A FAST, OPTIMAL PROCEDURE FOR THE WORK STATION ASSIGNMENT PROBLEM

Candace Arai Yano
Department of Industrial & Operations Engineering
University of Michigan
Ann Arbor, Michigan 48109

Technical Report 85-38

August 1985
Revised September 1986

A FAST, OPTIMAL PROCEDURE FOR THE WORK STATION ASSIGNMENT PROBLEM

Candace Arai Yano
Department of Industrial and Operations Engineering
University of Michigan
Ann Arbor, Michigan 48109-2117

August 1985
Revised September 1986

A FAST, OPTIMAL PROCEDURE FOR THE WORK STATION ASSIGNMENT PROBLEM

ABSTRACT

The problem of assigning clerks to work stations arises in the context of mail order sales operations, airline reservation facilities, utility service offices and other similar operations. The use of large numbers of part-time employees and long hours of operation can make the assignment problem difficult to solve optimally using manual procedures. Minimization of work stations also provides significant savings over a "one work station per person" policy under these conditions.

This paper describes an efficient procedure based upon graph coloring which is shown to be optimal. We also present computational results which compare this procedure with an known procedure. The results indicate that the new procedure permits much larger problems to be solved optimally, even on a personal computer.

1. Introduction

Many service and direct marketing operations require staff answering calls to be seated at work stations equipped with a telephone and possibly a video display terminal for reference and/or for input. If the staff work full-time shifts and the number of hours of operation is less than two full-time shifts, each person must be assigned his or her own work station in order to avoid conflicting assignments. If, however, a large percentage of the work force is employed part-time, or if the hours of operation are long, several people may be assigned to a given seating location without a schedule conflict. Economies may be obtained by minimizing the number of positions required. Even during periods where the facility is not fully utilized, operating costs such as for heat and light can be reduced by clustering the workers as much as possible.

This work was motivated by a facility of a major communications firm where the work station assignment problem needed to be addressed.

The solution to the assignment problem can be used in conjunction with any standard workforce scheduling system. It is assumed here that each person has been assigned work tours for a given day or week (i.e., the workforce scheduling problem has been solved). By work tour we mean a continuous time interval between the moment a worker begins work and the time at which he leaves. It is sometimes referred to as a "shift," except that a shift usually connotes a work tour of specified length (e.g., 8 hours plus a lunch break).

These work tours which are the output of a workforce scheduling system may not satisfy a constraint on number of work stations, since such constraints are not always incorporated into the workforce scheduling algorithm. For example, a network flow approach to the problem developed by Segal (1974) permits constraints on the number of people working identical work tours, but cannot incorporate constraints on the total number of people working at any given time.

Constraints on total number of people also are ignored where the workforce scheduling problem is solved in a decentralized fashion (e.g., where each supervisor determines the schedule for a group manually). Of course, infeasible schedules may result, but in reality, these infeasibilities are resolved *ex post*. For instance, some absenteeism usually occurs, reducing the need for work stations, and facilities normally used for other purposes (such as training) can be appropriated in an emergency. Thus, from a practical perspective, it is sufficient to solve the workforce scheduling problem without constraints on the total number of people working at any point in time, and then to solve the work station assignment problem with the objective of minimizing the number of work stations needed.

In Section 2 we formulate the work station assignment problem. Section 3 describes an optimal solution procedure which has computational complexity $O(n^2)$. Section 4 describes a known procedure with computational complexity $O(n^4)$. An example problem is solved using both approaches in Section 5, computational results are presented in Section 6, and we conclude with a summary in Section 7.

2. Problem Formulation

We make the following assumptions:

(1) The work week can be divided into distinct days in such a way that there is no feasible work tour which spans parts of two consecutive days, and such that clerks can work only one tour a day.

(2) The work day can be divided into an integral number of time increments of equal length in such a way that all work tours begin and end at a multiple of this time increment.

The first condition assures separability of the problem by work day. In so doing, it limits the size of the problem to the number of individuals working on a given day. This condition is usually satisfied in existing operations. The second condition is not necessary for the solution of the seating assignment problem but is assumed in most workforce scheduling algorithms and is almost always satisfied in manual scheduling systems.

The objective of this problem is to assign people to positions in such a way that the minimum number of positions is used.

Let

$$x_{ipt} = \begin{cases} 1 \text{ if person } i \text{ occupies position } p \text{ and works during time period } t \\ 0 \text{ otherwise} \end{cases}$$

$$y_{ip} = \begin{cases} 1 \text{ if person } i \text{ is assigned to position } p \\ 0 \text{ otherwise} \end{cases}$$

The objective is to minimize the number of positions required or

$$\min \max_t \sum_p \sum_i x_{ipt}$$

Obtaining the value of the objective function is trivial: it is equal to the maximum number of people working at any point in time. The important decisions involve assignment of people to these positions.

There are two basic requirements. First, each position can be occupied by only one person at a given time. This can be expressed as the constraint

$$\sum_i x_{ipt} \leq 1 \quad \forall\, p, t$$

The second requirement is that each person be assigned to exactly one position. This will enable each person to remain at the same position for his or her entire work tour on a given day. Changing positions in the middle of a work tour is undesirable both administratively and from the perspective of maintaining productivity. The requirement is expressed through a combination of constraints:

$$\sum_p y_{ip} = 1, \forall i$$

$$(y_{ip} - 1) \sum_t x_{ipt} = 0, \forall i, p$$

and $$\sum_p \sum_t x_{ipt} = \text{tour length for person } i, \forall i$$

The first constraint says that each person is assigned to exactly one position. The second constraint will not permit a person to be assigned to position p in any period t unless he is assigned to position p (i.e., $y_{ip} = 1$). Finally, the third constraint says that each person must be assigned to a position in each period of his work tour. These constraints, when combined with the one above, ensure that all necessary assignments are made and that two people are not assigned to the same position if their work tours overlap.

Some values may be prespecified as follows:

If person i does not work in time period t, set

$$x_{ipt} = 0, \forall p$$

If person i must be assigned to position p_i, set

$$x_{ipt} = 0, \forall t, p \neq p_i$$

and $x_{ipt} = 1$, $t \in$ {work tour for person i}, $p = p_i$

This formulation yields a large nonlinear integer programming problem. In this form the problem is computationally intractable for most real problems.

The nonlinear constraints and the binary nature of the decision variables make this problem extremely difficult to solve using what would appear to be applicable techniques. For instance, the problem is a constrained two-dimensional bin-packing problem. The height of the bin is the number of periods in a day, and the width corresponds to the number of positions. The "cartons" to be packed are one unit wide, and for each person, the height is the number of periods in the work tour. One special constraint is that each carton must be placed in the correct horizontal (time) location in the bin.

Unfortunately, the general two-dimensional bin-packing problem is np-complete (Fowler, et al., 1981). The problem, however, can be formulated as a graph coloring problem and can be solved in this manner. In the graphical representation of the problem, each node corresponds to a continuous interval of time on a real time axis. The nodes are connected if the work tours overlap, i.e., if there is a non-empty intersection. The problem then is to color the nodes with the smallest number of colors in such a way that adjacent (connected) nodes do not have the same color. All persons associated with a node of a particular color can be assigned to the same work station.

We next describe a new optimal algorithm for the problem and show that the worst case computational complexity is $O(n^2)$. In the subsequent section we briefly describe a known procedure which has computational complexity $O(n^4)$. Actual computation times using both procedures are compared in section 6.

3. A Fast Optimal Procedure

The new procedure is quite simple. Sequence the workers in order of work tour start time, breaking ties arbitrarily. Starting with the workers having earliest start times, assign each to the lowest indexed position available. This sequential coloring algorithm takes advantage of the nature of the application and the fact that the representation is an interval graph to determine the sequence in which the nodes are colored. Sequential algorithms, usually based upon node degree, have been shown to provide good results in more general graphs (see Matula et al. (1972) and Matula and Beck (1983)). We next present a proof of the optimality of this procedure for non-cyclic (i.e., linear) interval graphs.

THEOREM: A minimum coloring of a linear interval graph can be obtained by sequencing nodes in non-decreasing order of the minimum value of the corresponding interval and sequentially coloring the nodes with the lowest indexed feasible color. (A feasible color is one which differs from the color of all adjacent nodes which are already colored).

PROOF:

Let s_i be the minimum value of the ith interval (start time for person i) and e_i be the maximum value of the ith interval (ending time for person i). Also let [j] denote the node in the jth position in a sequence. We will show that a sequence not satisfying $s_{[1]} \leq s_{[2]} \leq \cdots \leq s_{[N]}$ cannot be guaranteed to provide a minimum coloring and that the sequential coloring algorithm described above only uses an additional color when necessary. These two facts jointly ensure that the procedure provides a minimum coloring.

We first show that any sequence not satisfying $s_{[1]} \leq s_{[2]} \leq \cdots \leq s_{[N]}$ cannot be guaranteed to give a minimum coloring. Suppose that $s_j > s_i$ and we decide to color node j before node i with some color k which has already been used and which is feasible for both nodes. If $e_i \leq s_j$, the number of alternative colors which we can use for node i is unaffected by the coloring of node j. (Node i can still be colored with color k, or with any other feasible color which has already been used. In either case, no additional color is used unnecessarily). On the other hand, suppose $e_i > s_j$ and k is the only color which has already been used and is feasible for node i. Then assigning color k to node j may force us to use an additional color for node i unnecessarily. Such a situation arises if some other color k' (which has already been used) becomes available in $[s_i, s_j]$. In this case, assigning node i to color k first and then node j to color k' would reduce the number of colors required. Thus, assigning colors to nodes in any sequence which is <u>not</u> non-decreasing in $s_{[i]}$ cannot be guaranteed to be optimal.

We next show that the algorithm only uses an additional color when necessary, and thus provides a minimum coloring. Here we assume that the node sequence is as specified above. Suppose n nodes are colored so far and that C colors have been used already. Node [n+1] can be colored with color $k \leq C$ only if

$$\max \{e_{[j]} \mid j \leq n, [j] \text{ has color } k\} \leq s_{n+1} \qquad (1)$$

That is, node n+1 can be colored with color k if all nodes already having that color have e_i values less than or equal to s_{n+1} (node n+1 is not connected to any node already colored with k). Selection of any such color k will result in the same availability (in terms of quantity) of already used colors for all possible subsequent assignments. The algorithm simply selects the smallest such value of k.

Stated another way, if there were m > 0 already used colors which could have been used for node [n], m-1 of those colors are still available to be used for node [n+1], irrespective of the specific color assignment for node [n]. Therefore, any rearrangement of color assignments for nodes [1],...,[n] cannot increase the availability of already used colors for node [n+1] without increasing the number of colors used (which, of course, is not desirable). If there is no color k for which (1) holds, the minimum coloring requires at least C+1 colors. Thus, the algorithm uses another color only when necessary. ■

Conceptually, the algorithm finds minimum colorings for the series of subgraphs induced by the given sequence, where at each step the next node in the sequence is added (along with associated arcs). When the nodes are added in the specified order, an optimal solution procedure exists in which one only needs to consider the best thing to do with the current node. The preferred action is to color it with a color which has already been used, and a new color is used only when this cannot be done. It is not necessary to consider changing previous color assignments, since any such rearrangment cannot increase the availability of already used colors.

The worst case computational complexity of the algorithm is $O(n^2)$. The sequencing operation requires a sort which has computational complexity $O(n \log n)$. The coloring procedure requires that for each node, the color of all adjacent nodes already colored be checked. This has a worst case computational complexity of $O(n^2)$. Thus, the entire algorithm has computational complexity $O(n^2)$.

4. A Known Graph Coloring Approach

In this section we describe a known graph coloring approach which can be used to solve the problem. The graph coloring technique is not new, but the work station assignment problem is a new application of the technique.

Using results of Hajos (1958), Berge (1960), and Golumbic (1980), it can be shown that interval graphs are triangulated, that triangulated graphs are perfect graphs, and that perfect graphs have a perfect vertex elimination scheme. Gavril (1972) shows that a perfect vertex elimination scheme for a trianglated graph has worst case computational complexity

$$nd(n+1)(d-1)/4.$$

where d is the maximum degree and n is the number of vertices in the graph. Clearly $d \leq n-1$ so $nd(n+1)(d-1)/4 < n^4/4$, so the computational complexity in the worst case is $O(n^4)$.

The first step in solving this problem using this approach involves transforming output from a workforce scheduling algorithm into a graphical representation of the seating assignment problem. This involves fairly complex data management but is straightforward conceptually. We will not dwell on this process here.

Once a graphical representation has been constructed, the problem can be solved by first constructing a perfect vertex elimination scheme. Select a node ν

such that Adj(ν) (i.e., set of all adjacent nodes) is completely connected and eliminate it from the graph. (Such a node is called a simplicial). Continue this process until only one node remains. The nodes are colored in the reverse order of the perfect elimination scheme.

A minimum coloring can now be obtained by coloring the relabeled nodes sequentially (node with smallest label first), using the first feasible color. Colors are sequenced by the order in which each is first used. A new color is used only when colors already utilized are infeasible. This minimum coloring algorithm is due to Gavril (1972).

Each set of nodes with the same color represents the set of people assigned to one specific position. The relabeled nodes must be translated into their original designations in order to make appropriate assignments.

Before presenting computational results, we illustrate both procedures by way of an example.

5. Example

Assume that on a given work day, the following work tours have been assigned:

Person	Work tour (including lunch and breaks)
A	0800-1200
B	0800-1700
C	0900-1200
D	0900-1500
E	1000-1800
F	1200-1600
G	1300-1700
H	1500-1800

The graphical representation appears in Figure 1. Note that there is one node representing each person. Nodes are connected whenever the corresponding people cannot be assigned to the same work station because of overlapping work tour assignments. Therefore, node A is connected to nodes B, C, D, and E, but not to F, G, and H.

One possible perfect elimination scheme is {H,G,F,A,B,C,D,E}. Returning nodes in reverse order, the nodes are relabeled, as illustrated in Figure 2.

The nodes can be colored as follows:

Step 1.	Node 1 colored with color #1
Step 2.	Node 2 colored with color #2
Step 3.	Node 3 colored with color #3
Step 4.	Node 4 colored with color #4
Step 5.	Node 5 colored with color #5
Step 6.	Node 6 can be colored with any color except #1, #2, and #4. The first feasible color is #3; hence node 6 is colored with color #3.
Step 7.	Node 7 can be colored with any color except #1, #2, #3 and #4. Therefore, node 7 is colored with color #5.
Step 8.	Node 8 can be colored with any color except #1, #3, #4 and #5. Therefore, node 8 is colored with color #2.

Using this information, the following positions are assigned:

Position	Node numbers	People
1	1	E
2	2,8	D,H
3	3,6	C,F
4	4	B
5	5,7	A,G

Using the new procedure, we need to sequence the workers by their respective start times. One sequence which satisfies this requirement is alphabetical order (A,B,C,D,E,F,G,H). Assigning the workers to the first available position (unoccupied position with lowest index), we arrive at the following assignments:

Position	People
1	A,F
2	B
3	C,G
4	D,H
5	E

Persons A through E must be assigned to separate work stations. Person F is assigned to the first available position, which is position 1. Then, person G is assigned to the first available position, which is position 3. Finally, person H is assigned to the first available work station (position 4).

6. Computational Results

There are two objectives of our computational study. First, we wanted to determine whether problems of reasonable size can be solved on a microcomputer. Second, we wanted to compare the two procedures in terms of computation times since we only know the worst case computational complexity.

We randomly generated 50 different work schedules each for 20, 50, and 80 person situations. We assumed that the facility operates 16 hours per day (which is typical in real applications). For 25 of the work schedules, work tour lengths were randomly selected from 4 to 9 hours (integer values only), and for the others, the work tour lengths could vary from 3 to 6 hours (again, integers only). The start of each work tour was randomly generated from among feasible start times for that tour length. For instance, a work tour six hours in length could start from opening of the facility until ten hours hence. The programs were coded in FORTRAN77 and run on an IBM PC/XT.

Means and standard deviations of the solution times using the $O(n^4)$ graph coloring procedure for each set of 25 problems are reported in Table 1. It is evident that solution times increase rapidly with the number of persons, but it is still feasible to determine solutions for problems with 80 persons in less than 30 minutes using this approach.

TABLE 1

The computation times also differed considerably with the average tour length. One reason for this is that the problems with shorter tour lengths tend to be much less connected (lower average node degree), therefore requiring fewer positions than problems with longer tour lengths. This results in a more complex combinatorial problem (more alternatives to be considered). From a graph theoretic viewpoint, constructing a perfect elimination scheme takes longer in graphs with relatively low average node degree since it is more difficult to find a simplicial vertex.

The mean and standard deviation of computation times using the new procedure are reported in Table 2 for each set of 25 problems. The new procedure required only a fraction of the time required for the existing procedure. Moreover, the computing times are affected only insignificantly by the average node degree. Thus, it appears that this simple procedure, which can be implemented manually, can provide considerable improvements in computing time.

TABLE 2

7. Summary and Discussion

This paper details an existing graph coloring procedure and a much faster new procedure for a typical seating assignment problem with many potential applications. Computational experience indicates that the new procedure can reduce computation time significantly. It appears that taking advantage of the interval nature of the graph and information about the intervals can provide significant computational advantages.

We note briefly that the graphs used in the computational study fall into the class of random graphs, where the nodes are specified in advance, but the node incidences are random. In this application, the interval nature of the graph gives it more structure than a general random graph in which only node incidence probabilities are specified (see, for example, Johri and Matula (1982)). The results here may have some important implications for random interval graphs, but such an investigation extends beyond the scope of this paper and remains an area for further research.

Another problem for further research which is more closely related to our application is to characterize the distribution of work stations needed for a fixed number of workers with random work tours. Such a characterization would be useful in partitioning the work force and work stations into supervisory groups, and in capacity planning for these facilities.

Acknowledgements

The author is grateful to Professors Katta Murty and David Matula for discussions on this topic and to two anonymous referees for their comments on an earlier version of this paper.

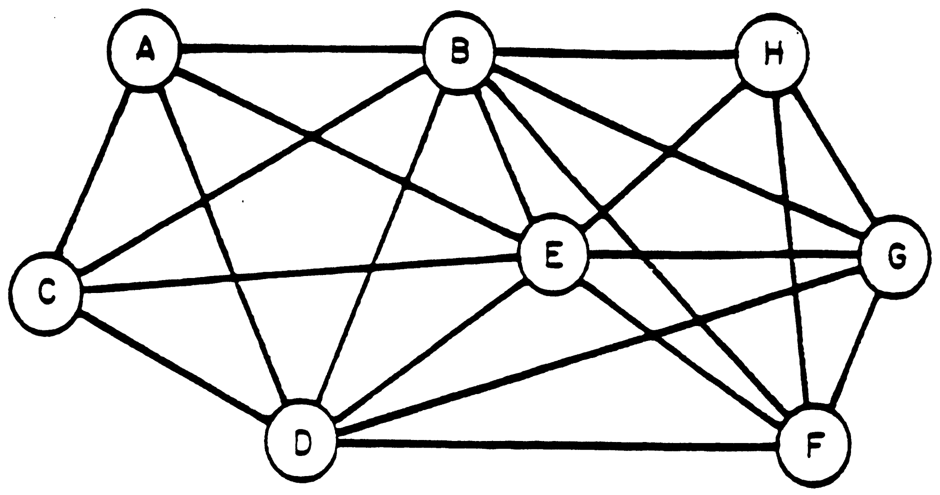

FIGURE 1

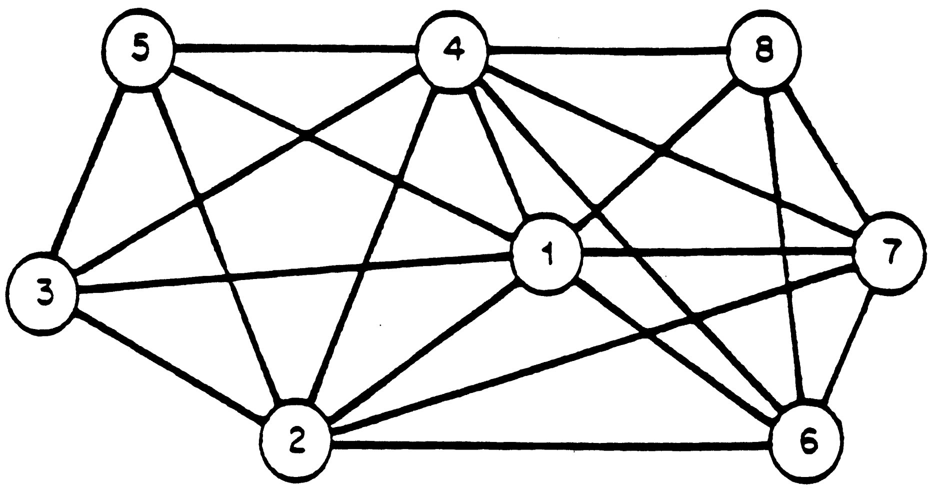

FIGURE 2

TABLE 1

Computation Time Statistics for $O(n^4)$ Procedure
(in seconds)

$\underline{n}$	Work Tour Length Range (hours) $\underline{3 - 6}$		$\underline{4 - 9}$	
	$\underline{x}$	$\underline{s}$	$\underline{x}$	$\underline{s}$
20	4.62	1.29	1.40	0.37
50	178.52	31.51	194.50	34.87
80	1184.59	228.75	352.14	54.72

TABLE 2

Computation Time Statistics for the New Procedure
(in seconds)

	Work Tour Length Range (hours)			
n	3 - 6		4 - 9	
	x	s	x	s
20	0.19	0.029	0.20	0.028
50	1.08	0.033	1.17	0.045
80	2.70	0.038	2.93	0.051

REFERENCES

Berge, Claude (1960). "Les problemes de colorations en theories des graphs." Publ. Inst. Statist. Univ. Paris 9, 123-160.

Fowler, R.J., M.S., Paterson, and S.L. Tanimoto (1981). "Optimal Packing and Covering in the Plane are NP-Complete", Info. Process. Letters 12, 133-137.

Gavril, Fanica (1972). "Algorithms for minimum coloring, maximum clique, minimum coloring by cliques and maximum independent set of a chordal graph." SIAM J. Comput. 1, 180-187.

Golumbic, Martin C. Algorithmic Graph Theory and Perfect Graphs, New York: Academic Press, 1980.

Hajos, G. (1958). "Uber eine Art von Graphen," Internat. Math. Nachr., p.65.

Johri, A. and D.W. Matula (1982), "Probabilistic Bounds and Heuristic Algorithms for Coloring Large Random Graphs," Technical Report 82-CSE-6, Department of Computer Science and Engineering, Southern Methodist University.

Matula, D. and L.L. Beck (1983), "Smallest-Last Ordering and Clustering and Graph Coloring Algorithms," J.A.C.M. 30(3), 417-427.

Matula, D., G. Marble, and J.D. Isaacson, "Graph Coloring Algorithms," in Graph Theory and Computing (R. Read, ed.), New York: Academic Press, 1972.

Segal, M. (1974), "The Operator-Scheduling Problem: A Network Flow Approach," Operations Research 22(4), 808-823.

Captions for Figures:

Figure 1: Graphical Representation of Workers and Seating Conflicts

Figure 2: Perfect Elimination Scheme