

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/223090176>

Developing a simulated annealing algorithm for the cutting stock problem

Article in *Computers & Industrial Engineering* · January 1997

DOI: 10.1016/S0360-8352(96)00205-7

CITATIONS

146

READS

2,402

2 authors, including:



Kin Keung Lai
Shaanxi Normal University

967 PUBLICATIONS 19,942 CITATIONS

SEE PROFILE



DEVELOPING A SIMULATED ANNEALING ALGORITHM FOR THE CUTTING STOCK PROBLEM

K. K. LAI and JIMMY W. M. CHAN

Department of Applied Statistics and Operational Research, City University of Hong Kong, Kowloon,
Hong Kong

(Received 1 May 1996)

Abstract—This paper presents an intuitive, simple and efficient simulated annealing searching technique to solve non-guillotine, two- or three-dimensional cutting stock problems. This algorithm considers the possibility of placing different sizes of small rectangles or boxes on a larger rectangle (pallet) or container, in such a way that the amount of trim loss is minimized. The algorithm we propose provides a basis for exploring the integration of the simulated annealing technique with artificial intelligence, and interval algebra. The algorithm is programmed in C and run on a personal computer with an Intel 486-based CPU. The algorithm is tested using randomly generated test cases and also using real data from a printing company in Hong Kong. Copyright © 1997 Elsevier Science Ltd

1. INTRODUCTION

The two-dimensional cutting stock problem consists of cutting a set of boxes or rectangles from a sheet of material in order to minimize away that trim loss. The problem arises in various production processes in the glass, steel, wood, paper and textile industries. The three-dimensional cutting problem is similar to packing boxes into a container. Thus, the problem is interesting not only from a theoretical but also from a practical point of view. In this research, a constrained version of two-dimensional and three-dimensional cutting stock problems with mixed box sizes is considered, for which a simulated annealing searching algorithm has been developed.

The simulated annealing algorithm is based on the analogy between the process of finding an optimal solution of a combinatorial optimization problem and the process of annealing a solid to its minimum energy state in statistical physics. Annealing is a process which finds the low energy state of a metal by melting it and then cooling it slowly. Temperature is the controlling variable in the annealing process and determines the randomness and the energy state.

Simulated annealing is a local search algorithm. The searching process starts with an initial solution perhaps chosen at random. A neighbor of this solution is then generated by some mechanism and the change in cost is calculated. For a general local search process, if a reduction in cost is found, the current solution is replaced by the generated neighbor, otherwise the current solution is retained. The process is repeated until no further improvement can be found in the neighborhood of the current solution, so the local search algorithm terminates at a local minimum. The simulated annealing searching process attempts to find a near-optimal minimum by occasionally accepting uphill moves, which increase the objective function values, and by employing probabilistic and deterministic acceptance strategies. More precisely, a simulated annealing algorithm accepts an uphill move ($\Delta C > 0$ where C is cost) from the current cutting pattern S to another S' with probability $e^{-\Delta C/T}$, where T is a positive control parameter called temperature. The acceptance probability decreases for increasing values of ΔC and for decreasing values of T . The T values are updated according to a defined cooling schedule. Also, pre-specified stopping criteria are used to terminate the searching process. Reviews of the theory of simulated annealing algorithms can be found in Section 3.

2. AIM OF THE STUDY

The cutting stock problem has many extensions and variants. In many industries, the cutting equipment operates in such a way that any cut made upon a rectangle must be in a straight line from one edge of the material to the opposite edge. This type of cut is referred to as a guillotine cut. Figure 1 illustrates different styles of guillotine cutting patterns.

However, some applications, such as cutting in flame, allow the use of non-guillotine cuts.

In this paper, we will consider the non-guillotine cutting stock problem. The objective of this study is to develop a cutting stock algorithm based on the simulated annealing technique to cut a number of rectangular pieces or conceptual boxes from a large confined space (in two or three dimensions) so as to minimize the total trim loss.

3. LITERATURE REVIEW

The cutting stock problem has been classified as NP-hard. This is because the simplest problems, i.e. the one-dimensional cutting problem and the pallet loading problem are already NP-hard. No polynomial algorithms exist for this class of problem. A complete discussion of the complexity of the problem in question is given in Blazewicz *et al.* (1989). The cutting stock problem has many real-life applications in industry. For instance, Yanasse (1991) presented an algorithm for the cutting stock problem in the wood industry, Gemmill (1990) formulated the inventory portfolios which consisted of determining the best combination of sheet sizes to keep in stock for the two-dimensional cutting stock problem, Ferreira *et al.* (1990) presented a heuristics approach for solving the non-linear cutting stock problems appearing in the iron and steel industry, and Roberts (1984) developed a heuristic technique to determine an appropriate cutting schedule for worktops of various shapes and sizes for the benefit of a furniture manufacturer.

The simulated annealing algorithm can be viewed as a general optimization technique for solving combinatorial optimization problems. The algorithm is based on randomization techniques. It also incorporates a number of aspects related to iterative improvement algorithms. Osman and Potts (1989) and Ogbu and Smith (1990) formulated the flowshop scheduling problem as a simulated annealing problem, Van Laarhoven *et al.* (1992) used simulated annealing to find the minimum makespan in jobshop scheduling problems, and Kampke (1988) considered the application of

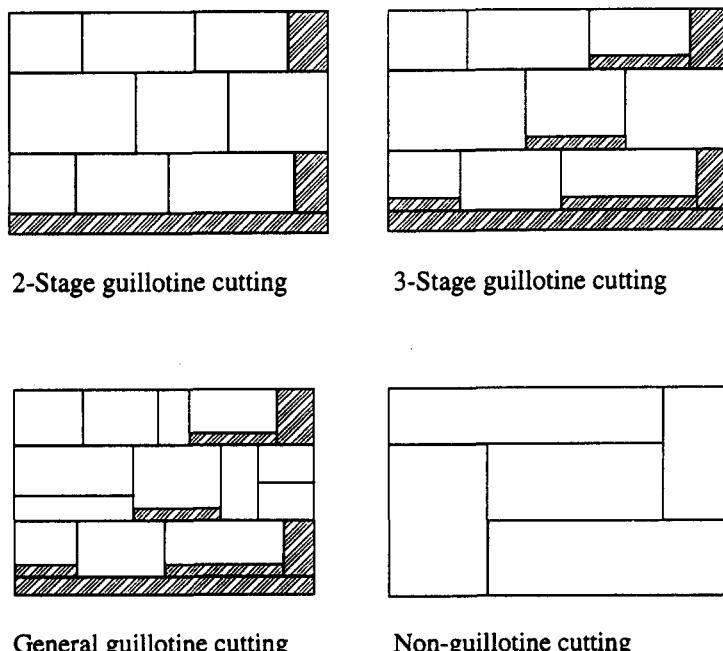


Fig. 1. Different styles of guillotine cutting patterns.

simulated annealing to bin packing problems. Cutting stock problems are combinatorial problems and as such we believe they are well suited to simulated annealing applications.

The cutting stock problem is an NP-hard problem so it is unlikely that an efficient exact method will be found to solve it. Exact methods can only solve small-sized problems, and practical-sized problems are often tackled by applying heuristics to obtain near-optimal or approximate solutions. As already mentioned, the problem is difficult and its complexity increases very rapidly with the number of elements (pieces of paper or boxes). To generate an acceptable solution and make an improvement in the solution involves a great many calculations, and it is easy to become trapped in a local minima.

The above reasoning led to the proposed use of simulated annealing by Kirkpatrick *et al.* (1983) and Cerny (1985). The simulated annealing algorithm imposes different randomized search steps in the feasible cutting state space. We have developed acceptance and stopping criteria for local search moves in order to escape poor quality local minima. Local search descent methods do not accept non-improvement moves at any iteration, whereas simulated annealing does with certain probabilities. These probabilities are determined by a control parameter called temperature. At high temperatures, most uphill moves are accepted with a high probability regardless of the increase in cost. Eventually, as temperature decreases, only downhill moves will be accepted, leading to near-optimal cutting patterns.

Recently, many successful simulated annealing applications have been found to work for other combinatorial optimization problems, such as the vehicle routing problem by Osman (1993) and Malek *et al.* (1989), airline seat inventory control by Dusan *et al.* (1993) and capacitated clustering problems by Osman and Christofides (1994).

In this paper, we apply the simulated annealing searching technique to two- or three-dimensional cutting stock problems with non-guillotine cut.

4. METHODOLOGY AND FORMULATION OF THE CUTTING STOCK PROBLEM

For every cutting pattern we generate, we can get a cutting order and obtain related trim loss for the pattern. Taking a two-dimensional cutting pattern as an example, we can use a cutting order to represent a cutting pattern. All the elements in a cutting order are the pieces of a rectangle.

Different positions of the pieces inside the cutting order form different cutting patterns. We can use the method of swapping any two pairs of cutting pieces in the cutting order to generate all the neighbors of the cutting patterns. This process can be extended to generate the whole cutting state space (all possible arrangements of cutting patterns). We can calculate the change in trim loss and replace the current solution to the generated neighbor. For example, see Fig. 2.

Swapping two elements inside the cutting order list is shown by Fig. 3. This example does not make any difference to the trim loss. However, it will affect the availability of space that can be used to assign the next rectangle. Therefore, we assume that the total area of all pieces placed in the demand pool should be greater than the area of the stock sheet. In the above example, you may ask why the element which is second in the cutting order should be placed on top of the rectangle a1. This involves the interval selection mechanism for which we will explain in the following sections.

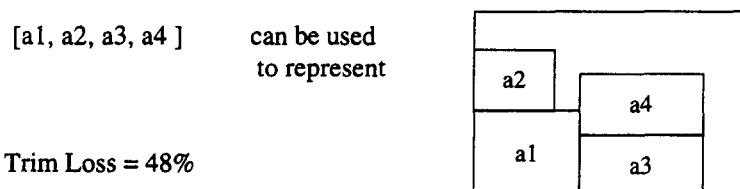


Fig. 2. The cutting pattern is illustrated by the cutting order [a1, a2, a3, a4].

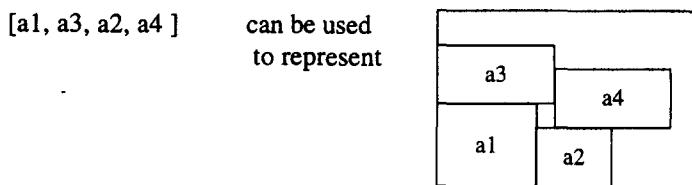


Fig. 3. The cutting pattern is illustrated by the cutting order [a1, a3, a2, a4].

To generalize the whole algorithm into two- and three-dimensional cutting stock problems, entails the following three steps:

- (1) Interval generation (formulate cutting problem)
 - (2) Interval selection (box packing or paper cutting)
 - (3) Cutting pattern generation (simulated annealing search process)

(1) Interval generation (formulate cutting problem)

Interval generation is used to generate the next cutting areas when one of the rectangles (boxes) has been cut from the stock sheet (container). Taking two-dimensional cutting as an example, suppose a rectangle a_1 has been assigned on the stock sheet, the possible areas that can be used to cut the next rectangle are $[(x_1, y_1), (x_2, y_2)]$ and $[(x_3, y_3), (x_2, y_2)]$ (Fig. 4).

To formulate the classic cutting stock problem into a state space searching environment and to generalize this concept to three-dimensional cutting stock problems, we employ the interval algebra technique. We will outline two basic processes in our interval generation below.

Difference process

The difference process involves the generation of possible new spaces after placing a box in a confined space. If we cut a conceptual box with a bottom left coordinate of (x_3, y_3, z_3) and an upper right coordinate of (x_4, y_4, z_4) into a three-dimensional confined space with a bottom left coordinate of (x_1, y_1, z_1) and an upper right coordinate of (x_2, y_2, z_2) , we have to consider three sets of projections.

We assume that $x_1 \leq x_3 \leq x_4 \leq x_2$, $y_1 \leq y_3 \leq y_4 \leq y_2$, and $z_1 \leq z_3 \leq z_4 \leq z_2$.

Difference: $[(x_1, y_1, z_1), (x_2, y_2, z_2)] - [(x_3, y_3, z_3), (x_4, y_4, z_4)]$.

10

$$[(x_1, y_1, z_1), (x_3, y_2, z_2)], \quad (1)$$

$$[(x_4, y_1, z_1), (x_2, y_2, z_2)], \quad (2)$$

$$[(x_1, y_1, z_1), (x_2, y_3, z_2)], \quad (3)$$

$$[(x_1, y_4, z_1), (x_2, y_2, z_2)], \quad (4)$$

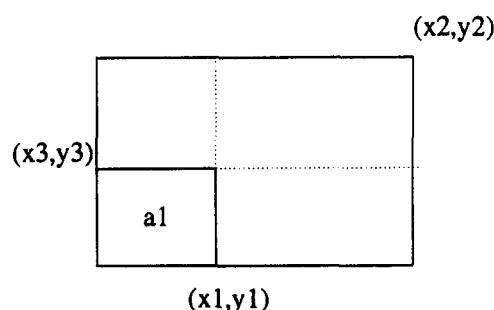


Fig. 4. Difference process in interval generation.

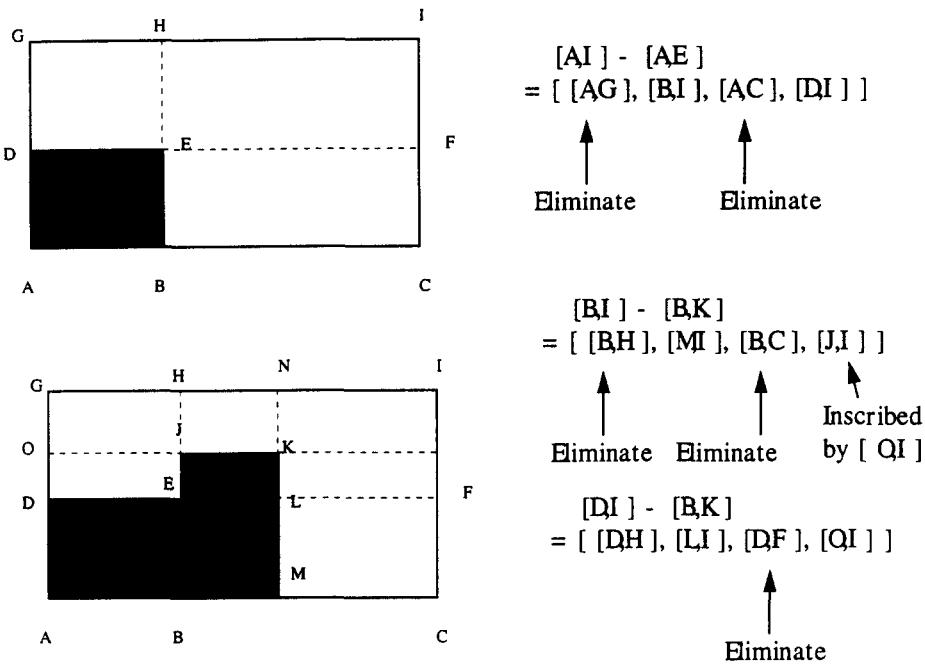


Fig. 5. Elimination process in interval generation.

$$[(x_1, y_1, z_1), (x_2, y_2, z_3)], \quad (5)$$

$$[(x_1, y_1, z_4), (x_2, y_2, z_2)] \quad (6)$$

The intervals of (1) and (2), (3) and (4) and (5) and (6) are generated based on the x, y and z projection respectively. We will now take a two-dimensional cutting example shown by Fig. 5.

Suppose a box with the interval [A, E] is packed in the stock sheet with the interval [A, I]. The interval list [[B, H], [M, I], [B, C], [J, I]] is generated after the difference process. The space intervals in the list represent the available choices for the next box to be assigned into a conceptual confined space, therefore the generated space intervals (maximal space intervals) overlap after using the difference process. The difference process continues to generate a new interval list after another box is packed on the stock sheet.

Elimination process

After the interval list has been generated by the difference process, some intervals should be eliminated in order to save computer memory storage space. Intervals with infinite thinness or those that are totally inscribed by the other space intervals will be removed from the list.

Cross-checking: Compare each space interval with the space interval elements in the list to see whether it is totally inscribed by other space intervals.

Eliminate $[(x_1, y_1, z_1), (x_2, y_2, z_2)]$ from $[(x_3, y_3, z_3), (x_4, y_4, z_4)]$
 if $(x_1 \geq x_3)$ and $(y_1 \geq y_3)$ and $(z_1 \geq z_3)$ and $(x_2 \leq x_4)$ and $(y_2 \leq y_4)$ and $(z_2 \leq z_4)$.

Self-elimination: Check whether the space interval is infinitely thin or not.

Eliminate $[(x_1, y_1, z_1), (x_2, y_2, z_2)]$ from the space interval list
 if $(x_1 = x_2)$ or $(y_1 = y_2)$ or $(z_1 = z_2)$ or $(x_3 = x_4)$ or $(y_3 = y_4)$ or $(z_3 = z_4)$.

The above process can easily be adapted to the two-dimensional problem by setting $z_1 = z_2 = z_3 = z_4 = 0.0$.

(2) Interval selection (box packing or paper cutting)

During the interval generation step, we may generate many possible cutting areas (intervals) in the stock sheet for cutting the next rectangle (box). We need some criteria to select which interval is used for the next cutting process. One of the selection criteria should be based on size. Thus, the interval should be large enough to hold the next cutting piece or rectangle. In addition, we need a selection strategy to choose the intervals. We use an ad hoc heuristic, which is called the “as compact as possible” packing heuristic. This heuristic is proposed because of its simplicity and commonality. That is, search for a space interval in the space interval list that is large enough to locate the cutting piece and has the shortest length of anchored vector. The anchor vector is defined as a vector that points from the origin to the bottom left corner of the cutting piece. In other words, the rectangles should be placed as near as possible to the bottom lower left hand corner [Fig. 6(a)].

Suppose rectangles a_1 and a_2 have been assigned on the stock sheet, and three space intervals have been generated: $[(x_3, y_3), (x_4, y_4)]$, $[(x_1, y_1), (x_2, y_2)]$ and $[(x_5, y_5), (x_2, y_2)]$. Rectangle a_3 is waiting to be assigned on the stock sheet [Fig. 6(b)].

Rectangle a_3 cannot be cut from the interval $[(x_5, y_5), (x_2, y_2)]$ because of its size. Therefore, rectangle a_3 can only be cut from the intervals: $[(x_3, y_3), (x_4, y_4)]$ and $[(x_1, y_1), (x_2, y_2)]$. Based upon the “as compact as possible heuristic”, the rectangle should be cut from the interval: $[(x_3, y_3), (x_4, y_4)]$, because the magnitude between $(0, 0)$ and (x_1, y_1) is greater than the magnitude between $(0, 0)$ and (x_3, y_3) .

(3) Cutting pattern generation (simulated annealing searching)

Based on the procedures of “Interval Generation” and “Interval Selection”, we can generate the state space of cutting patterns. In this section, we will illustrate how the simulated annealing searching technique can be used to find the optimal or near optimal cutting pattern in the cutting state space.

The simulated annealing searching procedure for the cutting problem we propose consists of two main loops. The outer loop checks that the stopping criteria have been met. Each time the inner loop is completed, the temperature is updated using an updated temperature function and the

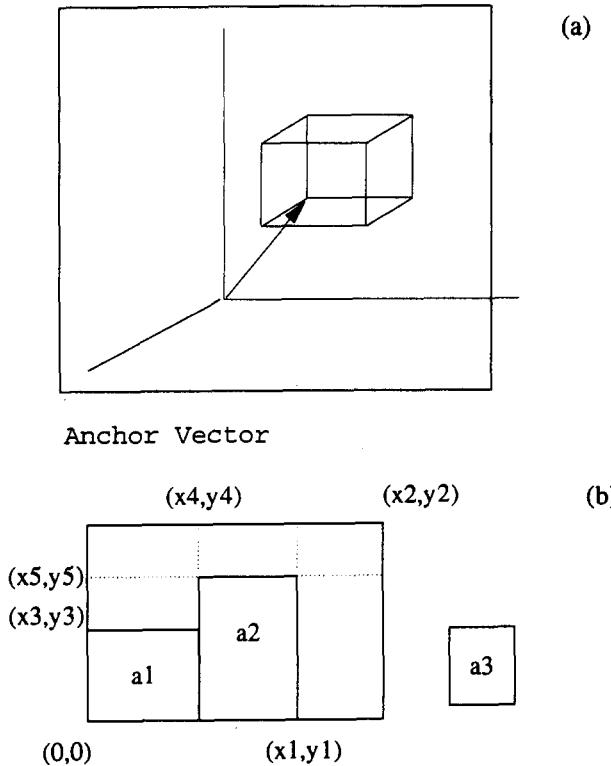


Fig. 6. (a) Anchor vector. (b). The rectangle a_3 is waiting to be assigned on the stock sheet.

stopping criteria are checked again. It is similar to a hill climbing searching approach, except that approach sometimes accepts the uphill move. Thus, it is useful to prevent being trapped into a local minimal situation.

```

WHILE (stopping criteria not met)
  WHILE (equilibrium not reached)
    Generate-next-cutting-pattern()
    IF (Accept (Temperature, Change-in-cost)) THEN
      Update-cutting-pattern()
    ENDWHILE
    Calculate-new-temperature()
  ENDWHILE

```

Moving process

Basically, the moving process is the Generate-next-cutting-pattern function. The algorithm will prompt a move until an equilibrium is reached. A possible move can be generated using a swapping strategy (i.e. swap the elements in the cutting order). Each move will lead to another cutting pattern in the state space. We define a swapping anchor vector distance to control the degree of pattern changes. Swapping anchor vector distance determines how many elements from the last element in the cutting order are available to move. For example,

$[a_1, a_2, a_3, a_4, a_5, a_6, a_7]$


From the above example, the anchor distance vector is 5. That means we can randomly select any two elements from the cutting order between a_3 and a_5 to swap. The longer the anchor vector distance, the higher the chance of a big difference between the cutting patterns. For example, see Fig. 7(a).

If the anchor vector distance is 2, it means only a_5 and a_6 can be used to be swapped [Fig. 7(b)].

Anchor Vector Distance $\alpha [e^{-1/\text{Temperature}}]$

Remark: [] means rounded up to nearest integer

The anchor vector distance is also determined by the temperature value. In our algorithm, we define the exponential relationship instead of using a linear relationship between the temperature and the anchor vector distance. Thus, this result provides more active elements in the cutting order at high temperature (Fig. 8).

Accept function

The accept function returns either true or false and it prompts a decision either to change to another cutting pattern or retain the current situation.

The accept function depends on two parameters: temperature and change in cost. Change in cost represents the change in the trim loss value of the cutting pattern.

Our accept function is defined as follows:

```

IF ( $\Delta C < 0$ ) RETURN TRUE
ELSEIF ( $e^{-\Delta C/T} > \text{random}(0, 1)$ ) RETURN (TRUE)
ELSE RETURN (FALSE)

```

The above function describes the hill climbing process. The ΔC represents the difference in change of trim loss from one cutting pattern to another. The cutting pattern change means two elements in the cutting order have swapped their positions. It also means the hill climbing search process moves forward a step in the cutting state space.

At very high temperatures, it is more likely to generate a solution with some loss, i.e. ΔC is positive in value. In hill climbing search terminology, it allows the uphill move even if it is looking for the global minimum. Eventually, as the temperature decreases, only downhill moves (i.e. cutting patterns with lower trim loss) will be accepted. The probability of the accept function is shown in Fig. 9.

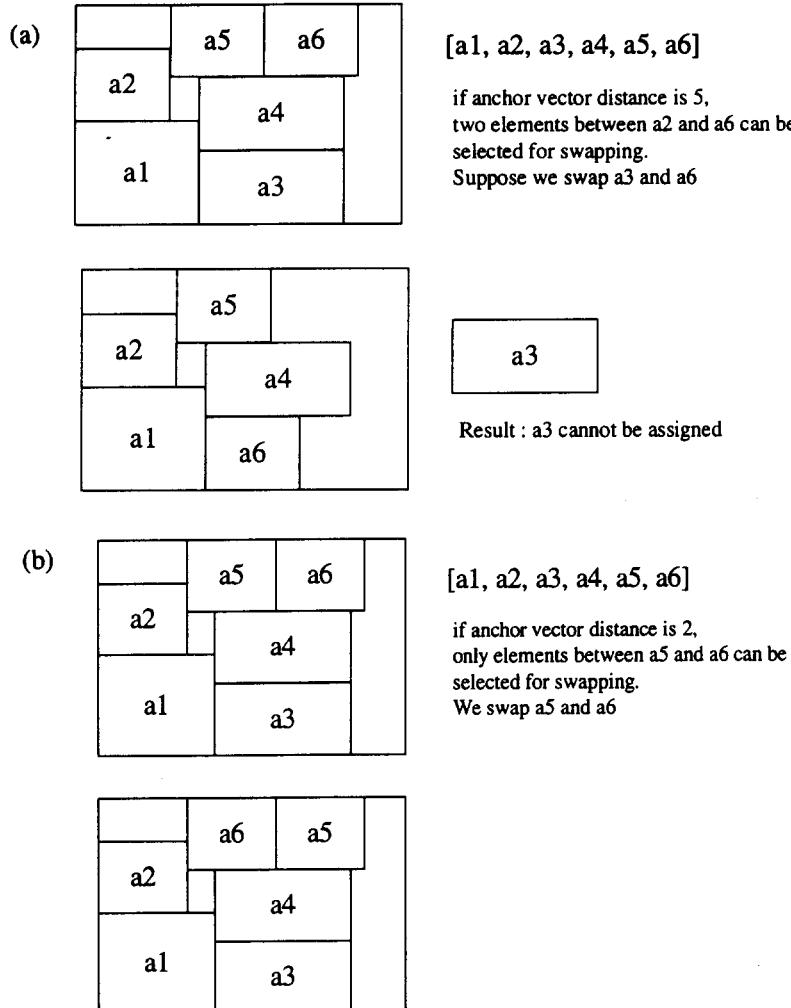


Fig. 7(a) If anchor vector distance is 5, any two elements between a2 and a6 can be selected for swapping.
 (b) If anchor vector distance is 2, only elements between a5 and a6 can be selected for swapping.

Cooling schedule

The performance of the simulated annealing algorithm also depends on the chosen cooling schedule, which is the temperature updating function. With a good cooling schedule, near optimal solutions can be achieved for many combinatorial problems. In our study, we employ the cooling schedule from Lundy and Mees (1986). The cooling schedule specifies a finite sequence of inhomogeneous Markov chains, and all chains in our algorithm have an equal length of one outer loop iteration. We set an initial temperature T_s to 100.0 where 100.0 is an upper bound on $\Delta_{\max} C$. This is because the maximum trim loss of a cutting pattern means no piece of paper can be cut from the stock sheet (i.e. the trim loss is 0.0% and the utilization rate is 100.0%). The temperature is reduced at each iteration by a smaller amount than the one before using a constant decrement ratio β , i.e. temperature T is updated according to the following rule:

$$T_{k+1} = \frac{T_k}{(1 + \beta \times T_k)}$$

According to the suggestion from Lundy and Mees (1986), the β value can be easily evaluated as:

$$\beta = \frac{T_s - T_f}{M \times T_s \times T_f}$$

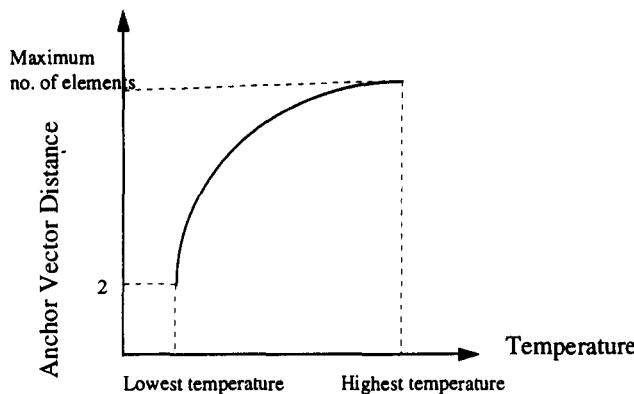


Fig. 8. Exponential relationship between the temperature and the anchor vector distance.

where T_s and T_f represent the initial and final temperatures respectively and M is the total number of iterations required to terminate the searching process.

Stopping criteria and equilibrium condition

Two stopping criteria are used to terminate the whole searching process. One is the maximum number of iterations M , and the other is fulfilment of an acceptable trim loss (e.g. trim loss $\leq 5.0\%$). Users can make a decision about the trade-off between the quality of the solution and the computation time. For each move, a new pattern is generated and the related trim loss of the pattern is found. The system will record the cutting pattern, P^* , which has the lowest trim loss found. It can be used to test whether the system reaches an equilibrium state or not. Equilibrium means the system cannot find a better solution comparing the trim loss value of P^* after a certain number of iterations.

5. COMPUTATIONAL RESULTS

The algorithm is tested using randomly generated test cases and also using real data from a printing company in Hong Kong.

Randomly generated test cases

Our experiments are conducted by generating different sets of perfect non-guillotine cutting patterns and using our algorithm to reassemble the cutting pattern. Perfect cutting means cutting rectangular pieces on the stock sheet without any trim loss. The algorithm is implemented on a 486 Intel-based micro-computer with 33 MHz. We define the initial and final temperatures as 100.0 and 10.0 respectively. The maximum number of iterations (stopping criteria) M is 10,000. Also,

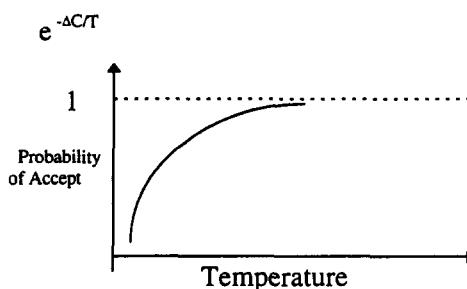


Fig. 9. Probability function for the annealing process.

Table 1. Summary of the simulation runs for a two-dimensional cutting stock problem (stock sheet size = 400 × 200)

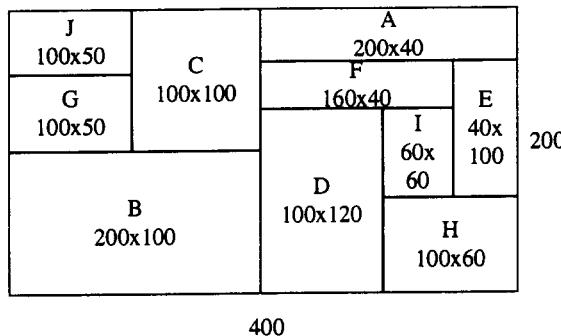
No. of pieces	No. of trial runs	Avg. CPU time (s)	No. of perfect cutting patterns that can be generated	Avg. No. of iterations
5	50	3.8	50	8
10	50	13.0	50	15
15	50	42.0	50	303
20	50	101.5	50	668
25	50	405.2	46	857
30	50	413.0	44	866
35	50	445.3	41	943

Table 2. Summary of the simulation runs for a two-dimensional cutting stock problem (stock sheet size = 400 × 400)

No. of pieces	No. of trial runs	Avg. CPU time (s)	No. of perfect cutting patterns that can be generated	Avg. No. of iterations
5	50	2.3	50	4
10	50	9.3	50	12
15	50	44.0	50	230
20	50	221.0	47	522
25	50	389.3	45	855
30	50	432.1	41	941
35	50	442.0	42	921

if the searching process obtains a cutting pattern with a zero trim loss value, the cutting process is stopped. Tables 1 and 2 give a summary of several sets of pieces to be cut on stock sheet with sizes 400 × 200 and 400 × 400.

Figures 10–12 show the cutting pattern and the change in trim loss during iteration. Figures 10 and 11 illustrate 10 and 15 cutting pieces on a stock sheet size of 400 × 200. Figure 12 shows 20 cutting pieces on a 400 × 400 stock sheet size.



400

200

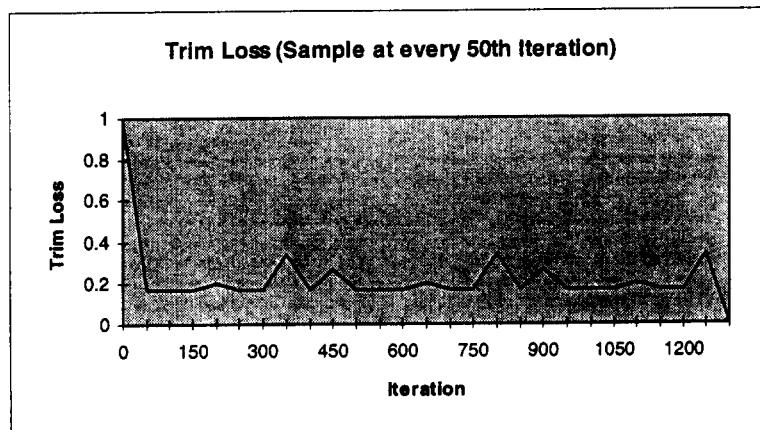


Fig. 10. Ten cutting pieces with a stock sheet size of 400 × 200.

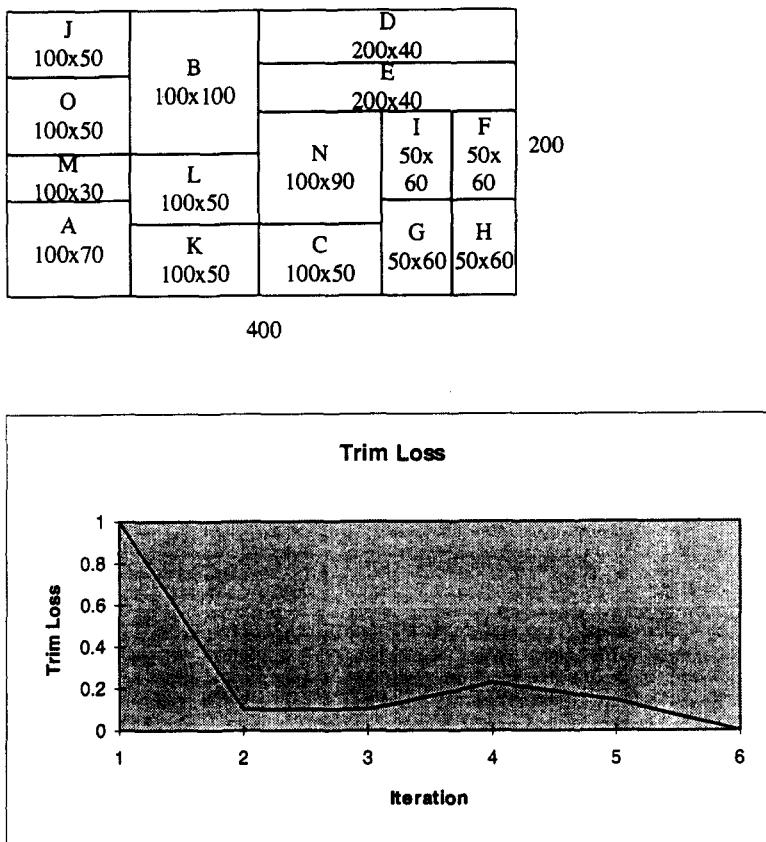


Fig. 11. Fifteen cutting pieces with stock sheet size 400×200 .

Figures 10–12 show graphically the changes in trim loss with increasing iterations of the two-dimensional cutting pattern.

Real data test cases

The simulated annealing technique described in the previous section has been tested using actual data from a printing company in Hong Kong. The company prints different types of leaflets, catalogues, bookmarks and carton labels for marketing use. The production procedure begins by classifying demand orders (rectangular pieces) into different demand pools based on quantity and other printing requirements. In other words, a demand pool will contain all the demand orders with similar quantity and printing characteristics, such as delivery date, color, paper quality and printing procedure. Then the demand orders (rectangular pieces) in the demand pool are looked at by an experienced planner to generate a cutting pattern. Afterwards, the cutting of all the rectangles from the stock sheet is considered. The finalized cutting pattern will be used to cut the zinc film for printing.

The algorithm is again implemented on a 486 Intel-based micro-computer with 33 MHz. We defined the initial and final temperatures as 100.0 and 10.0 respectively. The maximum number of iterations (stopping criteria) M is 10,000. Also, if the searching process obtains a cutting pattern with a trim loss value of less than 6%, the cutting process is stopped. The 6% trim loss is the planner's acceptable average trim loss. Table 3 gives the data for this real cutting stock problem and Table 4 gives a summary of the results of the experimental test with stock sheets of sizes 23.25×18.25 .

From the simulated test cases, we found that the computation time is increased tremendously when the number of pieces to be considered is increased at the initial stage. However, the computation time will not increase if the number of rectangles is increased. There are two reasons for this: (1) as the number of rectangles in the pool increases, it also increases the chance of generating a perfect cutting pattern. In other words, there is more than one perfect cutting pattern

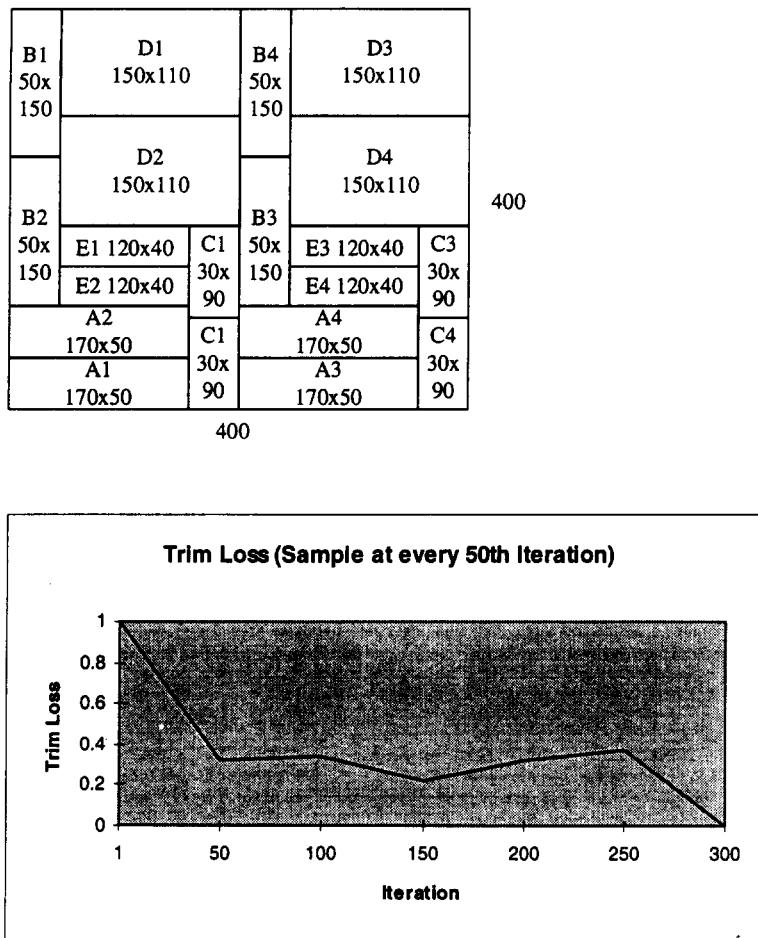


Fig. 12. Twenty cutting pieces with stock sheet size 400 x 400.

for a set of data, especially when many rectangles have the same size in the demand pool; (2) the stopping criteria will stop the searching process if the total searching iterations exceed a maximum limit or if it reaches an acceptable trim loss environment. This means that the algorithm will be more reliant on the pre-defined acceptable trim loss and the total number of iterations.

Table 3. Data for real cutting stock problem

Prob. No.	No. of rectangles in the demand pool	Sizes of the demand orders in the pool
PB1	10	(12, 11), (8, 6)x2, (8, 4)x2, (12, 8), (5, 5, 3)x2, (5, 4) and (8, 5, 6)
PB2 and (5, 3)	10	(10, 10), (10, 4)x2, (13.5, 8.5), (8, 5, 5), (4, 3), (2, 2), (7, 2), (8, 5)
PB3	15	(10, 7)x2, (10, 5)x4, (6, 6)x3, (7, 4)x2, (8, 5, 5) and (4, 3)x3
PB4	15	(7, 6)x4, (8, 4)x4, (6, 5)x5, (8, 4)x2
PB5	20	(7, 6)x4, (13.5, 8.5)x4, (12, 8)x3, (6, 5)x5, (5, 5, 3)x2 and (4, 3)x2

Table 4. Summary of the five test cases in a printing company for a two-dimensional cutting stock problem (stock sheet size = 23.25 x 18.25)

Prob. No.	No. of rectangles in the demand pool	CPU time (s)	Trim loss (%)	No. of rectangles to be assigned on the zinc film
PB1	10	15	5.2	8
PB2	10	18	2.9	10
PB3	15	356	7.8	11
PB4	15	165	5.6	12
PB5	20	160	5.4	11

From the practical test cases, we found that the performance of our proposed algorithm is good for small sized cutting problems (where there are less than 15 pieces to be cut). The existing cutting algorithm is not guaranteed to generate the optimal cutting pattern, but it can obtain an efficient and acceptable near optimal solution for the planner.

Besides the pre-defined parameters (such as temperature, alpha and equilibrium values), we found that the results depended greatly on the initial solution of the cutting pattern. In this paper, we used an arbitrary initial cutting pattern, i.e. the cutting order is: 1, 2, 3, 4, . . . , n. We firmly believe that a good estimated initial cutting pattern can lead to a significant improvement in results.

6. CONCLUDING REMARKS AND FUTURE WORK

In this paper we have proposed a simulated annealing technique for the cutting stock problem. Most of the two-dimensional papers referred to did not solve cutting stock problems by using simulated annealing searching models. This paper contributes to the problem formulation and a solution of small sized cutting stock problems. This technique was implemented on a microcomputer and it can be applied efficiently for moderate sized problems (i.e. not more than 20 rectangles to be considered on one stock sheet). Although we have not tested the performance on three-dimensional container packing, we have illustrated the formulation and the feasibility of handling three-dimensional cutting stock problems. From the experimental results in Section 5, we can conclude that the performance of this algorithm greatly depends on the maximum number of iterations and the acceptable trim loss value. Thus, we can improve our algorithm by implementing an adaptive simulated annealing approach to adjust the acceptable trim loss value automatically. The author is currently working in this direction.

REFERENCES

- J. Blazewicz, M. Drozdowski, B. Soniewicki and R. Walkowiak. Two dimensional cutting problem: basic complexity results and algorithms for irregular shapes. *Found. Contr. Eng.* **14** (1989).
- V. Cerny. A thermodynamic approach to the travelling salesman problem: an efficient simulated annealing algorithm. *J. Optim. Theory Appl.* **45**, 41–51 (1985).
- T. Dusan, K. N. Emina and S. Goran. Airline seat inventory control: an application of the simulated annealing method. *Transport. Planning Technol.* **17**, 219–233 (1993).
- J. Ferreira, Soeiro Neves, M. Antonio and P. Fonseca e Castro. A two-phase roll cutting problem. *Eur. J. Op. Res.* **44**, 185–196 (1990).
- D. Gemmill and J. L. Sanders. Approximate solutions for the cutting stock 'Porfolio' problem. *Eur. J. Op. Res.* **44**, 167–174 (1990).
- P. Ghandforoush and J. J. Daniels. A heuristic algorithm for the guillotine constrained cutting stock problem. *ORSA J. Computing* **4**, Summer (1992).
- T. K. Kampke. Simulated annealing: use of a new tool in bin packing. *Ann. Ops Res.* **16**, 327–332 (1988).
- S. Kirkpatrick, C. D. Gelatt and P. M. Vecchi. Optimization by simulated annealing. *Science* **220**, 671–680 (1983).
- K. K. Lai and W. M. Chan. Tabu search approach for cutting stock problem. *Proc. of the Third Conf. of the Operational Research Society of Hong Kong*, pp. 232–242 (1993).
- S. Lundy and A. Mees. Convergence of an annealing algorithm. *Math. Prog.* **21**, 498–516 (1986).
- M. Malek, M. Guruswamy, M. Pandya and H. Owens. Serial and parallel simulated annealing and tabu search algorithms for the traveling salesman problem. *Ann. Ops Res.* **21**, 59–84 (1989).
- N. Metropolis, A. N. Rosenbluth, M. N. Rosenbluth, A. H. Teller and H. Teller. Equation of state calculation by fast computing machines. *J. Chem. Phys.* **21**(6), 1087–1092 (1953).
- F. A. Ogbu and D. K. Smith. The application of the simulated annealing algorithm to the solution of the n/m/c subscript max flowshop problem. *Computers Ops Res.* **17**, 243–253 (1990).
- I. H. Osman. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Ann. Ops Res.* **41**, 421–451 (1993).
- I. H. Osman and N. Christofides. Capacitated clustering problems by hybrid simulated annealing and tabu search. *Int. Trans. Opl Res.* **1**, (1994).
- I. H. Osman and C. N. Potts. Simulated annealing for permutation flow-shop scheduling. *Omega* **17**, 551–557 (1989).
- S. A. Roberts. Application of heuristic techniques to the cutting-stock problem for worktops. *J. Opl Res. Soc.* **35**, 369–377 (1984).
- P. J. M. Van Laarhoven, E. H. L. Aarts and J. K. Lenstra. Job shop scheduling by simulated annealing. *Ops Res.* **40**, 113–125 (1992).
- H. H. Yanasse, A. S. I. Zinober and R. G. Harris. Two-dimensional cutting stock with multiple stock sizes. *J. Opl Res. Soc. (OQT)* **42**, 673–683 (1991).