm

4 Combinatorial Benders decomposition algorithm (Algorithm CBD)

5 Computational experiments
• Classical Benders decomposition algorithm: (Benders, 1962; Rahmaniani et al., 2017)
  • Given a MILP $P : \min \{cy + dx : Ay + Bx \geq b, y \geq 0, x \in X\}$
  • The Benders decomposition algorithm first fixes $x \in X$, then solves the slave problem $SP : \min \{cy : Ay \geq b - Bx, y \geq 0\}$, which can be solved by means of the dual slave problem $SD : \max \{u(b - Bx) : uA \leq c, u \geq 0\}$
  • If $SD$ has an optimal solution $\bar{u}$, then an optimality cut $z \geq \bar{u}(b - Bx)$ is constructed
  • If $SD$ is unbounded, a feasibility cut $0 \geq \bar{u}(b - Bx)$ is formed
  • When some variables in the subproblems are required to be integer, standard duality theory cannot be applied to derive the classical Benders cuts
Combinatorial Benders decomposition algorithm

- Combinatorial Benders decomposition algorithm (Codato and Fischetti, 2006)
  - Do not use the dual information to generate cuts
  - It can handle problems where the MP is a 0-1 integer program and the subproblem is a feasibility problem ($c = 0$)
  - The slave problem $SP$ can be used as a feasibility check on the system $\{Ay + B\bar{x} \geq b, y \geq 0\}$
  - If $\bar{x}$ is not a feasible solution for at least one variable $x_j$ causing infeasibility, then this variable must take a different value from $x_j$
  - If $\bar{x}$ is a feasible solution for $SP$, then it is feasible and optimal for $P$
Combinatorial Benders decomposition algorithm

- Schematic of CBD

![Schematic of CBD](image)

**Figure 2**: From Li et al. (2022)

- Numerous applications of CBD
  - Cutting and packing problems: Cote et al. (2014); Cote et al. (2021)
  - Assembly line balancing problems: Akpınar et al. (2017); Huang et al. (2022); Sikora and Weckenborg (2022)
  - Scheduling problems: Verstichel et al. (2015); Li et al. (2022)
Combinatorial Benders decomposition algorithm

- Decompose our problem into the following master and slave problems:
  - **The master problem**: determine the allocation of parts into batches without the two-dimensional packing constraints
  - **The slave problems**: determine the existence of feasible packing solutions for the allocated parts in each batch
- If the packing solution is infeasible for some slave problem, generate combinatorial Benders cuts to forbid the current allocation plan of parts, and add such cuts to the master problem
- Continue such process until all slave problems become feasible, and the solution of the master problem become optimal to the original problem
The master problem

\[ \begin{align*}
\text{[master]} & \quad \min & C_{\text{max}} \\
\text{s.t.} & & \sum_{b \in B} u_{ib} = 1 & \quad \forall i \in I \quad (1a) \\
& & h_i \cdot u_{ib} \leq h_b & \quad \forall i \in I, b \in B \quad (1b) \\
& & \sum_{i \in I} w_i \ell_i \cdot u_{ib} \leq W \cdot L & \quad \forall b \in B \quad (1c) \\
& & \sum_{i \in I} u_{ib} \leq M \cdot z_b & \quad \forall b \in B \quad (1d) \\
& & z_b \leq \sum_{i \in I} u_{ib} & \quad \forall b \in B \quad (1e) \\
& & \sum_{i \in I} u_{i(b+1)} \leq M \cdot \sum_{i \in I} u_{ib} & \quad \forall b \in B \setminus \{n\} \quad (1f) \\
& C_b \geq C_{b-1} + VT \cdot \sum_{i \in I} v_i \cdot u_{ib} + HT \cdot h_b + ST \cdot z_b & \quad \forall b \in B \quad (1g) \\
& C_{\text{max}} \geq C_b & \quad \forall b \in B \quad (1h) \\
u_{ib}, z_b \in \{0, 1\} & \quad \forall i \in I, b \in B \quad (1i) \\
h_b, C_b \geq 0 & \quad \forall b \in B \quad (1j) 
\end{align*} \]
The slave problems

- Let \( S = \{ u_{ib}^*, z_b^* \} \) be the solution of the master problem, and \( C_{max}^* \) be the corresponding makespan

- Denote \( \bar{I}_b = \{ i \in I | u_{ib}^* = 1 \} \) as the set of parts allocated into batch \( b \)

\[
\begin{align*}
[\text{slave}(b)] & \quad \min & 0 \\
\text{s.t.} & & x_i + w_i \leq W & \forall i \in \bar{I}_b \\
& & y_i + \ell_i \leq L & \forall i \in \bar{I}_b \\
& & \text{left}_{ij} + \text{left}_{ji} + \text{below}_{ij} + \text{below}_{ji} \geq 1 & \forall i, j \in \bar{I}_b, i \neq j \\
& & x_i + w_i \leq x_j + W \left( 1 - \text{left}_{ij} \right) & \forall i, j \in \bar{I}_b, i \neq j \\
& & y_i + \ell_i \leq y_j + L \left( 1 - \text{below}_{ij} \right) & \forall i, j \in \bar{I}_b, i \neq j \\
& & \text{left}_{ij}, \text{below}_{ij} \in \{0, 1\} & \forall i, j \in \bar{I}_b, i \neq j \\
& & x_i, y_i \geq 0 & \forall i \in \bar{I}_b
\end{align*}
\]
The algorithmic outline of Algorithm CBD

**Algorithm 1** The algorithmic outline of Algorithm CBD

1: Initialization: flag ← 1.
2: **while** flag = 1 **do**
3: FeasibleBatchcounter ← 0.
4: Solve the master problem to obtain its solution $x^*$, and the corresponding number of batches $B$.
5: **if** the MP is feasible **then**
   6: **for** $b = 0$ to $B$ **do**
   7: Solve the corresponding slave problems for batch $b$, i.e., slave(b).
   8: **if** slave(b) is infeasible **then**
   9: Add the corresponding combinatorial Benders cuts to the master problem.
   10: **break**
   11: **else**
   12: FeasibleBatchcounter ← FeasibleBatchcounter +1.
   13: **end if**
14: **end for**
15: **if** FeasibleBatchcounter $=$ $B$ **then**
16: All slave problems are feasible. Set flag ← 0.
17: Output the current solution $x^*$.
18: **end if**
19: **else**
20: The original problem is infeasible.
21: **break**
22: **end if**
23: **end while**
Combinatorial Benders cuts: No-good cuts

- Let $\tilde{I}_b = \{i \in I | u_{ib}^* = 1$, and slave(b) is infeasible\}
- One trivial combinatorial Benders cut can be derived:

$$\sum_{i \in \tilde{I}_b} u_{ib} \leq |\tilde{I}_b| - 1 \quad \forall b \in B. \quad (1)$$

- When the number of parts allocated into such infeasible batch is large, the above Benders cut could be quite loose (no-good cuts)

```
\Rightarrow u_{1b} + u_{2b} + u_{3b} + u_{5b} \leq 3 \quad \forall b \in B
```

Batch 1
Combinatorial Benders cuts: Next-fit-based cuts

- For any given order of parts, we pack each part subsequently to check its feasibility
- If feasible, we continue such process by adding the next unpacked part
- Otherwise, we obtain an infeasible set of parts, and a corresponding Benders cut can be generated
- Can only exclude some of the infeasible allocation plans
- Obtain an upper bound on the number of batches to be opened

$$u_1u_2 + u_2u_3 + u_3u_4 \leq 3$$
$$\forall b \in B$$
Combinatorial Benders cuts: Minimal infeasible subset cuts

- Alternative approach: \textit{enumeratively examine all subsets of the parts}, and check its feasibility
- The method of generating the MIS cuts:
  - We start enumerating each subset of this batch with a cardinality of $n_s = 2$, and check its feasibility
  - Each time when an infeasible subset is obtained, we generate a new Benders cut with respect to this subset
  - All supersets that include this subset will be excluded
  - We continue such process by gradually increasing the cardinality of the subset from 2 to $N$ until no more action can be made
An illustrative example for generating the MIS cuts

$n = 2$

$n = 3$

$n = 4$

$n = 5$
• Such procedure can output all MIS cuts
• The computational time will be exponentially increasing
• May not be practical when the total number of parts is large
• Balance between the quality of Benders cuts and computational time ⇒ Generate part or all MIS
• MIS-based heuristic cuts:
  • Given any infeasible batch with $\overline{N}$ parts
  • one-layer: only find infeasible subsets with $n_s = \overline{N} - 1$
  • two-layer: only find infeasible subsets with $n_s = \overline{N} - 1$ and $n_s = \overline{N} - 2$
  • all-layer: find all infeasible subsets with $n_s$ from 2 to $\overline{N}$
• For large-sized instances, the computational time remains unsatisfactory
• Let $LB(I)$ be the lower bound on the number of batches for a given set of parts $I$

• Trivial bound: $LB(I) \geq \left\lceil \sum_{i \in I} w_i l_i / WL \right\rceil$

• Considerable literature on designing different approximation algorithms for the two-dimensional bin packing problem (e.g., the hybrid first fit algorithm, HFF (Chung et al., 1982))
  
  • Let $HFF(I)$ be the number of bins used in an approximation algorithm HFF, and $\alpha$ is the approximation ratio of HFF
  
  • $OPT(I) \geq \left\lceil HFF(I) / \alpha \right\rceil$

• $LB = \max \left\{ \left\lceil \frac{\sum_{i \in I} w_i l_i}{WL} \right\rceil , \left\lceil \frac{HFF(I)}{\alpha} \right\rceil \right\}$
Accelerating strategy 2: Introducing a secondary objective

- The infeasibility of the slave problem is usually caused by the allocation of too many parts into the same batch.
- We introduce a secondary objective in the master problem to minimize the deviation of the number of parts across all batches while preserving the value of the primary objective.
- Distribute the parts into batches as equally as possible under the same makespan.
- The revised master problem:

\[
\begin{align*}
\min & \quad C_{\text{max}} + \varepsilon \cdot (\bar{O} - \underline{O}) \\
\text{s.t.} & \quad \text{Constraints (1b) - (1k)} \\
\quad & \quad O_b = \sum_{i \in I} u_{ib} \quad \forall b \in B \\
\quad & \quad \bar{O} \geq O_b \quad \forall b \in B \\
\quad & \quad \underline{O} \leq O_b \quad \forall b \in B
\end{align*}
\]
Accelerating strategy 3: Applying Steinberg’s Theorem

- We can also use Steinberg’s Theorem to directly verify whether the allocated parts can be feasibly packed into the batch.
- We calculate and compare the values in the conditions of Steinberg’s Theorem instead of solving the slave problem, and speed up the solution process of Algorithm CBD.
1 Introduction

2 Problem description

3 Approximation algorithm

4 Combinatorial Benders decomposition algorithm (Algorithm CBD)

5 Computational experiments
Computational environments

- The dataset provided by Che et al. (2021): parts with different orientations and various sizes
- We choose the first orientation of each part in their dataset and output the characteristics of this part, i.e., height, length, width and volume
- We randomly generate various parts based on the above data (repeat selections are allowed)
- The work of Kucukkoc (2019) have provided the additive machine-related parameters: the scanning time, recoating time, setup time
- We consider three different types of additive machines

<table>
<thead>
<tr>
<th>machine type</th>
<th>$VT \ (hr/cm^3)$</th>
<th>$HT \ (hr/cm)$</th>
<th>ST ($hr$)</th>
<th>$L \ (cm)$</th>
<th>$W \ (cm)$</th>
<th>$H \ (cm)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>small (S)</td>
<td>0.030864</td>
<td>0.7</td>
<td>2</td>
<td>15</td>
<td>15</td>
<td>32.5</td>
</tr>
<tr>
<td>medium (M)</td>
<td>0.030864</td>
<td>0.7</td>
<td>2</td>
<td>17.5</td>
<td>17.5</td>
<td>32.5</td>
</tr>
<tr>
<td>large (L)</td>
<td>0.030864</td>
<td>0.7</td>
<td>2</td>
<td>20</td>
<td>20</td>
<td>32.5</td>
</tr>
</tbody>
</table>
• We consider the following combinations of the number of parts $n$ and the type of machines:

$$\{(n, \text{type}) : n \in \{15, 20, 30, 40\}, \text{type} \in \{S, M, L\}\}.$$

• For each combination, we randomly generate 10 instances, for a total of $4 \times 3 \times 10 = 120$ instances

<table>
<thead>
<tr>
<th>combination</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>machine type number of parts</td>
<td>S</td>
<td>M</td>
<td>L</td>
<td>S</td>
<td>M</td>
<td>L</td>
<td>S</td>
<td>M</td>
<td>L</td>
<td>S</td>
<td>M</td>
<td>L</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>40</td>
<td>40</td>
<td>40</td>
</tr>
</tbody>
</table>

• We conduct our experiments on a computer with a 2.8GHz Intel Core i7 processor and 16 GB of RAM running the Windows 10 operating system

• We set a time limit of 7200 seconds for each experiment
Comparison of performance with different acceleration strategies

- Computational of performance between different acceleration strategies with $n = 20$ on S-type machine

<table>
<thead>
<tr>
<th>Instance</th>
<th>MILP</th>
<th>Algorithm CBD without any strategy</th>
<th>Algorithm CBD with strategy 1</th>
<th>Algorithm CBD with strategy 2</th>
<th>Algorithm CBD with strategy 3</th>
<th>Algorithm CBD with all strategies</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Obj</td>
<td>Time</td>
<td>Gap</td>
<td>Obj</td>
<td>MP</td>
<td>SP</td>
</tr>
<tr>
<td>1</td>
<td>97.45</td>
<td>–</td>
<td>6.00%</td>
<td>97.45</td>
<td>121</td>
<td>1785</td>
</tr>
<tr>
<td>2</td>
<td>93.03</td>
<td>–</td>
<td>5.45%</td>
<td>93.03</td>
<td>16</td>
<td>256</td>
</tr>
<tr>
<td>3</td>
<td>92.45</td>
<td>–</td>
<td>5.03%</td>
<td>92.45</td>
<td>58</td>
<td>827</td>
</tr>
<tr>
<td>4</td>
<td>79.59</td>
<td>–</td>
<td>4.62%</td>
<td>79.59</td>
<td>31</td>
<td>538</td>
</tr>
<tr>
<td>5</td>
<td>81.92</td>
<td>–</td>
<td>6.20%</td>
<td>81.64</td>
<td>48</td>
<td>828</td>
</tr>
<tr>
<td>6</td>
<td>92.15</td>
<td>–</td>
<td>11.60%</td>
<td>87.42</td>
<td>3</td>
<td>42</td>
</tr>
<tr>
<td>7</td>
<td>83.51</td>
<td>–</td>
<td>4.81%</td>
<td>83.51</td>
<td>12</td>
<td>202</td>
</tr>
<tr>
<td>8</td>
<td>78.95</td>
<td>–</td>
<td>7.23%</td>
<td>78.95</td>
<td>427</td>
<td>4421</td>
</tr>
<tr>
<td>9</td>
<td>90.94</td>
<td>–</td>
<td>3.12%</td>
<td>90.94</td>
<td>21</td>
<td>305</td>
</tr>
<tr>
<td>10</td>
<td>89.64</td>
<td>–</td>
<td>2.93%</td>
<td>89.36</td>
<td>9</td>
<td>164</td>
</tr>
<tr>
<td>Avg</td>
<td>87.96</td>
<td>–</td>
<td>5.70%</td>
<td>87.4374</td>
<td>60936.80</td>
<td>205.37</td>
</tr>
</tbody>
</table>

- The results show that these three strategies and their combinations can significantly reduce the CPU time
- The average CPU time is about half of the one without considering any acceleration strategy
Comparison of performance with different types of Benders cuts

- Algorithm CBD0: the combinatorial Benders decomposition algorithm that only uses the no-good cuts
- Algorithm CBD1: the one uses both the no-good cuts and the next-fit-based heuristic cuts
- Algorithm CBD2: the one with no-good and NF-based heuristic cuts and the one-layer MIS cuts
- Algorithm CBD3: the one with no-good and NF-based heuristic cuts and the two-layer MIS cuts
- Algorithm CBD4: the one with no-good and NF-based heuristic cuts and the all-layer MIS cuts
Comparison of performance with different types of Benders cuts

- Computational of performance between different types of Benders cuts with $n = 20$ on S-type machine

<table>
<thead>
<tr>
<th>Instance</th>
<th>MILP Obj</th>
<th>MILP Time</th>
<th>MILP Gap</th>
<th>Algorithm CBD0 MP Iter</th>
<th>Algorithm CBD0 SP Iter</th>
<th>Algorithm CBD0 Time</th>
<th>Algorithm CBD1 MP Iter</th>
<th>Algorithm CBD1 SP Iter</th>
<th>Algorithm CBD1 Time</th>
<th>Algorithm CBD2 MP Iter</th>
<th>Algorithm CBD2 SP Iter</th>
<th>Algorithm CBD2 Time</th>
<th>Algorithm CBD3 MP Iter</th>
<th>Algorithm CBD3 SP Iter</th>
<th>Algorithm CBD3 Time</th>
<th>Algorithm CBD4 MP Iter</th>
<th>Algorithm CBD4 SP Iter</th>
<th>Algorithm CBD4 Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>97.45</td>
<td>6.00%</td>
<td>97.45</td>
<td>123</td>
<td>482</td>
<td>313.22</td>
<td>134</td>
<td>532</td>
<td>287.54</td>
<td>112</td>
<td>1580</td>
<td>121.90</td>
<td>106</td>
<td>4273</td>
<td>92.79</td>
<td>124</td>
<td>10866</td>
<td>144.64</td>
</tr>
<tr>
<td>2</td>
<td>93.03</td>
<td>5.45%</td>
<td>93.03</td>
<td>21</td>
<td>75</td>
<td>8.67</td>
<td>21</td>
<td>84</td>
<td>10.27</td>
<td>5</td>
<td>100</td>
<td>4.53</td>
<td>10</td>
<td>386</td>
<td>13.19</td>
<td>8</td>
<td>738</td>
<td>10.98</td>
</tr>
<tr>
<td>3</td>
<td>92.45</td>
<td>5.03%</td>
<td>92.45</td>
<td>230</td>
<td>713</td>
<td>727.49</td>
<td>227</td>
<td>719</td>
<td>572.37</td>
<td>70</td>
<td>944</td>
<td>86.06</td>
<td>57</td>
<td>2039</td>
<td>96.47</td>
<td>61</td>
<td>3714</td>
<td>58.621</td>
</tr>
<tr>
<td>4</td>
<td>79.59</td>
<td>4.62%</td>
<td>79.59</td>
<td>96</td>
<td>356</td>
<td>90.07</td>
<td>104</td>
<td>395</td>
<td>87.84</td>
<td>34</td>
<td>451</td>
<td>45.95</td>
<td>22</td>
<td>846</td>
<td>42.35</td>
<td>21</td>
<td>1783</td>
<td>49.72</td>
</tr>
<tr>
<td>5</td>
<td>81.92</td>
<td>6.20%</td>
<td>81.64</td>
<td>44</td>
<td>140</td>
<td>23.46</td>
<td>45</td>
<td>157</td>
<td>22.18</td>
<td>26</td>
<td>415</td>
<td>29.09</td>
<td>23</td>
<td>982</td>
<td>40.97</td>
<td>34</td>
<td>2814</td>
<td>34.89</td>
</tr>
<tr>
<td>6</td>
<td>92.15</td>
<td>11.60%</td>
<td>87.42</td>
<td>3</td>
<td>9</td>
<td>1.19</td>
<td>3</td>
<td>20</td>
<td>1.44</td>
<td>3</td>
<td>34</td>
<td>3.37</td>
<td>3</td>
<td>77</td>
<td>4.74</td>
<td>3</td>
<td>285</td>
<td>7.55</td>
</tr>
<tr>
<td>7</td>
<td>83.51</td>
<td>4.81%</td>
<td>83.51</td>
<td>5</td>
<td>14</td>
<td>2.53</td>
<td>6</td>
<td>29</td>
<td>2.71</td>
<td>5</td>
<td>62</td>
<td>11.54</td>
<td>9</td>
<td>383</td>
<td>51.79</td>
<td>15</td>
<td>4287</td>
<td>111.94</td>
</tr>
<tr>
<td>8</td>
<td>78.95</td>
<td>7.23%</td>
<td>78.95</td>
<td>589</td>
<td>1767</td>
<td>812.37</td>
<td>588</td>
<td>1775</td>
<td>667.77</td>
<td>434</td>
<td>4469</td>
<td>473.00</td>
<td>424</td>
<td>12729</td>
<td>673.19</td>
<td>428</td>
<td>23038</td>
<td>410.84</td>
</tr>
<tr>
<td>9</td>
<td>90.94</td>
<td>3.12%</td>
<td>90.94</td>
<td>118</td>
<td>450</td>
<td>197.45</td>
<td>128</td>
<td>501</td>
<td>188.67</td>
<td>21</td>
<td>273</td>
<td>26.36</td>
<td>16</td>
<td>435</td>
<td>31.32</td>
<td>12</td>
<td>648</td>
<td>20.76</td>
</tr>
<tr>
<td>10</td>
<td>89.64</td>
<td>2.93%</td>
<td>89.36</td>
<td>41</td>
<td>144</td>
<td>39.30</td>
<td>47</td>
<td>168</td>
<td>27.82</td>
<td>8</td>
<td>103</td>
<td>11.13</td>
<td>7</td>
<td>182</td>
<td>15.13</td>
<td>7</td>
<td>646</td>
<td>19.77</td>
</tr>
<tr>
<td>Avg</td>
<td>87.96</td>
<td>&gt;7200</td>
<td>5.70%</td>
<td>87.43</td>
<td>127.00</td>
<td>415.00</td>
<td>221.57</td>
<td>130.30</td>
<td>438.00</td>
<td>186.86</td>
<td>71.80</td>
<td>843.10</td>
<td>81.29</td>
<td>67.70</td>
<td>2233.20</td>
<td>106.19</td>
<td>71.30</td>
<td>4881.90</td>
</tr>
</tbody>
</table>

- Any of the above combinatorial Benders decomposition algorithm can perform significantly better than solving the MILP model directly by Gurobi
• By imposing the NF-based heuristic cuts, the computational time of Algorithm CBD1 is generally smaller than Algorithm CBD0 (tighter upper bounds on the number of batches to be opened)

• By incorporating the MIS-based heuristic cuts, the number of iterations for the master problem in Algorithms CBD2-4 can be notably reduced compared to the ones in Algorithm CBD0, and the computational time decreases greatly when the number of parts increases

• The MIS-based heuristic cuts are quite effective in solving the SMSAM-2DP problem
END OF PRESENTATION

THANK YOU!

Please send your questions or comments to kfang@tju.edu.cn