Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2005), *19*, 33–38. Printed in the USA.
Copyright © 2005 Cambridge University Press 0890-0604/05 $16.00
DOI: 10.1017/S0890060405050031

PRACTICUM PAPER

Heuristic to optimize L-guillotine cutting operations

ALBERTO GÓMEZ, DAVID DE LA FUENTE, PAOLO PRIORE, AND JAVIER PUENTE

Departamento de Admon de Empresas y Contabilidad, Escuela Politécnica Superior de Ingeniería de Gijón, Universidad de Oviedo, Edificio de Energía, 33204 Gijón, Asturias, Spain

(RECEIVED October 14, 2003; ACCEPTED January 10, 2005)

Abstract

This study presents an application to optimize the use of an L-cut guillotine machine. The application has two distinct parts to it; first, a number of rectangular shapes are placed on as few metal sheets as possible by using genetic algorithms. Second, the sequence for cutting these pieces has to be generated. The guillotine's numeric control then uses this sequence to make the cuts.

Keywords: Genetic Algorithms; Heuristic; Optimization; Packing

1. INTRODUCTION

This study describes an application that our work team made for a company in the metalwork industry. The aim was to make a computer program to minimize the number of sheets of steel used by the company in the course of its production process.

The application that was designed focuses in particular on optimizing the use of the L-type guillotine that the company owns. In contrast to traditional guillotines, which can only cut vertically or horizontally, the L-type guillotine can cut in both these directions simultaneously, which lends greater flexibility to cutting, and leads to more efficient use of the steel sheets.

The company operates in the following way: orders for supplying rectangular pieces are received from the clients, and the order of production and numeric control of the guillotine are decided in view of these orders. The parts are cut from a fixed-size, rectangular base surface (generally referred to as the metal sheet) measuring 2995×1250 mm.

Based on the above premises, the program to be implemented must situate the different pieces that the client has ordered on the metal sheet, bearing in mind that orders are generally made for more than one unit of a particularly dimensioned piece, and that more than one metal sheet is generally required to complete any given order. Positioning should be such as make the maximum use of the sheet material, which is tantamount to minimizing the number of metal sheets that are used.

Once positioned, the program provides the guillotine's numeric control with the cutting sequence of the sheets (the order in which the pieces are to be cut). The guillotine begins cutting in the top, right-hand corner of the metal sheet, and successive cuts leave the pieces and the leftover material. To do this, an algorithm was designed based on positioning the pieces within the base surface and generating the corresponding cutting sequence for numeric control.

2. GENETIC ALGORITHMS (GAs)

GAs are a new approach to solving complex problems. As a result of the enormous increase in the capacity of computer technology, applying GAs, in recent years in particular, has become more and more widespread, because the problem of the cost of using computer facilities that might have arisen, is in reality only a minor one.

GAs came to the fore in the 1970s, through the work of Holland (1975). The aim was to provide solutions to the many problems that arose in different industrial and service sectors that proved extremely difficult to solve with the methods that were known at the time. The main contribution of GAs is solving optimization and search problems by providing a solution that is not the optimal one but that is, nevertheless, a good approximation to the optimal one.

Reprint requests to: Alberto Gomez, Departamento de Admon de Empresas y Contabilidad, Escuela Politécnica Superior de Ingeniería de Gijón, Universidad de Oviedo, Edificio de Energía, Campus de Viesques, 33204 Gijón, Asturias, Spain. E-mail: agomez@epsig.uniovi.es

https://doi.org/10.1017/S0890060405050031 Published online by Cambridge University Press

How GAs work, which is also summarized in Figure 1, will next be explained. The starting point is an initial population of solutions, which is usually randomly generated. This population undergoes a number of transformations designed to improve the solutions provided. Such transformations are made in the main loop of the algorithm, and have three basic stages: selection, reproduction, and replacement (Goldberg, 1989; Michalewicz, 1996).

The selection stage consists of sampling the initial population, thereby obtaining a new population with the same number of individuals as the initial one. This stage aims at improving the quality of the population by favoring those individuals that are more adequate for a particular problem (the "quality" of an individual is gauged by calculating its fitness, which indicates how good a solution is). Although there are numerous techniques around for the selection process, the most widely used one is *Stochastic Universal Sampling* (Baker, 1987).

The reproduction stage employs what are called "generic operators." "Crossover" and "mutation" operators are the ones that are most widely applied. Generally speaking, the crossover operator works by taking two parents and exchanging part of their strings, thereby generating two new individuals, the offspring. For its part, the mutation operator is applied to an offspring and alters its string in some way, for example, by changing the order of some of its genes. A further offspring is obtained in this way. At the end of the reproduction phase there are two independent populations: the parents and the offspring. The final stage of the process is called replacement, and it consists of forming a new population by mixing the two initial ones.

```
Procedure evolution program
Begin
    t←0
    initialize P(t)
    evaluate P(t)
    while (not termination-condition) do
    Begin
        t←t+1
        select P(t) from P(t-1)
        alter P(t)
        evaluate P(t)
    End
End
```

Fig. 1. The structure of a genetic algorithm adapted from Michalewicz (1996); $P(t)$, the population of the solution.

3. AN APPROACH TO THE PACKING PROBLEM

Cutting problems like the one this particular company has are common to the textile, leather, shipbuilding, metal, and many other industries where obtaining pieces from a base plate is a recurrent activity. Since the 1960s, many researchers have tackled the problem and attempted to design algorithms to solve it.

The cause of the problem lies in the fact that the raw materials that industry uses are available in certain standard sizes that usually need cutting down before they can be used in the industrial process. The obvious aim of this cutting phase is to make the maximum use of the raw material. Outstanding among early work in this field was that carried out by Gilmore and Gomory (1961, 1963) in resolving single dimension problems; the scope of this work was widened in 1965 to 2-dimensional problems (Gilmore & Gomory, 1965). Analyses of these problems have since spread rapidly, although no global solution has been offered for all of them because of their complexity.

One way of solving this type of problem is to divide it into several subproblems (Dyckhoff, 1990), and try to solve each of them separately. This study will focus in particular on one such subproblem, the packing problem, which fits perfectly into the kind of industrial process that a solution is needed for.

Various authors have confronted this particular problem, although none have found a method that provides the optimum solution for every case (Hopper & Turton, 2001; Lodi et al., 2002). Healy et al. (1999) presented an efficient algorithm for identifying the empty space that can be used to cut a new piece when the existing cutting pattern does not have the bottom-left property. A cutting pattern has the bottom-left property when each cut piece is as far down, and as far left, as possible in the stock rectangle.

Lai and Chan (1997) presented a heuristic algorithm based upon an evolutionary strategy approach. Their algorithm includes an improvement (hill-climbing) procedure based on dividing the ordered list into active (cut) and inactive (uncut) pieces and seeing if any (currently) uncut pieces can be cut. Their mutation process involves swapping two pieces in the ordered list. Leung et al. (2001) used the ordered representation of Lai and Chan (1997) and discussed producing a cutting pattern from it using both the difference process algorithm of Lai and Chan (1997) and a standard bottom-left algorithm (Jakobs, 1996). They showed that problems exist such that the optimal solution cannot be found by these two approaches. They presented a simulated annealing heuristic with a move corresponding to either swapping two pieces in the ordered representation or mov-

https://doi.org/10.1017/S0890060405050031 Published online by Cambridge University Press

ing a single piece to a new position in the representation. They also presented a genetic approach involving five different crossover operators.

4. A DESCRIPTION OF THE PROBLEM

Before analyzing the solution that we have proposed, more in-depth comments should first be made on the characteristics of the problem to be solved.

The first point to be noted is that the dimensions of the metal sheets are always the same. This simplifies the problem a great deal, as pieces are always placed on the same type of sheet. Furthermore, as customers generally order pieces that are of large dimensions, the number of pieces that fit onto a single metal sheet generally ranges between 5 and 20.

Each customer order is generally made up of several hundred pieces, and as they are usually large, more than 50 metal sheets are generally involved in each order.

Furthermore, the particular conditions of the L-type guillotine used by the company entails space on the metal sheets being lost, which is an important point to be kept in mind. On its first approximation to the metal sheets the guillotine makes two "approximation cuts" of 5 mm, one vertically and the other horizontally, besides which the clamping system used to hold the metal causes a further loss of 20 mm horizontally. Figure 2 shows the material loss that results from using this particular L-type guillotine.

The initial dimensions of the surface are 2992×1250 mm; but, considering the above factors, the new dimensions of the metal sheets are $(2992 - 5) \times (1250 - 20 - 5)$ mm.

Having commented on the particular nature of the problem that was analyzed, two independent algorithms are developed next: the position algorithm and the cutting sequence algorithm.

5. THE SOLUTION PROPOSED FOR CUTTING

This section will analyze the GA designed to minimize the number of steel sheets that were used. The steps that were followed to do this will be described next. The codification used in the algorithm is based on integer numbers; each piece to be placed on the metal sheet is assigned a number, and an individual chromosome is formed with a string of numbers. The order in which the numbers appear in the string represents the positions on the metal sheet of the pieces.

An example will be worked through to explain codification clearly: imagine you have four rectangular pieces to be placed on the metal sheet, and that each of these pieces is assigned a correlative number, the first piece is assigned the number *1*, the second is number *2*, and so forth. The individual (1, 3, 4, 2) could be a solution to the problem, which would mean that piece 1 is first onto the metal sheet, followed by 3, then 4, and finally by 2.

In the case being analyzed in this study, this codification needs some modification, because, as has already been mentioned, not all the pieces fit onto the same metal sheet, and several sheets must be used to make all the pieces ordered by the client.

The first problem to solve is the way in which a piece is assigned to a metal sheet when the individual is analyzed. When the first value of the individual (the first piece) is read, a metal sheet is "activated," and the piece is places on the sheet; the individual' second value is then read, and an analysis determines whether it can be fitted onto the sheet or not (the "free fall with replacement algorithm" is used to determine if a piece fits in a sheet; see the next paragraph). If the piece fits, then that piece is assigned to the metal sheet and the third value is read off. This reading process continues until one of the pieces fails to fit onto the metal sheet. On detecting that a piece cannot be assigned to a metal sheet because the space available is not large enough, a second metal sheet is "activated" and the piece is assigned to this second sheet. The process is iterated until all the pieces have been placed.

For the positioning of the pieces on the sheets, a modification of the "bottom-left algorithm" (Jakobs, 1996) was used. The reason for using this modification lies in the fact that Jakob's algorithm does not permit certain positions of pieces on the sheet (Gómez et al., 2000).

This algorithm, called the "free fall with replacement algorithm," works as follows: the first piece assigned to the metal sheet is placed in the bottom left-hand corner (piece 1, Fig. 3), and the others are places as low down in the space as they can possibly fit. Once the piece has been placed, a check is made as to whether there is a space below it. If

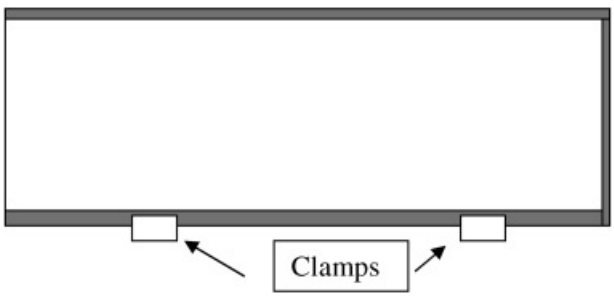


Fig. 2. Leftovers.

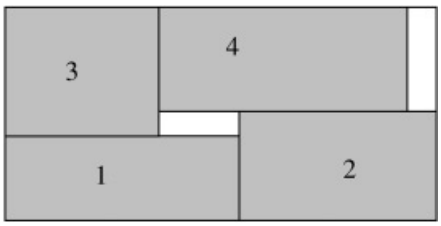


Fig. 3. Free fall with replacement.

https://doi.org/10.1017/S0890060405050031 Published online by Cambridge University Press

there is a space, then a search ensues to see if one of the rectangles still to be placed fits into it. If such a rectangle shows up, it is slotted into the space.

Having established the methods used to codify the placement of the pieces and to calculate their fit, the type of crossover and mutation to be used must next be decided on. In the light of earlier work that we have carried out (Gómez & De la Fuente, 2000), a partial matching-type crossover was opted for, along with an order-based mutation.

The *partial matching crossover* (PMX; Goldberg & Lingle, 1985) is a crossover type that is widely used in published work on how to resolve the travelling salesman problem by using GAs with decimal coding. Adapting this crossover to packing problems is simple, and will now be described. Given two parent chromosomes, the operator copies a substring of one of the parents directly into the same positions in the offspring. The remaining positions are then filled with the values that have yet to be used, in the same order as they occur in each of the parents.

The mutation, called *order-based mutation*, is based on the work of Davis (1991). It consists of interchanging the positions of two rectangles of the same individual.

6. CUTTING THE PIECES

As has already been pointed out, a second algorithm had to be applied to instruct the guillotine's numeric control on how to cut the metal sheets to obtain the pieces. It is useful to explain the way the guillotine approaches the steel sheets to understand how the algorithm works. The guillotine starts at the top, right-hand corner of the sheet, and after making the initial two cuts that have already been mentioned in Section 4, it proceeds to cut the pieces.

The algorithm responsible for carrying out this task acquires the distribution of the pieces in the metal sheet from the GA, and generates an output file that tells the guillotine's numeric control in what order to cut the pieces. The algorithm simulates the guillotine working process.

To indicate this sequence the *X* and *Y* coordinates of the different cuts need to be known. Thus, for example, to cut pieces 1 and 2 of Figure 4, the coordinates of the bottom left-land corner of both pieces should be given. The guillotine first cuts up to coordinates *X* and *Y* of the second piece and then cuts the first piece.

The problem of this cut lies in the areas where there is no piece assigned, which also has to be cut if the pieces ordered by the client are to be produced. Figure 5 highlights the problems that cutting these pieces involves; if the useful pieces are the rectangles (1, 2, 3, 4), and (A, B) represent the wasted parts of the metal sheet, then piece B cannot be obtained from a single cut, and must be divided in two because of its nonrectangular shape.

The task of the sequencing algorithm that was designed consists of detecting these problematical parts and of deciding how to divide them. To do this, the algorithm simulates how the guillotine works. First, it acquires the positions of the pieces to be cut from the GA and then it fictitiously places the guillotine in the top, right-hand corner of the metal sheet and attempts to start cutting the different pieces one by one. When it detects that a piece cannot be cut because there is a waste piece in the guillotine's cutting area, it tells numeric control to eliminate this leftover piece before proceeding to cut the customer's piece.

Figure 5 serves to illustrate this process. First, an attempt is made to cut piece 1. To see if this piece can be cut, the guillotine is "placed" in the lower left-hand corner of this same piece, and the metal sheet is "cut" in this position. If the piece is produced cleanly, the following piece is tried; if it is not, then the extra material on piece 1 is eliminated. In this example, the piece is obtained without extra material and the cut is accepted; once liberated, the simulator attempts to cut the next piece, which in this particular case is piece 2. However, a problem arises, because if the guillotine is placed in the bottom, left-hand corner of this piece and the cut is made, the piece that is produced is L-shaped and is made up of pieces A, 2, and part of B. Once this problem is detected the algorithm must determine how to obtain piece 2 without extra material. In this case, the algorithm tells numeric control that first piece A should be cut, and for this to occur it provides numeric control with the relevant coordinates for the bottom, left-hand corner; then one part of piece B must be cut, so the algorithm must provide numeric control with the coordinates. When these cuts have been made, piece 2 can easily be cut out. Piece 3 is the next to be cut; but before doing so, the part of piece B that has yet to be cut must be cut; finally, piece 4 is cut out.

7. RESULTS

The computer program that was already being used by the company in question was used in the study this article

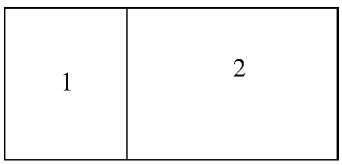


Fig. 4. An example of the cutting sequence.

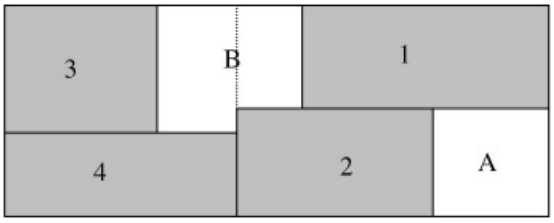


Fig. 5. The cutting sequence.

https://doi.org/10.1017/S0890060405050031 Published online by Cambridge University Press

describes. As this program was also involved in trying to minimize the number of metal sheets being used, it is useful to compare the solutions provided by this program with the solutions that our study came up with.

A number of experiments have been carried out using real data provided by the company, and our algorithms and the one previously used by the company were contrasted. Table 1 provides the results of this comparison.

The parameters applied during the different tests by the GA are crossover probability = 0.7, mutation probability = 0.3, special mutation probability = 0.4, number of generations = 3, and population size = 10 (only 30 calculations will be made using these parameters). The low number of generations and the small population size are both a result of the type of solution that was demanded. The company required a "quick fix," where within 3 min the algorithms would generate a solution that was similar to the solution provided by the original system as shown in Table 1 for solution 1.

Table 1 highlights how the heuristic proposed provide better than or at least equal results as those offered by the guillotine's original program (the old algorithm) in most cases. It also indicates how GA solutions get better as the number of pieces involved gets higher.

Tests were run to generate random solutions (column 4, Table 1) to check whether GA was really necessary. The *random* solution was reached by generating 30 random solutions and using the free fall with replacement heuristic to analyze the number of sheets needed to place and cut all the rectangles. The results shown in this column confirm the need for GAs.

8. CONCLUSIONS

This study presents a technique to solve an industrial problem involving the automation of an L-type guillotine. This automation meant designing a program capable of distributing a number of rectangular shapes ordered by a customer in such a way as to minimize the number of metal sheets that were used. Furthermore, the program had to tell the guillotine's numeric control what sequence the different pieces were to be cut in.

To check the quality of the solutions provided by the algorithm, a set of examples was applied. It was shown that the quality of the solutions provided by our algorithm is better in most cases, and it only obtains similar results when the number of rectangles to be placed is small, in which case exact methods can be used anyway.

Table 1. *Experimental results*

Example	Pieces	Old Algorithm	Random	GA
1	547	69	71	67
2	229	45	48	45
3	52	21	21	21
4	16	5	4	4
5	228	42	45	42
6	722	110	107	103
7	683	151	149	149
8	732	176	175	173
9	798	114	115	111
10	754	136	133	131
11	580	82	79	79
12	1346	223	219	208
13	192	42	42	42
14	573	53	57	52
15	529	92	89	87

Column 2 shows the number of pieces in the experiment, and the other columns show the number of metal sheets required to produce the pieces using each of the algorithms respectively

REFERENCES

Baker, J.E. (1987). Reducing bias and inefficiency in the selection algorithm. *Proc. Second Int. Conf. Genetic Algorithms*, pp. 14–21.

Davis, L. (1991). *Handbook of Genetic Algorithms*. New York: Van Nostrand Reinhold.

Dyckhoff, H. (1990). A typology of cutting and packing problems. *European Journal of Operational Research 44*, 145–159.

Gilmore, P.C., & Gomory, R.E. (1961). A linear programming approach to the cutting-stock problem. *Operations Research 9*, 724–746.

Gilmore, P.C., & Gomory, R.E. (1963). A linear programming approach to the cutting stock problem. *Operations Research 11*, 863–888.

Gilmore, P.C., & Gomory, R.E. (1965). Multi stage cutting stock problems of two and more dimensions. *Operations Research 13*, 94–1120.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization & Machine Learning*. Reading, MA: Addison–Wesley.

Golberg, D.E., & Lingle, R. (1985). Alleles, loci, and the TSP. *Proc. First Int. Conf. Genetic Algorithms*, pp. 154–159. Hillsdale, NJ: Erlbaum.

Gómez, A., & De la Fuente, D. (2000). Resolution of strip-packing problems with genetic algorithms. *Journal of the Operational Research Society 51*, 1289–1295.

Gómez, A., De la Fuente, D., & Priore, P. (2000). Resolución del problema de strip-packing mediante la metaheurística algoritmos genéticos. *Boletín de la SEIO 16(2)*, 12–16.

Healy, P., Creavin, M., & Kuusik, A. (1999). An optimal algorithm for rectangle placement. *Operations Research Letters 24*, 73–80.

Holland, H.J. (1975). *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press.

Hopper, E., & Turton, B.C.H. (2001). A review of the application of meta-heuristic algorithms to 2D strip packing problems. *Artificial Intelligence Review 16(4)*, 257–300.

Jakobs, S. (1996). On genetic algorithms for the packing of polygons. *European Journal of Operational Research 88*, 165–181.

Lai, K.K., & Chan, J.W.M. (1997). Developing a simulated annealing algorithm for the cutting stock problem. *Computers & Industrial Engineering 32*, 115–127.

Leung, T.W., Yung, C.H., & Troutt, M.D. (2001). Applications of genetic search and simulated annealing to the two-dimensional non-guillotine cutting stock problem. *Computers & Industrial Engineering 40*, 201–214.

Lodi, A., Martello, S., & Monaci, M. (2002). Two-dimensional packing problems: a survey. *European Journal of Operational Research 141(2)*, 241–252.

Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs*. New York: Springer–Verlag.

Alberto Gómez attained his BS in industrial engineering in 1994 and his PhD in 2001, both from the University of Oviedo. He is currently an Assistant Professor in the School of Industrial Engineering at Oviedo University. Dr. Gomez' research interests include applications of genetic algorithms in production problems, and information systems.

https://doi.org/10.1017/S0890060405050031 Published online by Cambridge University Press

David de la Fuente graduated with a BS in industrial engineering in 1983 at the University of Madrid and received his PhD from the University of Oviedo in 1987. He is currently a Professor in the School of Industrial Engineering at Oviedo University. His current research interests are in the areas of artificial intelligence, forecasting, and manufacturing. Dr. de la Fuente is a member of APICS.

Paolo Priore attained his BS in industrial engineering in 1991 and his PhD in 2001, both from the University of Oviedo. He is currently a Professor in the School of Industrial Engineering at Oviedo University. His research interests include machine learning applications in production problems, simulation, and manufacturing. Dr. Priore is a member of EUROMA (European Operations Management Association).

Javier Puente graduated with a BS in industrial engineering in 1994 and attained his PhD in 2002, both from the University of Oviedo. He is currently an Assistant Professor in the School of Industrial Engineering at Oviedo University. Dr. Puente's current research interests are in the areas of fuzzy logic in production problems and total quality management.

https://doi.org/10.1017/S0890060405050031 Published online by Cambridge University Press