Journal of the Operational Research Society (2007) 58, 429–438

www.palgrave-journals.com/jors

A local search approach to a circle cutting problem arising in the motor cycle industry

KA Dowsland[1,2]*, M Gilbert[3] and G Kendall[1]

[1]*The University of Nottingham, Nottingham, UK;* [2]*Gower Optimal Algorithms Ltd, Swansea, UK; and* [3]*Kitcross, Presles, Belgium*

This paper is concerned with the development of a customized circle packing algorithm for a manufacturer of sprockets for the motor cycle industry. Practical constraints mean that the problem differs somewhat from those tackled elsewhere in the literature. In particular, the layouts need to conform to a given structure. This is achieved by using a local search algorithm with an appropriate starting solution and a series of neighbourhoods designed to preserve the layout structure. Empirical evidence based on real data shows that the quality of the resulting solutions closely matches that of cutting patterns currently produced by human experts. Computation times average around 20–30 s per order as compared to several hours for an equivalent manual solution.

Journal of the Operational Research Society (2007) **58,** 429–438. doi:10.1057/palgrave.jors.2602170
Published online 22 February 2006

Keywords: heuristics; local search; cutting stock problem; production

Introduction

The problem of cutting a set of pieces from one or more rectangular stock-sheets arises in many manufacturing industries and cutting and packing problems have been widely researched for over 50 years (Kantorovitch, 1939; Gilmore and Gomory, 1961; Dowsland and Dowsland, 1992; Sweeney and Paternoster, 1992; Lodi *et al*, 2002). Although there have been many attempts to produce generic algorithms for specific classes of piece types (eg circles, rectangles, polygons, etc), the objectives and constraints imposed by real-world problems vary so much that there are many examples of customized solutions for a given problem. The focus of this paper is a local search-based packing algorithm designed specifically to meet the requirements of a circle cutting operation arising in a Belgian company involved in the manufacture of sprockets (toothed wheels used in a chain drive) for the motorcycle industry.

Although circle packing has long been of interest to mathematicians (Koebe, 1936), these types of problem have received less attention than many other stock-cutting problems such as rectangle packing (Beasley, 1985; Burke *et al*, 2004; Dowsland *et al*, 2006) and irregular packing (Dowsland *et al*, 1998; Bennell and Dowsland, 2001; Dowsland *et al*, 2002; Burke *et al*, forthcoming). Much of the literature on circle packing is of a theoretical nature (Williams, 1979; Steinhaus, 1999; Collins and Stephenson, 2003; Stephenson, 2003; Stoyan and Yas'kov, 2004; Birgin *et al*, 2005), while other publications are limited in scope—for example, packing circles into larger circles (Kravitz, 1967; Wells, 1991; Huang *et al*, 2001, 2003), packing circles into a square (Graham and Lubachevsky, 1996; Nurmela and Östergård, 1997; Boll *et al*, 2000), or dealing with three-dimensional (sphere) problems (Conway and Sloane, 1992; Chen *et al*, 2001).

Publications motivated by practical circle cutting or packing problems are less common. Examples include: collating cylindrical products into rectangular cases that provide efficient palletisation (Dowsland, 1991), packing long cylindrical items into shipping containers (George *et al*, 1995; Correia *et al*, 2000; Huang *et al*, 2005), packing optical fibres into tubes of minimum diameter (Wang *et al*, 2002), and punching circular blanks from a silicon sheet in the manufacture of rotors and stators for electric motors (Cui, 2005).

In this paper, we tackle a real-world problem. Superficially the problem is similar to many of those above in that it involves packing circles of different diameters into one or more containing rectangles of known dimensions. However, none of the above approaches are directly applicable for two reasons. Firstly, the number of circles to be cut is not fixed, as additional circles can be produced for inventory. Secondly, from a practical point-of-view it is desirable to produce layouts that are similar to those currently produced manually by experienced workers. This makes the problem unique. We have therefore developed our own approach, based on local search. It is widely accepted that utilizing domain knowledge can often improve the success of a search algorithm (see, for example, Wildemuth, 2004), and this knowledge can often be exploited when making

*Correspondence: KA Dowsland, Gower Optimal Algorithms Ltd, 5 Whitestone Lane, Newton, Swansea SA3 4UH, UK.
E-mail: k.a.dowsland@btconnect.com

the problem-specific decisions concerning the definition of the solution space, the evaluation function, the initial solution and the neighbourhood structure (see for example Reeves, 1993; Glover and Laguna, 1997; Glover and Kochenberger, 2003). In this paper, we address all four decisions. In particular, we show how the use of appropriate starting solutions and neighbourhood moves allows us to produce solutions with the required characteristics, and we examine the potential benefits and drawbacks of relaxing some of the constraints in the definition of the solution space. The result is an algorithm that is able to produce high-density layouts that meet all the practical requirements of the company.

In the next section, we define the problem together with the operating environment in which the solution is to be implemented. This is followed by a description of the solution procedure and a discussion on balancing the flexibility of the search with the need to home in on *feasible* solutions. An empirical evaluation of different variants of the algorithm based on experiments on nine problems selected from the company's archives is then presented and the performance of the best variant is confirmed on a further sample of 100 problems. The paper concludes with some general comments and suggestions for further work.

Problem definition

The current situation

The company's production process is shown in Figure 1. It starts with aluminium sheets of dimensions 1600 mm × 2840 mm, in one of the four different gauges (thicknesses). The first process cuts discs of given diameters. These discs are then further processed so as to produce the full range of sprockets by transferring the discs to a tooth cutter, where they are effectively transformed into cogs. At this stage, any surplus production can be held as inventory. The remainder of the process is concerned with further processing of the cogs. In this paper, we are concerned with the first phase of the production process, that of disc cutting.

Each order is treated separately. The set of discs that must be cut is determined by checking the inventory for any unallocated cogwheels of the required dimensions, and then setting up a disc-cutting plan in order to accommodate the remaining items in the order. An order may consist of more than one gauge and within each gauge there are typically between 60 and 2000 discs that need to be cut. The number of discs that can be cut from each sheet is typically between 60 and 100, depending on the diameters. Once a sheet has been used it is discarded (ie there is no possibility of using partially used sheets for future orders). However, the total order book consists of a set of standard diameters and, in order to improve sheet utilization, the company will cut extra discs to be held in inventory and used to meet future orders.

Cutting is executed by a milling cutter of diameter 10 mm. In order to avoid the sheet splitting or breaking 3 mm has to be left between extruded circles. Around the border of the sheet 5.5 mm must also be left to allow for it to be secured during the cutting process. The width of the cutter and the gap between the circles can be accommodated by adding 23 mm to every circle diameter. As this will also leave a gap of 1.5 mm along each edge of the sheet; the 5.5 mm gap around the border can be achieved by reducing the dimensions of the sheet by 8 mm. Although the set-up time for the disc-cutting machine is independent of the diameters cut, the tooth-cutting machine needs to be reset for each different diameter. Therefore, any diameter D_i for which there are already sufficient cog-wheels in stock will be supplied from stock. For all other diameters, the objective is to define a set of cutting patterns for at least $q_i = b_i - s_i$ circles of diameter d_i, where $d_i = D_i + 23$, b_i is the number of circles of diameter D_i in the order and s_i is the number of cogwheels of diameter D_i in stock, in such a way as to minimize the amount of waste material, while keeping the number of cogwheels made for inventory to a *reasonable* amount.

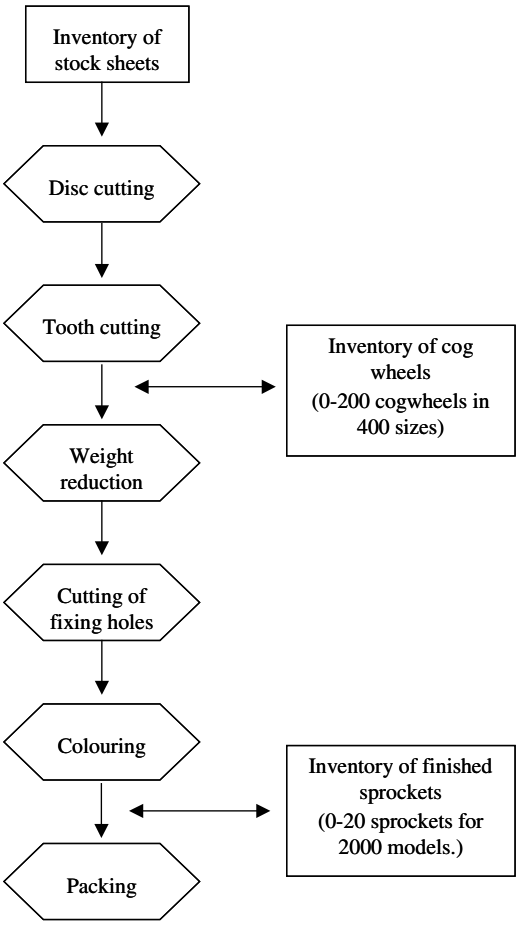


Figure 1 The production process.

The cutting patterns are currently produced manually by experienced staff, who plan and draw the new patterns using AutoCAD. For a typical order, about two-thirds of the

sheets re-use existing patterns, while the remaining third are produced from scratch. Each new sheet takes around one and a half man-hours to plan. Typically around 32–34% of the material is waste, including the material wasted by the milling-cutter, the gaps between the discs, and the waste material necessary for fixing the sheet. This translates to about 81–83% density for the circle packing problem using the adjusted diameters and the reduced sheet dimensions. Figure 2 shows a typical set of layout patterns for a small order requiring five sheets.

Formalization of the problem

The objective of this study is not to produce a computerized system to replace the manual operation but to provide an automatic system that is able to *support* the current expertise. For example, by relieving the pressure during busy times, or covering periods when experienced staff are unavailable due to holidays, illness, etc. In order to build an automatic system, the objectives and constraints on the problem need further formalization. In particular, two issues need to be addressed. The first concerns the types of layout that are required and the second concerns the balance between the objectives of minimizing waste and keeping inventory down to a 'reasonable' level.

Figure 2 is typical of the types of layouts currently in use. The most obvious feature of these layouts is that they are based on a hexagonal lattice made up of blocks of circles of the same diameter, with some adjustments within rows or around the edges. Although the operation of the cutting equipment does not impose any restrictions on the structure of the layout, it was deemed desirable to produce layouts similar to those in Figure 2 for two reasons. Firstly, after cutting, the pieces are sorted manually for loading onto the tooth cutter. The advantage of layouts consisting of large blocks of circles of equal diameter is self-evident. Secondly,

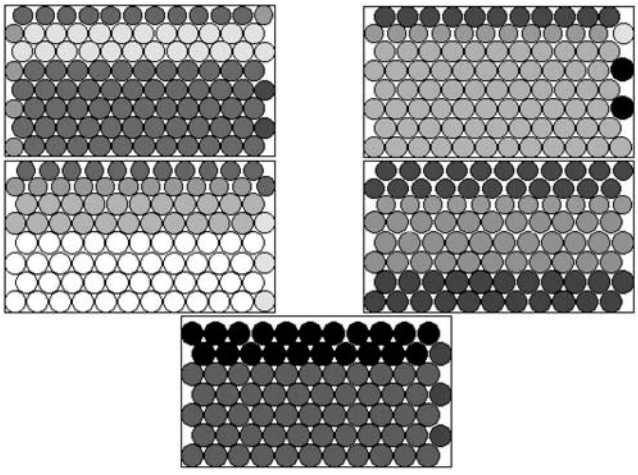


Figure 2 A typical layout produced by the current manual solution approach.

as the system was to be run in parallel with the manual operation, with the layouts being archived for potential re-use as a part of the cutting plan for future orders, there was a general consensus that the layouts should be of a similar type. It is also worth noting that such layouts are unlikely to compromise the quality of the solutions in terms of sheet utilization. This is because orders tend to consist of large numbers of circles of a given diameter where the range of diameters is usually small. In these circumstances, layouts based on hexagonal patterns are also likely to yield dense packings.

The balance between low inventory and waste minimization also needs addressing. Intuitively inventory will be minimized if the number of sheets used is the minimum required to meet the order quantities. Conversely extra sheets will allow more flexibility in the quantities of each diameter to be cut and this may result in improved sheet utilization. Furthermore, due to the set-up times on the tooth-cutting machine, sufficient inventory to supply small quantities of a particular diameter is viewed as a positive advantage. The company stated that they were satisfied with the levels of wastage and inventory produced by the current system. However, no figures for the relationship between inventory costs and material costs, or for the mean or maximum amount of inventory that would be acceptable, were available.

The current manual approach does not involve any formal guidelines, but consultation with the layout designers allowed us to estimate the amount of over-production they would expect to produce in different circumstances. This typically translates to between 5 and 30% of the total area packed, with the higher figures corresponding to orders with several small order quantities. However, for very small orders (ie less than five sheets) the surplus can amount to more than 50%. Typically, the manual approach starts with the design of an initial cutting plan based on hexagonal nestings. These are built up row-by-row, and all the circles of a given diameter are allocated before the next diameter is considered. Incomplete rows are filled with surplus circles of that diameter. This procedure determines the number of sheets that will be used, which in turn has a direct influence on the surplus production.

Our approach is similar in that we determine an upper bound, Q, on the minimum number of sheets required by applying a fast packing heuristic based closely on the logic used in the manual approach. Full details of this heuristic are given in the next section. Our objective is then to find an efficient packing of the Q sheets that includes a sufficient number of circles of each diameter. Thus, the problem can now be stated formally as follows.

Let D_i, d_i, b_i, s_i and q_i be defined as in the previous section. Given a set of n diameters d_i such that $b_i > s_i$ let Q be the number of sheets required to pack at least q_i circles of diameter d_i using an appropriate packing heuristic. Then our objective is to find a set of Q layouts of the required format

(one for each sheet) that maximizes

$$\frac{\sum_{j=1}^{Q} u_j}{Q} \text{ such that } \sum_{j=1}^{Q} a_{ij} \geqslant q_i, \quad \text{for } i = 1, n$$

where a_{ij} is the number of copies of circle diameter d_i in layout j and u_j is the packing density of layout j. Note that the set of diameters under consideration is restricted to those for which $q_i \geqslant 1$ in order to avoid additional set-up times on the tooth cutter.

The solution approach

Local search, or hill climbing, is a heuristic approach that starts from an initial solution, s, and tries to improve it by making a series of small changes or perturbations. When an improvement is found the improved solution becomes the incumbent solution and the process is repeated until a solution with no improving perturbations is reached. The set of solutions that can be obtained by applying an allowable perturbation to a solution, s, is known as the neighbourhood of s, and the replacement of s by one of its neighbours is known as a neighbourhood move. The local search process can be summarized as follows.

Local search for minimisation

Definitions:

S = space of feasible solutions
$N(s)$ = the neighbourhood of s
$f(s)$ = cost of solution s
$N^+(s) = \{t \in N(s): f(t) < f(s)\}$

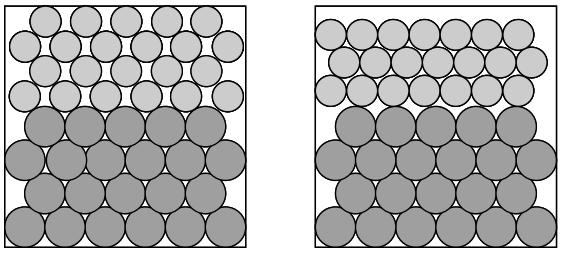
Procedure:
{Select $s_0 \in S$
Set $s_c = s_0$
While $N^+(s) \neq \varnothing$
{Select $s \in N^+(s_c)$
Set $s_c = s$}
Endwhile
Stop (s_c is the approximation to the optimal solution)}

Local search is often criticized for being unable to move sufficiently far from the starting solution without the use of uphill moves and/or additional diversification strategies such as those inherent in simulated annealing (Reeves, 1993) or tabu search (Glover and Laguna, 1997). However, in this case, this limitation can be viewed as a positive advantage, in that if we select an initial layout with the right characteristics we can attempt to improve it with a series of small changes that will not be disruptive enough to destroy its underlying structure.

In order to implement the above local search algorithm for a given problem, it is necessary to determine problem-specific decisions for the solution space, S, the starting solution s_0, the neighbourhood structure, N, and the evaluation function f. The way in which a suitable neighbour, $s \in N^+(s_c)$ is selected must also be defined. This section focuses on the problem-specific decisions. We start by considering three variants of a construction heuristic for the initial solution based on the current manual approach. We then go on to develop a series of complex neighbourhood moves designed to allow sufficient flexibility to find improved solutions while ensuring that the layouts produced are similar to those in Figure 2. As the quantity of each circle in the final solution is not known *a priori* these moves must allow for these quantities to change. This leads us to consider relaxing the definition of the solution space, S, to include solutions that violate the lower-bound constraints on the quantities of each diameter to be cut and to develop an evaluation function, f, that penalizes these violations in such a way as to guide the search to high-quality feasible solutions.

The initial solution

The initial solution is based on the initial solution produced by the manual approach and is used to determine the number of sheets, Q. It is generated deterministically and is composed of blocks of circles, each block consisting of circles of a single diameter, nested in a hexagonal pattern. Three variants of this approach are considered. The first two differ in the way the changeover between blocks is handled. The best of these two options is then subdivided according to whether the circles types are ordered randomly, or by diameter. In all cases the initial layout is constructed so as to obey the lower-bound constraints by taking each piece type in turn, and, starting from the bottom of the stock sheet, filling rows in the required pattern until q_i circles are packed. If the last row is incomplete then it is completed with additional circles of the same type. The process then continues with the next type, starting a new sheet whenever the current sheet cannot accommodate a row of the required diameter. The requirement to nest circles of different sizes on the same sheet needs special consideration as the hexagonal lattice given by the centres of the circles will obviously change with the circle diameter. There are two ways of dealing with this, as shown in Figure 3. The first, which we will denote *NEST* is to continue the nest as the diameters change by maintaining the same x-coordinate for the centres, as in Figure 3a. The second, which we will denote *BLOCK*, is to start a new nest for each type by placing the bottom of the first row of the new type along a line defined by the top of the last row of the previous type, as in Figure 3b. Both approaches have their disadvantages. In *NEST*, space will be wasted as the horizontal distance between circles in each row will be determined by the largest circle on the sheet. In *BLOCK*, space will be wasted at the changeover between types, which may result in fitting a smaller number of rows.



(a) (b)

Figure 3 Layout strategies for a change in diameter. (a) The bottom of the lattice of small circles is nested into the top of the lattice of larger circles and the x-coordinates of the original hexagonal lattice are preserved. (b) The two lattices are separated, enabling a tighter lattice for the smaller circles.

Experiments on 100 randomly selected orders showed that *NEST* required a total of 1044 sheets while *BLOCK* required a total of 1074 sheets, with *BLOCK* requiring less sheets than *NEST* for only two out of the 100 orders. For this reason, and because *NEST* more closely matches the manual approach, we chose *NEST* as the basis of our initial construction. The simplest way of applying *NEST* is to sort the circles into decreasing diameter order so that the largest diameter on each sheet is packed first. This also has the advantage that circles of similar diameter will appear on the same sheet, thereby minimizing the size of the gaps between the smaller circles. This is the strategy adopted for all the experiments reported here. Experiments with different permutations of the diameters were also carried out. The results showed that these random permutations increased computation times and failed to give any significant improvement in solution quality. They were therefore discarded in favour of the decreasing diameter option.

Layouts produced by the above procedure have two obvious sources of wasted space. On the earlier sheets where diameters are relatively large there is often space for a row of smaller circles to be placed at the top of the sheet, while the last sheet usually contains a significant area of waste above the last row packed. The procedure was therefore modified to reduce this waste. Firstly, when there is not sufficient space to fit another row of the current circle type, i, instead of moving immediately to a new sheet the remaining circle types are scanned for the type k of largest diameter that could form a final row. If such a k exists then a row of type k is added and the packing then continues with type i on the new sheet. Secondly, if there is space on the final sheet we add circles for inventory as follows. The circle type that has the smallest surplus packed to date is chosen, and an additional row of this type is added. Owing to the ordering of the circles this is likely to be larger than the circle in the previous row. For this reason, and because our objective in forming the initial layout is to form large hexagonal blocks of circles, this row is not necessarily inserted into the last sheet but is placed above the last row of circles of its own type. All subsequently placed rows are then moved accordingly, even where this involves moving to the next sheet. In this case, the number per row is adjusted as necessary. The process is repeated until there is no circle type for which a row can be added without requiring a new stock sheet. Empirical evidence, based on the experiments described in the next section show that the density of an initial layout produced in this way lies in the region of 77–79%.

As stated above the layout produced by this initialization phase is used to determine Q, the number of sheets to be used, which then remains fixed for the subsequent improvement phase.

The improvement stage

An examination of the layouts in Figure 2 and other layouts produced by the company, together with discussions with the current layout designers, suggested that the perturbations needed to improve the initial layouts to form patterns that are consistent with current practice can broadly be divided into two types: those altering or adding whole rows and those adding or changing circles along the vertical edges of the sheet. It is also apparent that very few simple moves, that is, moves based on just one such perturbation, are likely to lead to an improvement on their own. For example, changing the size of a single row of circles will only lead to an improvement in packing density if the new type is larger. Nevertheless replacing a circle by a smaller type may be a good move in the long run as it may allow space to add additional circles. We therefore define a neighbourhood move as a three-stage operation.

(1) A randomly chosen *row operation* aimed at changing one or more rows in the layout.
(2) An *add operation*, which attempts to add one or more new rows to the sheet(s) involved in the row operation.
(3) A *completion operation* that seeks further improvement by exhaustively working through a list of moves that allow some aspects of the hexagonal pattern to be destroyed.

The cost of a move is given as the decrease in the average density over all the sheets, once all three phases have been completed. The details of the moves allowed in each phase are detailed below.

Row operations

Five potentially useful row operations were identified. These are

R1 *Swap the last rows of two different sheets*. This involves taking the top rows of two different sheets and swapping them. In order to preserve the ordering of the diameters on the sheet, the swapped rows are

slotted into the appropriate place in the layout rather than being placed along the top. The exception to this is where the diameter of the new row is larger than the maximum diameter on the sheet. In this case, the new row is placed along the top edge of the sheet, effectively separating it from the nested pattern in the rest of the sheet and thereby allowing the hexagonal lattice in the rest of the sheet to be preserved. If the resulting layout exceeds the boundaries of the sheet then the move is not executed. Otherwise, the move will have created additional space along the top of one of the sheets, which may be sufficient for an extra row.

R2 *Swap the last two rows of two different sheets*. This move is similar to the above except that the two top rows are involved in the swap. This has the potential of creating twice as much space as R1, but it does not replace it as it is more likely to be infeasible.

R3 *Replace one or two rows by a circle of different diameter*. This simply replaces the top one or two adjacent rows by circles of a randomly selected diameter. Once again the new rows are inserted in the appropriate place to maintain the ordering of diameters in the sheet. This move will make a substantial change to the numbers of circles of each type, but is effective in situations, where just one sheet has a poor utilization, and the swap moves would result in a worse utilization for the remaining sheets.

R4 *Remove the row with the biggest circles*. This move stems from the observation that when there is a sheet with just one row of large circles the packing density tends to be poor, due to the gaps enforced in all other rows by the hexagonal arrangement. If this row is removed the remaining pieces can be moved closer together, leaving space along the vertical edges that could be utilized by pieces of the same or another type. Note that this phase of the move includes redefining the horizontal spacing of the lattice according to the largest remaining diameter but does not include adding extra circles to each row as this will be attempted during the completion phase.

R5 *Repeat the row with the biggest circles*. This move is the opposite of (R4) and is based on the observation that all rows apart from those with the biggest circles have wasted space between adjacent pieces. This move is designed to extend that part of the stock sheet covered by the densest packing. The new row is inserted on top of the last row with the given diameter and the rest of the layout is moved upwards. Any rows that exceed the top of the sheet are removed.

Add operation

After each row operation, all modified sheets are subjected to the add operation. This attempts to add a row. The circle types are considered randomly without replacement, until a diameter that will allow an extra row to be fitted is found or the list is exhausted.

Sheet completion

Whether or not the add operation was successful, each of the altered sheets is then considered for the completion operation. This consists of a series of moves that are considered in order, and accepted when an improvement is made. All the moves in the list are attempted whether or not an improvement has been found. The order in which the moves are considered is important and the list is ordered so that the moves that are likely to have the greatest impact are considered first. The moves are described below.

C1 *Add a column of extra circles*. Because the layouts are built in a hexagonal pattern no attempt is made to spread the layout across the width of the sheet. As the layout is built from the bottom left this means that there is often free space on the right of the sheet. This move attempts to fill this space with a column of circles.

C2 *Add a column of extra circles after making additional space*. In some cases there is not sufficient space to execute move C1. However, if the first column is replaced by circles of a smaller diameter, then the remaining circles can be shifted to the left, and sufficient space for an extra column may result.

C3 *Replace the last column by a column of larger circles*. If there is insufficient space for a new column the packing density may be increased by removing the last column and replacing it by a column of larger circles.

C4 *Stretch the pattern*. In some cases it may be better to utilize the full width of the sheet by stretching the pattern so that the circles are nested at an angle greater than 60°. The effect of this will be to create extra space at the top of the sheet, which may be sufficient for an additional row. This move involves stripping the layout of all but those pieces that conform to the hexagonal row layout, stretching that, and then attempting to add an extra row, or rows of circles.

C5 *Filling the top right corner*. Because the layout is built from the bottom left there is often free space in the top right corner. If there is sufficient space to fit an extra circle then a circle is added. If not, an attempt is made to increase the size of the circle closest to this corner.

In all cases, the circles considered for each move are selected randomly without replacement from those diameters that satisfy the move definition.

Maintaining feasibility

The above moves will change the number of circles of each diameter in the packing. This is obviously necessary in order to improve the packing density in a fixed set of sheets. However, it is possible that any search based on the unrestricted use of these moves will visit solutions that do not obey the lower-bound constraints on the quantities that need to be produced. In order to balance the conflicting demands of maintaining feasibility while ensuring sufficient flexibility to seek out good solutions a number of approaches are possible. These include

(1) Restricting the definition of the search space, S, to the set of feasible solutions by rejecting all infeasible moves.
(2) Restricting S the set of feasible solutions by repairing infeasible solutions.
(3) Redefining S to include infeasible solutions but penalizing these in the evaluation function $f(s)$.

Each of these has its own advantages and drawbacks. Option 1 is straightforward to implement, but computational effort will be wasted when evaluating infeasible moves and, more importantly, if the density of such moves is high it may have an adverse effect on the flexibility of the search in seeking out good-quality solutions. Option 2 can be effective if a simple repair operator is available. In our case it is often possible to repair infeasible layouts by replacing larger circles in surplus with smaller ones for which there is a deficit. If the smallest feasible circle is removed in each swap then the repair will be achieved at minimal cost. However, there are a number of potential problems. Firstly, not all layouts will be repairable. Secondly, there will be a choice of which circles to replace, with different choices at intermediate points of the search resulting in different outcomes. Thirdly, the repair will involve changing partial rows in the hexagonal lattice, thus making the definition of neighbourhood moves in subsequent iterations more complex. Option 3 is widely used in a number of practical situations. However, as illustrated by a number of studies (Wright, 1991; Abramson and Dang, 1993) it can be very difficult to set the penalties for binding constraints. This is particularly problematic when the search mechanism disallows uphill moves. Too high a penalty will effectively exclude all infeasible solutions from the solution space, while too low a penalty will tend to cause the search to converge towards infeasible solutions. In our case we can overcome this problem using a penalty based on the repair operator. Rather than physically replacing an infeasible layout with a repaired one we calculate the cost of the repaired layout and use that as the evaluation function. Where a layout cannot be repaired (eg if the largest circle is in deficit) then a high penalty is allocated.

In view of the above discussion three approaches to maintaining feasibility were selected for further experimentation and comparison. These are

- Allow infeasible solutions in the solution space and repair the final solution only.
- Restrict the solution space to the set of feasible solutions by disallowing infeasible moves.
- Relax the definition of S to allow infeasible solutions in the search space but penalise them with a penalty based on the cost of repair. The final solution is then repaired physically.

Experiments with each of these options are reported in the next section.

Experiments

A number of variants of a local search algorithm based on the initialization approach and improvement strategies outlined above were compared empirically. For this purpose, nine data sets were chosen from the company's archives. These data sets were used to identify the best variant, which was then subjected to further testing on 100 problem instances. Of the initial nine problems, the smallest had a requirement of 75 circles and the largest over 1500. The number of sheets used to accommodate the orders varied from 2 to 20. The objective of the first set of experiments was to determine the usefulness of the different row operations within a random descent framework (ie sampling from N until a member of N^+ is identified). In these experiments, the lower-bound constraints on the quantities were relaxed. Each of the nine data sets was run using 10 random number seeds, and continuing until 70 moves had passed without an improvement. (Experiments with up to 100 moves without improvement suggested that this was an appropriate stopping point.) Six different combinations of row moves were considered. These were

(1) R1 (swap top rows of two sheets),
(2) R2 (swap top two rows of two sheets),
(3) R1 and R2,
(4) R3 (change one or two rows on a single sheet),
(5) R1, R2 and R3,
(6) R1, R2, R3, R4 and R5 (ie all row operations).

In each case a move type is selected randomly, and within each move all choices (ie sheets, diameters, etc) are again selected randomly from those available. After a row move has been performed the row addition and completion phases are executed as outlined in the previous section. The first six rows of Table 1 summarize the results. Although there are some anomalies, the results suggest that supersets of row moves tend to give better results than their subsets, and in all cases variant 6 (ie using all row operations) performs better than any of the other five combinations. This variant was therefore used for all subsequent developments and experiments.

Observation of the progress of the algorithm also showed that the size of the improvement varied considerably from

Table 1 Mean packing density over 10 runs for different neighbourhoods and search strategies

Data set	1	2	3	4	5	6	7	8	9
Sheets	3	16	5	2	5	5	20	2	12
Solutions for search without lower-bound restrictions									
R1	81.70	82.21	81.60	83.10	81.47	83.07	81.10	80.72	80.36
R2	81.70	82.15	81.84	82.90	81.41	83.06	81.04	80.72	80.27
R1&R2	81.70	82.28	81.89	83.10	81.45	83.09	81.04	80.72	80.31
R3	81.70	82.44	81.99	82.90	81.82	83.11	81.46	80.90	80.38
R1,R2&R3	81.70	82.40	82.22	83.10	81.72	83.09	81.38	80.84	80.38
All	81.74	82.77	82.42	83.10	81.97	83.22	81.52	81.88	80.71
PSD—All	81.70	82.64	82.55	83.10	82.12	83.29	81.67	82.32	80.90
Solutions obeying lower bound obtained using penalty approach									
Penalty	81.29	82.23	82.05	82.51	81.70	82.86	81.09	81.86	80.41

move to move, suggesting that spending time looking for better neighbours, that is, a steeper descent strategy could be beneficial. Given the size and complexity of the neighbourhoods a true steepest descent approach, in which all neighbours are evaluated at each iteration, will be computationally expensive. Instead, we tried a form of probabilistic steepest descent (PSD) in which at each iteration 10 elements of N are considered. If the best belongs to N^+, then it is selected. Otherwise, a further 10 moves are sampled, with this process continuing until a move is selected or N is exhausted. The results are shown in row PSD, in the table. While this does not always show an improvement the results are better for six out of nine instances and equal on a further one. What the table does not show is that in all cases except for data set 1, PSD always found the best solution at least once. Therefore, probabilistic steepest descent, based on 10 moves per iteration, and using all five row operations was used exclusively for the remaining experiments.

The remaining experiments are concerned with meeting the lower bounds on the number of circles of each diameter. As expected using the unrestricted neighbourhood, without any penalty or repair, frequently resulted in solutions that violated the lower-bound constraints. The frequency of feasible solutions was greater in the less-effective variants using limited neighbourhoods, especially R1 and R2. Again, this is as expected as these moves do not make such large changes in the numbers of circles of each type. Where solutions were infeasible, it was usually possible to repair them using the repair operator outlined in the previous section. The densities achieved after repairing the final solution were typically 1.5–2% lower than those shown in Table 1. As stated above two other approaches were tested. Experiments with a restricted solution space resulted in a drop in packing densities over those in Table 1 of around 3%, suggesting that the search does require the added flexibility of visiting infeasible solutions. The results using the repair-based penalty function proved to be the most successful. All runs resulted in final solutions that could be repaired, and packing densities dropped by only between 0.5 and 0.8% . These results are given in the final row (labelled Penalty) of Table 1.

As illustrated by the final row of Table 1, local search incorporating all five row operations using probabilistic steepest descent based on sampling 10 neighbours per iteration and penalising violations of the lower-bound constraints using a penalty based on the repair cost, is able to produce solutions with packing densities that are close to that of a human expert.

The algorithm is coded in Visual C++ and typical run times on an Intel Pentium 4 2.2 GHz processor vary between 2 and 30 s depending on the size of the problem, whereas typical man-hours required to produce manual cutting plans are around 10.5 h for a problem of 20 sheets (assuming approximately seven of these would need to be produced from scratch). The layouts produced by the local search also conform to the required format. Figure 4 shows the initial and final solutions for one of the runs on data set 6. The density of the initial solution is 76.6% while the density of the final solution shown is 81.7%. This solution violates the lower-bound constraint for the third largest circle for which there is a shortfall of five units. This can be repaired by replacing five circles of the next diameter (which is in surplus). The density of the repaired layout is 81.4%.

In order to confirm this performance a further 100 problem instances were solved using this variant of the algorithm. The mean sheet utilization using the adjusted dimensions was 81.76%, which translates to 67.14% when the waste due to the thickness of the cutter, the space between disks and the unuseable area around the border of the sheet is taken into account. This falls within the range of 32–34% waste currently generated by the company. The amount produced for inventory also falls within the guidelines with a total of 17% of production being surplus to that ordered. For eight of the 100 orders, the surplus exceeded 30% but these were all orders requiring five or less sheets, and the surplus is therefore in line with current performance.

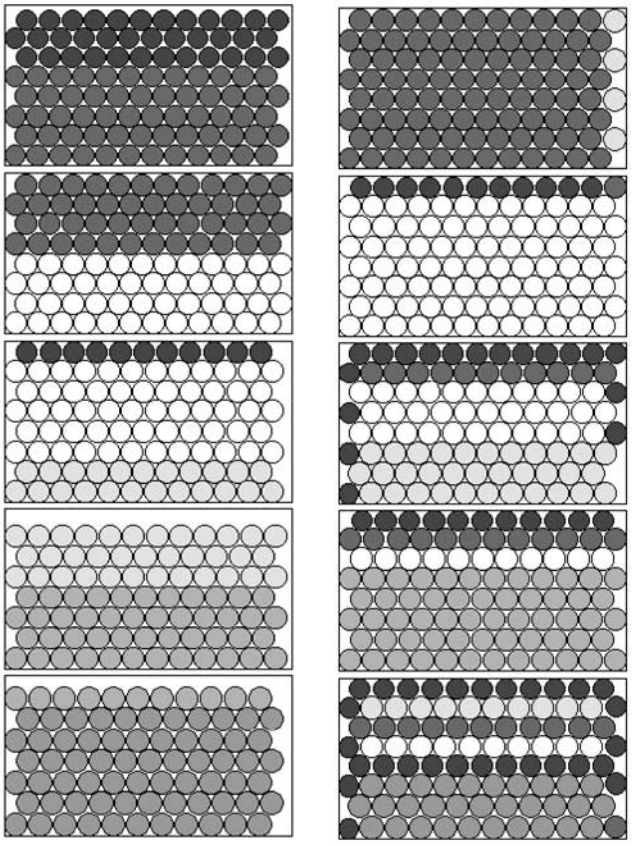


Figure 4 The initial solution (left) and final solution (right) for a typical data set.

A closer examination of the remainder of the sample showed that occasionally the surplus was significantly more than that produced manually. Not surprisingly this situation arose when relatively small numbers of demanded circles were placed on the final sheet in the initialization process. The manual workers would sometimes try and swap these circles for surplus circles on the previous sheets, thus reducing the number of sheets used by one. It would be possible for us to address such a situation in a similar way and to discard the last sheet, penalizing the shortfall in the initial solution appropriately. However, given the average performance of our heuristic the client deemed that this would not be necessary and we have not implemented these changes to date.

Conclusions and further developments

This paper has focussed on the design and testing of a circle packing algorithm for a manufacturer of sprockets. Practical considerations mean that the problem is somewhat different to those previously tackled in the literature and necessitated the development of a customized solution. Local search, based on an initial solution of the required form and a series of neighbourhood moves designed to perturb the layout without destroying the underlying structure, proved to be an effective way of meeting the requirement to produce layouts that are similar in structure to those currently produced by a manual solution method. An approach that allowed solutions that violated the lower-bound constraints, but penalized these with a penalty based on the repair costs, was shown to perform better than limiting the search to those solutions that obey all the constraints. The resulting layouts closely match those produced by human experts in terms of sheet utilization. Indeed the densities achieved suggest that this could be a competitive approach for problems with similar distributions of circle sizes even if there is no constraint on the format of the layouts. (For example, the density of layouts produced in (Huang *et al*, 2005), which often involve much smaller circles than those dealt with here, lie in the range of 80–85%.)

The solutions produced are certainly good enough to meet our objectives (ie a back-up system to complement the current manual approach). It was however noticeable that the algorithm sometimes produced more circles for inventory than the manual solutions, although the quantities involved did not suggest that this would be a major problem. A possible solution to this was suggested in the previous section and a more general investigation into the relationship between surplus production and material utilization could make an interesting study for the future. Given the very low computation times required by the algorithm (20–30 s as opposed to 10.5 h for a manual solution), it may also be possible to make further small improvements to the packing densities by using the local search framework as the basis for other neighbourhood-based search methods such as simulated annealing, tabu search or GRASP, or to incorporate further intelligence into the choices made at each iteration rather than relying on entirely random choices at each stage.

References

Abramson D and Dang H (1993). School timetables: a case study using simulated annealing. In: Vidal RVV (ed). *Applied Simulated Annealing*. *LNEMS 396*. Springer Verlag, Berlin, pp 104–124.

Beasley JE (1985). An exact two-dimensional non-guillotine cutting tree search procedure. *Opns Res* **33**: 49–64.

Bennell J and Dowsland K (2001). Hybridising tabu search with optimisation techniques for irregular stock-cutting. *Mngt Sci* **47**: 1160–1172.

Birgin EG, Martinez JM and Ronconi DP (2005). Optimizing the packing of cylinders into a rectangular container: a nonlinear approach. *Eur J Opl Res* **160**: 19–33.

Boll DW, Donovan J, Graham RL and Lubachevsky BD (2000). Improving dense packings of equal disks in a square. *Electronic J Combin* **7**: R46.

Burke EK, Hellier RSR, Kendall G and Whitwell G (forthcoming). A new bottom-left algorithm for the two-dimensional irregular packing problem. *Opns Res* (accepted for publication).

Burke EK, Kendall G and Whitwell G (2004). A new placement heuristic for the orthogonal stock cutting problem. *Opns Res* **52**: 655–671.

Chen DZ, Hu X, Huang Y and Li Y (2001). Algorithms for congruent sphere packing and applications. In: Souvaine D (ed). *Proceedings of the Seventeenth Annual Symposium on Computational Geometry, Medford, Massachusetts, United States*. ACM Press, New York, pp 212–221.

Collins CR and Stephenson K (2003). A circle packing algorithm. *Comput Geometry: Theory Appl* **25**: 233–256.

Conway JH and Sloane NJA (1992). *Sphere Packings, Lattices, and Groups*, 2nd edn. Springer-Verlag: New York.

Correia MH, Oliveira JF and Ferreira JS (2000). Cylinder packing by simulated annealing. *Pesquisa Operacional* **20**: 269–286.

Cui Y (2005). A cutting stock problem and its solution in the manufacturing industry of large electric generators. *Comput Opns Res* **32**: 1709–1721.

Dowsland KA (1991). Optimising the palletisation of cylinders in cases. *OR Spektrum* **13**: 204–212.

Dowsland KA, Bennell J and Dowsland W (1998). Jostling for position—local improvement for irregular cutting patterns. *J Opl Res Soc* **49**: 647–658.

Dowsland KA, Herbert E, Kendall G and Burke EK (2006). Using tree search bounds to enhance a genetic algorithm approach to two rectangle packing problems. *Eur J Opl Res* **168**: 390–402.

Dowsland KA, Vaid S and Dowsland W (2002). An algorithm for polygon placement using a bottom-left strategy. *Eur J Opl Res* **141**: 371–381.

Dowsland KA and Dowsland WB (1992). Packing problems. *Eur J Opl Res* **56**: 2–14.

George JA, George JM and Lamar BW (1995). Packing different-sized circles into a rectangular container. *Eur J Opl Res* **84**: 693–712.

Gilmore PC and Gomory RE (1961). A linear programming approach to the cutting-stock problem. *Opns Res* **9**: 849–859.

Glover F and Laguna M (1997). *Tabu Search*. Kluwer Academic Publishers: Boston, USA.

Glover FW and Kochenberger GA (2003). *Handbook of Metaheuristics*. Kluwer Academic Publishers: Boston, USA.

Graham RL and Lubachevsky BD (1996). Repeated patterns of dense packings of equal disks in a square. *Electronic J Combin* **3**: R16.

Huang WQ, Li Y and Xu RC (2001). Local search based on a physical model for solving a circle packing problem. In: de Sousa JP (ed). *Proceedings of the fourth Metaheuristics International Conference 2001 (MIC'2001), July 2001*. Kluwer, Boston, pp 455–459.

Huang WQ, Li Y, Akeb H and Li CM (2005). Greedy algorithms for packing unequal circles into a rectangular container. *J Opl Res Soc* **56**: 539–548.

Huang WQ, Li Y, Jurkowiak B, Li CM and Xu RC (2003). A two-level search strategy for packing unequal circles into a circle container. In: Rossi F (ed). *Proceedings of the Ninth International Conference on Principles and Practice of Constraint Programming (CP'03), Kinsale, Ireland. Lecture Notes in Computer Science 2833*. Springer Verlag, Berlin, pp 868–872.

Kantorovitch LV (1939, 1960). Mathematical methods of organising and planning production. *Mngt Sci* **6**: 366–422.

Koebe P (1936). Kontaktprobleme der Konformen Addildung, Abh. Sächs. Akad. Wiss. Leipzig. *Math. Natur. Kl.* **88**: 141–164.

Kravitz S (1967). Packing cylinders into cylindrical containers. *Math Magazine* **40**: 65–70.

Lodi A, Martello S and Monaci M (2002). Two-dimensional packing problems: a survey. *Eur J Opl Res* **141**: 241–252.

Nurmela KJ and Östergård PRJ (1997). Packing up to 50 Equal Circles in a square. *Discrete Comput Geometry* **18**: 111–120.

Reeves C (1993). *Modern Heuristic Techniques for Combinatorial Problems*. Blackwell Scientific Press: Oxford, UK.

Steinhaus H (1999). *Mathematical Snapshots*, 3rd edn. Dover: New York, p 202.

Stephenson K (2003). Circle packing: a mathematical tale. *Notices—Am Math Soc* **50**: 1376–1388.

Stoyan YG and Yas'kov G (2004). A mathematical model and a solution method for the problem of placing various-sized circles into a strip. *Eur J Opl Res* **156**: 590–600.

Sweeney E and Paternoster RE (1992). Cutting and packing problems: a categorized, application-orientated research bibliography. *J Opl Res Soc* **43**: 691–706.

Wang H, Huang W, Zhang Q and Xu D (2002). An improved algorithm for the packing of unequal circles within a larger containing circle. *Eur J Opl Res* **141**: 440–453.

Wells D (1991). *The Penguin Dictionary of Curious and Interesting Geometry*. Penguin: London, pp 30–31.

Wildemuth BM (2004). The effects of domain knowledge on search tactic formulation. *J Am Soc Inform Sci Technol* **55**: 246–258.

Williams R (1979). *The Geometrical Foundation of Natural Structure: A Source Book of Design*. Dover Publications: New York, pp 34–47.

Wright MB (1991). Scheduling English cricket umpires. *J Opl Res Soc* **42**: 447–452.

Received March 2005;
accepted December 2005 after two revisions