Tool Carousel Design via Integer Programming

L R Foulds*
Department of Management Systems
University of Waikato
New Zealand
lfoulds@waikato.ac.nz

J M Wilson
Business School
Loughborough University
Great Britain
j.m.wilson@lboro.ac.uk

Abstract

We describe a branch and bound algorithm for an assignment problem subject to a special set of side constraints. The problem has application in the design of tool carousels for certain flexible manufacturing systems. The resulting model represents a special case of the restricted facilities layout problem in which it is forbidden to locate any facility in certain zones. The bounds for the algorithm are generated by relaxing the side constraints and using the Hungarian method to solve the resulting assignment problem. Partitioning in a manner similar to subtour elimination for the travelling salesman problem leads to encouraging computational results.

Keywords:
algorithms; assignment; heuristics; integer programming; side constraints

1. Introduction

For a flexible manufacturing system tools will frequently be located in a carousel as shown in Figure 1. The carousel has a given number of pockets, of equal area, located around its perimeter into which tools are to be located. We now make three assumptions:

- the pockets are all of unit area and the tools are of various (integral) areas;
- it is forbidden to locate any tools in certain pockets (denoted in Figure 1 by hatched areas) for technical reasons;
- the total area of the set of tools to be located is no greater than the number of available pockets.

An adjacency rating is known for each pair of tools, representing the desirability that the pair be located in adjacent pockets in the carousel. The problem we will consider is:

> assign all tools to the pockets (with each tool occupying a continuous sequence of pockets equal to its area) so that the sum of the adjacency ratings of tools is a maximum.

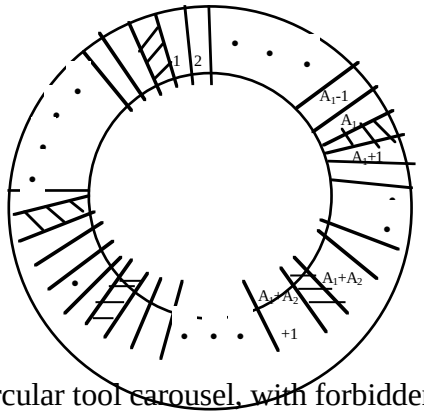
Other models of tool carousel design for flexible manufacturing systems have been discussed by Wilson [9] and Foulds and Wilson [4].

It should be noted that if there are no forbidden zones in the problem it reduces to the travelling salesman problem (TSP) whereas if it has exactly one forbidden zone it reduces to the TSP path problem (i.e. one where it is not necessary to return to the starting city). Hence the problem is NP-hard (see for instance Garey and Johnson [5]).

The problem to be discussed in this paper can be formulated as an assignment problem with a particular set of side constraints. The contribution of this paper is: to show that the problem is a special case of the restricted facilities layout problem with forbidden location zones and to report on computational experience in solving the model by a branch and bound algorithm. In the next section we introduce a model of the tool carousel design problem.

2. A Model

Consider the assignment of tools to the pockets of a flexible manufacturing system of the type shown in Figure 1.



1. The pockets of a circular tool carousel, with forbidden zones hatched.

As can be seen from Figure 1 there are m sequences of adjacent pockets, each separated in turn by a forbidden zone. The pth such sequence is called the pth *slot* and comprises A_p pockets. Let the forbidden zones be numbered 1,2, ... ,m. We assume there are n tools to be located in the slots and that the ith tool must occupy a continuous sequence of a_i pockets in exactly one slot, i = 1, 2, ..., n. We shall also assume that there exists a feasible allocation of tools to slots.

We further assume that the problem to be described is *balanced* in the sense that

$$\sum_{i=1}^{n} a_i = \sum_{p=1}^{m} A_p \ . \qquad (2.1)$$

If the right hand side of equation (2.1) is strictly greater than the left hand side then we introduce dummy tools each requiring one slot to fill empty slots.

There is an *adjacency rating* r_{ij} (i = 1, 2, ..., n; j = i+1, i+2, ..., n). It is considered to be
a measure of the suitability of locating tools i and j adjacently in the same slot in such a manner that one of the pockets in which tool i is located and one of the pockets in which tool j is located are consecutive in the pocket numbering. The pocket with the lower number (and hence the tool located in it) will be described as *immediately to the left* of the higher numbered pocket. It will be advantageous if a certain tool is immediately to the left of another one when it was the previous one to be used in the production cycle. The forbidden zones will sufficiently break up the advantages of adjacency that a tool immediately to the left of a forbidden zone and a tool immediately to the right of the same forbidden zone will have zero adjacency advantage. In addition, it can be assumed that pockets are numbered so that a forbidden zone lies to the immediate left of pocket 1.

The higher the value of the r_{ij} rating, the more desirable it is considered to locate tools i and j adjacently. (The adjacency ratings defined between dummy tools n+1, n+2, ..., if they exist, and all other tools are defined to be zero, but in what follows we assume that there are no such tools. The objective is to locate all the n tools in the slots so as to maximise the sum of the adjacency ratings of pairs of tools which are located in adjacent pockets. To this end we define the decision variable x_{ij} such that:

$$x_{ij} = \begin{cases} 1, \text{ if tool i is located immediately to the left of tool j, and} \\ 0, \text{ otherwise,} \end{cases}$$

$$(\quad i = 1, 2, ..., n; \ \ j = 1, 2, ..., n; \ \ i \neq j)$$

Then the objective is to

$$\text{Maximise} \quad Z(X) = \sum_{i=1}^{n} \sum_{j=1}^{n} r_{ij} \, x_{ij} \qquad (2.2)$$

Objective (2.2) is constrained by: each tool can be immediately to the left of at most one other tool, i.e.

$$\sum_{j=1}^{n} x_{ij} \leq 1, \quad i = 1,2,\ldots,n. \tag{2.3}$$

Each tool can have at most one other tool to its immediate left, i.e.

$$\sum_{i=1}^{n} x_{ij} \leq 1, \quad j = 1,2,\ldots,n. \tag{2.4}$$

Two further conditions are also required.

(a) Because we assume the problem is balanced, to each slot p, say, must be allocated tools whose total number of pockets is A_p.

(b) If tool i is immediately to the left of tool j and tool i is in slot p, then tool j must be in slot p and if tool i is immediately to the left of tool j and tool i is not in slot p, then tool j cannot be in slot p.

The modelling of (a), and (b), will give rise to further linear constraints and binary decision variables, which will not be presented as they will be relaxed in a subsequent formulation and not required in explicit form.

Thus the problem is to maximise objective (2.2) subject to constraints (2.3), (2.4), and conditions (a), and (b). We term this *Problem 1*. On referring to Figure 1 it can be seen that this problem is a facilities layout problem in which each tool represents a facility and the set of pockets represents the area in which they are to be laid out. Recent work on facilities layout models and techniques has been surveyed by Heregu [7] and by Foulds [2]. The problem at hand represents a special case of those just referenced in that it is, in essence, one dimensional, and involves the restriction that the placement of facilities in certain zones is forbidden. Such restricted facilities layout problems have been analysed by Foulds and Hamacher [3], and Hamacher and Nickel [6]. We now go on to devise a branch and bound algorithm for Problem 1. It should be noted that, if the constraints of Problem 1 were to be formulated as an integer programming model, several thousand constraints would be required even if n takes a value as small as 10. Such a model would be hard to solve in reasonable computational time.

3. A Branch and Bound Algorithm

Problem 1 can be transformed into an assignment problem with side constraints, and we show how to devise a branch and bound algorithm for it. We now augment Problem 1 with additional variables and provide values for certain r_{ij} ratings. Further, to associate with tools placed in the last (highest numbered) pocket of a slot, we introduce a dummy tool into each forbidden zone, and we introduce additional variables

x_{ij}, i = 1, 2, ..., n; j = n + 1, ..., n + m and i = n + 1, ..., n + m; j = 1, 2, ..., n

such that

$x_{i,n+k}$ = 1, (1 ≤ k ≤ m) if tool i is to the immediate left of forbidden zone k,
= 0, otherwise,

and

$x_{n+k,j}$ = 1 (1 ≤ k ≤ m) if forbidden zone k is to the immediate left of tool j,
= 0, otherwise.

We further define

r_{ii} = -M, i = n + 1, ..., n + m,
r_{ij} = 0, i = 1, 2, ..., n; j = n + 1, ..., n + m
and i = n + 1, ..., n + m; j = 1, 2, ..., n,
and r_{ij} = -M, i = n + 1, ..., n + m; j = n + 1, ..., n + m,
where M is an arbitrarily large number.

The additional x variables are introduced into (2.2), (2.3), and (2.4), but are not involved in conditions (a), or (b) of Section 2 and cause the generation of m additional constraints in each of (2.3) and (2.4). By removing constraints (a), and (b) and replacing the inequalities in (2.3) and (2.4) by equations, Problem 1 becomes:

Problem 2

$$\text{Maximize} \quad Z(X) = \sum_{i=1}^{n+m}\sum_{j=1}^{n+m} r_{ij}\, x_{ij} \tag{3.1}$$

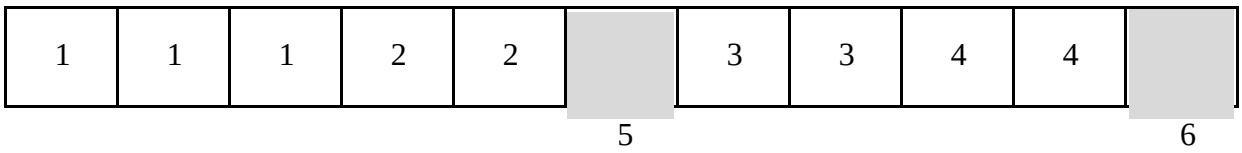
subject to

$$\sum_{i=1}^{n+m} x_{ij} = 1,\ j = 1, 2, ..., n + m, \quad (3.2)$$

$$\sum_{j=1}^{n+m} x_{ij} = 1,\ i = 1, 2, ..., n + m, \quad (3.3)$$

$$x_{ij} = 0 \text{ or } 1,\ i = 1, 2, ..., n + m;\ j = 1, 2, ..., n + m;\ (i \neq j) \quad (3.4)$$

This is the classical Assignment Problem which can be solved efficiently by the Hungarian method, first devised by Kuhn [8]. Solutions to this relaxed problem act as bounds in the branch and bound algorithm to be explained. These bounds are calculated as follows. Figures 2 and 3 illustrate a simple tool problem and its solution, denoted by X'.



2. A 2 slot 4 tool problem with A_1=5, A_2=4, a_1=2, a_2=3, a_3=2, a_4=2, shown as a strip.

	j	1	2	3	4	5	6
i							
1			*				
2						*	
3					*		
4							*
5				*			
6		*					

3. Non-zero occurrences of X_{ij} for 2 slot 4 tool problem.

Consider a solution X'. If $x'_{ij} = 1$ in X', then we say that tools i and j are in the same *adjacency set* ($1 \leq i, j \leq n$) and hence must both be assigned to the same slot. A tool i ($1 \leq i \leq n$) for which $x'_{ij} = 0$ for j = 1, 2, ..., n and $x'_{ij} = 1$ for some $j \in \{n + 1, ..., n + m\}$ will not be adjacent to any other tools and will be assigned to a single adjacency set comprising itself, and will be the only tool assigned to a particular slot. There will be at least m adjacency sets in any solution X'; exactly m are required of a solution to Problem 1. This notion provides a partitioning procedure for a branch and bound procedure. Suppose X' produces the adjacency sets: S_p, p = 1, 2, ..., m, say. The number of pockets, N_p, required in total by the tools of each S_p can be calculated as:

$$N_p = \sum_{i \in S_p} a_i \quad , p = 1, 2, ..., m. \tag{3.5}$$

If the N_p's can be renumbered as say $N'_1, N'_2, ..., N'_m$, so that

$$N'_p = A_p \quad , p = 1, 2, ..., m, \tag{3.6}$$

then a feasible solution to Problem 1 has been identified, as condition (a) of Section 2 will be satisfied. This is because of (3.6) and the fact that the satisfaction of constraints (3.2) and (3.3) by X' imply that constraints (2.3) and (2.4) will be satisfied, respectively. It will be convenient to assume, without loss of generality,

$$N'_p \geq N'_{p+1} \text{ and } A_p \geq A_{p+1} \quad p = 1, 2, ..., m - 1.$$

Thus if (3.6) does not hold there must exist at least one adjacency set, the qth say, for which

$$N'_q > A_p \text{ for } p = q + 1, ..., m. \tag{3.7}$$

Suppose that the N'_q identified corresponds to the values:

$x'_{i_1 i_2} = x'_{i_2 i_3} = ... = x'_{i_{b-1} i_b} = 1$, all other $x'_{ij} = 0$, in N'_q.

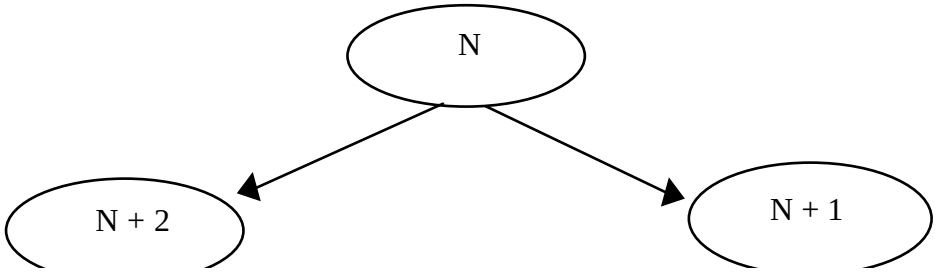
That is, tools $i_1, i_2, ..., i_{b-1}, i_b$ are to be allocated to a slot, in that order, with i_1 located first, with i_2 adjacent to it, i_3 next, and i_b located last. Unfortunately the total number of pockets required by this set of tools is greater than that for any slot. Thus it is impossible for this set of tools to compromise an adjacency set. This leads to a valid partition of the set of feasible solutions to Problem 1 represented by the node N, say, in the branch and bound decision tree represented by X' as follows. A branch is made from node N, where the variables in Problem 2 are at the values given by X', but with the following variable set:

node N + 2: $x_{i_s i_{s+1}} = 0$, where s is chosen such that

$$r_{i_s i_{s+1}} = \min(r_{i_t i_{t+1}} : t = 1, 2, ..., b - 1),$$

provided $x_{i_s i_{s+1}}$ has not already been fixed at node N or one of its predecessors. Node N + 1 has $x_{i_s i_{s+1}}$ set to 1. Nodes with higher node numbers will be developed before nodes with lower node numbers.

This partitioning procedure is illustrated in Figure 4.



$x_{i_s\ i_{s+1}} = 0$ $\qquad\qquad$ $x_{i_s\ i_{s+1}} = 1$

4. Branching on variable $x_{i_s\ i_{s+1}}$ from node N

If there are more than m adjacency sets, those that contain a subset of tools, none of which are immediately to the left of any forbidden zone, are identified. Each such set is identified as N'_q in turn, and the index s is chosen which corresponds to the minimal adjacency value over all such sets. The partitioning then proceeds as set out above, but two further elaborations are introduced.

First

At any branch of the tree it may be true that for some values of p = 1, 2, ..., m, $N'_p = A_p$, thus partially satisfying (3.6). Then as soon as the step that involves (3.7) is used, there is the possibility that the set of values of p for which $N'_p = A_p$ will change. For this reason a branch-and-fix phase has been introduced into the algorithm to fix a subset of tools in a slot at some node.

If it turns out that particular "fixes" generate descendants of a node leading to infeasible or bounded solutions, then the corresponding "unfixed" part of the branch and bound tree remains unexplored and available to possibly yield a feasible or optimal solution to the complete problem.

Second

It was found useful to introduce the following fathoming test to detect when "fixing" was likely to become too restrictive on further development of the branch and bound tree. At each node a record is kept of any region A_p which is being "fixed" at that node. Thus corresponding to each node there will be a set of regions A_p which are not fixed. Similarly, because of the fixing of x variables at nodes, there will be a set of tools at each node which have not been allocated (fixed). The test is:

> at node N, if the number of unallocated tools which require an odd number of pockets is smaller than the number of unfixed slots which contain an odd number of pockets, then node N is fathomed.

4. Example

Consider the following example with m=3, n=8 and A_1=6, A_2=5, A_3=5.

a = (3 2 4 1 1 3 1 1).

r =

-M	17	-M	28	23	17	66	38
*	-M	49	94	30	40	10	33
*	*	-M	49	31	-M	67	23
*	*	*	-M	15	69	84	32
*	*	*	*	-M	27	30	55
*	*	*	*	*	-M	14	99
*	*	*	*	*	*	-M	94
*	*	*	*	*	*	*	-M

The following is the sequence of nodes explored.

N	P	slt1 (size)	slt2 (size)	slt3 (size)	slt4 (size)	slt5 (size)	soln	action	decision
0	-	3,7(5)	1(3)	5(1)	6,8(4)	2,4(3)	453		fix(3,7,10)

2	0	3,7(5)	1(3)	5(1)	6,8(4)	2,4(3)	453	fixed(3,7)	
4	2	3,7(5)	1(3)	5(1)	6,8(4)	2,4(3)	453	fixed(7,10)	
6	4	3,7(5)	1(3)	5(1)	6,8(4)	2,4(3)	453	fixed (10,3)	break slt4 or slt5
8	6	3,7(5)	5,2,4(4)	1(3)	6,8(4)		389	unfixed(4,2)	break slt4
10	8	2,4,6,8,5(8)	3,7(5)	1(3)			384	unfixed(8,6)	break slt1
12	10	2,4,6,8,1(10)	3,7(5)	5(1)			367	unfixed(8,5)	break slt1
14	12	3,7(5)	1(3)	5(1)	2,4,6,8(7)		362	unfixed(8,1)	break slt4
16	14	5,2,4,6,8(8)	3,7(5)	1(3)			359	unfixed(8,2)	break slt1
18	16	2,4,6,8(7)	3,7(5)	1,5(4)			352	unfixed(5,2)	break slt1
20	18	2,4,1(6)	3,7(5)	5,6,8(5)			315	unfixed(4,6)	feasible solution

Backtracking now commences until optimal solution found.

……

108	106	2,4,6(6)	3,5(5)	1,7,8(5)	354

Then optimality is established in a total of 142 nodes.

Key:

N	=	node number
P	=	preceding node in tree
soln.	=	current solution value
unfixed	=	set to 0
fixed	=	set to 1
(size)	=	total size of tools allocated to the slot
slt1,slt2, ...,slt5	=	tools assigned to slots 1-5

It should be noted that only slots 1,2 and 3 are genuine slots and slots 4 and 5, if occupied, contain a cycle.

5. Computational Experience

The algorithm described in Section 3 was programmed in FORTRAN 77 and implemented on a Honeywell Series 9000 computer. The assignment algorithm used to solve the relaxed problem was the version of Burkard and Derigs [1]. One typical problem to be solved had the following specifications m = 10, n = 20, $\sum A_p = 70$ and was solved at node number 4437, with optimality proved at node 4729 in a total of 23.6 CPU seconds.

A set of similar problems with the dimensions m = 10, n = 30 was randomly generated as follows. Data values were chosen randomly from uniform integer distributions. Each value of r_{ij} was chosen from the distribution (10,99), each value of A_p from the distribution (5,10) and each value of a_i from the distribution (2,5). Only results for feasible problems are reported and a limit of 20000 branch and bound nodes was set. Table I summarises the results of testing 16 problems. On average, the problems required about 8000 nodes and 50 seconds of CPU time. In Table I the problems are divided into four sets according to the variability of the a_i or A_p values. L indicates problems for which the standard deviation, S1, of the a_i values (respectively, S2 for A_p values) was on or below the median of the problems tested and H indicates problems for which S1 (S2 respectively) was above the median. From Table I, it appears that problems with less variability in a_i values converge more rapidly to a solution than those problems with higher variability.

Table I Performance of test problems

		nodes to optimal solution		nodes to completion		CPU seconds	
S1	S2	lowest	highest	lowest	highest	lowest	highest
L	L	49	13109	87	13747	0.4	78.5
L	H	49	13781	93	14963	0.4	83.9
H	L	8309	12045	12999	18107	80.7	103.2
H	H	2923	14297	3299	17869	13.7	105.3

7. Summary and Conclusions

We have described a problem in the design of tool carousels for flexible manufacturing systems and modelled it as a zero-one integer programming model. We showed how the model can be transformed into an assignment problem subject to a particular set of side constraints. We detailed a branch and bound algorithm for this problem. The bounds for this problem are calculated by relaxing the side constraints and solving the resulting assignment problem by the Hungarian method. Partitioning in the algorithm is carried out in a manner akin to subtour elimination for the travelling salesman problem. The algorithm is capable of solving problems of small to moderate size in reasonable computing time.

References

[1] Burkhard, R. E., and U. Derigs, *Assignment and matching problems: solution methods with Fortran programs*, in: Lecture Notes in Economics and Mathematical Systems 184, Springer, Berlin (1980)

[2] Foulds, L. R., *Graph theoretic-based decision support for facilities layout*, IFIP Transactions B: Applications in Technology B11, (1993), pp145-158

[3] Foulds, L.R. and H. Hamacher, *A new integer programming approach to (restricted) facilities layout problems allowing flexible facility shapes*, Research Report 1992-3, Department of Management Systems, University of Waikato, New Zealand (1992)

[4] Foulds, L. R. and J. M. Wilson, *Formulation and solution of problems of tool positioning on a single machine centre*, International Journal of Production Research 31, (1993), pp2479-2485

[5] Garey, M. R. and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco (1979)

[6] Hamacher, H. and S. Nickel, *Restricted Planar Location Problems and Applications*, Research Report 1992-18, Department of Management Systems, University of Waikato, New Zealand (1992)

[7] Heregu, S. S., *Recent models and techniques for solving the layout problem*, European Journal Operational Research 57, (1992), pp136-144

[8] Kuhn, H. W., *The Hungarian method for the assignment problem*, Naval Research Logistics Quarterly 2, (1956), pp83-97

[9] Wilson, J. M., *Formulation and solution of a set of sequencing problems for FMS*, Transactions of the Institution of Mechanical Engineers 20, (1987), pp247-249