1 Description of the algorithm

1.1 Assumptions

1. Goal of the algorithm: minimizing the number of pallet load units.
2. The basis for planning is the PRODUCT_ID set in the order, the number of which is smaller than the standard number.
3. The number of PRODUCT_ID units in a shipment (WMS) is not subject to division.
4. Adding subsequent PRODUCT_ID layers to the mix takes into account:
   - height (H) of the layer,
   - degree of layer filling (fw),
   - fragility (stackability) parameter (CL),
   - weight (W).

1.2 Input data

- order (ORDER_ID)
- place of shipment (DB)
- product name (PRODUCT_ID)
- of product units in the order (BU_QUANTITY)
- number of product units per standard volume (Quantity_Max)
- number of layers in the normative pjl (Layers_Norm)
- product fragility (CAPACITY_LOAD_CLASS)

1.3 Output

- pallet loading unit number (LOADUNIT_ID)
- weight of pallet loading unit (WEIGHT) with carrier [mm]
- height of pallet loading unit (HEIGHT) with carrier [kg]
- type of pallet loading unit (TOP or BASE) due to the possibility of creating sandwich pallets (SNDW)

1.4 Indexes

- i – product included in the order, where $i = 1, 2, \ldots n, \ldots I$
- j – pallet mix type loading unit
- w – layer on a pallet mix loading unit
- z – client's order

1.5 Parameters

- CL_i – fragility class of the ith product (according to the standard)
- CL_j – fragility class of the jth pallet loading unit determined as the highest CLi value of the PRODUCT_ID items included in the jth pallet loading unit mix
- fw – layer filling degree; parameter subject to control $fw =< 0.7; 1 >$
- H_i – unit height of the ith product (according to the standard)
- H_j – height of the jth pallet load unit mix
- ^{Hmax} – maximum height of a pallet load unit mix (system parameter)
- H^{diff} – difference between the maximum height of the pallet loading unit mix $Hmax$ and the height of the jth pallet loading unit mix H_j
- LP_i – number of units of the ith product in the order
- LP_i^{nor} – number of units of the ith product on a standard pallet loading unit
- LW_i^{nor} – number of layers of the ith product in a standard pallet loading unit
- S_j – status of the $j-th$ pallet loading unit mix due to the possibility of being included in the planning of the construction of the SNDW T – pallet loading unit TOP, i.e. it can only be placed on the top of the SNDW; B – BASE pallet loading unit, i.e. can be placed as a base or intermediate layer
- w_i – number of layers of the ith product in the order, where $w_i = w_i^{INT} + w_i^R$
- w_i^{INT} – total value of the number of layers of the ith product in the order
- w_i^R – non-integer value of the number of layers of the ith product in the order
- W_i – weight of the ith product in the order

- W_i^{norm} – standard weight of a pallet loading unit with the ith product
- W_j – weight of the jth pallet load unit mix
- W_j^{zpk} – the difference in the weight of the standard pallet loading unit with the ith Winorm product and the weight of the jth mix pallet loading unit, determined when breaking the fragility parameter
- W_n – weight of n products in the order
- W^{max} – maximum weight of a pallet load unit mix (system parameter)
- W^{diff} – difference between the maximum weight of the pallet loading unit mix W^{max} and the weight of the jth pallet loading unit mix W_j
- ZP – a set of products for planning a pallet load unit mix

1.6 Suggested parameter values

- $H^{max} = 1700[mm]$
- $W^{max} = 700[kg]$
- $f_w \in [0.7\ 1.0]$

2 Algorithm

Algorithm 1 Palet Stacking algorithm

Input: $z, CL(i), ZP, H^{max}, W^{max}, f_{wmin}, f_{wmax}, h_{diff}$
Output: $SetSofpalletlabeledas, , Top''or, , Base''$

1: $j \leftarrow 0$
2: $S \leftarrow \emptyset$
3: **for each** $z \in$Z **do**
4: **for each** $i \in z$ **do**
5: $l_i^{INT} \leftarrow int(\frac{LP_i}{LP_i^{NOR} * LW_i^{NOR}})$
6: $l_i^R \leftarrow \frac{LP_i}{LP_i^{NOR} * LW_i^{NOR}} - l_i^{INT}$
7: $l_i^{Planned}(1 : l_i^{INT} + l_i^R) \leftarrow false$
8: $Added_i \leftarrow false$
9: $l_i = l_i^{INT} + l_i^R$
10: $fw_i^{INT} \leftarrow 0$ ▷ resztkowy spolczynnik wypelnienia warstwy pelnej
11: $fw_i^R \leftarrow 1 - l_i^R$ ▷ resztkowy wspolczynnik wypelnienia warstwy niepelnej
12: **end for each**
13: **for each** $i \in$z **do**
14: **if** $fw_i^{INT} = 0$ **and** $fw_i^R = 0$ **then**
15: $S \leftarrow S \cup Make\_full\_layers\_pallet(i, CL(i), H^{max}, W^{max}, j)$ ▷ Tworzenie palet z warstw pelnych
16: $Added_i = true$
17: **go to** 22
18: **else**
19: **go to** 22
20: **end if**
21: **end for each**
22: **for each** $i \in$z **do**
23: $sort(i, CL(i), 1)$ ▷ Sortowanie rosnaco po klasie kruchosci
24: $sort(i, CL(i), w_i, -1)$ ▷ W obrebie kazdej klasy kruchosci sortowanie malejaco po liczbie warstw w zamowieniu li
25: **end for each**
26: **for each** $i \in$z **do**
27: **if** $\exists\, i : l_i^{Planned}(:) = false \& l_i^{Planned}(:) = true$ **then**
28: **go to** 50
29: **else**
30: **if** $\exists_i : l_i^{Planned}(1 :) = false$ **then**
31: **go to** 74
32: **else**
33: **for each** $j \in$S **do**
34: **if** $|W_j - W_{\neq j}| < W^{diff} \& |H_j - H_{\neq j}| < H^{diff}$ **then**
35: **go to** 300
36: **else**
37: $W^{max} \leftarrow 0.99 * W^{max}$

38: $H^{max} \leftarrow 0.99 * H^{max}$
39: **go to** 13
40: **end if**
41: **end for each**
42: **end if**
43: **end if**
44: **for each** $j \in$S **do**
45: $W_j^{mix} = \frac{\sum_{j=1}^{J} W_j}{W_{max}}$
46: $H_j^{mix} = \frac{\sum_{j=1}^{J} H_j}{H_{max}}$
47: **end for each**
48: $W^{max} = max_j(W_j^{mix})$
49: $H^{max} = max_j(H_j^{mix})$
50: **for each** $j \in$S **do**
51: $W_j^{diff} = W^{max} - W_j$
52: $H_j^{diff} = H^{max} - H_j$
53: **end for each**
54: **for each** $i \in$z **do**
55: **if** $W_i <= W^{diff} \| H_i <= H^{diff}$ **then**
56: **go to** 62
57: **else**
58: $S_j \leftarrow, , BASE''$ ▷ Pallet is builded
59: **go to** 26
60: **end if**
61: **end for each**
62: **for each** $j \in$S **do**
63: $H_j^{diff} = H^{max} - H_j$
64: $W_j^{diff} = W^{max} - W_j$
65: **for each** $i \in$z **do**
66: **if** $H_i <= H_{diff} \| W_i <= W_{diff}$ **then**
67: **go to ??**
68: **else**
69: $S_j \leftarrow, , BASE''$ ▷ Pallet is builded
70: **go to** 26
71: **end if**
72: **end for each**
73: **end for each**
74: **for each** $i \in$z **do**
75: **if** $Added_i = false \& fw_i \in [0.7 : 1.0]$ **then**
76: **go to** 82
77: **else**
78: **go to** 175
79: **end if**
80: **end for each**
81: $count \leftarrow 0$

```
82:     for each i ∈z do
83:         if fw_i ∈ [0.7 : 1.0] then
84:             count ← count + 1
85:         end if
86:     end for each
87:     if count >= 2 then
88:         go to 92
89:     else
90:         go to 143
91:     end if
92:     for each i ∈ z&fw_i ∈ [0.7 : 1.0] do
93:         if ∃_n : ||H_i − H_{i+n}|| <= 10 then ▷ Warunek polaczenia dwoch pozycji w warstwe celem dodania do palety
94:             if Σ_{k=1}^{I} W_k > W^{max} || Σ_{k=1}^{I} H_k > H^{max} then
95:                 go to 74 ▷ Dodanie polaczonych warstw przekroczylo wartosci maksymalne
96:             end if
97:             if Σ_{K=1}^{I} W_k <= W^{max} & Σ_{k=1}^{I} H_k <= H^{max} then
98:                 if fw_i + fw_{i+n} > 1.0 then ▷ Stopien wypelnienia warstwy wspolnej
99:                     go to 74
100:                else
101:                    if ∃_i : planned_i = false then ▷ Czy sa kolejne pozycje w zamowieniu
102:                        for each j ∈ S do
103:                            if W_j+W_i >= W_max || W_j+W_i >= W_max || CL_i >= CL_j then ▷ Mozliwosc dodania do istniejacej palety
104:                                if fw_j + fw_i <= 1 then
105:                                    S_j ∪ Product_ID(S_j, i)
106:                                    if fw_i + fw_{i+n} < 0.7 then
107:                                        i \ i + n
108:                                        go to 74
109:                                    else
110:                                        S_j = BASE
111:                                        go to 30
112:                                    end if
113:                                else
114:                                    if fw_i + fw_{i+n} < 0.7 then
115:                                        i \ i + n
116:                                        go to 74
117:                                    end if
118:                                    if fw_i + fw_{i+n} ∈ [0.7; 1.0] then
119:                                        S_j = BASE
120:                                        go to 30
121:                                    end if
122:                                end if
```

```
                        else
                            if  fw_i + fw_{i+n} < 0.7  then
                                i \ i + n
                                go to 74
                            else
                                if  fw_i + fw_{i+n} ∈ [0.7; 1.0]  then
                                    S_j = BASE
                                    go to 30
                                end if
                            end if
                        end if
                    end for each
                else
                end if
            end if
        end if
    else
        go to 30
    end if
end for each
if  i : fw_i ∈ [0.7; 1.0]&∃_n CL_i < CL_{i+n} then
    if  fw_i + fw_{i+n} <= 1  then
        if  Σ_{i=1}^{I} W_i < W^{max} & Σ_{i=1}^{I} H_i < H^{max}  then
            if  fw < 0.7  then
                S_j ∪ Product_ID(S_j, i)
                S_j = TOP
                go to 30
            end if
            if  fw ∈ [0.7; 1.0] then
                S_j ∪ Product_ID(S_j, i)
                if  Try_Pillars(S_j) = true  then
                    S_j ← TOP
                    go to 30
                else
                    S_j ← BASE
                    go to 30
                end if
            end if
        end if
    end if
else
    Make_new_Pallet(Product_ID_i)
    j ← j + 1
    if  fw_i < 0.7  then
        S_j = TOP
```

168: **go to** 30
169: **end if**
170: **if** $fw_i \in [0.7; 1.0]$ **then**
171: $S_j = base$
172: **go to** 30
173: **end if**
174: **end if**
175: **for each** $i \in z$ **do**
176: **if** $\exists_n : \|H_i - H_{i+n}\| <= 10$ **then** ▷ Warunek polaczenia dwoch pozycji w warstwe celem dodania do palety
177: **go to** 182
178: **else**
179: **go to** 30
180: **end if**
181: **end for each**
182: $W^{sum} \leftarrow 0$
183: $top\_layer \leftarrow \emptyset$
184: $fw_{top\_layer} \leftarrow 0$
185: **for each** $i \in z$ **do**
186: **if** $\|H_i - H_{i+n} ¡ 10$ **then** ▷ Mozliwosc polaczenia
187: $top\_layer \leftarrow top\_layer \cup i + n$
188: $W^{sum} \leftarrow W^{sum} + W_{i+n}$
189: $fw_{top\_layer} \leftarrow fw_{top\_layer} + fw_{i+w}$
190: **end if**
191: **end for each**
192: **if** $W^{sum} + W_j < W^{max}$ **then**
193: **if** $fw_{top\_layer} < 1 \& H_j + H_{i+n} < H^{max}$ **then**
194: $Create\_Pillars()$
195: **end if**
196: **if** $fw_{top\_layer} < 0.7$ **then**
197: $S_j \cup top\_layer$
198: $S_j = TOP$
199: **go to** 175
200: **end if**
201: **if** $fw \in [0.7; 1.0]$ **then**
202: $S_j \cup top\_layer$
203: **if** $try\_pillars(S_j) = true$ **then**
204: $S_j = TOP$
205: **go to** 175 ▷ Koniec budowy
206: **else**
207: $S_j = BASE$
208: **go to** 175
209: **end if**
210: **end if**
211: **else**
212: **while** $W^{sum} + W_j > W^{max}$ **do** ▷ usuwamy do limitu wagi

213: $top\_layer \leftarrow top\_layer \setminus i + n$
214: $W^{sum} \leftarrow W^{sum} - W_{i+n}$
215: $fw_{top\_layer} \leftarrow fw_{top\_layer} - fw_{i+w}$
216: **end while**
217: **if** $fw_{top\_layer} < 1 \& H_j + H_{i+n} < H^{max}$ **then**
218: $Create\_Pillars()$
219: **end if**
220: **if** $fw_{top\_layer} < 0.7$ **then**
221: $S_j \cup top\_layer$
222: $S_j = TOP$
223: **go to** 175
224: **end if**
225: **if** $fw \in [0.7; 1.0]$ **then**
226: $S_j \cup top\_layer$
227: **if** $try\_pillars(S_j) = true$ **then**
228: $S_j = TOP$
229: **go to** 175 ▷ Koniec budowy
230: **else**
231: $S_j = BASE$
232: **go to** 175
233: **end if**
234: **end if**
235: **end if**
236: $CL_j \leftarrow 0$
237: **for each** $Product\_ID \in S_j$ **do**
238: **if** $CL_i > CL_j$ **then**
239: $CL_j \leftarrow CL_i$
240: **end if**
241: **end for each**
242: **end for each**
243: **if** $Product\_ID \in LP_i = 0$ **then** ▷ Sprawdz liczbe pozycji w zamowieniu
244: **go to** 26
245: **end if**
246: **if** $Product\_ID \in LP_i = 1$ **then**
247: **go to** 252
248: **end if**
249: **if** $Product\_ID \in LP_i > 1$ **then**
250: **go to** 267
251: **end if**
252: **if** $CL_i < CL_j$ **then**
253: **if** $\exists_j : Considered_j = false$ **then**
254: $S_j = BASE$
255: $Considered_j = true$
256: **go to** 50
257: **else**

258: $S_j = BASE$
259: $j \leftarrow j + 1$
260: $S_j \cup Product\_ID$
261: **go to** 74
262: **end if**
263: **else**
264: $S_j \cup Product_ID$
265: **go to** 26
266: **end if**
267: **if** $CL_1 = CL_2 = CL_n = CL_i \& CL_i < CL_j$ **then**
268: **go to** 282
269: **end if**
270: **if** $CL_1 = CL_2 = CL_n = CL_i \& CL_i \geq CL_j$ **then**
271: **go to ??**
272: **end if**
273: **if** $CL_1 \neq CL_2 \neq CL_n \neq CL_i \& CL_i < CL_j$ **then**
274: **go to** 282
275: **end if**
276: **if** $CL_1 \neq CL_2 \neq CL_n \neq CL_i \& CL_i \geq CL_j$ **then**
277: **go to ??**
278: **end if**
279: **if** $CL_1 \neq CL_2 \neq CL_n \neq CL_i \& CL_i \geq CL_j \& CL_n < CL_j$ **then**
280: **go to ??**
281: **end if**
282: **if** $\exists_i : fw_i \in [0.7; 1.0]$ **then** ▷ Czy istnieje Product_ID o pelnej warstwie
283: **go to ??**
284: **else**
285: **if** $LP_i = 1$ **then** ▷ Sprawdz liczbe pozycji w zamowieniu
286: **if** $fw < 0.7$ **then**
287: **go to** 26
288: **end if**
289: **if** $fw \in [0.7; 1.0]$ **then**
290: $S_j \cup LP_i$ ▷ zgodnie z sekwencja CL
291: **go to** 26
292: **end if**
293: **else**
294: **if** $LP_i > 1$ **then**
295: **go to ??**
296: **end if**
297: **end if**
298: **end if**
299: **end for each**
300: *STOP*