2-D Stock Cutting Problem

# Quantities Defined

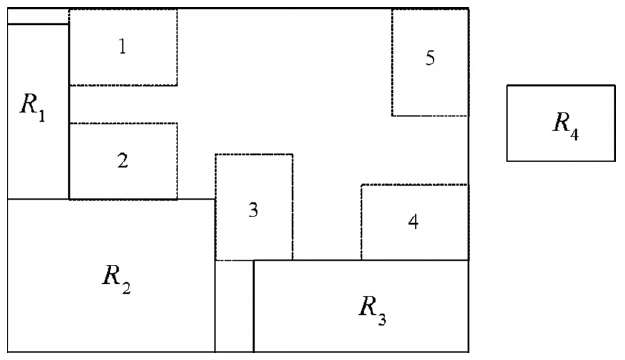
## Configuration:-

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.

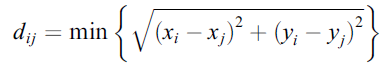


*Example of Candidate corner-occupying action (CCOA) for rectangle R4*

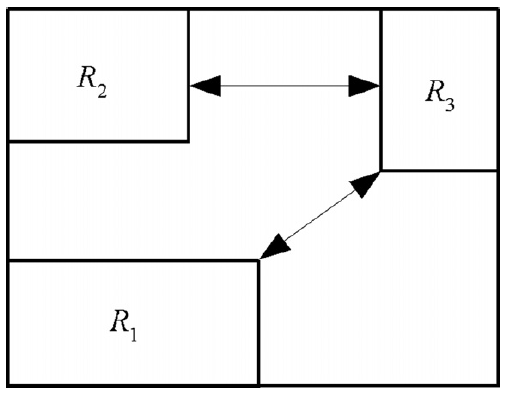
Obviously, the rectangle to be packed has two possible orientation choices at each empty corner, that is, the rectangle can be placed with its longer side laid horizontally or vertically. A CCOA can be represented by a quadri-tuple (*i, x, y, h*), where (*x, y*) is the coordinate of the bottom-left corner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



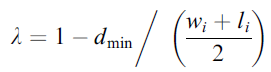
In Figure, R3 is packed on the position occupying the corner formed by the upper side and the right side of the container. As shown in Figure, the minimal distance between R3 and R1, and the minimal distance between R3 and R2 are illustrated, respectively.



*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:



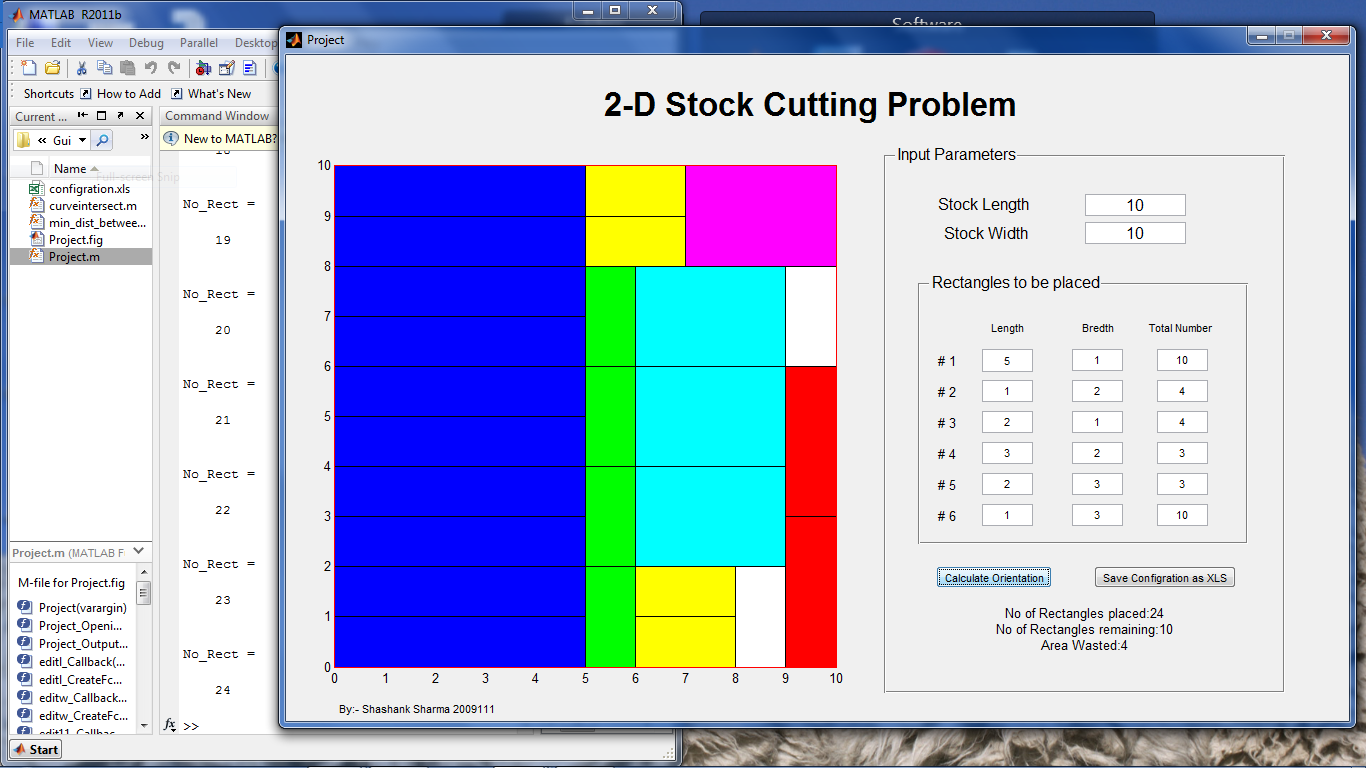
where *wi* and *li* are the width and the length of rectangle *i*, and *dmin* is the minimal distance from rectangle *i* to other rectangles in *M* and sides of the container (excluding *u* and *v*), that is,



where s1, s2, s3 and s4 are the four sides of the container.

# Algorithm:-

# 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