J Intell Manuf (2012) 23:787–795
DOI 10.1007/s10845-010-0433-0

Sequential quadratic programming method with a global search strategy on the cutting-stock problem with rotatable polygons

M. T. Yu · T. Y. Lin · C. Hung

Received: 4 December 2007 / Accepted: 2 July 2010 / Published online: 14 July 2010

Abstract The cutting-stock problem, which considers how to arrange the component profiles on the material without overlaps, can increase the utility rate of the sheet stock. It is thus a standard constrained optimization problem. In some applications the components should be placed with specific orientations, but in others the components may be placed with any orientation. In general, the methods used to solve the cutting-stock problem usually have global search strategies to improve the solution, such as the Genetic Algorithm and the Simulated Annealing Algorithm. Unfortunately, many parameters, such as the temperature and the cooling rate of the Simulated Annealing method and the mutation rate of the Genetic Algorithm, have to be set and different settings of these parameters will strongly affect the result. This study formulates the cutting-stock problem as an optimization problem and solves it by the SQP method. The proposed method will make it easy to consider different orientations of components. This study also presents a global search strategy for which the parameter setting is easy.

Keywords Cutting-stock problem · Material saving · Rotatable · Sequential quadratic programming · Global optimization

M. T. Yu · C. Hung
Department of Mechanical Engineering, National Chiao Tung University, 1001 Ta Hsueh Rd, Hsinchu 300, Taiwan, ROC

T. Y. Lin (✉)
Department of Mechatronic, Energy and Aerospace Engineering, National Defense University, Tahsi, Taoyuan 335, Taiwan, ROC
e-mail: tsylin0912@hotmail.com

Introduction

The cutting-stock problem is a key consideration in many manufacturing industries, such as textile, garment, metalware, paper, ship-building, and sheet metal industries. It considers how to arrange objects on the material, with the intention of increasing the material utility rate and avoiding overlaps between objects.

The cutting-stock problem can be classified according to different characteristics of objects, such as the shape, the number, and their orientation. Some applications, such as in the paper industry, consider only rectangular objects. However, most applications, such as garment manufacture, ship-building, and the sheet metal work, consider irregular objects. When classifying by the object number, the cutting-stock problem may be considered as a mass production problem or a small production problem. Many objects with the same shape will be cut from the material in the mass production problem. In recent years customized products have become popular; thus, products will be manufactured on a small scale. In the small production problem, there will be many different kinds of objects, and this problem is more complex than the mass production problem. The other classification in this study is made by the orientation. In some applications the object can be arranged only in one desired orientation. For example, clothes may have some straight lines, and the lines have to be aligned with one another on the material. This is called "orientation constraint" in the cutting-stock problem. The orientation constraint may be ignored in other applications, such as arranging the profiles of components of a computer case on a sheet metal. However, little research on this has been done.

Therefore, this study proposes a method of solving the cutting-stock problem with rotatable irregular objects in small-scale production. Because this kind of problem is complex and popular, the literature about it is limited at present.

Literature review

The relative positions where one object contacts another object can be represented as a polygon called a “no-fit polygon”. For solving the cutting-stock problem, Dowsland et al. (2002) used a “no-fit polygon” to obtain the closest position between two objects, and used a bottom-left strategy to arrange objects on the sheet stock. In this method, the arrangement sequence governs the resulting arranged pattern. Thus, deciding the arrangement sequence is very important. Gomes and Oliveira (2002) changed the position of objects in the sequence to generate a new solution based on the original one. These methods can be used to solve the cutting-stock problem when objects should be arranged in a special orientation, because the object orientation is fixed when finding the no-fit polygon. To allow the full rotation of objects, Yu et al. (2009) formulated the cutting-stock problem as a standard constrained optimization problem. The cost function is the summation of the distance of all objects, and the constraints are the overlap depths between objects which should be less than or equal to zero. The design variables are the positions and orientations of objects, and the problem is solved by the Sequential Quadratic Programming method (Arora 2004). This approach allows objects to be arranged in any orientation, but it finds only the local optimum solution of the cutting-stock problem.

Poshyanonda and Dagli (2004) represented objects as binary matrices, and used an artificial neural network and the Genetic Algorithm to solve the cutting-stock problem. Ratanapan et al. (2007) used an evolutionary algorithm to solve the cutting-stock problem. The evolutionary algorithm has a fitness function similar to the Genetic Algorithm, and has some operators designed by the authors to escape from the local optimum trap. Although the Genetic Algorithm (Onwubolu and Mutingi 2003) is a popular and widely-use method for searching for the global optimum solution, it is not easy to use. It has many parameters that need to be decided, such as the crossover rate and the mutation rate. Also, each parameter setting will greatly affect the result, as shown by Poshyanonda and Dagli (2004). Every design variable is transferred to a binary code, and the resolution of the binary code will affect the result. This is another disadvantage of the Genetic Algorithm.

The Simulated Annealing Algorithm is another popular global optimum method. When using the Simulated Annealing Algorithm, various operators used to determine a new solution have to be designed for “searching”. The solution is updated to the new solution if the cost function is decreased. If the cost function of the new solution is larger than or equal to the original one, a random number σ will be generated as $0 \leqq \sigma 1$. If

$$\sigma \leq e^{\frac{-\Delta E}{T}}, \tag{1}$$

where T is the temperature of the Simulated Annealing, and ΔE is the increment of the cost function, the solution will be updated to a new one. The temperature here is not a real temperature. It is a parameter used to simulate a real annealing process, while the initial temperature, final temperature, and cooling rate have to be set when using the Simulated Annealing Algorithm. Marques et al. (1991) and Szykman and Cagan (1995) used the Simulated Annealing Algorithm to conduct a global search for solving the cutting-stock problem. Leung et al. (2003) combined the Genetic Algorithm and the Simulated Annealing Algorithm, and compared the results with the pure Genetic Algorithm. As shown by Marques et al. (1991), the parameter setting proved important for obtaining a good solution compared to the Genetic Algorithm.

Bennell and Dowsland (2001) used the Tabu Search and Linear Programming to solve the cutting-stock problem. The standard deviation of the results from 100 runs is small, implying it is a stable method.

The methods, such as bottom-left strategy, Genetic Algorithm, etc., used to arrange objects govern the search process, and the geometry treatments govern the objects can be rotated or not. There are four kinds of the geometry treatment in the cutting-stock problem. They are no-fit polygon, matrix representation, Φ-function, and direct method. The no-fit polygon is mentioned above. For obtaining the no-fit polygon, an object contacts another fixed object first, and then slides on the boundary of the fixed object with a fixed orientation. The sliding path can be represented as a polygon, and it is the no-fit polygon. This method is not suitable for rotating the object in an arbitrary orientation because the no-fit polygon will be changed if the relative orientation of these two objects is changed. Thus the no-fit polygon has to be calculated for every orientation. The matrix representation usually represents an object as a binary matrix. It is easy to be rotated with 90, 180, and 270 degree because the matrix can be treated as a set of rectangles. For arbitrary rotation, the binary matrix has to be encoded for all orientations. The Φ-function is present by Stoyan et al. (2001). They use the same concept of no-fit polygon, but represent the no-fit polygon as some equations not a set of vertices. The forth method use the overlap area (Petridis and Kazarlis 1994) or overlap length (Bennell and Dowsland 1999) to consider the overlap, and the arrangement method will eliminate the overlap by moving objects. This is the most suitable method for rotatable polygons because all objects can be rotated in arbitrary orientations.

Three common cases: “Dagli”, “Shapes2”, and “Shirts” are usually used in the literatures. The best results of Dagli, Shapes2, and Shirts are 84.35% (Poshyanonda and Dagli 2004), 79.13% (Gomes and Oliveira 2002), and 85.58% (Gomes and Oliveira 2002) respectively. Although the result of Dagli is good, it depends on the parameter setting of the Genetic Algorithm. There are 9 kinds of parameters proposed by Poshyanonda and Dagli (2004), and the difference between the worst and the best one is 7.26%.

A review of the literature shows that the Genetic Algorithm and the Simulated Annealing Algorithm are commonly-used methods for solving the cutting-stock problem. However, many parameters need to be set for both algorithms, and the results are strongly affected by the choices of these parameters.

Therefore, this study will propose a global search strategy that does not require any parameters to be set, and is easy to use. This study also presents a formulation of the cutting-stock problem with rotatable irregular objects and a fixed width stock. This formulation is in a constrained optimization problem form, making it easy to consider different orientations of objects.

Method

The cutting-stock problem is a type of optimization problem. This study formulates it into a constrained optimization problem form and solves it by using the Sequential Quadratic Programming (SQP) method, a local search strategy. For improving the solution, a global search strategy will be used after obtaining a local optimum solution.

Formulation

The cutting-stock problem is formulated as a standard form of the constrained optimisation problem as follows:

$$\text{cost function:} \quad \text{minimize } f = \sqrt[n]{\sum_{i=1}^{N} x_{ui}^{n}} \tag{2}$$

$$\text{design variables} \quad x_i, y_i, \theta_i; \quad i = 1 \sim N \tag{3}$$

$$\text{constraints} \quad g1_i = D_{Mijk}(x_j, y_j, \theta_j, x_k, y_k, \theta_k) \le 0; \quad \text{where } i = 1 \sim \frac{N(N-1)}{2}, \quad j = 1 \sim (N-1), \; k = (j+1) \sim N \tag{4}$$

$$g2_i = -x_{li} \le 0; \quad i = 1 \sim N \tag{5}$$

$$g3_i = -y_{li} \le 0; \quad i = 1 \sim N \tag{6}$$

$$g4_i = y_{ui} \le y_{width}; \quad i = 1 \sim N \tag{7}$$

where N is the number of objects; n is an even number; x_i is the x-coordinate value of the reference point of object i; y_i is the y-coordinate value of the reference point of object i; θ_i is the orientation of object i; x_{li} is the lower bound of object i in the x-direction; x_{ui} is the upper bound of object i in the x-direction; y_{li} is the lower bound of object i in the y-direction; y_{ui} is the upper bound of object i in the y-direction; y_{width} is the width of the sheet stock.

The cost function will cause objects be arranged near the lower boundary of the sheet stock as close to each other as possible. The even number n in the cost function is used to enhance the effect of objects far from the lower boundary of the sheet stock. The design variables will denote the position and the orientation of objects. A reference point is used to denote the position of an object. The reference point of object i is calculated by

$$x_i = \frac{1}{Vn}\sum_{j=1}^{Vn} x_j \tag{8}$$

$$y_i = \frac{1}{Vn}\sum_{j=1}^{Vn} y_j \tag{9}$$

where Vn is the vertex number of object i. Figure 1 shows an example of a reference point.

D_{Mjk} is the maximum depth between object j and object k. A depth is the distance between a vertex of an object and its projection point on an edge of another object. The projection direction is parallel to the vector connecting the reference points of these two objects. The depth is defined as positive if the vertex is in another object, and vice versa. The maximum depth of object j and k is the largest depth of all vertices of object j and k. The calculation of the maximum depth method can be described with Fig. 2. Point B on edge CD is the projection point of Vertex A projected along vector O_jO_k. Line AB can be represented as equations and edge CD can be represented as equations.

$$x_B = x_A + (x_{Ok} - x_{Oj})\,t_1 \tag{10}$$

$$y_B = y_A + (y_{Ok} - y_{Oj})\,t_1 \tag{11}$$

$$x_B = x_C + (x_D - x_C)\,t_2 \tag{12}$$

$$y_B = y_C + (y_D - y_C)\,t_2 \tag{13}$$

t_1 and t_2 are parameters used to represent point B on the lines. $x_{Oj}, y_{Oj}, x_{Ok}, y_{Ok}, x_A, y_A, x_B, y_B, x_C, y_C, x_D$, and y_D are used to represent the position of points O_j, O_k, A, B, C, and D. O_j and O_k are the reference point of object j and k. Solve the equations by (10)=(12) and (11)=(13), the point B can be obtained. If $0 \leqq t_2 \leqq 1$ and $t_1 \leqq 0$, the depth is set as the distance between vertex A and point B. If $0 \leqq t_2 \leqq 1$ and $t_1 > 0$, the depth is set as the minus value of the distance between vertex A and point B. If $t_2 > 1$ or $t_2 < 0$, point B is not on edge CD. The depth is ignored. All vertices and all edges of both objects have to be checked each other, and the maximum one is the “maximum depth” of these two

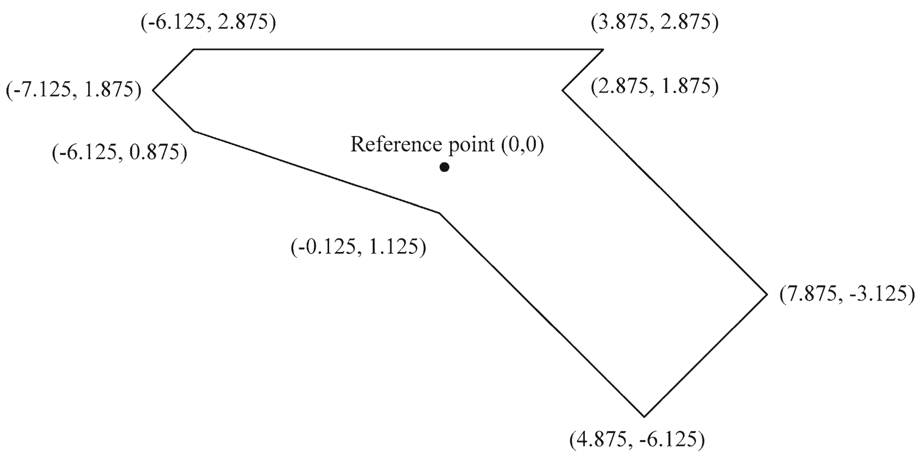

Fig. 1 The reference point of an object

objects. The algorithm form of obtaining the maximum depth of object Obj1 and Obj2 is shown as follows.

```
The position of Obj1 is $\begin{Bmatrix} x1 \\ y1 \end{Bmatrix}$ and the orientation of Obj1 is o1
The position of Obj2 is $\begin{Bmatrix} x2 \\ y2 \end{Bmatrix}$ and the orientation of Obj2 is o2
1. Set $\theta = 90 - \tan^{-1}\left(\frac{y1-y2}{x1-x2}\right)$,
   where $\begin{Bmatrix} x1 \\ y1 \end{Bmatrix}$ is the position of Obj1 and $\begin{Bmatrix} x2 \\ y2 \end{Bmatrix}$
   is the position of Obj2
   $\begin{Bmatrix} Tx1_i \\ Ty1_i \end{Bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{Bmatrix} x1_i \\ y1_i \end{Bmatrix} + \begin{bmatrix} cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{Bmatrix} x1-x2 \\ y1-y2 \end{Bmatrix}$
   $\begin{Bmatrix} Tx2_i \\ Ty2_i \end{Bmatrix} = \begin{bmatrix} cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{Bmatrix} x2_i \\ y2_i \end{Bmatrix}$,
   where $\begin{Bmatrix} x1_i \\ y1_i \end{Bmatrix}$ and $\begin{Bmatrix} x2_i \\ y2_i \end{Bmatrix}$ are the positions of the i-th vertex
   of Obj1 and Obj2 respectively; $\begin{Bmatrix} Tx1_i \\ Ty1_i \end{Bmatrix}$ and $\begin{Bmatrix} Tx2_i \\ Ty2_i \end{Bmatrix}$ are the
   positions of the $i$-th vertex of templates TObj1 and TObj2
   respectively.
2. For (k=0; k<2; k++)
     For (i=0; i<vertex number of TObj2; i++)
       For (j=0; j<vertex number of TObj1; j++)
         if (($Tx2_i > Tx1_j$) and ($Tx2_i < Tx1_{j+1}$))
           Set $y = \frac{(Ty1_{j+1}-Ty1_j)(Tx2_i-Tx1_j)}{(Tx1_{j+1}-Tx1_j)} + Ty1_j$
           if (first time to obtain Length)
             Length=$Ty2_i - y$
           else if (Length < ($Ty2_i - y$))
             Length=$Ty2_i - y$
           End if
         End if
       End For
     End For
     Set Temp as TObj1
     Set TObj1 as TObj2
     Set TObj2 as Temp
   End For
3. Obtain Length as the result
```

The maximum depth is used to evaluate the overlap of these two objects. The maximum depth is used to evaluate the overlap of these two objects. The D_{Mjk} will be negative if two objects have no overlap but have a gap between them. The constraint group $g1$ considers the overlap of objects; $g2$, $g3$, and $g4$ constrain the objects to avoid them to be arranged outside the boundaries of the sheet stock.

Local search strategy

After formulating the problem as a constrained optimization problem, it can be solved by using the SQP method, a numerical method for solving optimization problems. The procedure of a numerical method is an iterative process of finding a "search direction" and a "step size".

To solve the optimization problem by the SQP method, the Karush-Kuhn-Tucker (KKT) conditions (Arora 2004) of the Lagrange function are used. The Lagrange function is defined as follows:

$$L(d, \mu) = f(d) + \mu^T g \tag{14}$$

where μ is the vector form of the Lagrange multipliers, and d is a collection of design variables: x_i, y_i, θ_i. The numerical solving process of the KKT conditions is an iterative process of calculating the new solution $d^{(k+1)}$,

$$d^{(k+1)} = d^{(k)} + \Delta d^{(k)} \tag{15}$$

where k is the iteration number, and $\Delta d^{(k)}$ is the change in design variables. It is also the search direction of the SQP method. The SQP method defines a QP sub-problem to calculate the search direction. The flowchart of the SQP method (Arora 1984) is shown in Fig. 3. The detailed descriptions can be found in the references, and the derivation can be found in the work by Arora (2004) and Liao (1990). There are several programs such as MOST (Tseng 1989) and IDESIGN (Arora 1988) that use the SQP method to solve the constrained optimization problem.

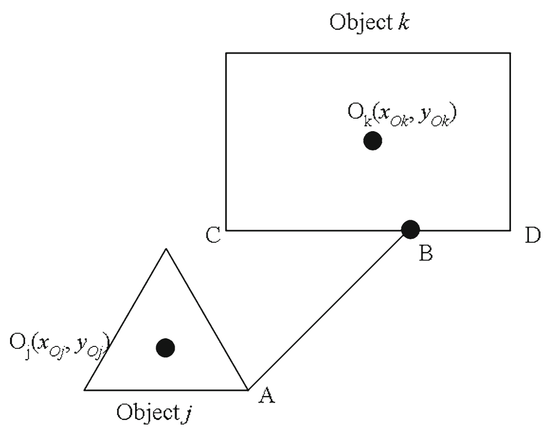

Fig. 2 The overlap depth of object j and k

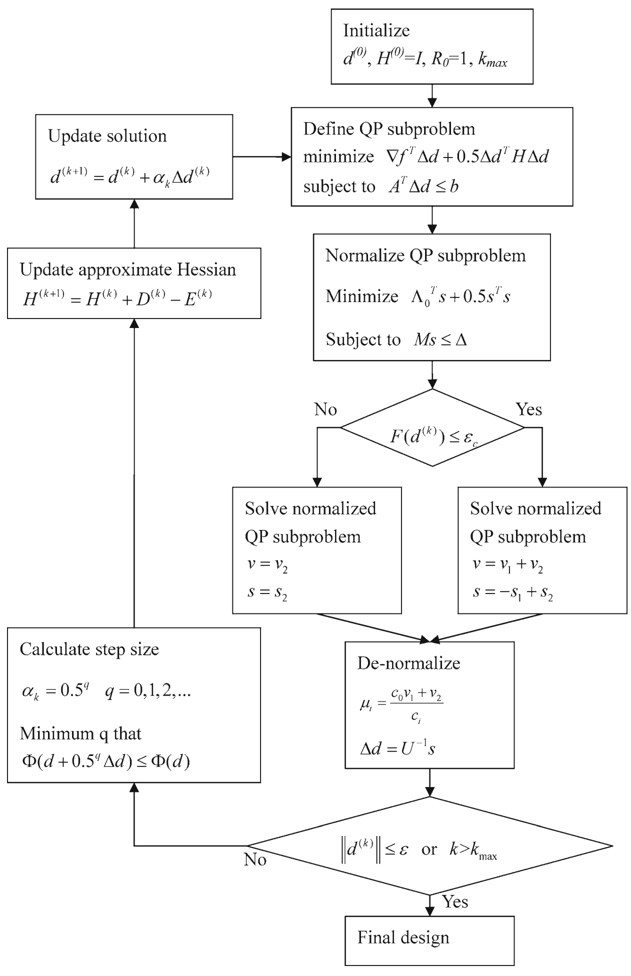
Initialize
$d^{(0)}, H^{(0)}=I, R_0=1, k_{max}$
Define QP subproblem
minimize $\nabla f^T \Delta d + 0.5\Delta d^T H \Delta d$
subject to $A^T \Delta d \le b$
Update solution
$d^{(k+1)} = d^{(k)} + \alpha_k \Delta d^{(k)}$
Update approximate Hessian
$H^{(k+1)} = H^{(k)} + D^{(k)} - E^{(k)}$
Normalize QP subproblem
Minimize $\Lambda_0^T s + 0.5 s^T s$
Subject to $Ms \le \Delta$
No
Yes
$F(d^{(k)}) \le \varepsilon_c$
Solve normalized QP subproblem
$v = v_2$
$s = s_2$
Solve normalized QP subproblem
$v = v_1 + v_2$
$s = -s_1 + s_2$
Calculate step size
$\alpha_k = 0.5^q \quad q = 0,1,2,...$
Minimum q that
$\Phi(d + 0.5^q \Delta d) \le \Phi(d)$
De-normalize
$\mu_i = \frac{c_0 v_1 + v_2}{c_i}$
$\Delta d = U^{-1} s$
$\|d^{(k)}\| \le \varepsilon$ or $k > k_{max}$
No
Yes
Final design

Fig. 3 The flowchart of SQP method

Global search strategy

The global search strategy of this study includes "escaping the local optimum trap" and "sometimes accepting a bad solution". Two objects will be swapped after finding a local optimum, and objects will be re-arranged by the SQP method.

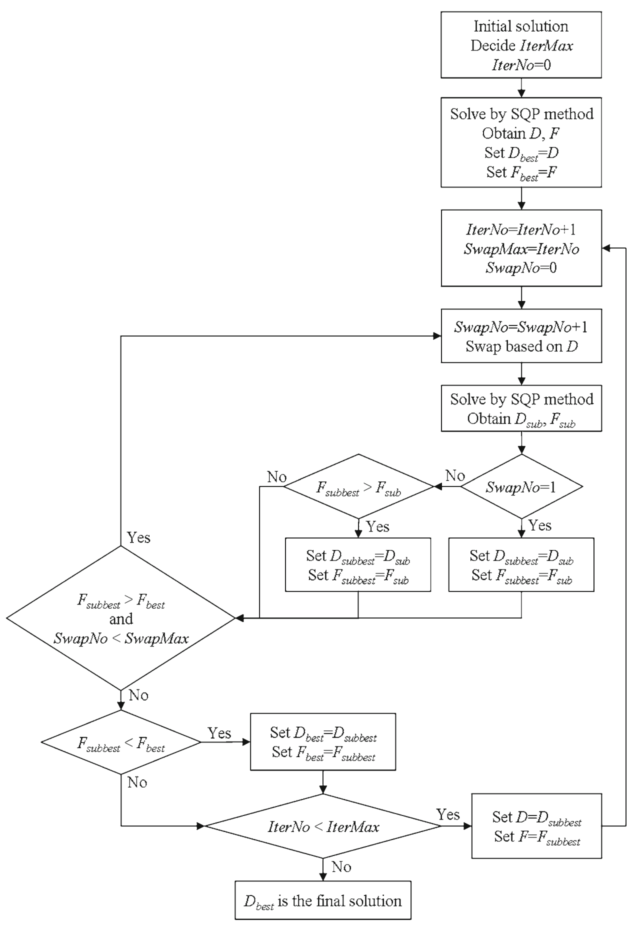

Fig. 4 The flowchart of whole approach

This will help to search for a solution in another region and escape the local optimum trap. If there is no solution better than the original one after several swaps, the best one in these swaps will be updated as the new solution. This is similar to the Simulated Annealing method that accepts a bad solution according to Eq. (1) described above.

The whole approach of this study is shown in Fig. 4, and includes the following steps:

1. Arrange all objects on the sheet stock randomly, and decide the maximum iteration number. The random arrangement is the initial solution. The index "*IterNo*" indicates the iteration number now, and it is set as 0 at the beginning.
2. Use the SQP method to arrange objects, and the solution "D" and the cost function value "F" are obtained. And then, initialize the best solution "D_{best}" and the best cost function value "F_{best}" as D and F.
3. Update the "*IterNo*" and initialize the "*SwapNo*". Set the maximum swap number as the iteration number.
4. Update the "*SwapNo*" first, and swap the position of two objects of solution D. The two objects are selected randomly.

5. There will be some overlap after swap two objects. Thus the SQP method is used to re-arrange all objects, and the solution of the swap sub-process "D_{sub}" and its cost function value "F_{sub}" are obtained.
6. If it is the first swap or the result of swap is better than the best solution in the swap sub-process, the best solution of the swap sub-problem "$D_{subbest}$" and its cost function value "$F_{subbest}$" are updated.
7. If "$F_{subbest}$" is larger than "F_{best}", i.e., the best solution of the swap sub-process is worse than the best one of total process, and it will be better to try another swap, or "*SwapNo*" is less than "*SwapMax*", i.e., another swap is allowable, go to step 4 to do another swap.
8. If another swap is not necessary or not allowable in the swap sub-process, check if "$F_{subbest}$" is better than "F_{best}" or not. If a solution better than the best one of the total process is obtained in the swap sub-process, update the best solution and its cost function.
9. If "*IterNo*" is less than "*IterMax*", update the swap base "D" as the best solution of the swap sub-process and go to step 3 to continue the process. If not, the best solution is the final solution.

The maximum swapping number is set as the iteration number, because the bad solution may be accepted easily in the beginning of the solving process. The acceptance of bad solution will become more and more difficult in the solving process. This characteristic is similar to the concept of Simulated Annealing Algorithm for global searching. As the maximum swapping number is increased during every iteration in the process, the number of bad solution acceptances is decreased. It is similar to the "cooling down" in the Simulated Annealing Algorithm, but no additional parameter has to be set, such as the temperature and the cooling rate of the Simulated Annealing Algorithm and the mutation rate of the Genetic Algorithm. It is friendly and easy to use.

Experimental results

The object information of the cases used in this study can be downloaded from the ESICUP website. Some cases are shown in Table 1. Before testing the results of the proposed approach, two parameter settings were tested first on an example named Dagli.

The first parameter was the exponent n in Eq. (2) of the optimization problem formulation. The maximum iteration number was set as 10, and four levels of n were tested. It is selected as an even number because it is easy for programming. The infinite n is implemented by taking the maximum bound of all objects in the x-direction. Ten results were obtained for every level because the initial solution of the whole process was generated randomly. The results are shown in Table 2. When n equals 2, the best material utility rate in the ten results is 80.43% and the average material utility rate is 78.18%. When n increases to 4, i.e., the effect of objects far from the origin becomes large, the best material utility rate increases to 81.05%, and the average increases to 79.46%. The best result increases the utility rate a little further to 82.61%, but the average decreases marginally to 79.15% when n increases to 8. When n is set as infinite, the best and average utility rate are decreased a little. Therefore, a larger exponent may improve the material utility rate, but an infinite exponent may not be good. It may be because the object with a larger upper bound in the x-direction will has a larger effect for the cost function with a larger n, and this may help to reduce the necessary stock length. When n is infinite, the movement of all objects will not affect the cost function unless the object with the largest x upper bound. If it does not affect the cost function to move an object toward the lower boundary of the stock in x-direction, the movement will not be executed and the object will not be arranged toward the x lower boundary. This may not be good for arranging all objects toward the x lower boundary as close as possible. However the difference is not remarkable, and the best one

Table 1 The information of cases

Case name	Number of different kind objects	Total number of objects	Total area of objects	Stock width
Dagli	10	30	3043.28	60
Shapes2	7	28	324	15
Marques	8	24	7,194	104
Mao	9	20	3758550.93	2,550
Shirts	8	99	2,160	40

Table 2 Results of different n with maximum iteration number equals to 10

No.	$n=2$ (%)	$n=4$ (%)	$n=8$ (%)	$n=\infty$ (%)
1	75.00	79.69	82.61	78.57
2	80.26	79.94	75.45	78.21
3	78.19	78.19	79.50	78.40
4	80.43	79.79	78.41	77.44
5	76.27	78.19	81.97	77.27
6	79.39	81.05	78.34	78.88
7	77.95	80.79	79.53	77.97
8	76.74	79.81	78.12	79.56
9	79.09	79.10	78.61	77.46
10	78.51	78.07	78.95	76.67
Best utility	80.43	81.05	82.61	79.56
Average utility	78.18	79.46	79.15	78.04
Deviation utility	1.66	1.01	1.91	0.82

Table 3 Results of four cases with different n and maximum iteration number equals to 10

Case	Utility		
	Best (%)	Average (%)	Deviation (%)
Shapes2			
$n=2$	71.18	68.37	2.17
$n=4$	69.63	67.28	1.88
$n=8$	72.23	68.65	1.97
$n=\infty$	72.34	68.39	2.14
Marques			
$n=2$	75.73	71.21	2.85
$n=4$	76.75	71.77	2.71
$n=8$	77.98	73.37	3.21
$n=\infty$	74.69	71.98	1.80
Mao			
$n=2$	71.37	68.48	2.00
$n=4$	72.96	69.14	2.06
$n=8$	70.90	68.01	1.41
$n=\infty$	71.10	67.67	2.18
Shirts			
$n=2$	69.70	67.53	2.02
$n=4$	73.79	69.59	2.50
$n=8$	71.22	67.53	3.56
$n=\infty$	73.06	68.17	4.19

Table 4 Results of different maximum iteration number with $n=8$

No.	Maximum iteration number			
	10	20	30	40
1	82.61%	83.25%	83.85%	82.35%
2	75.45%	82.29%	84.28%	82.90%
3	79.50%	79.98%	82.54%	82.51%
4	78.41%	80.44%	81.89%	83.16%
5	81.97%	79.97%	83.04%	82.43%
6	78.34%	82.40%	84.24%	83.73%
7	79.53%	81.60%	82.63%	83.13%
8	78.12%	80.30%	80.49%	83.69%
9	78.61%	81.51%	82.13%	82.40%
10	78.95%	79.20%	81.83%	83.72%
Best utility	82.61%	83.25%	84.28%	83.73%
Average utility	79.15%	81.09%	82.69%	83.00%
Deviation utility	1.91%	1.24%	1.14%	0.54%
Average time	1 m 29.42 s	6 m 17.22 s	14 m 7.16 s	28 m 48.9 s,

in the test cases will be used. The number n was selected as 8 in the latter cases.

The results of different cases with different n are also shown in Table 3. The difference between different n is not remarkable.

Table 5 Results of five cases

Case	Iteration number	Utility			Time
		Best (%)	Average (%)	Deviation (%)	
Dagli	10	82.61	79.15	1.91	1 min 29.42 s
	20	83.25	81.09	1.24	6 min 17.22 s
	30	84.28	82.69	1.14	14 min 7.16 s
	40	83.73	83.00	0.54	28 min 48.9 s
Shapes2	10	72.23	68.65	1.97	1 min 9.33 s
	20	73.50	71.53	1.20	4 min 56.94 s
	30	79.78	78.26	0.93	11 min 33.55 s
	40	79.86	78.59	0.65	23 min 11 s
Marques	10	77.98	73.37	3.21	53.3 s
	20	80.02	76.65	2.13	2 min 55.83 s
	30	82.10	80.44	1.16	8 min 21 s
	40	84.32	81.99	1.59	13 min 59.6 s
Mao	10	70.90	68.01	1.41	58.46 s
	20	74.76	71.62	1.69	3 min 29.45 s
	30	78.97	75.05	2.31	9 min 43.07 s
	40	81.31	79.04	0.88	15 min 6.4 s
Shirts	10	71.22	67.53	3.56	12 min 42.91 s
	20	75.28	72.84	1.59	53 min 12.58 s
	30	77.19	75.65	0.90	2 h 25 min 27.35 s
	40	80.48	78.79	0.88	4 h 4 min 6.01 s

The other parameter was the maximum iteration number. The case Dagli was also used to test this parameter, also with four levels for testing. The results are shown in Table 4. The best and average material utility rates are 82.61 and 79.15%, respectively, when the maximum iteration number is 10. The utility rate deviation and average time of these ten results are 1.91% and 1 min 29.42 s. The best utility rate increases to 83.25% and the average increases to 81.09% when the maximum iteration number increases to 20. The deviation decreases to 1.24% and the average time is 6 min 17.22 s. The best and average material utility rate both increase and the deviation decreases when the maximum iteration number increases to 30. The average time is 14 min 7.16 s. When the iteration number is 40, the best utility rate decreases a little, but the average utility rate also increases. The average time is 28 min 48.9 s. It is obvious that a tendency toward the best and average material utility rate increase and the deviation decreases as the maximum iteration number increases. The time cost increases not linearly when the iteration number increases. It is because the allowable swap number is larger in later iteration. The time spent for completing the iteration No. 40, allows 40 swap actions, which is usually larger than it is for iteration No. 10, that allows 10 swap actions. This corresponds with the notion that the solution of the optimization

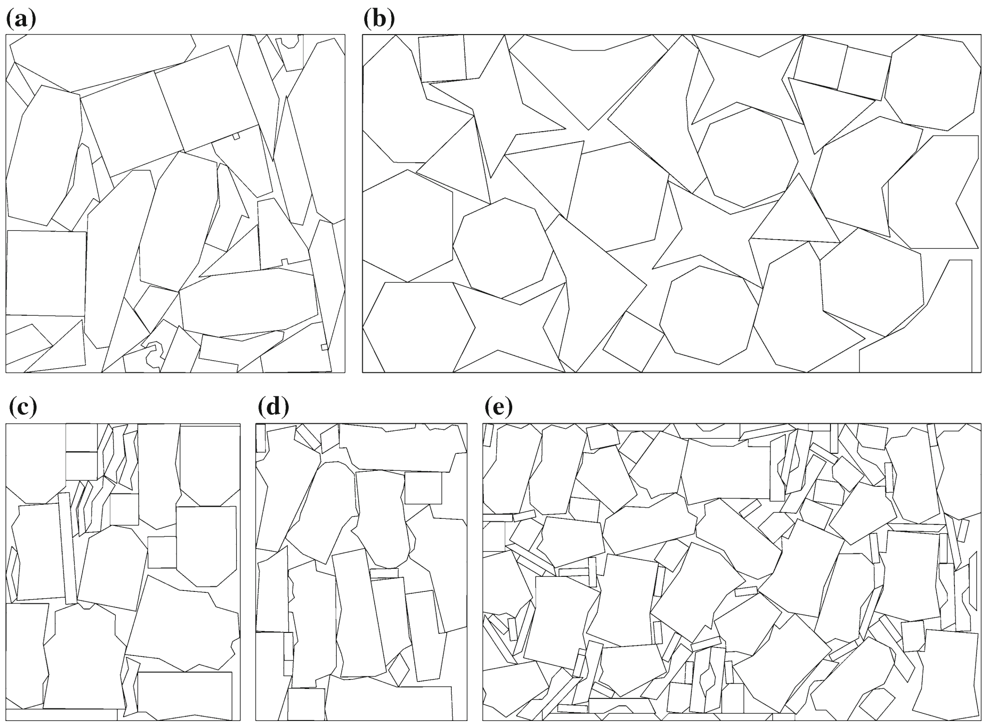

Fig. 5 The arranged patterns of the best run of five cases. **a** Dagli, **b** Shapes2, **c** Marques, **d** Mao, **e** Shirts

method will be improved and convergence if the iteration number increases.

Five cases called Dagli, Shapes2, Marques, Mao, and Shirts were used to test the effect of the proposed approach. The object information of these cases can be downloaded from the ESICUP website. The test results are shown in Table 5 and the best arranged patterns are shown in Fig. 5.

When the maximum iteration number is 30, the best utility rate for the case Dagli is 84.28%. It is a little worse than the 84.35% reported by Poshyanonda and Dagli (2004), but the parameter setting of this study is much easier than that used by Poshyanonda and Dagli (2004). The difference of the worst and the best results of Dagli is 3.79%, and it is almost a half of the difference in Poshyanonda and Dagli (2004). The average utility rate in this study is 82.69%, larger by about 4.11% than the 78.58% reported by Ratanapan et al. (2007). When the maximum iteration number increases to 40, the best utility decreases, but the average utility rate increases. The deviation is also improved.

The best utility rate for the case Shapes2 is 79.78% when the maximum iteration number is 30, which is a small improvement over the 79.12% reported by Gomes and Oliveira (2002). The average utility rate and the deviation of this case are 78.28 and 0.93%, and are better than the 76.58 and 1.12%, respectively, shown by Bennell and Dowsland (2001). The best utility rate, average utility rate, and deviation are improved when the maximum iteration number increases to 40.

The case Marques was used by Marques et al. (1991), but its results are not represented as a percentage, and the width of sheet stock is not shown, either. Although the results cannot be used to compare with those of this study, the best and average utility rate of this study are larger than 80%. Although the deviation increases 0.43%, the best and average utility rates are improved when the maximum iteration increases from 30 to 40. The results from case Mao are about 79%, while the best one is 81.31% in ten runs. They are similar to the results of other instances in this study.

The fifth case Shirts is a case with a large model. There are total 99 objects in this case. The average solving time is about 2.5 h when the maximum iteration number is 30, and increases to about 4 h when maximum iteration number increases to 40. The best utility rate is 80.48%, and it is worse than the result reported by Gomes and Oliveira (2002). The average utility rates and the deviation are 78.79 and 0.88%. The average is also worse than the 81.42% shown by Bennell and Dowsland (2001), it is because the model becomes large and difficult to obtain a good solution. But the proposed approach is more stable because the deviation is less than the 1.11% in Bennell and Dowsland (2001). All these cases showed the same characteristic that the solution of the optimization method will be improved and convergence if the iteration number increases.

Conclusions

The cutting-stock problem, which is a type of constrained optimization problem, is considered in many manufacturing industries. The cutting-stock problem in small-scale production has become popular in recent years because of the customization of products. The methods used to solve the cutting-stock problem are usually not easy when considering different orientations of objects. Some methods combine the Genetic Algorithm or the Simulated Annealing Algorithm to achieve a global search, but some parameters need to be set in these methods, and the parameter settings greatly affect the results. Therefore, this study proposes:

1. Formulating the cutting-stock problem as a standard constrained optimization problem and solving it by the Sequential Quadratic Programming method in order to consider different orientations of object easily.
2. The approach in this study has only two parameters to set, and the effect of parameter *n* is negligible. The other parameter is the maximum iteration number. Although it is better to set the maximum iteration number as large as possible to obtain the best result, ten iterations already yield good results.
3. The deviation from ten runs of every parameter setting in every case is small, which means that the approach proposed in this study has high stability.
4. The experimental instances used in this study show that the proposed approach in this study can improve the results.

References

Arora, J. S. (1984). An algorithm for optimum structural design without line search. In Atrek, E., Gallagher, R. H., Ragsdell, K. M., & Zienklewiz, O. C. (Eds.), *New directions in optimum structural design* (Chap. 20). New York: John Wiley and Sons.

Arora, J. S. (1988). *IDESIGN software, optimal design laboratory, college of engineering*. Iowa City: The University of Iowa.

Arora, J. S. (2004). *Introduction to optimum design* (2nd ed.). London: Elsevier/Academic Press.

Bennell, J. A., & Dowsland, K. A. (1999). A tabu thresholding implementation for the irregular stock cutting problem. *International Journal of Production Research, 37*(18), 4259–4275.

Bennell, J. A., & Dowsland, K. A. (2001). Hybridising tabu search with optimisation techniques for irregular stock cutting. *Management Science, 47*(8), 1160–1172.

Dowsland, K. A., Vaid, S., & Dowsland, W. B. (2002). An algorithm for polygon placement using a bottom-left strategy. *European Journal of Operational Research, 141*, 371–381.

ESICUP website (http://paginas.fe.up.pt~esicup/tiki-index.php).

Gomes, A. M., & Oliveira, J. F. (2002). A 2-exchange heuristic for nesting problems. *European Journal of Operational Research, 141*, 359–370.

Leung, T. W., Chan, C. K., & Troutt, M. D. (2003). Application of a mixed simulated annealing-genetic algorithm heuristic for the two-dimensional orthogonal packing problem. *European Journal of Operational Research, 145*, 530–542.

Liao, W. C. (1990). *Integrated software for multifunctional optimization*. Master Thesis, National Chiao Tung University, Taiwan, ROC.

Marques, V. M. M., Bispo, C. F. G., & Sentieiro, J. J. S. (1991). A system for the compactation of two-dimensional irregular shapes based on simulated annealing. In *Proceedings of the 1991 International Conference on Industrial Electronics, Control and Instrumentation-IECON* (Vol. 91, pp. 1911–1916).

Onwubolu, G. C., & Mutingi, M. (2003). A genetic algorithm approach for the cutting stock problem. *Journal of Intelligent Manufacturing, 14*(2), 209–218.

Petridis, V., & Kazarlis, S. (1994). Varying quality function in genetic algorithms and the cutting problem. In *Proceedings of the IEEE Conference on Evolutionary Computation* (pp. 166–169)

Poshyanonda, P., & Dagli, C. H. (2004). Genetic neuro-nester. *Journal of Intelligent Manufacturing, 15*(2), 201–218.

Ratanapan, K., Dagli, C. H., & Grasman, S. E. (2007). An object-based evolutionary algorithm for solving nesting programs. *International Journal of Production Research, 45*(4), 845–869.

Stoyan, Y. G., Terno, J., Scheithauer, G., Gil, N., & Romanova, T. (2001). Φ-functions for primary 2D-objects. Studia informatica universalis. *International Journal on Informatics, 2*(1), 1–32.

Szykman, S., & Cagan, J. (1995). A simulated annealing-based approach to three dimensional component packing. *Transactions of the ASME, 117*, 308–314.

Tseng, C. H. (1989). MOST software, Applied Optimal Design Laboratory, Department of Mechanical Engineering, The National Chiao-Tung University, Taiwan, ROC.

Yu, M. T., Lin, T. Y., & Hung, C. (2009). Active-set sequential quadratic programming method for the multi-polygon mass production cutting-stock problem with rotatable polygons. *International Journal of Production Economics, 121*, 148–161.