2-D Stock Cutting Problem

Quantities Defined

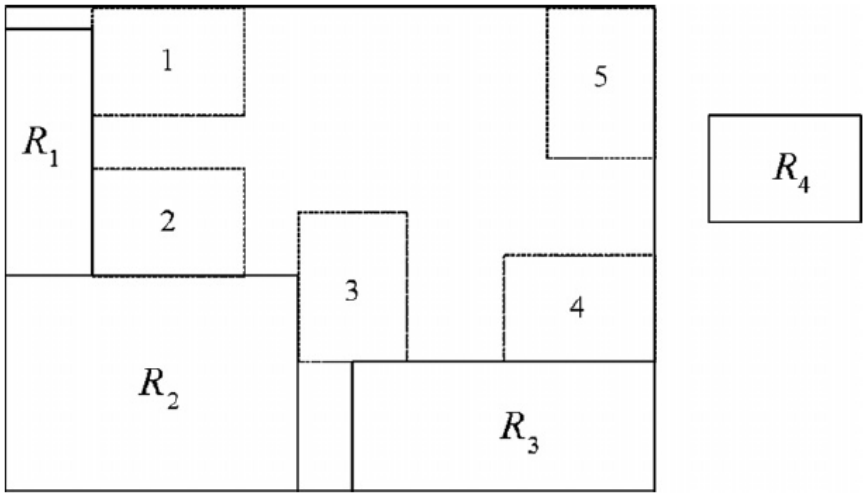
Configuration:-

A configuration *C* is a pattern (layout) where m ($0 < m < n$) rectangles have been already packed inside the container without overlap, and $n - m$ rectangles remain to be packed into the container.

A configuration is said to be successful if $m = n$, i.e., all the rectangles have been placed inside the container without overlapping. A configuration is said to be failure if $m < n$ and none of the rectangles outside the container can be packed into the container without overlapping. A configuration is said to be final if it is either a successful configuration or a failure configuration.

Candidate corner-occupying action (CCOA):-

Given a configuration with *m* rectangles packed, there may be many empty corners formed by the previously packed rectangles and the four sides of the container. Let rectangle *i* be the current rectangle to be packed, a candidate corner-occupying action (CCOA) is the placement of rectangle *i* at an empty corner in the container so that rectangle *i* touches the two items forming the corner and does not overlap other previously packed rectangles (an item may be a rectangle or one of the four sides of the container). Note that the two items are not necessarily touching each other.



Example of Candidate corner-occupying action (CCOA) for rectangle R_4

Obviously, the rectangle to be packed has two possible orientation choices at each empty corner, that is, the rectangle can be placed with its longer side laid horizontally or vertically. A CCOA can be represented by a quadri-tuple (*i, x, y, h*), where (*x, y*) is the coordinate of the bottom-left corner of the suggested location of rectangle *i* and *h* is the corresponding orientation.

Minimal distance between rectangles:-

Let *i* and *j* be two rectangles already placed in the container, and (*xi, yi*), (*xj, yj*) are the coordinates of an arbitrary point on rectangle *i* and *j*, respectively. The minimal distance *dij* between *i* and *j* is:

$$d_{ij} = \min\left\{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}\right\}$$

In Figure, R3 is packed on the position occupying the corner formed by the upper side and the right side of the container. As shown in Figure, the minimal distance between R3 and R1, and the minimal distance between R3 and R2 are illustrated, respectively.

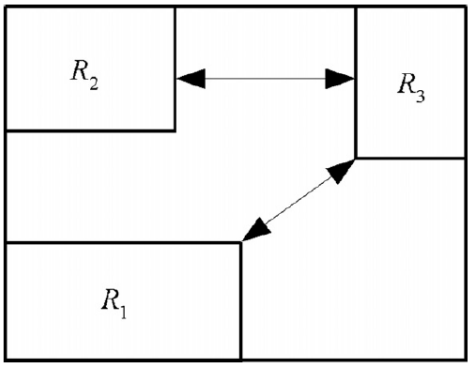


Illustration of distance

Degree of CCOA:-

Let *M* be the set of rectangles already placed in the container. Rectangle *i* is the current rectangle to be packed, (*i, x, y, h*) is one of the CCOAs for rectangle *i*. If corner-occupying action (*i, x, y, h*) places rectangle *i* at a corner formed by two items (rectangle or side of the container) *u* and *v*, the degree *k* of the corner-occupying action (*i, x, y, h*) is defined as:

$$\lambda = 1 - d_{\min} \Big/ \left(\frac{w_i + l_i}{2}\right)$$

where w_i and l_i are the width and the length of rectangle *i*, and d_{min} is the minimal distance from rectangle *i* to other rectangles in *M* and sides of the container (excluding *u* and *v*), that is,

$$d_{\min} = \min\left\{d_{ij} | j \in M \cup \{s_1, s_2, s_3, s_4\}, j \neq u, v\right\}$$

where s_1, s_2, s_3 and s_4 are the four sides of the container.

Algorithm:-

INPUT

- Stock- Length and Breadth
- Rectangles- Length, Breadth and Numbers

Step 1

- Generate initial CCOA

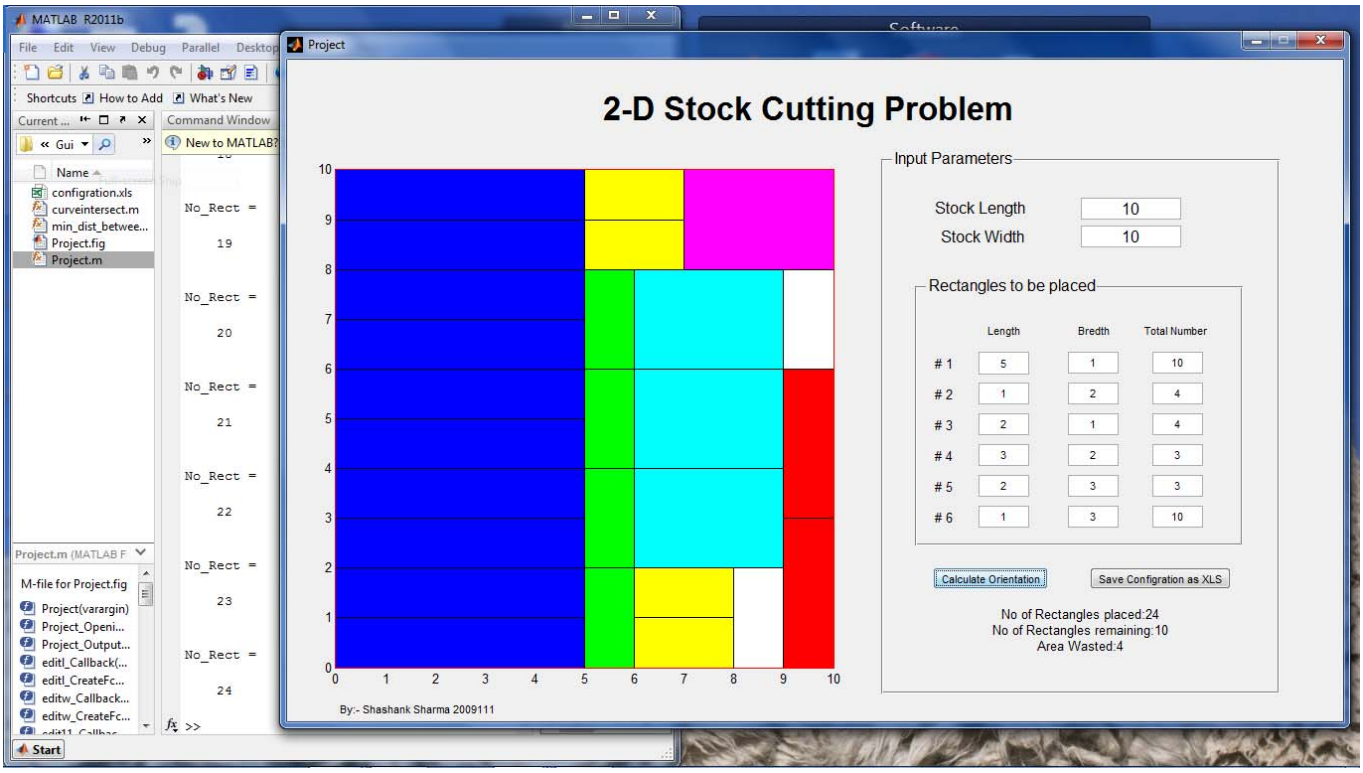
Step 2

- Repeat till no CCOA available
  - Calculate Degree of CCOA available
  - Selecting Correct CCOA and inserting it into Configuration
  - Check for CCOA for next iteration by:-
    - traversing over the area of Stock
    - Checking for Minimum Distance with other entities in the Configuration
    - Checking for Overlapping with other entities in the Configuration

OUTPUT

- Plotting the Final Configuration Matrix

Implementation:-



Program:-

```
% 2-D STOCK CUTTING PROBLEM
% x-axis=-4
% y-axis=-3
% ||to x-axis=-2
% ||to y-axis=-1

clc
clear all
close all

% Input for stock and rectangles
stock_l=6;
stock_w=8;
Rect=xlsread('rectangles',-1);
[Type,y]=size(Rect);

axis_n4.x=[0
    stock_l];
axis_n4.y=[0
    0];
axis_n3.x=[0
    0];
axis_n3.y=[0
    stock_w];
axis_n2.x=[0
    stock_l];
axis_n2.y=[stock_w
    stock_w];
axis_n1.x=[stock_l
    stock_l];
axis_n1.y=[0
    stock_w];

C=[];
CCOA=[];

% Generate initial CCOA
for i=1:Type
    if Rect(i,1)<stock_l && Rect(i,2)<stock_w
        CCOA=[CCOA
            i 0 0 0 -3 -4];
    end
    if Rect(i,1)<stock_w && Rect(i,2)<stock_l
        CCOA=[CCOA
            i 0 0 1 -3 -4];
    end
end

[a,b]=size(CCOA);
rect_i=[];
rect_j=[];
d_min=1000;
No_Rect=0;
Degree=[];
on=0;
overlap=[];
noverlap=0;
t1=0;t2=0;

% Loop executes till CCOA are available
while a ~= 0
    % loop for all CCOA to calculate degree from 1 to last CCOA
    for i=1:a
        %extracting rect coordinates according to orientation
        if CCOA(i,4)==0
            rect_i.x=[CCOA(i,2)
                CCOA(i,2)
                CCOA(i,2)+Rect(CCOA(i,1),1)
                CCOA(i,2)+Rect(CCOA(i,1),1)];
```

```
            rect_i.y=[CCOA(i,3)
                CCOA(i,3)+Rect(CCOA(i,1),2)
                CCOA(i,3)+Rect(CCOA(i,1),2)
                CCOA(i,3)];
        else
            rect_i.x=[CCOA(i,2)
                CCOA(i,2)
                CCOA(i,2)+Rect(CCOA(i,1),2)
                CCOA(i,2)+Rect(CCOA(i,1),2)];
            rect_i.y=[CCOA(i,3)
                CCOA(i,3)+Rect(CCOA(i,1),1)
                CCOA(i,3)+Rect(CCOA(i,1),1)
                CCOA(i,3)];
        end

        %checking distances with edges and getting MIN DISTANCE
        % x-axis=-4        y-axis=-3         ||to x-axis=-2           ||to y-axis=-1
        if CCOA(i,5)~=-1 || CCOA(i,6)~=-1
            d=min_dist_between_two_polygons(axis_n1,rect_i);
            if d<d_min
                d_min=d;
            end
        end
        if CCOA(i,5)~=-2 || CCOA(i,6)~=-2
            d=min_dist_between_two_polygons(axis_n2,rect_i);
            if d<d_min
                d_min=d;
            end
        end
        if CCOA(i,5)~=-3 || CCOA(i,6)~=-3
            d=min_dist_between_two_polygons(axis_n3,rect_i);
            if d<d_min
                d_min=d;
            end
        end
        if CCOA(i,5)~=-4 || CCOA(i,6)~=-4
            d=min_dist_between_two_polygons(axis_n4,rect_i);
            if d<d_min
                d_min=d;
            end
        end

        %extracting rect coordinates of other rect in configration and
        %getting MIN DISTANCE from other rectangles
        for j=0:No_Rect-1
            rect_j.x=[C(j+1,1)
                C(j+1,1)
                C(j+1,3)
                C(j+1,3)];
            rect_j.y=[C(j+1,2)
                C(j+1,4)
                C(j+1,4)
                C(j+1,2)];
            if CCOA(i,5)~=j+1 || CCOA(i,6)~=j+1
            d=min_dist_between_two_polygons(rect_j,rect_i);
            if d<d_min
                d_min=d;
            end
            end
        end

        %Calculation of DEGREE for EACH CCOA
        Degree(i)=1-d_min/((Rect(CCOA(i,1),1)+Rect(CCOA(i,1),2))/2);
    end

    [val,ind] = max(Degree);
    No_Rect=No_Rect+1
    % Insertion of OPTIMUM CCOA into CONFIGURATION
    if CCOA(ind,4)==0
        C=[C
            CCOA(ind,2) CCOA(ind,3) CCOA(ind,2)+Rect(CCOA(ind,1),1)
CCOA(ind,3)+Rect(CCOA(ind,1),2) CCOA(ind,1)];
    else
        C=[C
            CCOA(ind,2) CCOA(ind,3) CCOA(ind,2)+Rect(CCOA(ind,1),2)
CCOA(ind,3)+Rect(CCOA(ind,1),1) CCOA(ind,1)];
```

```
    end
    % Decreasing the quantity of available rect of type inserted in CCOA
    Rect(CCOA(ind,1),3)=Rect(CCOA(ind,1),3)-1;
    CCOA=[];

    % Loop over the different types of rectangles
    for i=1:Type
        % Check if rectangle of specific type is available or not
        if Rect(i,3) ~=0
            % Traversing over the area to find possible CCOA
            for j=0:1:stock_l-Rect(i,1)
                for k=0:1:stock_w-Rect(i,2)
                    % All entities in the CONFIGRATION are checked if
                    % they touch the prospective CCOA or not
                    for l=-4:1:No_Rect-1
                        rect_i.x=[j
                            j
                            j+Rect(i,1)
                            j+Rect(i,1)];
                        rect_i.y=[k
                            k+Rect(i,2)
                            k+Rect(i,2)
                            k];
                        if l<0
                            if l==-4
                                rect_j.x=axis_n4.x;
                                rect_j.y=axis_n4.y;
                            elseif l==-3
                                rect_j.x=axis_n3.x;
                                rect_j.y=axis_n3.y;
                            elseif l==-2
                                rect_j.x=axis_n2.x;
                                rect_j.y=axis_n2.y;
                            elseif l==-1
                                rect_j.x=axis_n1.x;
                                rect_j.y=axis_n1.y;
                            end
                        else
                            rect_j.x=[C(l+1,1)
                                C(l+1,1)
                                C(l+1,3)
                                C(l+1,3)];
                            rect_j.y=[C(l+1,2)
                                C(l+1,4)
                                C(l+1,4)
                                C(l+1,2)];
                        end
                        d=min_dist_between_two_polygons(rect_j,rect_i);

                        % check if prospective CCOA touches any entity and
                        % if it does it stores its type
                        if d==0
                            on=on+1;
                            if on==1
                                t1=l;
                            else
                                t2=l;
                            end
                        end
                    end

                    % Excutes when a prospective CCOA which touch more than 2 entities is
found
                    if on>=2
                        % Loop checks if prospective CCOA overlaps with
                        % any entity in the CONFIGRATION or not
                        for l=0:1:No_Rect-1
                            rect_j.x=[C(l+1,1)
                                C(l+1,1)
                                C(l+1,3)
                                C(l+1,3)];
                            rect_j.y=[C(l+1,2)
                                C(l+1,4)
                                C(l+1,4)
                                C(l+1,2)];
```

```
overlap=polybool('intersection',rect_j.x,rect_j.y,rect_i.x,rect_i.y);
                            [yyy,zzz]=size(overlap);
                            if zzz ~= 0
                                noverlap=noverlap+1;
                            end
                        end
                        % If no overlap CCOA is confirmed
                        if noverlap == 0
                            CCOA=[CCOA
                                i j k 0 t1 t2];
                        end
                    end
                    on=0;
                    noverlap=0;
                end
            end

            % repeat to check CCOA for perpendicular orientation
            for j=0:1:stock_l-Rect(i,2)
                for k=0:1:stock_w-Rect(i,1)
                    for l=-4:1:No_Rect-1
                        rect_i.x=[j
                            j
                            j+Rect(i,2)
                            j+Rect(i,2)];
                        rect_i.y=[k
                            k+Rect(i,1)
                            k+Rect(i,1)
                            k];
                        if l<0
                            if l==-4
                                rect_j.x=axis_n4.x;
                                rect_j.y=axis_n4.y;
                            elseif l==-3
                                rect_j.x=axis_n3.x;
                                rect_j.y=axis_n3.y;
                            elseif l==-2
                                rect_j.x=axis_n2.x;
                                rect_j.y=axis_n2.y;
                            elseif l==-1
                                rect_j.x=axis_n1.x;
                                rect_j.y=axis_n1.y;
                            end
                        else
                            rect_j.x=[C(l+1,1)
                                C(l+1,1)
                                C(l+1,3)
                                C(l+1,3)];
                            rect_j.y=[C(l+1,2)
                                C(l+1,4)
                                C(l+1,4)
                                C(l+1,2)];
                        end
                        d=min_dist_between_two_polygons(rect_j,rect_i);
                        if d==0
                            on=on+1;
                            if on==1
                                t1=l;
                            else
                                t2=l;
                            end
                        end
                    end
                    if on>=2
                        for l=0:1:No_Rect-1
                            rect_j.x=[C(l+1,1)
                                C(l+1,1)
                                C(l+1,3)
                                C(l+1,3)];
                            rect_j.y=[C(l+1,2)
                                C(l+1,4)
                                C(l+1,4)
                                C(l+1,2)];

overlap=polybool('intersection',rect_j.x,rect_j.y,rect_i.x,rect_i.y);
```

```
                        [yyy,zzz]=size(overlap);
                        if zzz ~= 0
                            noverlap=noverlap+1;
                        end
                    end
                    if noverlap == 0
                        CCOA=[CCOA
                            i j k 1 t1 t2];
                    end
                end
                on=0;
                noverlap=0;
            end
        end

    end
end
[a,b]=size(CCOA);
end

% Plotting of CONFIGURATION
[a,b]=size(C);
for i=1:1:a
    rect_j.x=[C(i,1)
        C(i,1)
        C(i,3)
        C(i,3)
        C(i,1)];
    rect_j.y=[C(i,2)
        C(i,4)
        C(i,4)
        C(i,2)
        C(i,2)];
    switch C(i,5)
        case 1
            patch(rect_j.x,rect_j.y,'b');
            hold on;
        case 2
            patch(rect_j.x,rect_j.y,'g');
            hold on;
        case 3
            patch(rect_j.x,rect_j.y,'r');
            hold on;
        case 4
            patch(rect_j.x,rect_j.y,'c');
            hold on;
        case 5
            patch(rect_j.x,rect_j.y,'m');
            hold on;
        case 6
            patch(rect_j.x,rect_j.y,'y');
            hold on;
        otherwise
            patch(rect_j.x,rect_j.y);
            hold on;
    end
end
```