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

Summary

- 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

Introduction

- 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)

- The production process of SLM/DMLS

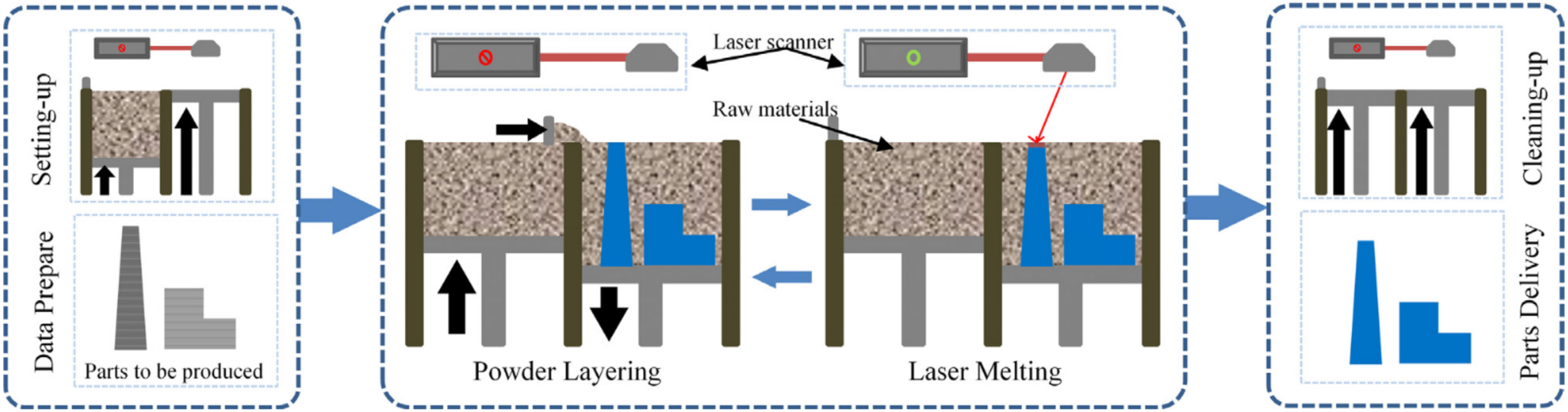


Figure 1: From Li et al. (2017)

- Pre-processing operations (data preparation, filling of powder materials, adjustment of AM machine, filluing 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 ℓ_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

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
  - left_{ijb}: 1 if part i is located left of part j in batch b, 0 otherwise
  - 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_{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

$$C_{\max} \geq C_b \quad \forall b \in B$$

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

Reference	Problem type	Constraint	Objective	Method
Freens et al. (2016)	S	SM	Min. cost function	MILP
Kucukkoc et al. (2016)	S	RM	Min. production costs	MILP+Heuristic
Kim et al. (2017)	S	PM/PA	Min. makespan	MILP+GA
Ransikarbum et al. (2017)	OAS	RM	Multiobjective	MILP
Li et al. (2017a)	S	RM	Min. average production costs	MILP+ Heuristic
Oh et al. (2018c)	NS	R	Min. cycle time	Heuristic
Dvorak et al. (2018)	NS	RM	Min. makespan, tardiness	CP+Heuristic
Kucukkoc et al. (2018)	S	RM	Min. maximum lateness	GA
Fera et al. (2018)	S	SM	Min. lateness/earliness costs	GA
Chergui et al. (2018)	NS	PM	Min. tardiness	MILP
Li et al. (2018)	OAS	RM	Max. profit	MILP
Griffiths et al. (2019)	NS	SM/BO	Min. build costs	ITSP
Stein et al. (2019)	OAS	RM	Max. revenue	MILP

Problem description

Reference	Problem type	Constraint	Objective	Method
Kucukkoc (2019)	S	R	Min. makespan	MILP
Li et al. (2019b)	OAS	R	Max. profit	MILP
Wang et al. (2019)	NS	R	Max. nesting rate	Heuristic
Luzon and Khmelnitsky (2019)	S	SM, F	Min. exp. makespan, flowtime	Queueing theory
Fera et al. (2020)	S	SM	Min. lateness/earliness costs	TS
Zhang et al. (2020)	NS	R	Min. makespan	EA
Kim and Kim (2020)	S	P/PA/SU	Min. makespan	MILP
Alicastro et al. (2021)	S	SM	Min. makespan	ILS
Che et al. (2021)	NS	PM/BO	Min. makespan	MILP+SA
Kapadia et al. (2021)	OAS	PM/BO	Max. profit	GA
Rohaninejad et al. (2021)	S	R	Min. weighted tardiness	Hybrid GA, LS
Altekin et al. (2021)	S	R	Multiobjective	MILP+Pareto
Aloui and Hadj-Hamou (2021)	NS	R	Min. total lateness	MILP+Heuristic
Kucukkoc et al. (2021)	NS	R	Min. total tardiness	GA
Zipfel et al. (2021)	NS	PM	Min. total weighted tardiness	ILS
Altekin and Bukchin (2022)	NS	RM	Min. makespan	MILP
Lee and Kim (2023)	NS	RM	Min. makespan	MILP+GA, PSO
Hu et al. (2022)	NS	RM/BO	Min. makespan	MILP+ALNS

Problem description

Problem type:

- S-Scheduling
- NS-Nesting & scheduling
- OAS-Order Acceptance and Scheduling

Constraint:

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

Objective:

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

Method:

- MILP
- Heuristics: GA, TS, SA, EA, LS…
- CP, Pareto
- Approximation Algorithm
- Exact Algorithm

Approximation algorithm

- 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

$$P = \sum_{b \in B} P_b = \underbrace{VT \sum_{i \in I} v_i}_{\text{total scanning time}} + \underbrace{HT \cdot \sum_{b \in B} h_b}_{\text{total recoating time}} + \underbrace{ST \cdot \sum_{b \in B} z_b}_{\text{total setup time}}$$

- The optimal makespan only depends on the total recoating time and the total setup time

- Suppose σ^* 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 $(k = 1, \ldots, t)$
- $C_{\max}(\sigma^*) = VT \sum_{i \in I} v_i + HT \cdot \sum_{k=1}^{t} h_k^* + ST \cdot t$

- Divide all the parts into three groups

$$I_1 = \left\{ i : w_i \le \frac{1}{2}W \ \ \& \ \ \ell_i \le \frac{1}{2}L \right\},$$
$$I_2 = \left\{ i : w_i > \frac{1}{2}W \right\},$$
$$I_3 = \left\{ i : w_i \le \frac{1}{2}W \ \ \& \ \ \ell_i > \frac{1}{2}L \right\}.$$

- 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_k^i as the kth part in group I_i $(h_{j_1^i} \ge h_{j_2^i} \ge \cdots \ge h_{j_{n_i}^i})$

Approximation algorithm: Algorithm GreedyPack

Initialize: $\widetilde{A} \leftarrow 0$, $\widetilde{L} \leftarrow 0, \widetilde{W} \leftarrow 0$.
for $i = 1$ to 3 **do**
  if $I_i \neq \emptyset$ **then**
    Open a new batch so as to pack the parts for each I_i. Let $s_i \leftarrow 1$.
  end if
end for

greedy packing for group I_1

for $k = 1$ to n_1 **do**
  $\widetilde{A} \leftarrow \widetilde{A} + w_{j_k^1} \cdot \ell_{j_k^1}$.
  if $\widetilde{A} \leq \frac{1}{2} W L$ **then**
    Put part j_k^1 into the current batch.
  else
    Close the current batch. Open a new batch and put part j_k^1 into the new batch.
    $s_1 \leftarrow s_1 + 1, \widetilde{A} \leftarrow w_{j_k^1} \cdot \ell_{j_k^1}$.
  end if
end for

greedy packing for group I_2

for $k = 1$ to n_2 **do**
  $\widetilde{L} \leftarrow \widetilde{L} + \ell_{j_k^2}$.
  if $\widetilde{L} \leq L$ **then**
    Put part j_k^2 into the current batch.
  else
    Close the current batch. Open a new batch and put part j_k^2 into the new batch.
    $s_2 \leftarrow s_2 + 1, \widetilde{L} \leftarrow \ell_{j_k^2}$.
  end if
end for

greedy packing for group I_3

for $k = 1$ to n_3 **do**
  $\widetilde{W} \leftarrow \widetilde{W} + w_{j_k^3}$.
  if $\widetilde{W} \leq W$ **then**
    Put part j_k^3 into the current batch.
  else
    Close the current batch. Open a new batch and put part j_k^3 into the new batch.
    $s_3 \leftarrow s_3 + 1, \widetilde{W} \leftarrow w_{j_k^3}$.
  end if
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 $w_i \leq 1/2W$ and $\ell_i \leq 1/2L$, 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 kth batch in group I_i

Lemma

$s_1 \leq 4t$

Proof.

- A new batch can be opened only if $\widetilde{A} + w_i \ell_i > \frac{1}{2} WL$
- The total area of parts in any two consecutive batches must be at least $1/2WL$
- 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 $h^1_{4k-3} \leq h^*_k \Rightarrow \sum_{k=1}^{s_1} h^1_k \leq 4 \cdot \sum_{k=1}^{t} h^*_k$

Proof.

- when $k = 1$, obviously true as h^*_1 must be the largest height
- We have $h^1_1 \geq \cdots \geq h^1_{4k-5} \geq h^1_{4k-4} \geq h^1_{4k-3} \geq \ldots$
- 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
- $h_{i'} \geq h^1_{4k-3} \Rightarrow h^*_k \geq h_{i'} \geq h^1_{4k-3}$

$\square$

Lemma

$s_2 \leq 2t$ and $s_3 \leq 2t$

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 $\widetilde{L} + \ell_i > L$, where $\widetilde{L}$ 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

$\square$

Lemma

For any $k \geq 0$, we have $h^2_{2k-1} \leq h^*_k$ and $h^3_{2k-1} \leq h^*_k$
$\Rightarrow \sum_{k=1}^{s_2} h^2_k \leq 2 \cdot \sum_{k=1}^{t} h^*_k$, *and* $\sum_{k=1}^{s_3} h^3_k \leq 2 \cdot \sum_{k=1}^{t} h^*_k$

Proof.

- when $k = 1$, obviously true
- $h^2_1 \geq \cdots \geq h^2_{2k-3} \geq h^2_{2k-2} \geq h^2_{2k-1} \geq \ldots$
- 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 $h_{i'} \geq h^2_{2k-1} \Rightarrow h^*_k \geq h_{i'} \geq h^2_{2k-1}$
- Similar results hold for group I_3

Theorem

The approximation ratio of Algorithm GreedyPack is at most 8

Proof.

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

$$
\begin{aligned}
C_{\max}(\sigma) = & VT \sum_{i \in I} v_i + HT \cdot (\sum_{k=1}^{s_1} h_k^1 + \sum_{k=1}^{s_2} h_k^2 + \sum_{k=1}^{s_3} h_k^3) + ST \cdot \sum_{i=1}^{3} s_i \\
& \leq VT \sum_{i \in I} v_i + HT \cdot 8 \sum_{k=1}^{t} h_k^* + ST \cdot 8t \\
& \leq 8 \cdot (VT \sum_{i \in I} v_i + HT \cdot \sum_{k=1}^{t} h_k^* + ST \cdot t) \\
& = 8 \cdot C_{\max}(\sigma^*) \quad \text{(quite loose!)}
\end{aligned}
$$

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 $\overline{x} \in X$, then solves the slave problem $SP : \min\{cy : Ay \geq b - B\overline{x}, y \geq 0\}$, which can be solved by means of the dual slave problem $SD : \max\{u(b - B\overline{x}) : uA \leq c, u \geq 0\}$
    - If SD has an optimal solution $\overline{u}$, then an optimality cut $z \geq \overline{u}(b - Bx)$ is constructed
    - If SD is unbounded, a feasibility cut $0 \geq \overline{u}(b - Bx)$ 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 (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 $\{Ay + B\overline{x} \geq b, y \geq 0\}$
    - If $\overline{x}$ is not a feasible solution for at least one variable x_j causing infeasibility, then this variable must take a different value from $\overline{x}_j$
    - If $\overline{x}$ is a feasible solution for SP, then it is feasible and optimal for P

- Schematic of CBD

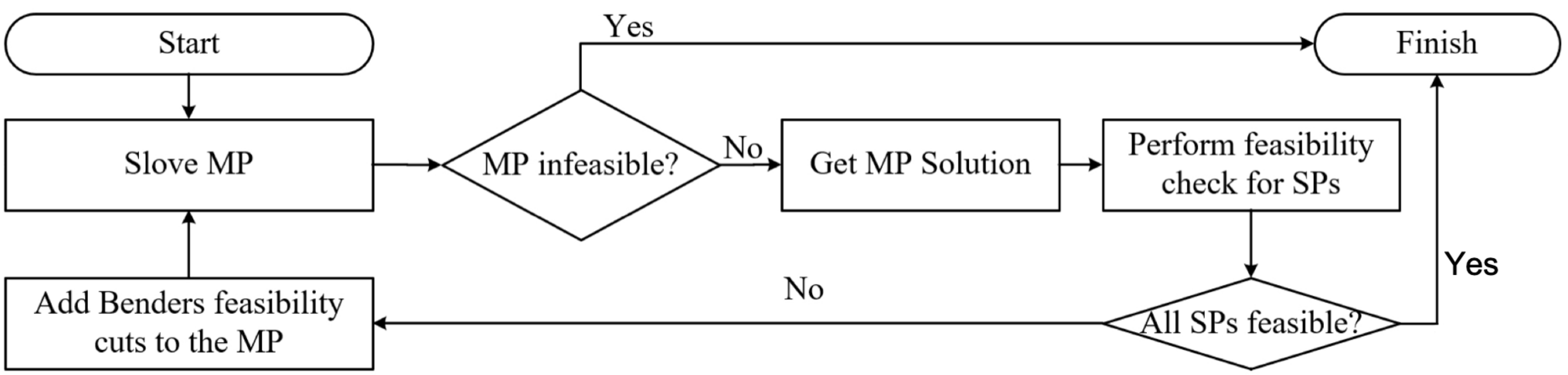


Figure 2: From Li et al. (2022)

- Numerous applications of CBD
  - Cutting and packing problems: Cote et al. (2014); Cote et al. (2021)
  - Assebly line balancing problems: Akpinar 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 probelm

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

area-restriction cuts → (1d)

- 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

$$
\begin{aligned}
\textbf{[slave(b)]} \quad & \min \quad 0 && && \text{(2a)} \\
& \text{s.t.} \quad x_i + w_i \le W && \forall i \in \bar{I}_b && \text{(2b)} \\
& y_i + \ell_i \le L && \forall i \in \bar{I}_b && \text{(2c)} \\
& \text{left}_{ij} + \text{left}_{ji} + \text{below}_{ij} + \text{below}_{ji} \ge 1 && \forall i, j \in \bar{I}_b,\ i \ne j && \text{(2d)} \\
& x_i + w_i \le x_j + W\left(1 - \text{left}_{ij}\right) && \forall i, j \in \bar{I}_b,\ i \ne j && \text{(2e)} \\
& y_i + \ell_i \le y_j + L\left(1 - \text{below}_{ij}\right) && \forall i, j \in \bar{I}_b,\ i \ne j && \text{(2f)} \\
& \text{left}_{ij},\ \text{below}_{ij} \in \{0,\ 1\} && \forall i, j \in \bar{I}_b,\ i \ne j && \text{(2g)} \\
& x_i,\ y_i \ge 0 && \forall i \in \bar{I}_b && \text{(2h)}
\end{aligned}
$$

Algorithm 1 The algorithmic outline of Algorithm CBD

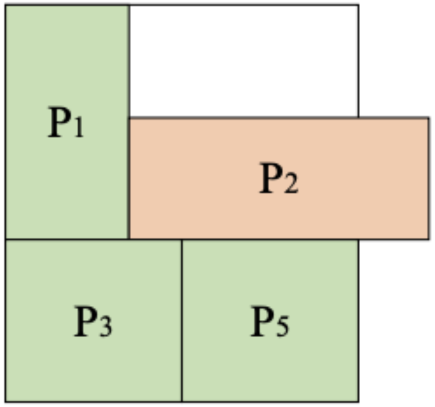
```
Initialization: flag ← 1.
while flag = 1 do
    FeasibleBatchcounter ← 0.
    Solve the master problem to obtain its solution x*, and the corresponding number of batches B.
    if the MP is feasible then
        for b = 0 to B do
            Solve the corresponding slave problems for batch b, i.e., slave(b).
            if slave(b) is infeasible then
                Add the corresponding combinatorial Benders cuts to the master problem.
                break
            else
                FeasibleBatchcounter ← FeasibleBatchcounter +1.
            end if
        end for
        if FeasibleBatchcounter = B then
            All slave problems are feasible. Set flag ← 0.
            Output the current solution x*.
        end if
    else
        The original problem is infeasible.
        break
    end if
end while
```

Combinatorial Benders cuts: No-good cuts

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

$$\sum_{i \in \widetilde{I}_b} u_{ib} \leq |\widetilde{I}_b| - 1 \qquad \forall b \in B. \tag{1}$$

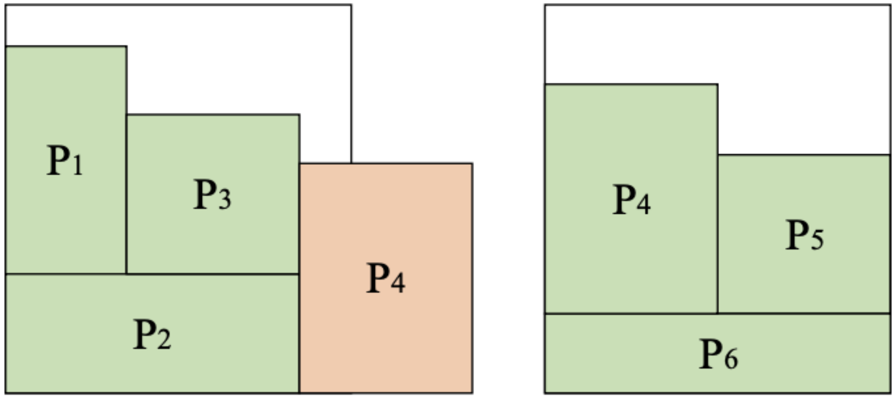
- When the number of parts allocated into such infeasible batch is large, the above Benders cut could be quite loose (no-good cuts)



Batch 1

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

- 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



Batch 1 Batch 2

$$\Rightarrow u_{1b} + u_{2b} + u_{3b} + u_{4b} \leq 3 \quad \forall b \in B$$

- Alternative approach: 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

- 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 $\Rightarrow$ 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 α is the approximation ratio of HFF
    - $OPT(I) \geq \lceil HFF(I)/\alpha \rceil$
- $LB = \max \left\{ \left\lceil \frac{\sum_{i \in I} w_i l_i}{WL} \right\rceil, \left\lceil \frac{HFF(I)}{\alpha} \right\rceil \right\}$

- 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:

$$\min \quad C_{\max} + \varepsilon \cdot \left(\overline{O} - \underline{O}\right) \tag{3a}$$

$$s.t. \quad \text{Constraints } (1b) - (1k) \tag{3b}$$

$$O_b = \sum_{i \in I} u_{ib} \qquad \forall b \in B \tag{3c}$$

$$\overline{O} \geq O_b \qquad \forall b \in B \tag{3d}$$

$$\underline{O} \leq O_b \qquad \forall b \in B \tag{3e}$$

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 vlaues in the conditions of Steinberg's Theorem instead of solving the slave problem, and speed up the solution process of Algorithm CBD

- 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

machine type	VT (hr/cm^3)	HT (hr/cm)	ST (hr)	L (cm)	W (cm)	H (cm)
small (S)	0.030864	0.7	2	15	15	32.5
medium (M)	0.030864	0.7	2	17.5	17.5	32.5
large (L)	0.030864	0.7	2	20	20	32.5

Computational environments

- 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

combination	1	2	3	4	5	6	7	8	9	10	11	12
machine type	S	M	L	S	M	L	S	M	L	S	M	L
number of parts	15	15	15	20	20	20	30	30	30	40	40	40

- 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

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

	MILP			Algorithm CBD without any strategy				Algorithm CBD with strategy 1			Algorithm CBD with strategy 2			Algorithm CBD with strategy 3			Algorithm CBD with all strategies		
Instance	Obj	Time	Gap	Obj	MP Iter	SP Iter	Time	MP Iter	SP Iter	Time	MP Iter	SP Iter	Time	MP Iter	SP Iter	Time	MP Iter	SP Iter	Time
1	97.45	–	6.00%	97.45	121	1785	255.82	151	2129	172.16	102	1492	193.94	121	1742	234.40	112	1580	121.90
2	93.03	–	5.45%	93.03	16	256	29.51	34	520	66.77	17	270	17.18	16	239	23.24	5	100	4.53
3	92.45	–	5.03%	92.45	58	827	115.87	70	1024	86.18	74	1049	148.37	58	763	132.10	70	944	86.06
4	79.59	–	4.62%	79.59	31	538	82.59	28	512	50.34	35	508	69.94	31	504	89.76	34	451	45.95
5	81.92	–	6.20%	81.64	48	828	75.52	24	430	31.06	25	442	55.98	48	776	81.14	26	415	29.09
6	92.15	–	11.60%	87.42	3	42	4.48	3	42	3.10	3	42	3.95	3	34	4.85	3	34	3.37
7	83.51	–	4.81%	83.51	12	202	831.55	3	54	29.71	6	111	59.95	12	193	850.56	5	62	11.54
8	78.95	–	7.23%	78.95	427	4421	518.10	422	4314	422.98	425	4344	561.54	427	4413	521.16	434	4469	473.00
9	90.94	–	3.12%	90.94	21	305	116.89	23	330	26.90	21	303	56.61	21	283	128.29	21	273	26.36
10	89.64	–	2.93%	89.36	9	164	23.36	16	211	16.77	9	133	20.17	9	147	24.96	8	103	11.13
Avg	87.96	>7200	5.70%	87.43	74.60	936.80	205.37	77.40	956.60	90.60	71.70	869.40	118.76	74.60	909.40	209.05	71.80	843.10	81.29

- 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

- 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

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

Instance	MILP				Algorithm CBD0			Algorithm CBD1			Algorithm CBD2			Algorithm CBD3			Algorithm CBD4		
	Obj	Time	Gap	Obj	MP Iter	SP Iter	Time	MP Iter	SP Iter	Time	MP Iter	SP Iter	Time	MP Iter	SP Iter	Time	MP Iter	SP Iter	Time
1	97.45	–	6.00%	97.45	123	482	313.22	134	532	287.54	112	1580	121.90	106	4273	92.79	124	10866	144.64
2	93.03	–	5.45%	93.03	21	75	8.67	21	84	10.27	5	100	4.53	10	386	13.19	8	738	10.98
3	92.45	–	5.03%	92.45	230	713	727.49	227	719	572.37	70	944	86.06	57	2039	96.47	61	3714	58.621
4	79.59	–	4.62%	79.59	96	356	90.07	104	395	87.84	34	451	45.95	22	846	42.35	21	1783	49.72
5	81.92	–	6.20%	81.64	44	140	23.46	45	157	22.18	26	415	29.09	23	982	40.97	34	2814	34.89
6	92.15	–	11.60%	87.42	3	9	1.19	3	20	1.44	3	34	3.37	3	77	4.74	3	285	7.55
7	83.51	–	4.81%	83.51	5	14	2.53	6	29	2.71	5	62	11.54	9	383	51.79	15	4287	111.94
8	78.95	–	7.23%	78.95	589	1767	812.37	588	1775	667.77	434	4469	473.00	424	12729	673.19	428	23038	410.84
9	90.94	–	3.12%	90.94	118	450	197.45	128	501	188.67	21	273	26.36	16	435	31.32	12	648	20.76
10	89.64	–	2.93%	89.36	41	144	39.20	47	168	27.82	8	103	11.13	7	182	15.13	7	646	19.77
Avg	87.96	>7200	5.70%	87.43	127.00	415.00	221.57	130.30	438.00	186.86	71.80	843.10	81.29	67.70	2233.20	106.19	71.30	4881.90	86.97

- 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