# Multiple-type, Two-dimensional Finite Bin Packing Problem

**Members of group 3**
Member 1: Phan Dinh Nhat 20210654
Member 2: Chu Minh Ha 20210293
Member 3: Do Quang Minh 20210579
Member 4: Nguyen Huu Duan 20214951

**Course dates**
October 2022 - March 2023
Professor: PhD. Pham Quang Dung

## 1 Introduction and Objectives

$K$ trucks $1, 2, ..., K$ are available for transporting $N$ packages $1, 2, ..., N$. Each truck $k$ has the container size of $W_k * H_k$. The dimensions of each package $i$ are $w_i * h_i$. Packages that are placed in the same container must not overlap. Assume that the number $K$ can be large, leading to a great number of trucks that are not being used. $C_k$ represents the cost of using truck $k$. Find a solution that loads all the packages into those given trucks such that the total cost of trucks used is minimal.

**The data format**

- First line: $N$ and $K$
- Next $N$ lines: $w_i$ and $h_i$
- Last $K$ lines: $W_k$, $H_k$ and $C_k$

**Generated data**
*The input size is denoted by (number_of_items * number_of_bins).*

- 1 sample test: $(7 * 3)$
- 45 tests: $(10 * 10)$, $(11 * 11)$, .., $(53 * 53)$, $(54 * 54)$
- 10 tests: $(90 * 90)$, $(120 * 120)$, $(150 * 150)$, .., $(330 * 330)$, $(360 * 360)$
- 10 tests: $(550 * 550)$, $(600 * 600)$, .., $(950 * 950)$, $(1000 * 1000)$
- 09 tests mainly for heuristic testing: $(2000 * 2000)$, $(3000 * 3000)$, ..., $(10000 * 10000)$

## 2 Modeling the problem

### 2.1 CP model

In this model. Let:

- $N_{bins}$: the number of bins given
- $N_{items}$: the number of items given
- $W_j$, $H_j$, $C_j$: the width, height, and cost of bin $j$, respectively
- $w_i$, $h_i$: the width and height of item $i$, respectively
- $l_i$, $r_i$, $b_i$, $t_i$: left, right, bottom and top coordinates of item $i$

#### 2.1.1 Decision Variables

- $X_{ij} = 1$: item $i$ packed in bin $j$

$$\Rightarrow \sum_{i=1}^{N_{items}} X_{ij} \geq 1 \Leftrightarrow Z_j = 1 : \text{bin j has been used} \tag{1}$$

- $R_i = 1$: item $i$ rotated 90 degree

- **Item's Coordinate**:
  - First way to approach:
    * if item i not rotated: $R_i = 0$
$$\Rightarrow \begin{cases} r_i = l_i + w_i \\ t_i = b_i + h_i \end{cases} \tag{2}$$
    * if item i rotated: $R_i = 1$
$$\Rightarrow \begin{cases} r_i = l_i + h_i \\ t_i = b_i + w_i \end{cases} \tag{3}$$
  - Another way to approach:
    * if item i not rotated: $R_i = 0$
$$\Rightarrow \begin{cases} w_i = w_i \\ h_i = h_i \end{cases} \tag{4}$$
    * if item i rotated: $R_i = 1$
$$\Rightarrow \begin{cases} w_i = h_i \\ h_i = w_i \end{cases} \tag{5}$$

### 2.1.2 Constraints

- Each item has to be packed in exactly 1 bin:
$$\sum_{j=1}^{N_{bins}} X_{ij} = 1 \forall i \in \{1, 2, ..., N\_items\} \tag{6}$$
- No two items overlap:
if $X_{i_1 j} = X_{i_2 j} = 1$
$$r_{i_1} \le l_{i_2} \text{ or } r_{i_2} \le l_{i_1} \text{ or } t_{i_1} \le b_{i_2} \text{ or } t_{i_2} \le b_{i_1} \tag{7}$$
- Items cannot exceed the bin:
if $X_{ij} = 1$
$$\Rightarrow \begin{cases} w_i \le r_i \le W_j \\ h_i \le t_i \le H_j \end{cases} \tag{8}$$

### 2.1.3 Objective Function

$$\min \sum_{j=1}^{N_{bins}} Z_j * C_j \tag{9}$$

## 2.2 MIP model

In this model. Let:

- Constant value $M$
- $N_{bins}$: the number of bins given
- $N_{items}$: the number of items given
- $W_j$, $H_j$, $C_j$: the width, height, and cost of bin $j$, respectively
- $w_i$, $h_i$: the width and height of item $i$, respectively
- $l_i$, $r_i$, $b_i$, $t_i$: left, right, bottom and top coordinates of item $i$

#### 2.2.1 Decision Variables

- $X_{ij} = 1$: item $i$ packed in bin $j$

$$\Rightarrow \begin{cases} Z_j \leq \sum_{i=1}^{N_{items}} X_{ij} * M \\ Z_j * M \geq \sum_{i=1}^{N_{items}} X_{ij} \end{cases} \tag{10}$$

- $R_i = 1$: item $i$ rotated 90 degree
- **Item's Coordinate**:

$$\Rightarrow \begin{cases} r_i = l_i + w_i * (1 - R_i) + h_i * R_i \\ t_i = b_i + h_i * (1 - R_i) + w_i * R_i \end{cases} \tag{11}$$

#### 2.2.2 Constraints

- Each item has to be packed in exactly 1 bin:

$$\sum_{j=1}^{N_{bins}} X_{ij} = 1 \forall i \in \{1, 2, ..., N\_items\} \tag{12}$$

- No two items overlap:
  **New variable e such that**:

$$\begin{cases} e \geq X_{i_1 j} + X_{i_2 j} - 1 \\ e \leq X_{i_1 j} \\ e \leq X_{i_2 j} \end{cases} \tag{13}$$

$$\Rightarrow \begin{cases} r_{i_1} \leq l_{i_2} + M * (1 - (r_{i_1} \leq l_{i_2})) \\ r_{i_2} \leq l_{i_1} + M * (1 - (r_{i_2} \leq l_{i_1})) \\ t_{i_1} \leq b_{i_2} + M * (1 - (t_{i_1} \leq b_{i_2})) \\ t_{i_2} \leq b_{i_1} + M * (1 - (t_{i_2} \leq b_{i_1})) \\ (r_{i_1} \leq l_{i_2}) + (r_{i_2} \leq l_{i_1}) + (t_{i_1} \leq b_{i_2}) + (t_{i_2} \leq b_{i_1}) + (1 - e) * M \geq 1 \\ (r_{i_1} \leq l_{i_2}) + (r_{i_2} \leq l_{i_1}) + (t_{i_1} \leq b_{i_2}) + (t_{i_2} \leq b_{i_1}) \leq e * M \end{cases} \tag{14}$$

- Items cannot exceed the bin:

$$\Rightarrow \begin{cases} r_i \leq (1 X_{ij}) * M + W_j \\ t_i \leq (1 X_{ij}) * M + H_j \end{cases} \tag{15}$$

#### 2.2.3 Objective Function

$$\min \sum_{j=1}^{N_{bins}} Z_j * C_j \tag{16}$$

### 2.3 Heuristic

#### 2.3.1 Overview

**Heuristic algorithms**: Combine two algorithms, including Guillotine and Maximal Rectangles [1].

**Concept of free rectangles**: A list of free rectangles represents the free space of the bin. In Guillotine algorithm, these rectangles are pairwise disjoint.

#### 2.3.2 Sorting Input

**Bins**: Ascending order of density (cost/area), ties broken with the descending order of the longer side, followed by the descending order of the shorter side.

**Items**: Descending order of the longer side, ties broken with the descending order of the shorter side.

#### 2.3.3 Choosing the Destination for Items

**Destination Bin**: Bin First Fit rule – pack the item into the bin with the lowest index (after the process of sorting bins from 3.1.1); in other words, pack in the first bin that the item fits.

**Destination Free Rectangle of a Bin**: Best Short Side Fit rule – choose a free rectangle where the shorter remainder side after insertion is minimized; in other words, minimize the length of the shorter leftover side, ties broken with best longer side (longer leftover side is minimized).

#### 2.3.4 Packing process

**Guillotine**:

- Pack the item into the first free rectangle of the bin, which means the bin itself, starting with its bottom-left corner.
- For each insertion, split the initial free rectangle into smaller free rectangle(s) by the Guillotine split rule, then tracked in a list.
- Whenever a new item is inserted into the bin, choose a free rectangle (with the rule 3.2.2) and place the item into its bottom-left corner, then split the chosen rectangle using the Guillotine split rule to produce at most two new rectangles.
- Merge some free rectangles into larger ones if possible.
- **Splitting rule**: Best Short Side rule - split by horizontal axis if the free rectangle's width is less than its height; otherwise, split by vertical axis.
- **Rectangle merging**: If exists a pair of neighboring rectangles $F_i$ and $F_j$ such that the union $F_i \cup F_j$ can be exactly represented by a single bigger rectangle, merge these two into one.

**Maximal Rectangles**

- Rather than choosing one of the two split axes like in the Guillotine algorithm, the Maximal Rectangles algorithm picks both split axes at the same time to ensure that the largest possible rectangular areas are present in the list of free rectangles.
- Because the free rectangles are no longer pairwise disjoint, any free rectangle that intersects the area occupied by the newly inserted item is split such to remove the intersection.
- Delete every free rectangle which is fully overlapped by others in the list.

#### 2.3.5 Pseudo-code

**Algorithm 1** The Guillotine algorithm

Set $F = \{(W, H)\}$;
**for** each item $i = (w, h)$ in the list of inserted items of the bin **do**
    Decide the free rectangle $F_j \in F$ to pack the item into;
    Decide the orientation for the item and place it at the bottom-left of $F_j$;
    Use the guillotine split scheme to subdivide $F_j$ into two new free rectangles $F_{j1}$ and $F_{j2}$;
    Set $F \leftarrow (F \cup \{F_{j1}, F_{j2}\}) \setminus \{F_j\}$;
    **for** each ordered pair of free rectangles $F_{j1}$ and $F_{j2}$ in $F$ **do**
        **if** $F_{j1}$ and $F_{j2}$ can be merged together **then**
            $F_{merge} \leftarrow$ Merge $F_{j1}$ and $F_{j2}$;
            Set $F \leftarrow (F \cup \{F_{merge}\}) \setminus \{F_{j1}, F_{j2}\}$;
        **end if**
    **end for**
**end for**

**Algorithm 2** Maximal Rectangles Algorithm

Set $F = \{(W, H)\}$;
**for** each item $i = (w, h)$ in the list of inserted items of the bin **do**
    Decide the free rectangle $F_j \in F$ to pack the item into;
    Decide the orientation for the item and place it at the bottom-left of $F_j$;
    Use the maximal rectangles split scheme to subdivide $F_j$ into two new free rectangles $F_{j1}$ and $F_{j2}$;
    Set $F \leftarrow (F \cup \{F_{j1}, F_{j2}\}) \setminus \{F_j\}$;
    **for** each free rectangle $F_j$ in $F$ **do**
        Compute $F_j \setminus i$ and subdivide the result into at most four new free rectangles $F_{j1}, \ldots, F_{j4}$;
        Set $F \leftarrow (F \cup \{F_{j1}, \ldots, F_{j4}\}) \setminus \{F_j\}$;
    **end for**
    **for** each ordered pair of free rectangles $F_{j1}, F_{j2}$ in $F$ **do**
        **if** $F_{j1}$ contains $F_{j2}$ **then**
            Set $F \leftarrow F \setminus \{F_{j2}\}$;
        **end if**
    **end for**
**end for**

# 3 Result and analysis

## 3.1 Result

### 3.1.1 CP model and MIP model solver

**Exact solution**

| Input sizes | | CP 1 | | CP 2 | | MIP | |
|---|---|---|---|---|---|---|---|
| n_packs | n_bins | f | t(s) | f | t(s) | f | t(s) |
| 7 | 3 | 250 | 0.027932056 | 250 | 0.029741828 | 250 | 3.176666667 |
| 10 | 10 | 51 | 0.049313393 | 51 | 0.082288480 | 51 | 2.642000000 |
| 11 | 11 | 79 | 0.053943779 | 79 | 0.194204638 | 79 | 14.00300000 |
| 12 | 12 | 54 | 0.057868931 | 54 | 0.088900393 | 54 | 7.898000000 |
| 13 | 13 | 103 | 0.109191075 | 103 | 0.248680102 | F | F |
| 14 | 14 | 50 | 0.218952388 | 50 | 0.156991295 | 50 | 27.73466667 |
| 15 | 15 | 106 | 0.513134012 | 106 | 0.859766784 | F | F |
| 16 | 16 | 113 | 0.905138111 | 113 | 0.434518921 | F | F |
| 17 | 17 | 105 | 117.0229775 | 105 | 48.88790709 | F | F |
| 18 | 18 | F | F | F | F | F | F |
| 19 | 19 | 106 | 5.214479066 | 106 | 4.515408114 | F | F |
| 20 | 20 | 171 | 2.519827466 | 171 | 3.259184459 | F | F |
| 21 | 21 | 108 | 8.500796449 | 108 | 13.22533175 | F | F |

Table 1: Results only exact solution of 2 CP solver and MIP solver

**All solution**
Due to the size and complexity of the optimization problems, we only received feasible solutions for test sizes above $(21 * 21)$ for the CP solver and above $(15 * 15)$ for the MIP solver within a time limit of 300 seconds. To obtain reliable results, we ran each solver three times and calculated the average results.

| Input sizes | | CP 1 | | | | | CP 2 | | | | | MIP | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n_packs | n_bins | f_min | f_max | f_avg | std_dev | t_avg(s) | f_min | f_max | f_avg | std_dev | t_avg(s) | f_min | f_max | f_avg | std_dev | t_avg(s) |
| 7 | 3 | 250 | 250 | 250 | 0 | 0.027932056 | 250 | 250 | 250 | 0 | 0.02974183 | 250 | 250 | 250 | 0 | 3.17666667 |
| 10 | 10 | 51 | 51 | 51 | 0 | 0.049313393 | 51 | 51 | 51 | 0 | 0.08228848 | 51 | 51 | 51 | 0 | 2.64200000 |
| 11 | 11 | 79 | 79 | 79 | 0 | 0.053943779 | 79 | 79 | 79 | 0 | 0.19420464 | 79 | 79 | 79 | 0 | 14.0030000 |
| 12 | 12 | 54 | 54 | 54 | 0 | 0.057868931 | 54 | 54 | 54 | 0 | 0.08890039 | 54 | 54 | 54 | 0 | 7.89800000 |
| 13 | 13 | 103 | 103 | 103 | 0 | 0.109191075 | 103 | 103 | 103 | 0 | 0.24868010 | 103 | 103 | 103 | 0 | 300.245000 |
| 14 | 14 | 50 | 50 | 50 | 0 | 0.218952388 | 50 | 50 | 50 | 0 | 0.15699130 | 50 | 50 | 50 | 0 | 27.7346667 |
| 15 | 15 | 106 | 106 | 106 | 0 | 0.513134012 | 106 | 106 | 106 | 0 | 0.85976678 | 106 | 106 | 106 | 0 | 300.406667 |
| 16 | 16 | 113 | 113 | 113 | 0 | 0.905138111 | 113 | 113 | 113 | 0 | 0.43451892 | 113 | 113 | 113 | 0 | 300.472667 |
| 17 | 17 | 105 | 105 | 105 | 0 | 117.0229775 | 105 | 105 | 105 | 0 | 48.8879071 | 105 | 105 | 105 | 0 | 300.741667 |
| 18 | 18 | 118 | 118 | 118 | 0 | 300.0134580 | 118 | 118 | 118 | 0 | 300.009749 | 121 | 147 | 133 | 13.115 | 300.650000 |
| 19 | 19 | 106 | 106 | 106 | 0 | 5.214479066 | 106 | 106 | 106 | 0 | 4.51540811 | 106 | 106 | 106 | 0 | 300.789000 |
| 20 | 20 | 171 | 171 | 171 | 0 | 2.519827466 | 171 | 171 | 171 | 0 | 3.25918446 | 466 | 466 | 466 | 0 | 301.060667 |
| 21 | 21 | 108 | 108 | 108 | 0 | 8.500796449 | 108 | 108 | 108 | 0 | 13.2253317 | 210 | 210 | 210 | 0 | 301.076333 |
| 22 | 22 | 156 | 156 | 156 | 0 | 300.0168832 | 156 | 156 | 156 | 0 | 300.015485 | 461 | 623 | 569 | 93.531 | 301.211333 |
| 23 | 23 | 116 | 116 | 116 | 0 | 300.0116365 | 116 | 116 | 116 | 0 | 300.011188 | 1108 | 1108 | 1108 | 0 | 301.363333 |
| 24 | 24 | 158 | 158 | 158 | 0 | 300.0148793 | 158 | 160 | 158.67 | 1.1547 | 300.015304 | 356 | 356 | 356 | 0 | 301.663333 |
| 25 | 25 | 204 | 204 | 204 | 0 | 300.0179474 | 217 | 217 | 217 | 0 | 300.016161 | 1601 | 1601 | 1601 | 0 | 301.887000 |
| 26 | 26 | 158 | 158 | 158 | 0 | 300.0207270 | 158 | 158 | 158 | 0 | 300.019220 | 622 | 622 | 622 | 0 | 301.989667 |
| 27 | 27 | 158 | 158 | 158 | 0 | 300.0189365 | 158 | 158 | 158 | 0 | 300.015856 | N/A | N/A | N/A | N/A | N/A |
| 28 | 28 | 158 | 158 | 158 | 0 | 300.0196483 | 158 | 158 | 158 | 0 | 300.016423 | 1190 | 1190 | 1190 | 0 | 304.034333 |
| 29 | 29 | 167 | 178 | 174.33 | 6.3509 | 300.0187450 | 178 | 178 | 178 | 0 | 300.016640 | 1415 | 1415 | 1415 | 0 | 303.806667 |
| 30 | 30 | 181 | 182 | 181.67 | 0.5774 | 300.0173433 | 181 | 181 | 181 | 0 | 300.016748 | 2144 | 2144 | 2144 | 0 | 303.520000 |
| 31 | 31 | 215 | 215 | 215 | 0 | 300.0253246 | 215 | 215 | 215 | 0 | 300.022030 | 2249 | 2249 | 2249 | 0 | 303.887000 |
| 32 | 32 | 171 | 171 | 171 | 0 | 300.0176523 | 171 | 171 | 171 | 0 | 300.017343 | 1224 | 1224 | 1224 | 0 | 306.495333 |
| 33 | 33 | 165 | 165 | 165 | 0 | 300.0201768 | 165 | 165 | 165 | 0 | 300.019763 | 1429 | 1429 | 1429 | 0 | 306.537000 |
| 34 | 34 | 165 | 165 | 165 | 0 | 300.0260553 | 165 | 165 | 165 | 0 | 300.020847 | 803 | 803 | 803 | 0 | 306.058333 |
| 35 | 35 | 213 | 220 | 215.33 | 4.0415 | 300.0196759 | 220 | 220 | 220 | 0 | 300.023850 | 1723 | 1723 | 1723 | 0 | 309.124333 |
| 36 | 36 | 299 | 299 | 299 | 0 | 300.0296591 | 299 | 299 | 299 | 0 | 300.023764 | 2795 | 2795 | 2795 | 0 | 309.352000 |
| 37 | 37 | 163 | 163 | 163 | 0 | 300.0222829 | 163 | 163 | 163 | 0 | 300.021046 | 1750 | 1750 | 1750 | 0 | 308.550000 |
| 38 | 38 | 252 | 252 | 252 | 0 | 300.0218226 | 252 | 252 | 252 | 0 | 300.024474 | 1193 | 1193 | 1193 | 0 | 309.728667 |
| 39 | 39 | 229 | 229 | 229 | 0 | 300.0224621 | 229 | 229 | 229 | 0 | 300.022671 | 2895 | 2895 | 2895 | 0 | 308.145000 |
| 40 | 40 | 263 | 263 | 263 | 0 | 300.0266393 | 263 | 263 | 263 | 0 | 300.027277 | 2964 | 2964 | 2964 | 0 | 317.509667 |
| 41 | 41 | 225 | 228 | 226 | 1.7321 | 300.0271970 | 225 | 225 | 225 | 0 | 300.023711 | 1304 | 1304 | 1304 | 0 | 315.537000 |
| 42 | 42 | 367 | 367 | 367 | 0 | 300.0270539 | 375 | 375 | 375 | 0 | 300.030499 | 3115 | 3115 | 3115 | 0 | 317.383667 |
| 43 | 43 | 237 | 237 | 237 | 0 | 300.0364837 | 279 | 279 | 279 | 0 | 300.024842 | 3178 | 3178 | 3178 | 0 | 314.327667 |
| 44 | 44 | 283 | 304 | 296.33 | 11.590 | 300.0313213 | 283 | 292 | 286 | 5.1962 | 300.027890 | 2030 | 2030 | 2030 | 0 | 318.902667 |
| 45 | 45 | 220 | 220 | 220 | 0 | 300.0270809 | 220 | 220 | 220 | 0 | 300.029565 | N/A | N/A | N/A | N/A | N/A |
| 46 | 46 | 272 | 272 | 272 | 0 | 300.0306748 | 274 | 291 | 285.33 | 9.8150 | 300.030624 | N/A | N/A | N/A | N/A | N/A |
| 47 | 47 | 313 | 322 | 319 | 5.1962 | 300.0306518 | 322 | 322 | 322 | 0 | 300.030069 | 2182 | N/A | N/A | N/A | N/A |
| 48 | 48 | 269 | 269 | 269 | 0 | 300.0316954 | 269 | 269 | 269 | 0 | 300.029771 | N/A | N/A | N/A | N/A | N/A |
| 49 | 49 | 273 | 273 | 273 | 0 | 300.0297330 | 273 | 273 | 273 | 0 | 300.032365 | N/A | N/A | N/A | N/A | N/A |
| 50 | 50 | 291 | 293 | 291.67 | 1.1547 | 300.0346591 | 290 | 290 | 290 | 0 | 300.040313 | N/A | N/A | N/A | N/A | N/A |
| 51 | 51 | 328 | 328 | 328 | 0 | 300.0348325 | 328 | 391 | 370 | 36.373 | 300.035202 | N/A | N/A | N/A | N/A | N/A |

| Input sizes | | CP 1 | | | | | CP 2 | | | | | MIP | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **n_packs** | **n_bins** | **f_min** | **f_max** | **f_avg** | **std_dev** | **t_avg(s)** | **f_min** | **f_max** | **f_avg** | **std_dev** | **t_avg(s)** | **f_min** | **f_max** | **f_avg** | **std_dev** | **t_avg(s)** |
| 52 | 52 | 330 | 345 | 335 | 8.6603 | 300.0370899 | 330 | 334 | 332.67 | 2.3094 | 300.038526 | N/A | N/A | N/A | N/A | N/A |
| 53 | 53 | 382 | 382 | 382 | 0 | 300.0353604 | 359 | 384 | 373 | 12.767 | 300.037128 | N/A | N/A | N/A | N/A | N/A |
| 54 | 54 | 260 | 260 | 260 | 0 | 300.0374483 | 260 | 260 | 260 | 0 | 300.037056 | N/A | N/A | N/A | N/A | N/A |
| 90 | 90 | 567 | 585 | 573 | 10.392 | 300.0979588 | 565 | 570 | 567.33 | 2.5166 | 300.117140 | N/A | N/A | N/A | N/A | N/A |
| 120 | 120 | 760 | 800 | 776.33 | 20.984 | 300.1965074 | 744 | 784 | 770 | 22.539 | 300.192483 | N/A | N/A | N/A | N/A | N/A |
| 150 | 150 | 1169 | 1454 | 1301.3 | 143.58 | 300.3545031 | 856 | 994 | 911.67 | 72.762 | 300.442578 | N/A | N/A | N/A | N/A | N/A |
| 180 | 180 | 1810 | 1847 | 1825 | 19.468 | 300.7002796 | 1314 | 1787 | 1490 | 258.68 | 300.542184 | N/A | N/A | N/A | N/A | N/A |
| 210 | 210 | 2595 | 2828 | 2731 | 121.30 | 300.7788572 | 2270 | 2600 | 2481.7 | 183.73 | 300.749900 | N/A | N/A | N/A | N/A | N/A |
| 240 | 240 | 1861 | 1992 | 1941.7 | 70.571 | 301.0924356 | 2074 | 2434 | 2308.3 | 203.12 | 301.062326 | N/A | N/A | N/A | N/A | N/A |

Table 2: Average results of 2 CP solver and MIP solver

### 3.1.2 Heuristic solver

| Input sizes | | Heuristics | | | | |
|---|---|---|---|---|---|---|
| n_packs | n_bins | f_min | f_max | f_avg | std_dev | t_avg(s) |
| 7 | 3 | 300 | 300 | 300 | 0 | 0.000029500 |
| 10 | 10 | 51 | 51 | 51 | 0 | 0.000037000 |
| 11 | 11 | 79 | 79 | 79 | 0 | 0.000035500 |
| 12 | 12 | 54 | 54 | 54 | 0 | 0.000040500 |
| 13 | 13 | 145 | 145 | 145 | 0 | 0.000039000 |
| 14 | 14 | 55 | 55 | 55 | 0 | 0.000051000 |
| 15 | 15 | 106 | 106 | 106 | 0 | 0.000057500 |
| 16 | 16 | 130 | 130 | 130 | 0 | 0.000049000 |
| 17 | 17 | 105 | 105 | 105 | 0 | 0.000052500 |
| 18 | 18 | 121 | 121 | 121 | 0 | 0.000059000 |
| 19 | 19 | 129 | 129 | 129 | 0 | 0.000058500 |
| 20 | 20 | 188 | 188 | 188 | 0 | 0.000053500 |
| 21 | 21 | 120 | 120 | 120 | 0 | 0.000060000 |
| 22 | 22 | 171 | 171 | 171 | 0 | 0.000060500 |
| 23 | 23 | 147 | 147 | 147 | 0 | 0.000075500 |
| 24 | 24 | 181 | 181 | 181 | 0 | 0.000084500 |
| 25 | 25 | 237 | 237 | 237 | 0 | 0.000071500 |
| 26 | 26 | 161 | 161 | 161 | 0 | 0.000070500 |
| 27 | 27 | 180 | 180 | 180 | 0 | 0.000069000 |
| 28 | 28 | 158 | 158 | 158 | 0 | 0.000079000 |
| 29 | 29 | 186 | 186 | 186 | 0 | 0.000077000 |
| 30 | 30 | 182 | 182 | 182 | 0 | 0.000080000 |
| 31 | 31 | 215 | 215 | 215 | 0 | 0.000090000 |
| 32 | 32 | 187 | 187 | 187 | 0 | 0.000096500 |
| 33 | 33 | 178 | 178 | 178 | 0 | 0.000101500 |
| 34 | 34 | 182 | 182 | 182 | 0 | 0.000091500 |
| 35 | 35 | 237 | 237 | 237 | 0 | 0.000098000 |
| 36 | 36 | 345 | 345 | 345 | 0 | 0.000098500 |
| 37 | 37 | 163 | 163 | 163 | 0 | 0.000095000 |
| 38 | 38 | 260 | 260 | 260 | 0 | 0.000102000 |
| 39 | 39 | 229 | 229 | 229 | 0 | 0.000116500 |
| 40 | 40 | 236 | 236 | 236 | 0 | 0.000121500 |
| 41 | 41 | 251 | 251 | 251 | 0 | 0.000117500 |
| 42 | 42 | 412 | 412 | 412 | 0 | 0.000109000 |
| 43 | 43 | 243 | 243 | 243 | 0 | 0.000109000 |
| 44 | 44 | 356 | 356 | 356 | 0 | 0.000149500 |
| 45 | 45 | 239 | 239 | 239 | 0 | 0.000117000 |
| 46 | 46 | 301 | 301 | 301 | 0 | 0.000118000 |
| 47 | 47 | 284 | 284 | 284 | 0 | 0.000160000 |
| 48 | 48 | 287 | 287 | 287 | 0 | 0.000124500 |
| 49 | 49 | 275 | 275 | 275 | 0 | 0.000134500 |
| 50 | 50 | 314 | 314 | 314 | 0 | 0.000125000 |
| 51 | 51 | 291 | 291 | 291 | 0 | 0.000162000 |
| 52 | 52 | 358 | 358 | 358 | 0 | 0.000135500 |
| 53 | 53 | 404 | 404 | 404 | 0 | 0.000130500 |
| 54 | 54 | 278 | 278 | 278 | 0 | 0.000134000 |
| 90 | 90 | 591 | 591 | 591 | 0 | 0.000233000 |
| 120 | 120 | 701 | 701 | 701 | 0 | 0.000335500 |
| 150 | 150 | 770 | 770 | 770 | 0 | 0.000483500 |
| 180 | 180 | 965 | 965 | 965 | 0 | 0.000588500 |

| Input sizes | | Heuristics | | | | |
|---|---|---|---|---|---|---|
| n_packs | n_bins | f_min | f_max | f_avg | std_dev | t_avg(s) |
| 210 | 210 | 1098 | 1098 | 1098 | 0 | 0.000721500 |
| 240 | 240 | 1161 | 1161 | 1161 | 0 | 0.000794500 |
| 270 | 270 | 1243 | 1243 | 1243 | 0 | 0.001095500 |
| 300 | 300 | 1817 | 1817 | 1817 | 0 | 0.001208500 |
| 330 | 330 | 1942 | 1942 | 1942 | 0 | 0.001404500 |
| 360 | 360 | 2052 | 2052 | 2052 | 0 | 0.001529000 |
| 550 | 550 | 2841 | 2841 | 2841 | 0 | 0.003081500 |
| 600 | 600 | 3633 | 3633 | 3633 | 0 | 0.004040500 |
| 650 | 650 | 3775 | 3775 | 3775 | 0 | 0.004719500 |
| 700 | 700 | 3835 | 3835 | 3835 | 0 | 0.005014500 |
| 750 | 750 | 4017 | 4017 | 4017 | 0 | 0.005300000 |
| 800 | 800 | 4489 | 4489 | 4489 | 0 | 0.006496500 |
| 850 | 850 | 4503 | 4503 | 4503 | 0 | 0.007256500 |
| 900 | 900 | 4625 | 4625 | 4625 | 0 | 0.007605000 |
| 950 | 950 | 5308 | 5308 | 5308 | 0 | 0.008947000 |
| 10000 | 10000 | 53952 | 53952 | 53952 | 0 | 0.774925500 |
| 1000 | 1000 | 5579 | 5579 | 5579 | 0 | 0.010288000 |
| 2000 | 2000 | 11217 | 11217 | 11217 | 0 | 0.032937001 |
| 3000 | 3000 | 16019 | 16019 | 16019 | 0 | 0.057311000 |
| 4000 | 4000 | 21701 | 21701 | 21701 | 0 | 0.101622500 |
| 5000 | 5000 | 27568 | 27568 | 27568 | 0 | 0.127237998 |
| 6000 | 6000 | 32746 | 32746 | 32746 | 0 | 0.178141996 |
| 7000 | 7000 | 38368 | 38368 | 38368 | 0 | 0.224725999 |
| 8000 | 8000 | 43364 | 43364 | 43364 | 0 | 0.288765997 |
| 9000 | 9000 | 49632 | 49632 | 49632 | 0 | 0.362545997 |

Table 3: Results of Heuristics solver

## 3.2 Analysis

### 3.2.1 Exact solution

Our evaluation of the accuracy of each algorithm revealed that CP and MIP were capable of generating exact solutions for a subset of test cases:

- CP was able to produce exact solutions for tests with sizes: $(7 * 3)$, $(10 * 10)$, $(11 * 11)$, $(12 * 12)$, $(13 * 13)$, $(14 * 14)$, $(15 * 15)$, $(16 * 16)$, $(17 * 17)$, $(19 * 19)$, $(20 * 20)$ and $(21 * 21)$.
- MIP was only able to produce exact solutions for tests with sizes: $(7*3)$, $(10*10)$, $(11*11)$, $(12*12)$ and $(14 * 14)$.

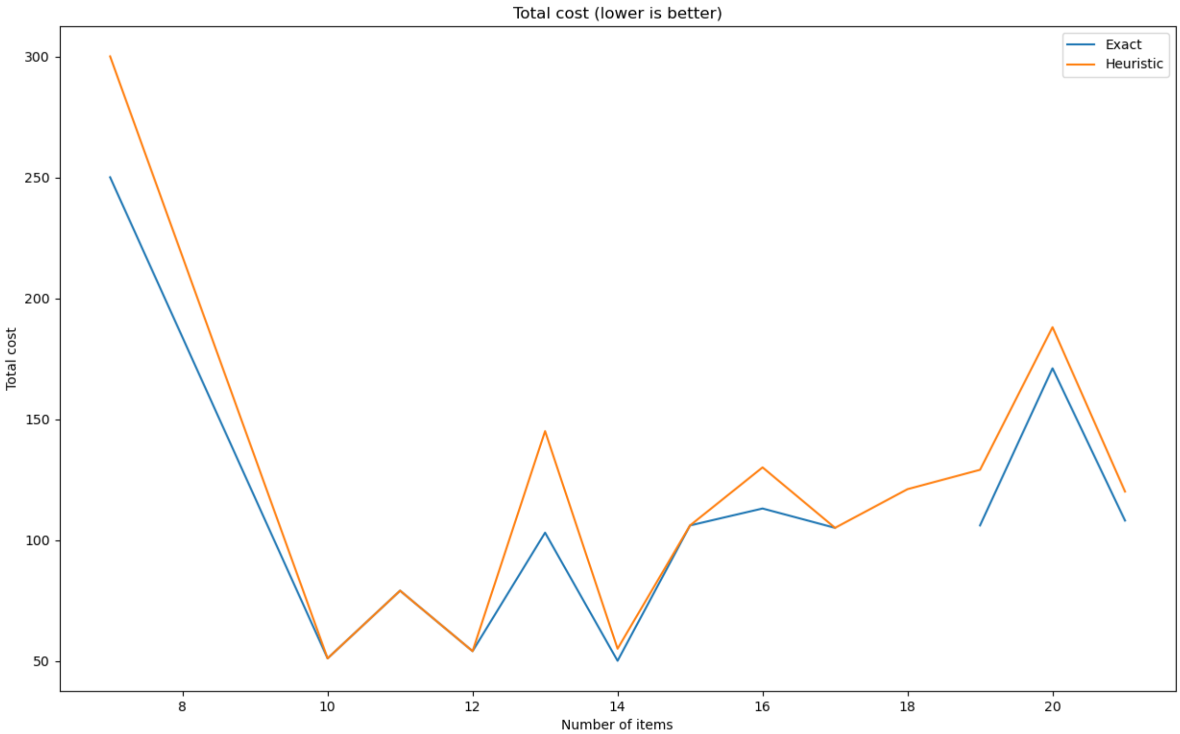



Figure 1: Results compare only exact solution

### 3.2.2 All solution

Our analysis of the solutions generated by three distinct algorithms revealed the following:

- CP is unable to handle data sets larger than $(240 * 240)$.
- MIP is unable to handle data sets larger than $(44 * 44)$.
- The Heuristic algorithm is capable of handling all test cases, including the largest test size of $(10,000 * 10,000)$.

In terms of total cost, our findings indicate:

- MIP performs the worst.
- CP1 and CP2 generate nearly equivalent results.
- With larger data sets, CP2 performs better than CP1.
- The Heuristic algorithm generates better results and significantly outperforms all other algorithms for all test cases larger than $(100 * 100)$ , although it is slightly inferior to CP for test cases smaller than $(100 * 100)$.

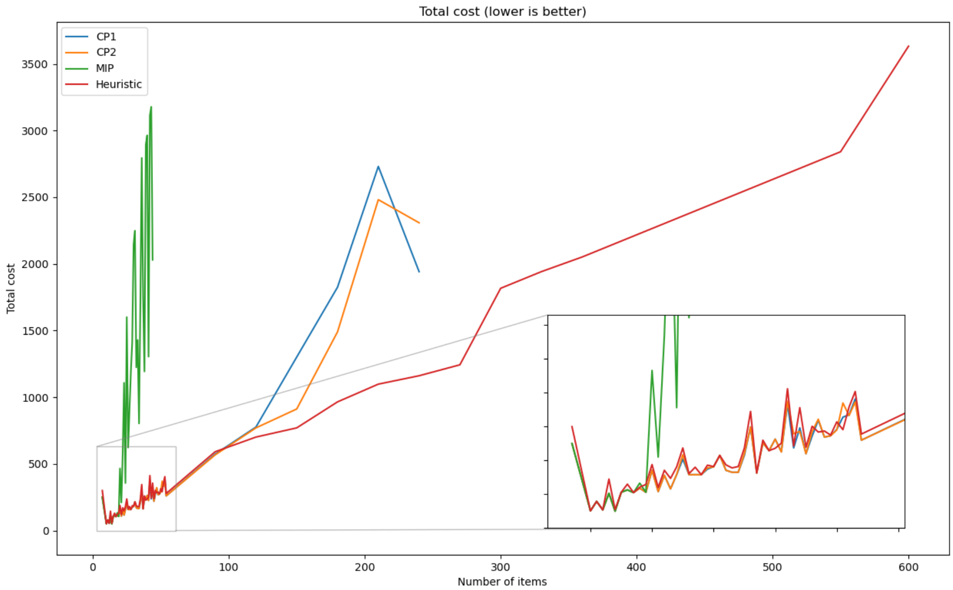



Figure 2: Results compare all solution

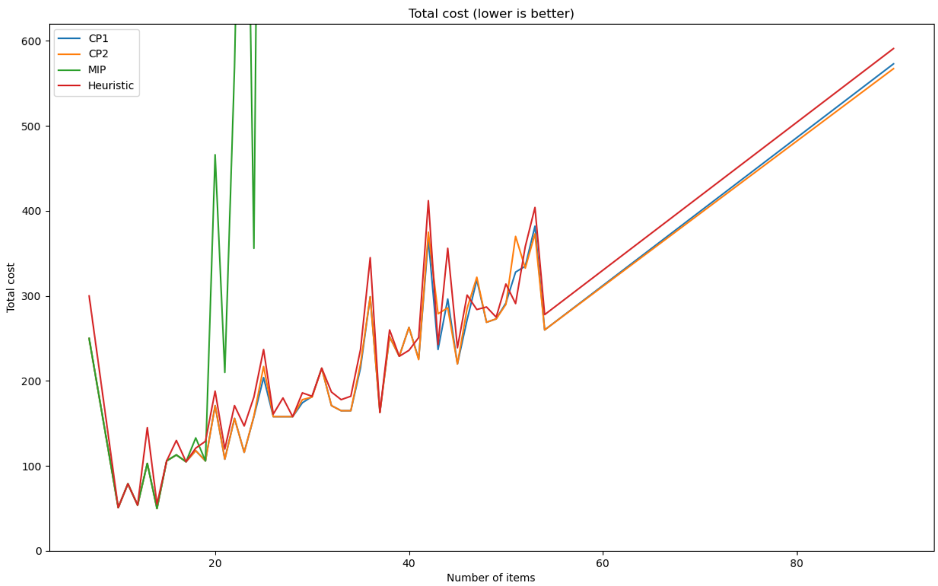



Figure 3: Zoomed results compare all solution

#### 3.2.3 Running time

The running time of each algorithm was also evaluated, and our findings indicate that:

- MIP reaches the time limit of 300 seconds for all tests with size greater than or euqal $(15 * 15)$.
- CP reaches the time limit of 300 seconds for all tests with size greater than or euqal $(22 * 22)$.
- The Heuristic algorithm has a remarkably short run time, with every test completing in under 1 second, even for the largest test size of $(10,000 * 10,000)$.

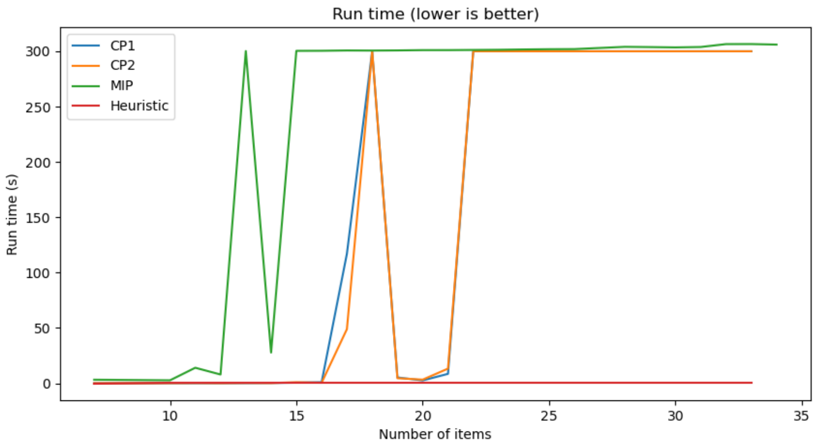



Figure 4: Run time of the first 25 tests

**Based on our analysis, we can conclude that:**

- **MIP consistently performs the worst, as it reaches the time limit of 300 seconds and generates the worst results.**
- **CP is a better option than MIP in terms of run time and cost, CP1 generating nearly equivalent results to CP2 and having a faster run time for smaller data sets.**
- **However, the Heuristic algorithm outperforms both MIP and CP in terms of efficiency, consistently generating good results and having a very short run time for all test cases.**

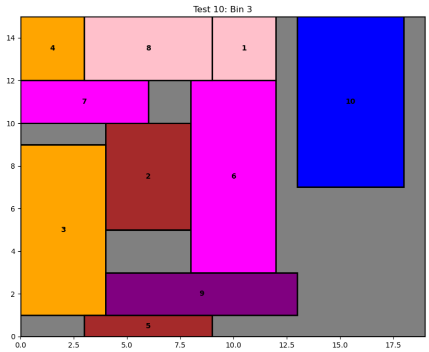
# 4 Conclusion

These results indicate that the Heuristic algorithm is not only highly accurate but also highly efficient, making it a valuable tool for practical applications where both accuracy and speed are essential. Future research could explore ways to further optimize the algorithm's performance while maintaining its accuracy and efficiency.

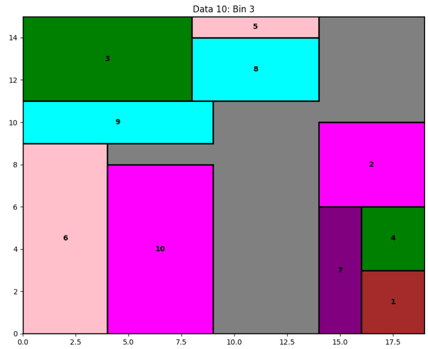
# 5 Visualization

We generated figures to display the results of the CP solver and the Heuristic solver. However, due to the MIP solver's long running time and poor performance, we did not generate any figures for it. Therefore, it is not necessary to include MIP solver results in the figures.

Below are some figures comparing the results of the CP and Heuristic solvers.

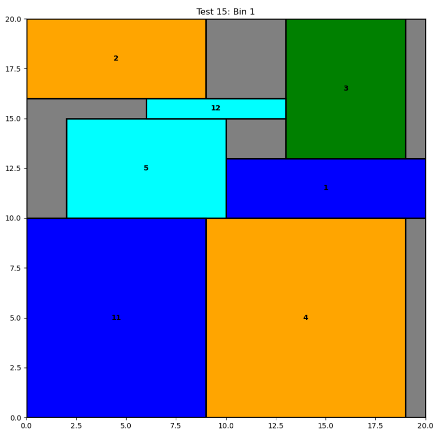



(a) CP solver.

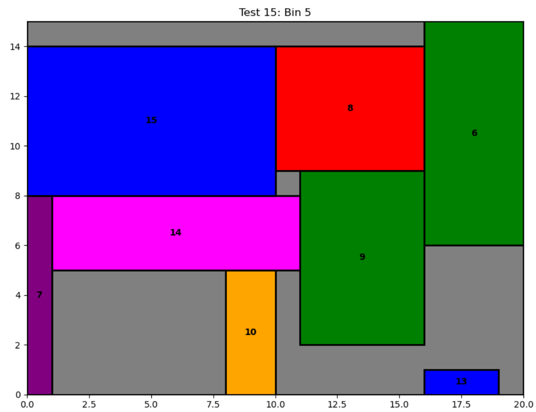



(b) Heuristic solver.

Figure 5: Result for test size $(10 * 10)$.

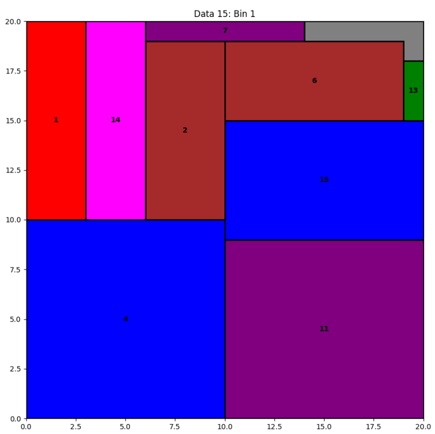



(a) First bin.

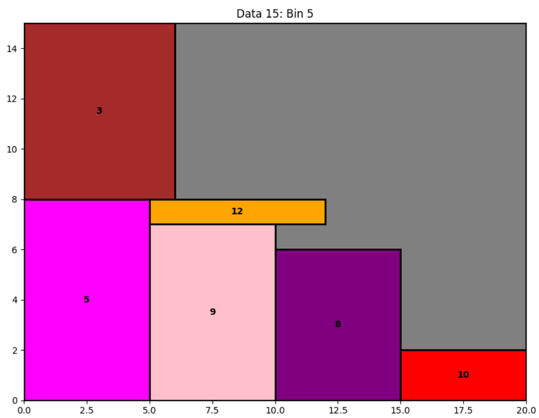



(b) Second bin.

Figure 6: Result for test size $(15 * 15)$ by CP solver.




(a) First bin.




(b) Second bin.

Figure 7: Result for test size $(15 * 15)$ by Heuristic solver.

## References

[1] Jukka Jylänki. A thousand ways to pack the bin - a practical approach to two-dimensional rectangle bin packing. 2010. Available online: `http://pds25.egloos.com/pds/201504/21/98/RectangleBinPack.pdf` (Accessed on February 8, 2023).