Hybridising Tabu Search with Optimisation Techniques for Irregular Stock Cutting
Author(s): Julia A. Bennell and Kathryn A. Dowsland
Source: *Management Science*, Vol. 47, No. 8 (Aug., 2001), pp. 1160-1172
Published by: INFORMS
Stable URL: http://www.jstor.org/stable/822601
Accessed: 05/03/2010 04:16

Hybridising Tabu Search with Optimisation Techniques for Irregular Stock Cutting

Julia A. Bennell • Kathryn A. Dowsland
Department of Management, University of Southampton, Highfield, Southampton SO17 1BJ, United Kingdom
European Business Management School, University of Wales Swansea, Singleton Park, Swansea SA2 8PP, United Kingdom
j.a.bennell@soton.ac.uk • k.a.dowsland@swansea.ac.uk

Sequential meta-heuristic implementations for the irregular stock-cutting problem have highlighted a number of common problems. The literature suggests a consensus that it is more efficient to allow configurations with overlapping pieces in the solution space and to penalise these in the evaluation function. However, depending on the severity of the penalty this relaxation results in a tendency to converge toward infeasible solutions or to seek out feasible solutions at the expense of overall quality. A further problem is encountered in defining a neighbourhood search strategy that can deal with the infinite solution space inherent in the irregular stock-cutting problem. The implementation in this paper adopts a hybrid tabu search approach that incorporates two very different optimisation routines that utilise alternative neighbourhoods to address the described problems.
(*Irregular Stock Cutting; Packing Problems; Tabu Search; Linear Programming*)

1. Introduction

Irregular stock-cutting problems arise whenever a set of irregular pieces are to be cut from a stock sheet of material. This paper is concerned with a common variant of the problem, that of minimising the length required to fit a given set of polygonal pieces into a rectangular sheet of known width. In addition to the complex geometry involved, the problem inherits all the combinatorial features encountered in its rectangular counterpart. It is therefore one of the more challenging two-dimensional problems in the field of cutting and packing. Different mixes of piece numbers, sizes, and shapes yield problem instances with different characteristics. Thus, in spite of a number of successful approaches to problems that focus on a specific data type, the development of a generic approach remains a challenging problem.

As with many other difficult optimisation problems, there has been considerable interest in the potential of metaheuristics such as genetic algorithms, simulated annealing, and tabu search. This paper is concerned with a tabu search approach to the problem. The approach illustrates how the strategic use of optimisation routines, which utilise alternative neighbourhoods designed to exploit the geometry of the problem, can significantly enhance the effectiveness of the search.

We start by summarising previous work using simulated annealing (SA) and tabu search (TS) for irregular stock cutting, and highlight a major problem inherent in many of these approaches. We then introduce a different local search, based on linear programming, that has proved successful for certain classes of irregular cutting problem, and show how this can be used to exploit the shortcomings observed in the

0025-1909/01/4708/1160$5.00
1526-5501 electronic ISSN

TS and SA approaches. The remainder of the paper deals with our own developments, starting with a description of our underlying search strategy, using a variant of tabu search. We then discuss two different ways of incorporating optimisation routines. Results of experiments with four variants are used to compare the approaches.

2. Lessons from Previous Work

A survey of SA and TS approaches to the problem reveals a variety of definitions of the solution space, neighbourhood structure, and evaluation function. TS appears to have been less popular in the solution of packing problems. This may be because of the infinite neighbourhoods implied by a continuous stock sheet, which favours the use of SA. However, there are a number of implementations that have addressed this problem in different ways. Blazewicz et al. (1993) deal with the objective of minimising the length of the stock sheet. They overcome the problem of searching large continuous neighbourhoods by exploiting the fact that only a limited set of moves will reduce the rightmost extent of the layout. If there are no such moves available, then further criteria are applied to limit the number of moves considered. Oliveira and Ferriera (1993) define solutions as permutations of the pieces. Each permutation is evaluated by packing the pieces in the given order using a fixed placement policy. This overcomes the problem of infinite neighbourhoods produced by moves within the stock sheet but has the disadvantage that much of the layout must be repacked at each iteration. Both these local search frameworks are designed to overcome the problems of converging to infeasible solutions by excluding such solutions from the search space.

Among the SA implementations one model stands out as being the simplest to implement and the most popular in practice. Solutions are defined as the set of coordinate positions of the pieces on the stock sheet with overlap between pieces being allowed but penalised in some way in the evaluation function (for example Dagli and Hajakbari 1990, Jain et al. 1992). In some implementations the length or area required to fit all the pieces is combined with an appropriate measure of overlap to form a single cost function (for example, Heckmann and Lengauer 1995). Depending on the relative weightings given to the different parts of the cost function, the result is a tendency either to converge toward infeasible solutions or to seek out feasible solutions at the expense of overall quality. In other implementations, the length of the sheet is fixed, and the search attempts to reduce the overlap to zero before proceeding with a reduced length (see, for example, Bennell and Dowsland 1999 and Oliveira and Ferreira 1993). Although this approach avoids the need to balance the relative weightings of different cost elements, it is limited by the need to find feasible solutions at each length. Both approaches tend to give rise to landscapes with many local optima corresponding to layouts with some overlap, often with sizeable gaps elsewhere.

A further problem encountered with many of the local search implementations is the way in which they handle the complex geometry, which results in an infinite solution space consisting of all possible positions of the pieces on the stock sheet. Many employ a discrete grid of placement positions. This has the advantage of providing a convenient basis for calculating overlap. However, a dense grid results in very large neighbourhoods, suggesting that large numbers of iterations will be required to search the space effectively. Conversely, if a coarser grid is used, then solution quality may be compromised.

Other researchers have had considerable success in producing good layouts using compaction routines based on optimisation techniques such as linear programming (Li and Milenkovic 1995, Stoyan et al. 1993). Such approaches need a good starting solution, as they cannot make significant changes in the relative positions of the pieces; for example, moving a piece positioned at one end of the layout to a position near the other end. Appropriate starting solutions can sometimes be achieved by exploiting problem characteristics. Li and Milenkovic (1995) restrict their investigation to trouser patterns in the garment industry. Initial solutions are generated by one of two approaches: first, by a heuristic that is designed to mimic the strategies employed by human experts, and second, through selecting a problem with similar data for which an efficient solution has been found, and adapting it by replacing pieces in the solution with those required for the current problem. They show how the same linear programming (LP) approach as used for compaction can be used to separate pieces to remove small amounts of overlap from a layout. Stoyan et al. (1993) use a similar technique of defining the feasible movement of pieces within an initial layout. A range of starting points are enumerated, each converging to different local optima.

In summary, the problems associated with the local search approaches include difficulty in handling infinite neighbourhoods and the likelihood of converging on infeasible solutions. The compaction approaches are not only hindered by the infinite nature of the solution space but cannot perform the wider search that is possible with SA and TS. Therefore, they are heavily reliant on an appropriate starting solution. This limitation is shown in Bennell (1998), who finds that the compaction of randomly generated solutions cannot compete with local search. Thus, it seems logical to combine the SA/TS and LP approaches. The former are able to move individual pieces anywhere on the stock sheet. They can, therefore, be used to find good starting positions for LP-based separation and compaction routines. The latter can take advantage of SA/TS local optima with gaps and small amounts of overlap, and translate them into good, feasible solutions. The combination of these approaches allows our initial, local search implementation to use a grid of placement positions and rely on the compaction routines to allow placements at intermediate points. As an alternative, we divide an infinite neighbourhood into a finite set of partitions. Although each partition includes an infinite number of neighbourhood solutions, the relationship between the member solutions allows an optimisation routine to determine the best solution in each.

3. The Choice of Tabu Thresholding

The idea of combining a neighbourhood search method such as SA or TS with a compaction routine is straightforward, but the implementation details raise a number of strategic questions. These involve the way in which the basic search is organised and at what points the compaction routine should be used. For example, we can develop a method that is essentially a compaction approach, using a neighbourhood search to identify good starting solutions. Conversely, we can develop a neighbourhood search and use compaction routines to improve local optima.

Here we take the latter view. Nevertheless, we need to ensure that the search will provide a variety of different local optima for the compaction routines to improve. We, therefore, ruled out simulated annealing as the underlying algorithmic, as it tends only to visit high-quality local optima toward the end of the search. The more aggressive nature of tabu search means that it is likely to visit many more candidates for compaction. Although it is possible to control the balance between aggression and diversification via the tabu list and other diversification strategies, more direct control can be achieved from a variant of tabu search known as tabu thresholding (TT) (Glover 1992). The main advantage of tabu thresholding is that it operates in two distinct phases—an improving phase to seek out local optima and a mixed phase to encourage diversification into new areas. The amount of time spent in the mixed phase is controlled directly by a user-defined parameter.

Our implementation was, therefore, based on tabu thresholding. The basic algorithm searches the solution space by moving one piece to a different position in the stock sheet and is described in Bennell and Dowsland (1999). Solutions containing overlap are permitted and penalised in the cost function. To simplify the calculation of overlap and provide a measure that is meaningful in terms of the search objectives, overlap is approximated as the sum of the horizontal extents of the overlap between each pair of pieces. The basic algorithm is enhanced by allowing some changes in the mixed phase to strengthen the diversification. A form of probabilistic best is introduced that allows the search to ignore the absolute best and, therefore, break cycles of moves. The result is an increase in the threshold of the search that encourages the exploration of new areas of the solution space. Another improvement is to discriminate between moves within the set of moves that create overlap with other pieces. A move that creates overlap with a cluster of small pieces generates the conditions in which the cluster can be broken up by subsequent moves. This is more desirable than moves that create overlap with a large piece that is unlikely to be able to be moved to a feasible position. Full details of the structure and development of the basic and enhanced algorithm can be found in Bennell and Dowsland (1999).

The investigation in this paper builds on the work in Bennell and Dowsland (1999). Two definitions of solution space are investigated. As a result of that investigation, only pieces that contribute to the evaluation function may be moved. However here, the length required to pack the pieces is not fixed but defined by the rightmost extent of the rightmost pieces. Thus, the algorithm can simultaneously reduce overlap and length by combining these to form the evaluation function. As a result of the equally weighted cost elements, there will exist a large number of suboptimal zero-overlap solutions that may be found by the search. The compaction phase will also generate zero-overlap solutions at intermediate points throughout the search. Diversification away from these solutions is achieved by introducing the mixed phase.

4. Incorporating the Optimisation Routines

Two features of the TT algorithm defined above hinder the search for good feasible solutions. Both concern the definition of the neighbourhood moves. The first difficulty arises when near-feasible solutions are densely packed. Reaching feasibility is often a matter of a number of pieces simultaneously shifting closer together. However, the TT search framework only allows the unilateral movement of a piece that is in an overlapping position. Thus, our first optimisation routine permits simultaneous movement of all pieces to compact and separate overlapping pieces to achieve feasibility. Second, without further modification to the neighbourhood, moving a piece within the layout implies an infinite solution space and infinite neighbourhood. Although there are a number of ways of approximating the solution space, for example, using a grid, this may mean that optimal solutions are ignored. Here, we describe an approach that is able to partition the neighbourhood into sets of more promising solutions and determine the best in each set through an optimisation routine.

Both optimisation routines rely on the use of a geometric concept known as the nofit polygon (NFP). In the context of cutting and packing problems the NFP is used as an efficient means of detecting overlap between polygonal pieces. In this paper, the properties of the NFP have been exploited not only to detect overlap, but also to select the optimal move from an infinite neighbourhood and to facilitate the compaction of the layout. The nofit polygon is found by combining the two-component polygons in such a way that overlap can be detected through a simple point-inclusion test. These polygons are calculated using Minkowski operators and deal with concavities in the pieces via an approach based on a method suggested by Ghosh (1991). Details of this approach can be found in Bennell et al. (2001). The compaction routines use the edges of the nofit polygons to define the constraints in the linear programming formulation, while the neighbourhood partitioning approach uses the edges to define the partitions.

4.1. Compaction/Separation at Local Optima

The basic tabu thresholding implementation is designed to visit a variety of local optima with some overlap. Whenever this overlap falls below a given threshold, the local optima may be made feasible and further improved through the implementation of a compaction-separation phase. This phase will be referred to as the optimisation phase for the remainder of the paper. Because the optimisation phase does not rely on a finite neighbourhood, a new neighbourhood structure is defined. As a result, the heuristic rules that trapped the search no longer apply, and the previously locally optimal solution can be further improved.

The optimisation phase is modeled and solved as a linear programme. The aim is to minimise the length of the stock sheet in which the pieces are packed, while removing any overlap between pieces. Therefore, the objective function penalises the length of the layout, and the constraints prevent or remove overlap between any pair of pieces. Although the improvement in a solution is measured by the horizontal distance moved, pieces are permitted to move linearly in any direction to achieve the overall minimum length. To maintain the integrity of the linear program, all positions on the trajectory of the pieces must be feasible. Thus, pieces are not allowed to pass through each other, which prevents large changes in the relative positions of the pieces. Although this phase cannot make significant changes to the local features of the solution neighbourhood, it can result in improvements that would either not be possible using the original, local-search framework or would require significant diversification.

Our LP formulation is based on that of Li and Milenkovic (1995). It uses the same approach to derive the overlap constraints, but it differs in other details. First, Li and Milenkovic's objective is to pack all the pieces as densely as possible with respect to the bottom left corner of the stock sheet. Our objective is stated explicitly as that of minimising the length required to pack all the pieces. Second, they define the positions of their pieces according to their centres, whereas we define the reference point to be the bottom left corner of the enclosing rectangle of the piece, i.e., at x_i, y_i, where x_i and y_i are the minimum x and y coordinates on piece i.

Letting (x_i, y_i) denote the position of the reference point of piece i, X and Y the length and width of the stock sheet, and w_i and h_i the width and height of the enclosing rectangle of piece i, the problem is formulated as follows:

$$\begin{aligned}
&\text{Minimise } X \\
&\text{s.t.} \quad X \geq x_i + w_i && \forall i, && (1)\\
&\qquad Y \geq y_i + h_i && \forall i, && (2)\\
&\qquad c_{i,j,k} \geq a_{i,j,k}(x_j - x_i) \\
&\qquad\qquad + b_{i,j,k}(y_j - y_i) && \forall i,j,k, && (3)\\
&\qquad |\delta x_i| \leq B && \forall i, && (4)\\
&\qquad |\delta y_i| \leq B && \forall i, && (5)\\
&\qquad x_i, y_i \geq 0 \quad \forall i.
\end{aligned}$$

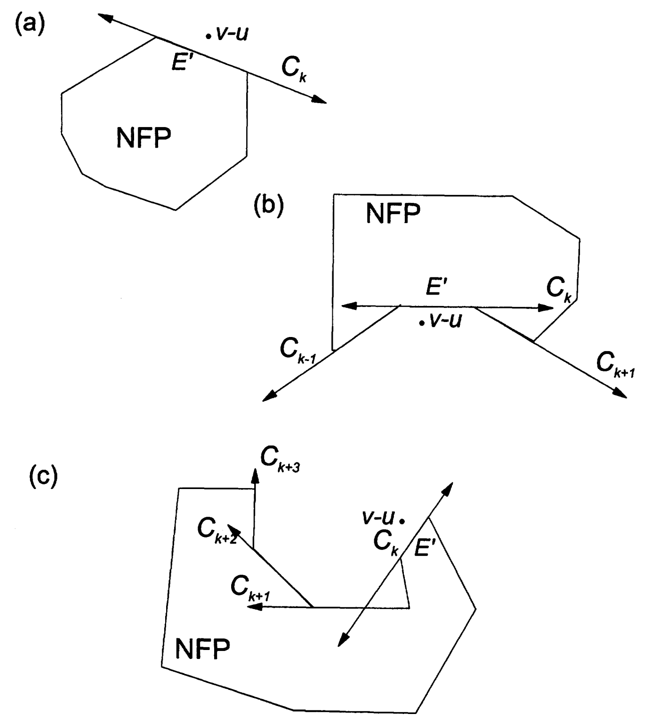
The objective is that of minimising the length X and constraint set (1) is included to ensure that X represents the length of the layout.

To ensure the LP solution is feasible a number of additional constraints are required. First, pieces must remain within the boundaries of the stock sheet. The top boundary is dealt with by constraint set (2). The bottom and left-hand boundary of the stock sheet will not be violated because x_i and y_i are constrained to be positive. Finally, because the objective is to minimise length, the right-hand end is dealt with implicitly.

The second set of feasibility constraints deals with pieces overlapping each other and is represented by constraint set (3). Its formulation is obtained directly from the following property of the NFP. If piece i is placed at point u and piece j is placed at point v, the problem of determining whether i and j intersect is reduced to the point-inclusion test of determining whether the point defined by $v-u$ lies inside the NFP of i and j (NFP_{ij}). Therefore, to ensure that piece i and piece j do not overlap, the resultant vectors of their positions denoted by $v-u$ must lie outside NFP_{ij}. This property can be used in the LP formulation to remove or prevent overlap between pieces. However, the relationship between $v-u$ and the NFP cannot be modelled directly as the space is not convex. Therefore, the formulation overconstrains the space and selects a convex subregion. This is achieved by identifying one or a series of the NFP edges that may be translated into linear constraints that form a convex feasible region.

For pieces i and j the edge or edges that derive these constraints are identified as follows. First, select the edge (E') of the NFP_{ij} that is closest to the point $v-u$. $v-u$ is found by subtracting the coordinate points as follows: $(x_j - x_i, y_j - y_i)$. The line that corresponds to the line segment E' becomes the first constraint (C_k). If the entire set of points that make up NFP_{ij} lie on one side of C_k, for example, if NFP_{ij} is convex, then no other constraints are required. An example of this is illustrated in Figure 1a. Otherwise, the next edge that forms a constraint is identified by travelling along C_k starting from E'. If C_k crosses another boundary edge into the interior of NFP_{ij}, then the next constraint (C_{k+1}) is defined by that edge. At this point the direction of travel follows C_{k+1} retaining the exterior of the NFP on the same side as C_k and reducing the feasible region. If this results in entry into the NFP, then a new edge is identified and the process continues. This is repeated in both directions, i.e., from E' in a clockwise direction and in a counterclockwise direction.

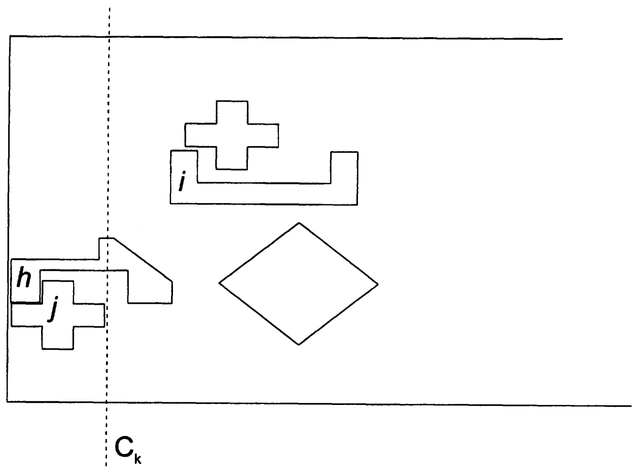
Figure 1 **The First Constraint (C_k) Is Derived from Edge E'. Subsequent Constraints Are Found by Travelling Along C_k Identifying Intersections with the Boundary of the NFP**



Figures 1b and 1c illustrate two examples of this process. In the case of 1b, the nearest edge that corresponds to the constraint C_k crosses the boundary of the NFP in both directions at the vertices of E'. Travelling clockwise the connecting edge derives the constraint denoted as C_{k+1}, and travelling counterclockwise C_{k-1} is derived in a similar way. In Figure 1c, travelling in the counterclockwise direction, three further edge constraints are derived when travelling in the counterclockwise direction. Note that for this example constraints are derived from nonconsecutive edges, illustrating the need to follow the constraint beyond a convex vertex.

If no edges are encountered in either direction, then no further constraints are required to form a valid, convex feasible region. Thus, for each pair of pieces i, j there are k edges, where $k \geq 1$, that correspond to constraints. These are formulated for the LP as follows: If $ax + by = c$ denotes the equation of a line, then the constraints are defined by replacing (x, y)

Figure 2 **Constraint C_k Between Pieces i and j Is Redundant but Still Restricts the Movement of Piece i Towards the Beginning of the Stock Sheet**



with $(x_j - x_i, y_j - y_i)$ and setting an inequality such that $v - u$ is on the nonoverlapping side of the line.

Constraints (4) bound the movement of each piece in the x and y directions. As a result of enforcing bounds, the LP routine is repeated several times, allowing the pieces to move in a sequence of short hops. Bounds are set for the following reasons. Theoretically, we need to define constraints for each pair of pieces. However, pieces that cannot touch each other through a feasible path also cannot overlap. Therefore, setting an upper bound on movement reduces the number of pairs of pieces that need to be considered in constraint set (3). Moreover, constraint set (3) can restrict the movement of pieces unnecessarily. For example, Figure 2 illustrates an edge constraint, C_k, between piece i and j. Although, C_k is a valid constraint, it is redundant as piece h is directly between i and j. However, C_k also prevents piece i from moving horizontally toward the beginning of the stock sheet. This problem is minimised only if pairs of pieces that can meet within the bound are considered in (3). Finally, as the edges of NFP_{ij} selected to act as constraints will depend on the relative positions of i and j, the constraint set will change after each call to the LP. The net result is that some pieces will move in a piecewise linear fashion, thus allowing them to travel around corners and access areas that would not be achievable in a single run.

In theory this method will be valid for removing overlap as well as for compacting layouts without overlap. However, Constraints (3) require overlap to be removed in one iteration. This may not be possible, either because greater movement than the bound of one or more pieces may be required or pieces may need to take a nonlinear path. To allow these situations to occur, the edge constraints on the overlapping pieces must be relaxed. One approach would be to penalise the amount of overlap in the cost function. However, this would result in complex calculations of overlap encoded into the LP and may lead to a trade-off in costs. Instead, a more direct approach is taken. If the LP fails because overlap is not removed, then the violated edge constraints are redefined to be closer to the point $v - u$, i.e., the overlap constraints causing problems are relaxed to enforce a reduction in overlap but not its total elimination. Thus, a successful solution to the new formulation would provide an improvement in overlap and allow the next iteration the opportunity to remove the remainder of the overlap. To minimise the need for this relaxation and to reduce the likelihood of the LP failing, an upper limit on the amount of overlap contained in the solution must hold before the optimisation phase is initialised. As a result, the described relaxation quickly reduces the amount of overlap; therefore, repetition of failure and relaxation of a constraint are only allowed three times. Inevitably there will be situations when the LP will fail to find a feasible solution, and in such cases the search algorithm simply continues from the minimum overlap solution found during the optimisation phase. If the optimisation phase does not successfully solve any LP formulation, then the search returns to the solution that initialised the optimisation phase.

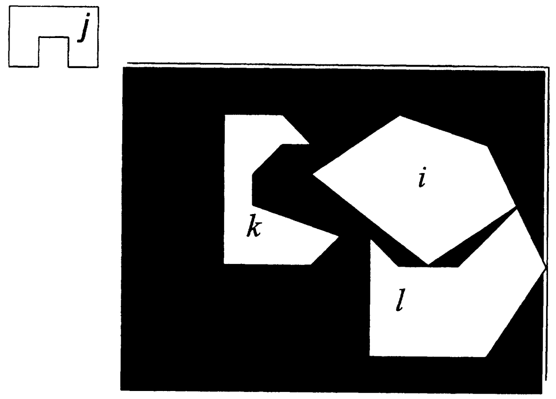
4.2. An Improved Approximation of the Solution Space

Our second modification is designed to eliminate the need for grid positions and to concentrate the search in areas that are likely to yield dense layouts. This is achieved by defining the neighbourhoods in such a way as to take account of the features of an optimal solution. In an optimal layout each piece will touch at least one other piece or an edge of the stock sheet. These touching positions are defined by the boundaries of the nofit polygons. For example, if we choose to place piece j next to piece i, then we can determine the position of j by choosing a point on NFP_{ij}. Although these boundaries are infinite, the relationship between the boundary of one NFP and the amount of overlap incurred from another NFP can be used to reduce the neighbourhood to a finite set of points.

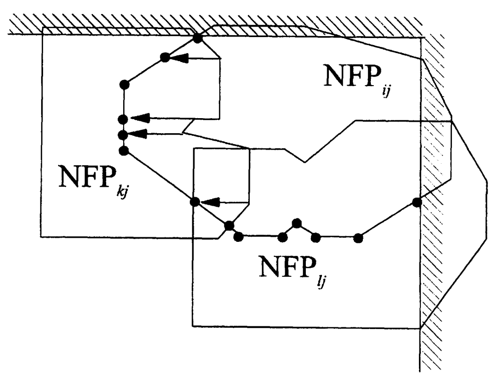
Recall that overlap between two pieces is approximated by its horizontal extent. This is equivalent to the distance that point $v - u$ has to move horizontally to reach the boundary of the NFP. To explain how the infinite neighbourhood is reduced, consider the example in Figure 3a. Piece j is the piece to be moved within the TT framework and originates from another area of the layout. Assume we want to evaluate the possibility of placing this piece next to piece i, which is

Figure 3 The Minimum Point of Overlap Can Be Found by Testing the Crossing Points and Vertices of the Nofit Polygon; These Points Are Illustrated by Dots

(a)



(b)

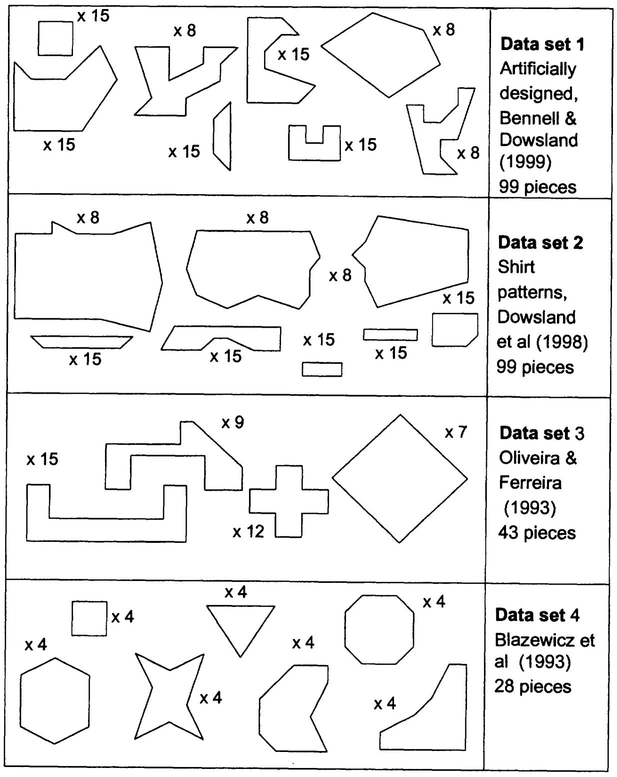


already adjacent to piece k, piece l, and the edge of the stock sheet. Figure 3b shows the NFPs of each of these pieces with piece j. These have been transposed, so that their origin lies at the origin of their respective fixed piece, positions i, k, l (i.e., transposed through vector u), resulting in the relationship that if j is placed inside NFP_{*j}, piece $*$ and j overlap, where $* = i, k, l$. Our intention is to identify the point on NFP_{ij} where overall overlap is minimised. The set of pieces overlapping j will change at the points where the boundaries of the NFPs cross. Within a segment between two such points the amount of overlap will be piecewise linear. The y coordinates of the breakpoints on this piecewise linear function are defined by the y coordinates of the vertices of NFP_{ij} and NFP_{*j}, where $*$ denotes any polygon that j intersects. Because overlap will increase or decrease linearly between these breakpoints and crossing points, then to find the point of minimum overlap, all that is required is to measure overlap at these points. In Figure 3 the breakpoints and crossing points are illustrated by dots; arrows indicate where the breakpoints originated. The scored lines indicate where NFP_{ij} lies with respect to the edge of the stock sheet. Placements outside the stock sheet are not permitted, and therefore search points beyond this line are not investigated.

5. Implementation and Experiments

Our intention is to assess the advantages of incorporating the optimisation phases described above by comparing their performance with that of the basic TT algorithm. Performance is measured with respect to solution quality and computational time. Experiments using both definitions of the solution space both with and without the optimisation phase were performed on four data sets from the literature. Details of the data sets appear in Figure 4. To investigate the compatibility of the tabu thresholding and the optimisation phase, experiments were run with graphics to observe the behaviour of the algorithm. These preliminary experiments highlighted the influential implementation issues. These are setting the bounds on movement within the LP, and deciding at which point the optimisation phase should be introduced and at

Figure 4 Experiment Data Sets Taken from the Literature



what frequency. All these decisions were found to be interdependent.

First, we examined when and how often the optimisation phase should be introduced. The obvious time of introduction is at local optima, i.e., the switch between the improving and mixed phases and at zero-overlap solutions. However, introduction at all local optima was counterproductive because many of these solutions contained large amounts of overlap, causing the LP to fail. Therefore, an upper limit was enforced on the maximum amount of overlap permitted in the initial LP solution. Because the capability of the LP to remove overlap is dependent on the bound on movement of each piece in the LP formulation, it is reasonable to assume the overlap limit should be a function of the bound. Our observations indicated that a limit of three times the bound performed well. The maximum overlap that can be removed between any two pieces is two times the bound. Given that overlap frequently exists between more than one pair of pieces and with the relaxation of constraints in place, this seems a sensible figure.

To set the bound, a trade-off must be resolved between the capability of the LP to compact and separate overlapping pieces in one iteration and the minimisation of redundant and obstructive constraints. A large bound would also require fewer iterations of the LP to reach the locally optimal solution. However, a small bound would encourage better solutions because it would be less likely to miss nesting positions by moving too far in one direction. Also, there would be fewer calls to the optimisation phase, thus only exploiting the good local optima. Experiments were carried out that involved 30 random number starts on each data set using four different bounds. These were minimum piece width multiplied by 0.25, 0.5, 0.75, and 1.0. Each increase in bound size increased the number of calls to the optimisation phase, thus increasing computational time. With respect to solution quality, on average 0.25 and 1.0 produced inferior results, and 0.5 outperformed 0.75, 57% of the time when directly comparing each random number start. Therefore, the bound was set to 0.5×minimum piece width for all further experiments.

Frequency of the optimisation phase was also examined because there was a tendency for many similar locally optimal solutions to be visited in close succession. For this phase to be most effective, solutions that initialise the optimisation phase should be in a different area of the solution space. Enforcing the condition that at least two executions of the mixed phase must be performed between each optimisation phase was sufficient to resolve this problem.

As a result of these preliminary experiments to investigate the interaction between the search heuristic and the optimisation phase, the optimisation phase is called at local optima provided one of the following two sets of conditions are satisfied: first, if the local optima does not contain overlap, and second, if the overlap is less then three times the bound and at least two executions of the mixed phase have occurred.

The full investigation comprised 100 randomly generated, initial solutions run with each of the four variants. Each experiment was allowed to run for a minimum of 5,000 moves; however, the algorithm was not permitted to terminate until at least 2,000 moves were performed with no improvement in the best zero-overlap solution. The results are presented graphically in Figures 5 and 6, and summary statistics are presented in Table 1. The variants are denoted as follows:

TTGrid =	Tabu thresholding with grid neighbourhood,
TTNofit =	Tabu thresholding with not polygon neighbourhood,
OPTGrid =	Hybrid; TTGrid and optimisation phase,
OPTNofit =	Hybrid; TTNofit and optimisation phase.

The graphs clearly show that the introduction of the optimisation phase has made a considerable improvement in packing efficiency over the pure TT algorithm for both definitions of the solution space. This is confirmed by the average percentage utilisation of the stock sheet shown in Table 1. It can also be seen that all the variants compare very favourably with other approaches found in the literature. It is interesting to observe that for the TTGrid variant the bar chart has uniform gaps between the bars. This highlights the problem of missed solutions as a result of the grid-defined placement positions. The optimisation phase is able to reduce this problem as these gaps are not so obvious for the OPTGrid variant. When the placement positions are defined by the nofit polygon, this problem does not exist.

Despite the improved packing efficiency, the nofit placement positions and the optimisation phase are shown to be computationally intensive when compared on CPU time. The additional complexity in the calculation of nofit polygon placement positions and the lack of uniformity in the spread of the placement positions severely slow down variants TTNofit and OPTNofit. It was anticipated that the optimisation phase would significantly increase computation time, but would compensate by reaching quality solutions more quickly. The termination criteria of the algorithm penalises the variants that include the optimisation phase because the minimum number of moves are fixed. Therefore, a new set of experiments was designed to terminate at a given solution quality. This is set for each data set as the lowest average utilisation across the variants, plus one standard deviation. The results can be found in Table 2. Here we

Figure 5 **Results of Experiments: x-axis Represents Decreasing Percentage Utilisation of Stock Sheet, and y-axis Represents Number of Experiments**

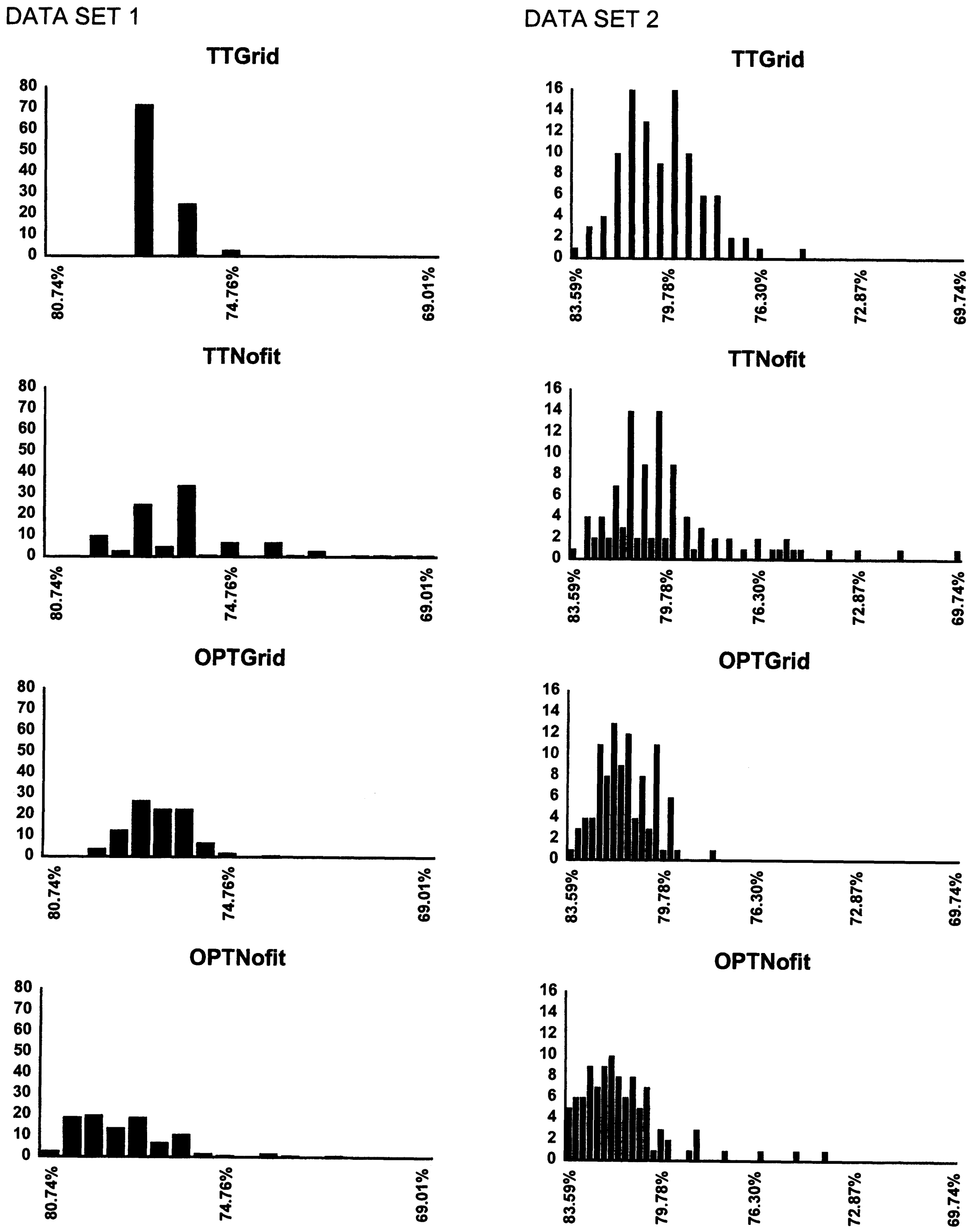


Figure 6 Results of Experiments: x-axis Represents Decreasing Percentage Utilisation of Stock Sheet, and y-axis Represents Number of Experiments

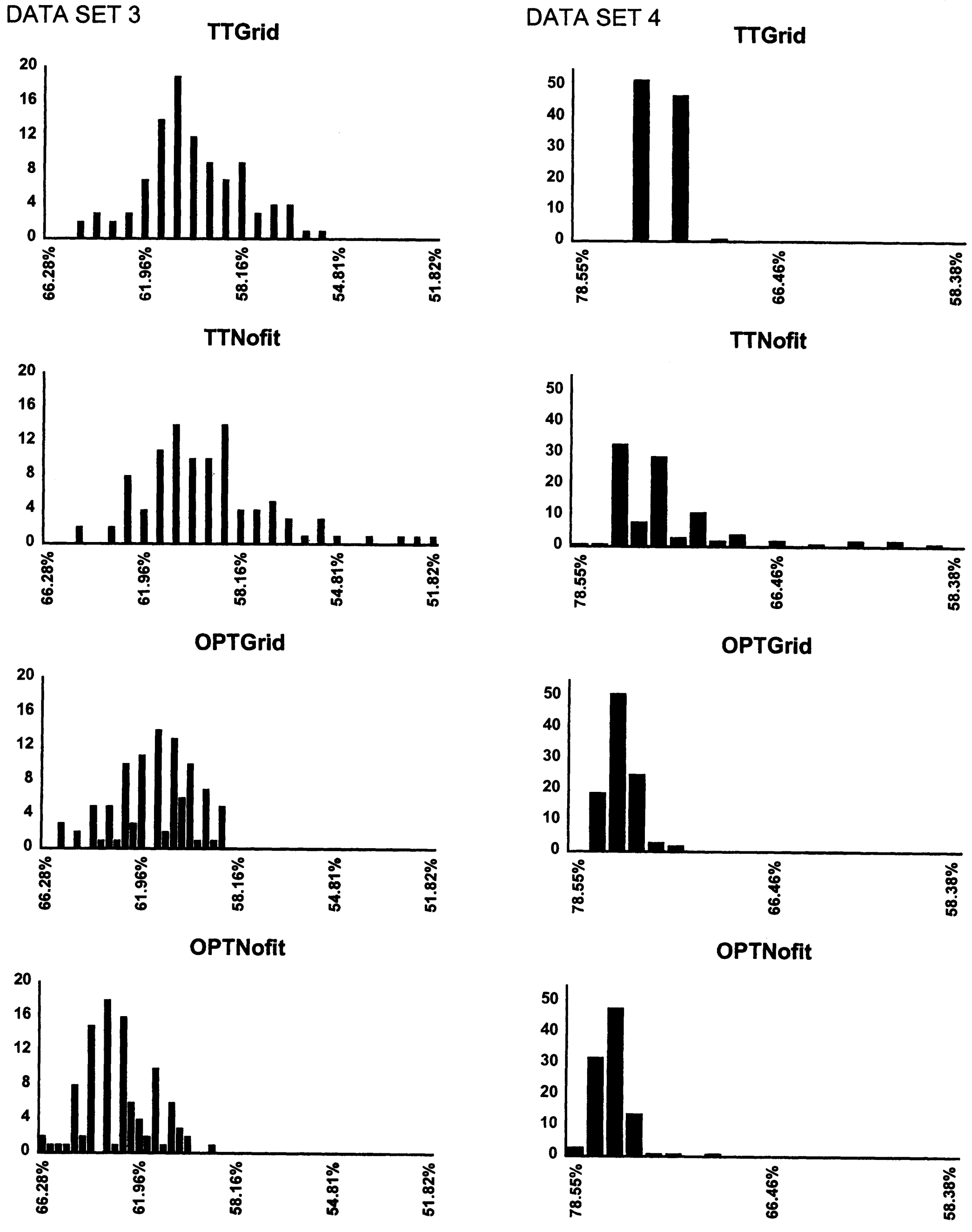


Table 1 Packing Efficiency of Variants with Respect to Percentage Utilisation of the Stock Sheet and CPU Time in Minutes

Data set 1	TTGrid	TTNofit	OPTGrid	OPTNofit	Dowsland	Oliveira
Average	77.47%	76.25%	77.24%	78.38%	74.8%	—
St. dev.	1.12%	2.14%	1.09%	1.83%	—	—
Ave. CPU	6.37	14.85	10.84	21.43	—	—
Data set 2	TTGrid	TTNofit	OPTGrid	OPTNofit	Dowsland	Oliveira
Average	80.03%	79.20%	81.42%	81.18%	80.5%	76.2%
St. dev.	1.59%	2.53%	1.11%	1.11%	—	—
Ave. CPU	5.96	19.97	17.08	28.12	—	—
Data set 3	TTGrid	TTNofit	OPTGrid	OPTNofit	Dowsland	Oliveira
Average	60.01%	59.39%	61.4%	62.81%	57.0%	58.2%
St. dev.	1.91%	2.46%	1.56%	1.46%	—	—
Ave. CPU	5.30	9.26	9.82	12.57	—	—
Data set 4	TTGrid	TTNofit	OPTGrid	OPTNofit	Oliveira	Blazewicz
Average	73.27%	71.56%	76.08%	76.58%	68.8%	69.1%
St. dev.	1.32%	3.75%	1.12%	1.37%	—	—
Ave. CPU	0.83	1.94	1.34	2.81	—	—

Notes. Table also includes results from literature as follows: Dowsland et al. (1998), Oliveira et al. (1996), and Blazewicz et al. (1993). These results correspond to identical data on identical width stock sheets with the exception of Oliveira et al. (1996) who use a wider stock sheet for data set 4 and Blazewicz et al. (1993), who originated data set 5 with some curved edges. Both Oliveira et al. (1996) and Blazewicz et al. (1993) allow rotation of the pieces.

can see that the execution times are much more competitive. It is worth noting that the LPs were solved using relatively old software, and it is possible that more recent LP code would be quicker. More detailed analysis compared the experiments on a run-for-run basis, and it was found that the variants that use the optimisation phase take fewer moves, 67% of the time.

Table 2 Results of Experiments with Percentage Utilisation as Termination Criteria

Data set 1	TTGrid	TTNofit	OPTGrid	OPTNofit
Ave. no. of moves	750.36	1226.80	1220.20	816.04
Ave CPU (mins)	1.078	4.815	2.594	3.979
Data set 2	TTGrid	TTNofit	OPTGrid	OPTNofit
Ave. no. of moves	505.84	1049.60	430.28	446.08
Ave CPU (mins)	1.165	5.569	1.512	2.661
Data set 3	TTGrid	TTNofit	OPTGrid	OPTNofit
Ave. no. of moves	691.56	891.72	359.64	210.68
Ave CPU (mins)	1.544	2.304	0.771	0.471
Data set 4	TTGrid	TTNofit	OPTGrid	OPTNofit
Ave. no. of moves	343.96	955.68	198.68	278.88
Ave CPU (mins)	0.095	0.571	0.065	0.183

6. Conclusion

This paper has explored the potential of hybridising a neighbourhood search technique with appropriate optimisation routines to provide an effective solution procedure for the solution of the irregular stock-cutting problem. Both approaches used fall within the scope of what Glover and Laguna (1997) refer to as *referent domains*. The first hybrid incorporates an additional phase into the two-phase tabu thresholding technique. This allows moves that do not fall within the scope of valid neighbourhood moves. The phased nature of tabu thresholding make it particularly suitable for this type of modification. The second hybrid uses an optimisation routine to find the best solution within an infinite neighbourhood.

The experimental results show that the inclusion of the optimisation phase improves the performance of the tabu thresholding packing algorithm. Also, tabu thresholding provides a good starting solution for the optimisation phase. As anticipated, including this phase incurs a computational overhead. However, OPTGrid and OPTNofit are shown on average to find more efficient solutions in fewer moves than TTGrid and TTNofit. The nofit polygon subsets did not produce the improvements expected and were computationally intensive. We suggest two possible explanations. First, a lack of uniformity in search positions with some areas searched repeatedly for one move. Second, noninteger placement positions on the boundary of the polygons may create problems associated with calculations of overlap in later moves of the algorithm. Further research is needed with this placement definition in iterative algorithms.

For the data sets examined here, the variants that included the LP-based compaction approach produced more efficient solutions and produced them more consistently.

References

Bennell, J. A. 1998. Incorporating problem specific knowledge into a local search framework for the irregular shape packing problem. Ph.D. thesis, European Business Management School, University of Wales, Swansea.

———, K. A. Dowsland. 1999. A tabu thresholding implementation for the irregular stock cutting problem. *Internat. J. Production Res.* **18** 4259–4275.

———, ———, W. B. Dowsland. 2001. The irregular cutting stock problem—A new procedure for deriving the nofit polygon. *Comput. OR* **28**(3) 271–287.

Blazewicz, J., P. Hawryluk, R. Walkowaik. 1993. Using tabu search for solving the two-dimensional irregular cutting problem. *Ann. Oper. Res.* **41** 313–325.

Dagli, C. H., A. Hajakbari. 1990. Simulated annealing approach for solving stock cutting problem. *Proc. IEEE Internat. Conf. Systems, Man, and Cybernet.* 221–223.

Dowsland, K. A., W. B. Dowsland, J. A. Bennell. 1998. Jostle for position—local improvement for irregular cutting patterns. *J. Oper. Res. Soc.* **49** (6) 647–658.

Ghosh, P. K. 1991. An algebra of polygons through the notion of negative shapes. *CVGIP: Image Understanding* **54** (1) 119–144.

Glover, F. 1992. Simple tabu thresholding in optimisation. Internal report, University of Colorado, Boulder, Colorado.

———, M. Laguna. 1997. Tabu search. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Heckmann, R., T. Lengauer. 1995. A simulated annealing approach to the nesting problem in the textile manufacturing industry. *Ann. Oper. Res.* **57** 103–133.

Jain, P., P. Fenyes, R. Richter. 1992. Optimal blank nesting using simulated annealing. *Trans. ACME* **114** 160–165.

Li, Z., V. Milenkovic. 1995. Compaction and separation algorithms for non-convex polygons and their application. *Eur. J. Oper. Res.* **84** (3) 539–561.

Oliveira, J. F., J. S. Ferreira. 1993. Algorithms for nesting problems. R. V. V. Vidal, ed. Applied Simulated Annealing, *Lecture notes in Economics and Mathematical Systems 396*, Springer Verlag 255–274.

———, A. M. Gomes, J. S. Ferreira. A new constructive algorithm for nesting problems. *OR Spektrum* Forthcoming.

———, J. S. Ferreira. 1994. Some experiments with tabu search for solving nesting problems. *EURO XIII/OR* **36** Glasgow.

Stoyan, Y. G., M. V. Novozhilova, A. V. Kartashov. 1993. Mathematical model and method of searching for a local extremum for the nonconvex oriented polygons allocation problem. Working paper. Ukranian Academy of Science, Institute for Problems in Machinery, Kharkov, Ukraine.

———, L. D. Ponomarenko. 1977. Minkowski sum and hodograph of the dense placement vector function. Working paper. Ukranian Academy of Science, Institute for Problems in Machinery, Kharkov, Ukraine.

Accepted by Th. M. Liebling; received July 2000. This paper was with the authors 2 months for 1 revision.