REFERENCES

[1] S. K. Card, "Studies in the psychology of computer text editing systems," Ph.D. dissertation, Carnegie-Mellon Univ., Pittsburgh, PA, 1978.

[2] S. K. Card, T. P. Moran, and A. Newell, "The keystroke level model for user performance time with interactive systems," Xerox Palo Alto Research Center, Palo Alto, CA, Tech. Rep. SSL 79-1, Jan. 1979.

[3] C. A. Ellis, R. P. Gibbons, and P. A. Morris, "The control of information in complex organizations: An analytic approach," working paper at Xerox Palo Alto Research Center, Palo Alto, CA, Sept. 1979.

[4] D. C. Englebart and W. K. English, "A research center for augmenting human intellect," *AFIPS Confer. Pro.*, vol. 33, pp. 395–410, 1968.

[5] M. Fein, "Job enrichment: A reevaluation," *Sloan Management Rev.*, Winter, 1974.

[6] C. Feinstein, P. Morris, and R. Smallwood, "Information trees," presented at the October 1978 Joint ORSA/TIMS Conf.

[7] T. P. Moran, "Introduction to the command language grammar: A representation for the user interface of interactive computer systems," Xerox Palo Alto Research Center, Palo Alto, CA, Tech. Rep. SSL 78-3, Oct., 1978.

[8] S. S. Oren, "A mathematical theory of man–machine text editing," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-4, pp. 258–267, May 1974.

[9] ——, "A mathematical theory of man–machine document assembly," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-5, pp. 520–526, Sept. 1975.

[10] C. A. Petri, *Kommunikation Mit Automaton*. Bonn, Germany: Univ. Bonn, 1962.

[11] M. F. Stankard, "Clerical service systems" in *McGraw-Hill Handbook of Problem Solving*. New York: McGraw-Hill, 1980, to appear.

[12] M. D. Zisman, "Representation, specification, and automation of office procedures," Ph.D. dissertation, Univ. Pennsylvania, Philadelphia, PA, circa 1977.

Optimal Allocation of Two-Dimensional Irregular Shapes Using Heuristic Search Methods

ANTONIO ALBANO AND GIUSEPPE SAPUPPO

Abstract—**A problem of relevant interest to some industrial processes is the one of allocating a specified number of two-dimensional regular or irregular generally different pieces on a stock of sheets of finite dimensions, the object being to minimize the amount of waste produced. The problem is present in industrial applications like shipbuilding, clothing manufacturing, or leather cutting. An approach is given where the solution is achieved with a heuristic search method typical of the artificial intelligence discipline. In this paper it is shown by the experimental results that this framework not only allows a simple formulation of the solution but represents an effective technique to obtain solutions competitive with the ones produced by hand.**

I. INTRODUCTION

THE PROBLEM of the allocation of regular and irregular shapes arises frequently in applications where it has to be determined how a set of two-dimensional pieces will fit into (or be cut from) a large stock sheet of finite dimensions the object being to maximize the value of pieces fitted (or cut) or alternatively minimize the waste. Moreover, in some applications, the large sheet to be used contains defective areas, and the allocated pieces must not overlap these defects. Such problems appear, for example, in stamping parts from steel strips, cutting cloth for suits, etc. Other similar cutting problems involve the cutting of rectangular pieces as, for example, the cutting of steel, wood, or glass plates, the cutting of paper board for the production of boxes, and several other applications involving cutting operations but also operations unrelated to cutting such as the multiprocessing of a batch of programs. These problems, however, will not be discussed here and for an updated bibliography on these subjects see [1], [2].

In recent years, a number of researchers have investigated the problem of the allocation of irregular shapes and the proposed approaches can be grouped in three categories: manual, automatic, and semiautomatic.

In the first case, the draftsman makes use of a computer with graphic input–output devices, capable of manipulating (rotating, translating, etc.) pieces on a working display [3], [4]. In the second category, two more classes of proposals can be distinguished. The first one utilizes a two-step procedure in which the problem is converted from one of placing irregularly shaped pieces to one of allocating rectangular modules obtained, in general, from the rectangular enclosures for various clusterings of irregular and rectangular pieces [5]–[8]. Although these investigations have been successful in handling specific

Manuscript received June 6, 1979; revised January 28, 1980. This work was supported in part by the Consiglio Nazionale delle Ricerche.

A. Albano is with the Istituto di Scienze dell'Informazione, Corso Italia, 40-56100 Pisa, Italy.

G. Sapuppo is with the Cassa di Risparmio di Firenze, Centro Elettronico, Via Bufalini, 6-50100 Firenze, Italy.

0018-9472/80/0500-0242$00.75

subsets of the general optimal two-dimensional allocation problem, their approach, however, has resulted in procedures that are marginally useful for applications with a small number of pieces where an exact ordering of the pieces is required.

This situation has been considered in the second class of the automatic approach, where a procedure is defined, which searches for an optimal arrangement by operating directly on the pieces [9]–[12]. These methods are similar insofar as they are based on a set of heuristic rules used by deterministic sequential placement techniques: once a piece has been placed in a certain position, it will not be reconsidered any more in respect to any consideration that may result later. This shortcoming is not present in the approach proposed by Adamowicz [13], [14] which involves the iterative solution of a linear programming problem followed by a fitting one, which can generate new constraints for the next iteration until the optimal solution is obtained. However, the experimental program was not completely operational, and its time performance and the quality of the solutions are not known.

In the semiautomatic approach, to alleviate the problem of a deterministic placement procedure, a system is proposed where a tentative solution is automatically generated and then interactive improvements are allowed by a conversational display unit, using a special set of commands to operate on the solution process [15].

In this paper an automatic approach is presented which transforms the allocation problem into a search process through a "space" of candidate solutions. Since this space is generally very large, heuristic methods will be employed for shortening the search.

In the next sections, after a brief description of the heuristic search methods developed in artificial intelligence, we will show how this framework allows the formulation of a nondeterministic solution of the allocation problem. Then some experimental results will be described and the effectiveness of the algorithm will be commented on.

The present paper makes the following assumptions.

- The pieces are irregular polygons without holes and the sheet is a rectangle, large enough to contain all the demanded pieces disregarding the quality of the solution.
- The pieces can be rotated; they must lie entirely within the sheet and must not overlap.
- The goal is to minimize the waste, or better, the length of the produced packing.

II. State Space Search Methods

Many problems in artificial intelligence and operations research are solved by a general problem-solving technique based on searching through a "space" of candidate solutions. In this section the main aspects and terminology of this approach will be recalled. For a complete presentation see Nilsson [16].

The first step consists in associating to the problem a set of *states* and *operators* transforming one state into another. The set of states reachable from the initial state can be seen as a direct graph containing nodes corresponding to the states and arcs corresponding to the operators. With this representation in mind, the solution is seen as a *search process* for finding a path from the initial to any member of a set of nodes called the *goal* or *final* nodes.

A great part of the search process can be organized in the following steps.

1) Put the start node on a list called GENERATED.
2) If GENERATED is empty, exit with failure; otherwise continue.
3) Select a node from GENERATED according to a rule R and put it on a list called EXPANDED. Call it n.
4) If n is a goal node, exit with the solution path; otherwise continue.
5) Expand n, that means generate all its successors. If there are no successors, go to step 2); otherwise put them on GENERATED, and go to step 2).

In order to specify the process completely, the rule R for selecting the node to be expanded must be decided. Two classes of algorithms can be distinguished according to whether or not they use information about the presented problem to speed up the search of the optimal path. This information is called *heuristic information*: when it is used, we talk about a *heuristic search method*. Hart, Nilsson, and Raphael [17], [18] gave conditions under which it is possible to obtain all the benefits of the heuristically guided search without loosing the optimality of the final path. They introduced an estimate that a node n can be on the optimal solution path, called *evaluation function*, defined as $\hat{f}(n)=\hat{g}(n)+\hat{h}(n)$. Here $\hat{g}(n)$ is an estimate of $g(n)$, the shortest path cost from the start node to n, and $\hat{h}(n)$ an estimate of $h(n)$, the shortest path cost from n to the final node. At each iteration, the node n whose total estimate $\hat{f}(n)$ is smallest will be selected. An obvious choice for $\hat{g}(n)$ is the minimum cost of the path from the start node to n, computed step by step during the search. On the other hand, the choice of $\hat{h}(n)$ strongly influences the properties of the search algorithm. These authors have shown that, if $\hat{h}(n)$ is a lower bound on the cost of the minimal cost path from node n to a goal node, the search will end up with an optimal path.

As it will be shown in the sequel, when the problem is complex and it is impossible to find a significant estimate of $h(n)$ different from the constant zero, and the time to produce the solution is also of importance in evaluating the results, then, in order to increase the search power, the above condition is often released at the expense of obtaining a generally not optimal solution which, however, proves to be "good" and efficient in terms of computer time.

III. Representation of the Allocation Problem

In this section we will show how the problem of finding the optimal allocation of a set of irregular pieces can be transformed into the problem of finding an optimal path through a space of problem states from the initial state to a goal state. Remember that the rectangular resource has to be large enough to contain all the demanded pieces disregarding the quality of the solution which will be produced.

Let an instantaneous allocation A_i on the resource be an ordered pair of lists (P,Q), such that P contains the position and orientation of all the pieces allocated, and Q contains the unallocated pieces. At the beginning P is empty and A_0 is called the *initial allocation*; while, when Q is empty, A_i is called a *final allocation* and is a solution of the problem. The allocation profile of A_i is the "front" produced by the rightmost borders of the rightmost pieces which are exposed to the part of the resource not yet used. Fig. 1 provides examples of profiles indicated by thicker lines.

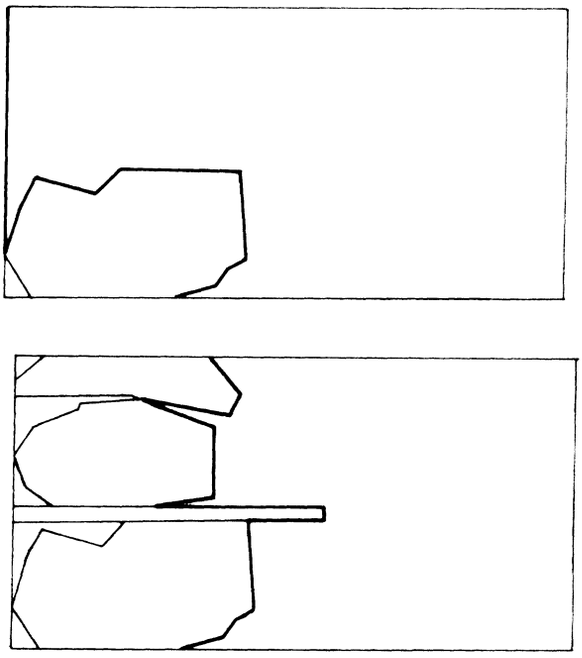


Fig. 1. Profile examples.

An allocation A_{i+1} is a *successor* of A_i if it is obtained arranging one of the unallocated pieces in A_i according to a prefixed *placement policy*.

The placement policy has to satisfy all the required constraints on the layout, e.g., a new piece should be arranged inside the resource so as not to overlap the preceding ones. The policy chosen in this proposal is the obvious although not necessarily the optimum one, considering the fact that the length of the final allocation has to be minimized: the piece is placed in contact with the profile of A_i in a position which produces the leftmost lowest allocation.

An *allocation process* is a finite sequence of instantaneous allocations such as the first is *initial*, the last is *final*, and each A_i, $i>0$, is a successor of A_{i-1}.

Given an instantaneous allocation A_i, let the *space available*, *space* (A_i), be the area on the right side of the profile, i.e., the area which may be used to allocate a piece p_i. As an example, in Fig. 2 the area S2 will not be considered usable any more. Then the *added waste* in A_i as a consequence of the allocation of p_i is defined as

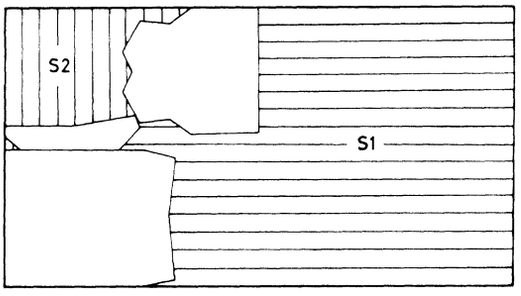


Fig. 2. Space available example.

$$\text{added waste } (A_i) = \text{space } (A_{i-1}) - (\text{space } (A_i) + \text{area of } (p_i)).$$

The waste in A_i is defined recursively as

$$\text{waste } (A_0) = 0$$

$$\text{waste } (A_i) = \begin{cases} \text{waste } (A_{i-1}) + \text{added waste } (A_i), \\ \quad \text{if } A_i \text{ is not final} \\ w \times l - \Sigma \text{ area of } (p_i), \\ \quad \text{otherwise.} \end{cases}$$

w and l are respectively the width of the resource and the length of the layout, that is, in the final allocation the whole area between the pieces and the rectangle with sides equal to the given width and a length sufficient to wholly include all the pieces, will be considered as waste.

In order to transform the optimal allocation problem into the search of an optimal path in a state space, each state corresponds to the allocation of a piece according to the placement policy. The initial state S_0 corresponds to the allocation of no piece and the final one to the allocation of all the demanded pieces. Each path between the initial state and any of the final ones is a possible final allocation. The cost of an arc between two states S_i and S_{i+1} is the added waste produced by the allocation of the piece associated to S_{i+1} to generate the allocation A_{i+1}, and the cost of a path from S_0 to S_{i+1} is the waste of the allocation A_{i+1}. With this representation the optimal allocation of a set of pieces is reduced to the problem of finding the path from the initial state to a final one while minimizing the cost.

The procedure to implement the heuristic search method is given in Fig. 3. The meaning of the employed evaluation function is explained in the next section together with the other heuristics used to increase, in a controlled way, the search power of the procedure.

```
begin
   initial conditions;
   input pieces and resource descriptions;
   let the CURRENT-NODE be the Initial State;
   while CURRENT-NODE ≠ goal node do
         for all pieces left to be allocated do
             for all orientations do
                 apply placement policy;
                 apply evaluation function and
                 append successor to the list GENERATED
             end
         end
   set CURRENT-NODE EXPANDED;
   let the CURRENT-NODE be the "best" successor in GENERATED;
   end
   plot the solution;
end
```

Fig. 3. Allocation procedure.

IV. Techniques to Increase the Heuristic Search Power

The described procedure cannot be applied to realistic applications without introducing a few modifications to reduce the increasing number of expanded nodes and to keep the computation time within reasonable limits. Although the optimal solution cannot be assured further, it will be shown with the experimental results, that the procedure achieves an interesting trade-off between the solution time and the quality of the solution produced.

1) Evaluation Function: In order to preserve the "admissibility" of the search algorithms, i.e., the property of finding a minimal-cost path, the evaluation function used to choose from the list of unplaced pieces the one to place next, should be expressed as sum of $\hat{g}$ and $\hat{h}$, two cost estimate functions. For expressing the first one, no problem will arise because it is given by the *waste* function previously defined. The choice of $\hat{h}$ is not so simple, because it should always be lower than the waste that would result in the optimal solution if the piece in question were to be included. A possibility is $\hat{h} \equiv 0$, but it has been preferred to drop the admissibility property because in this type of applications the optimality of the solution is unimportant and a "good" one will be enough. Therefore, a fixed percentage of the total area of the unplaced pieces has been introduced as expected waste, because this choice has also the property to anticipate the use of the largest pieces (parameter FUTURE WASTE).

2) Successor Limitation: In order to generate the successors of a node, for each of the n unplaced pieces, all the k allowed orientations should be considered and $n \times k$ successors should be added. Since this step is the one consuming most of the time, the following limitations have been introduced.

1) For each yet unplaced piece the placement policy is applied to all possible orientations, but only the one which produces the leftmost lowest allocation will be considered in step 2. Ties are resolved in favor of the allocation with the minimum extension to the right.
2) From the pieces chosen in the previous step, only a fixed number (parameter MAXGENERATED) of the leftmost allocated pieces are preserved for step 3. Ties are resolved in favor of the piece with the maximum extension to the right.
3) For the allocation obtained in the previous step, a fixed number of successors (parameter *N*-SUCCESSORS) will be generated according to the value of $\hat{f}$. Ties are resolved in favor of the piece presented first.
4) Since the list of generated nodes is finite (parameter LIST SIZE), when it becomes full, the tree so far generated by the search is pruned by erasing the node with the highest $\hat{f}$ value.

3) Evaluation Function Discretization: During the execution of the program the evaluating function assumes large values distributed in a narrow interval. Therefore it can happen that a successor of the current expanded node is by a few units worse than all the others in the list GENERATED. In this case, it is preferable to continue the search along the same path as far as the value of the evaluation function is still slightly higher. This has been accomplished by introducing a parameter PRECISION which represents the number of the most significant digits of the $\hat{f}$ values to be considered by the algorithm. The other digits are set to zero.

4) Expansion Band: When the search is at kth level, the next node to be expanded has to be at a level not lower than $(k-t)$, where t is a given threshold. It is interesting to note that, in general, if we increment t by a few units, the solution does not necessarily improve. In fact it may happen that the search proceeds too much along a promising path and because of the expansion band it is impossible to resume a better path search previously suspended.

5) Termination Condition: The search does not terminate when all the pieces have been allocated, but when the list of generated nodes is empty. That is, in the end the procedure develops all the possible search trees within the expansion band and the final solution will be the one which requires the least resource.

6) Profile Simplification: At each step the profile is simplified in order to exclude all the areas on the left side of the vertical line through the leftmost point of the last allocated pieces. This operation is shown in Fig. 4, and it has been included to simplify the most frequent and time consuming operations involving the profile.

V. Experimental Results

The algorithm has beeen programmed in ALGOLW, some parts in Assembler for the VM/370 running on a IBM 370/168.

The program uses some operators for the pieces and profile descriptions. Some of the more important ones are cited below.

Piece [Piece–Description]: Inputs the polygon description of a piece. A piece is described by providing a list of

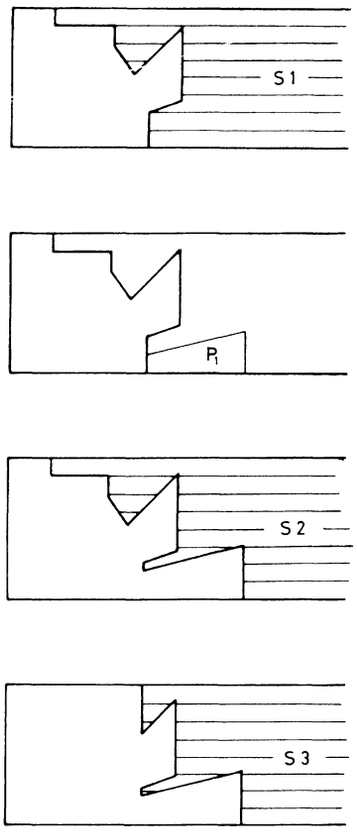


Fig. 4. Profile simplification.

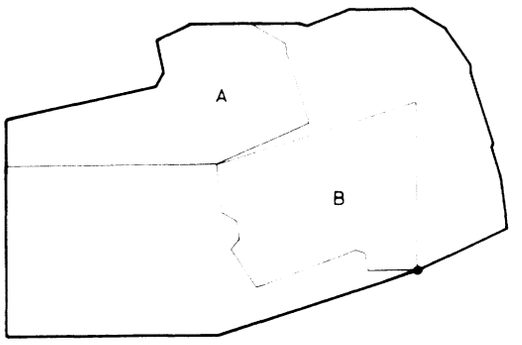


Fig. 5. No-fit-polygon example.

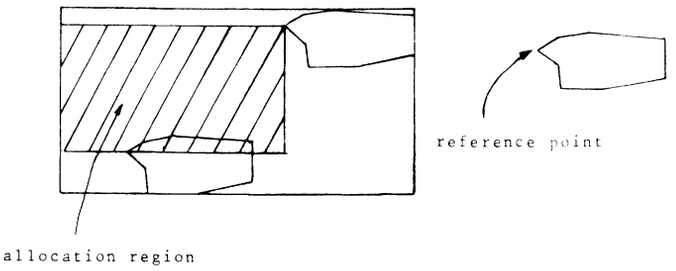


Fig. 6. Allocation region example.

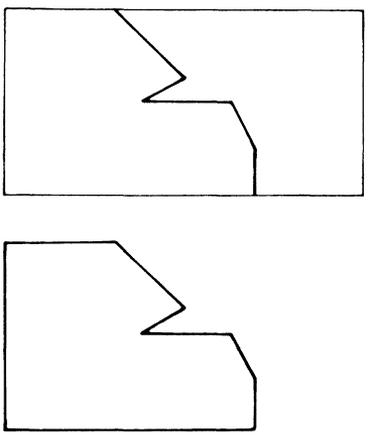


Fig. 7. Envelope example.

the points (vertices) of the polygon approximation given in a counterclockwise (CCW) direction. The co-ordinates of each point of the polygon are given relative to the polygon reference point, which may or may not be a point on the original polygon.

No-Fit-Polygon [*Piece A, Piece B*]: Computes the no-fit-polygon (NFP) for the given pair of pieces with the given orientations. The NFP completely describes all those positions where the reference point of *B* may be placed in order to have *B* touching *A* without overlapping it (see Fig. 5 from Adamowicz–Albano [19]).

Allocation Region [*Resource, Piece*]: Computes the perimeter of the area in which the reference point of the piece can validly fall (see Fig. 6).

Envelope [*Profile*]: Closes the profile with the border of the resource (see Fig. 7).

Allocation Point [*Profile, Piece*]: Computes the leftmost lower allocation point of the piece with respect to the current profile.

New Profile [*Profile, Piece, Allocation Point*]: Computes the new profile resulting by a combination of the old one and the piece placed with the reference point in the allocation point (see Fig. 8).

Normalization [*Piece*]: Normalizes the description of the pieces in such a way that the reference point, the leftmost lower vertex and the origin of the coordinates will coincide.

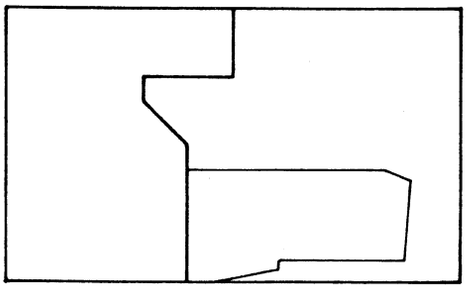
Area [*Piece*]: Computes the area of the piece.

Flip [*Piece*]: Reflects the shape about the vertical axis through the reference point.

Rotate [*Piece, Theta*]: Rotates the piece through of "theta" degrees CCW about the reference point.

Moreover, when the program is used interactively, it is possible to obtain information on the status of the search, on the use of the memory available and other information which make it possible to change the parameters which influence the search strategy. These facilities have been implemented to control the program execution during its development; they could be improved by the use of an interactive display system where an operator can interactively modify the layout [20].

In the following examples, the pieces can be only rotated by 180°, and for each test two different results are presented. In the first one, the search proceeds in a sequential and deterministic way, i.e., once a piece is placed in some position, it will not be removed anymore. The position achieved will be its position in the final



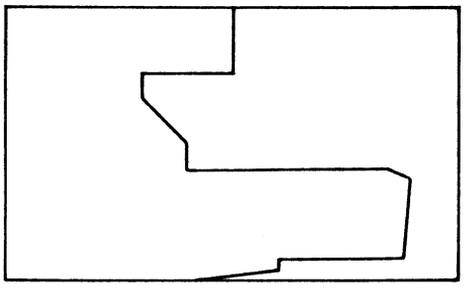


Fig. 8. New profile example.

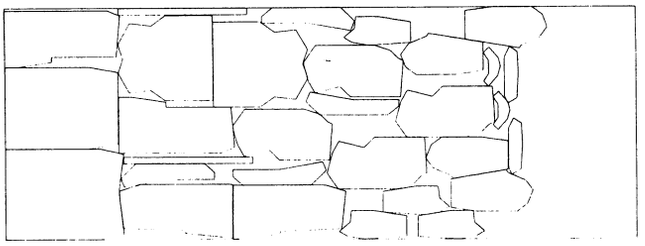


Fig. 9. Layout example (expansion band = 0).

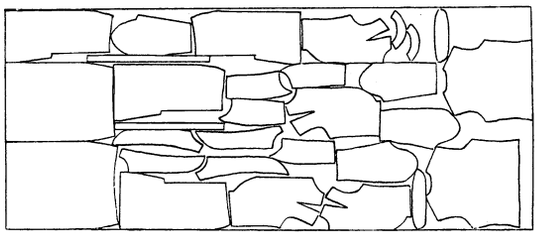


Fig. 10. Art layout example.

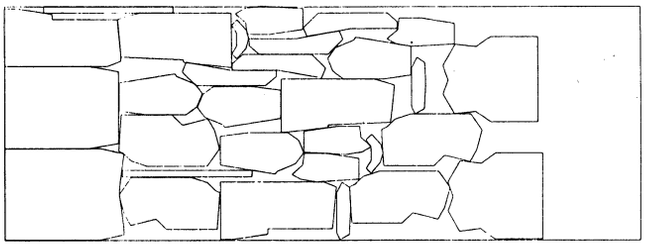


Fig. 11. Layout example (expansion band = 3).

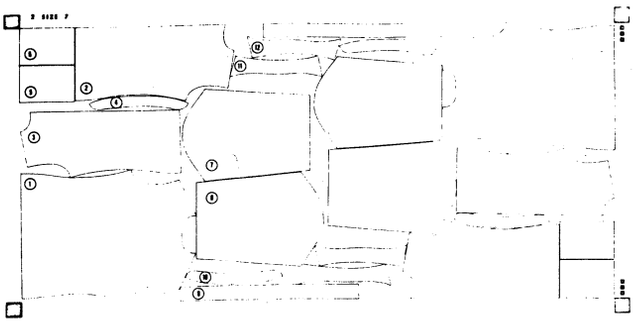


Fig. 12. Gurel layout example.

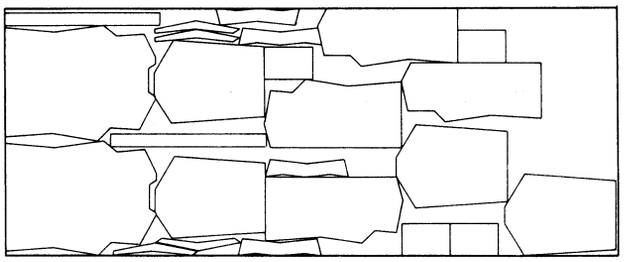


Fig. 13. Layout example (expansion band = 0).

Fig. 14. Layout example (expansion band = 3).

TABLE I

	Number of Pieces	Future Waste	Maxgenerated	N-Successors	Expansion Band	Final Waste	Computing Time (s)	Number of Expanded Nodes	Number of Generated Nodes
Fig. 9	30	20 percent	18	18	0	20.8 percent	27.7	30	243
Fig. 11	30	20 percent	9	5	3	19.4 percent	162.	344	1206
Fig. 13	24	20 percent	8	8	0	27.2 percent	8.5	24	114
Fig. 14	24	20 percent	5	4	3	17.4 percent	30.4	187	426

allocation. The other way of producing the solution, allows the process to backtrack, i.e., to suspend the current path search and to develop a new one. The backtracking is controlled by the size of the expansion band. This last facility is the main feature introduced with this approach which makes it more effective than the other proposals.

Fig. 9 provides an example of layouts generated using the pieces in Art's solution shown in Fig. 10. The values of the parameters and the results are given in Table I. In this test the search proceeds without backtracking as in Art's approach, and the waste produced is already lower. Exact figures cannot be given, because the piece descriptions were obtained from Art's drawings. On the same problem the search capability of the program was exploited allowing a backtracking with an expansion band of three levels. The results shown in Fig. 11 and in Table I, represent a sensitive improvement in the solution optimality at the expense of an increment of computer time.

Fig. 12 provides an example of handmade layouts given by Gurel [21]. The solutions in Fig. 13 and Fig. 14 have been obtained with the parameters given in Table I, and again it is shown how allowing more exploration in the problem space, the algorithm is able to improve the solution at the expense of increasing the computing time.

VI. Conclusion

An approach has been described to produce an optimal arrangement of irregular pieces into a rectangular resource. The problem has been reduced to a search of an optimal path in a graph and, using an heuristic search method, an algorithm has been implemented which produces an approximate solution which proves to be of good quality and efficient in terms of computer time. The balance between the solution quality and the computing time can be interactively controlled by the user of the program. Moreover, this approach appears to give a natural framework also to solve extensions of the problem where an irregular resource is considered and more complex constraints are included in the placement of the pieces. The algorithm has been tested throughly and its performance indicates that it favorably compares with some others designed to solve this problem.

References

[1] N. Christofides and C. Whitlock, "An algorithm for two-dimensional cutting problems," *Oper. Res.*, vol. 14, pp. 30–44, 1977.

[2] A. Albano and R. Orsini, "A heuristic solution of the rectangular cutting-stock problem," *Nota Scientifica* S78/16, Istituto di Scienze dell'Informazione, Nov. 1978, to appear in *Computer Journal*.

[3] E. Virgilietti, "Graphics in ship design," *Sperry Univac Symposium: Computers in Shipbuilding*, Roma, 1976.

[4] A. Catastini, C. Cavagna, U. Cugini, and P. Moro, "A computer-aided design system for pattern grading and marker making in the garment industry," *CAD 76 Proc.*, London, pp. 159–165, 1976.

[5] M. Adamowicz and A. Albano, "Two-stage solution of the cutting stock problem," in *Information Processing 71*. Amsterdam: North-Holland, 1972, pp. 1086–1091.

[6] M. Haims and H. Freeman, "A multistage solution of the template layout problem," *IEEE Trans. Syst. Sci. & Cybern.*, vol. *SSC-6*, pp. 145–151, 1970.

[7] H. Freeman, "On the packing of arbitrary-shaped templates," Division of Applied Science, New York University, New York, internal rep. 1974.

[8] M. Adamowicz and A. Albano, "A solution of the rectangular cutting-stock problem," *IEEE Trans. Syst., Man, Cybern.*, vol. *SMG-6*, pp. 302–310, 1976.

[9] R. C. Art, "An approach to the two-dimensional irregular cutting stock problem," IBM Cambridge Scientific Center, Cambridge, MA, Rep. No. 320-2006, 1966.

[10] M. Tanaka and T. Wachi, "Computerized marker making," *J. Textil Machinery Society of Japan*, vol. 19, pp. 74–81, 1973.

[11] O. Gurel, "Circular graph of marker layout," IBM New York Scientific Center, New York, NY, Rep. No. 320-2965, 1969.

[12] G. R. Moreau and P. T. DeSaint Hardavin, "Marker layout problem: An experimental attempt," IBM New York Scientific Center, New York, NY, Rep. No. 320-2978, 1969.

[13] M. Adamowicz, "The optimum two-dimensional allocation of irregular, multiply-connected shapes with linear, logical and geometric constraints," Ph.D. dissertation, Elec. Eng. Dep., New York University, New York, NY, 1969.

[14] ——, "Optimum allocation of two-dimensional shapes," in *Proc. Int'l Design Automation Conf.*, Toronto, Canada, 1971.

[15] A. Albano, "A method to improve two-dimensional layout," *Computer Aided Design*, vol. 9, pp. 48–52, 1977.

[16] N. J. Nilsson, "*Problem Solving Methods in Artificial Intelligence*," New York: McGraw-Hill, 1971.

[17] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost path," *IEEE Trans. Syst. Man, Cybern.*, vol. 4, pp. 100–107, 1968.

[18] ——, Correction to "A formal basis for the heuristic determination of minimum cost paths," *SIGART Newsletter*, pp. 28–29, 1972.

[19] M. Adamowicz and A. Albano, "Nesting two-dimensional shapes in rectangular modules," *Computer Aided Design*, vol. 8, pp. 27–33, 1976.

[20] G. Sapuppo, "Allocazione di forme irregolari su risorse rettangolari," Tesi di Laurea, Istituto di Scienze dell'Informazione, Università di Pisa, 1976.

[21] O. Gurel, Circular graph of marker layout," IBM New York Scientific Center, New York, NY, Rep. No. 320-2965, 1969.