Division of Research
Graduate School of Business Administration
The University of Michigan

March 1985

A COMPARATIVE EVALUATION OF HEURISTIC LINE BALANCING TECHNIQUES

Working Paper No. 415

F. Brian Talbot
The University of Michigan
and
James H. Patterson
University of Missouri-Columbia
and
William V. Gehrlein
University of Delaware

FOR DISCUSSION PURPOSES ONLY

None of this material is to be quoted or reproduced without the expressed permission of the Division of Research.

Abstract

In this paper, we report on a computational experiment designed to assess the efficacy of twenty-six heuristic decision rules which group work tasks into work stations along an assembly line such that the number of work stations required is minimized. The heuristic decision rules investigated vary from simple list processing procedures that consider a single attribute of each work task for assignment, to procedures which are optimal seeking, but which have had their search terminated through the imposition of a limit on the amount of computation time that can be devoted to each search. Also included are heuristic decision rules which backtrack in an attempt to locate an improved solution, and decision rules which probabilistically search for improved solutions. Our investigation differs from those reported previously, in that the objective in balancing each line is to determine the minimum number of work stations for a given limit on the time available for assembly at each work station (the cycle time). Previous approaches have investigated the problem of determining the minimum cycle time for a given line length. We also compare the results obtained with the optimal solution for a subset of the problems investigated. Both problems which have appeared in the open literature and problems which have been solved for the first time are included. Since some of the results reported in this paper differ from those reported previously, we suggest why these differences have occurred. Guidelines are also given to those balancing industrial assembly lines on the choice of the heuristic decision rule to use whether one is attempting to obtain a minimum station balance given a limit on the time available for assembly at each work station, or whether one is attempting to minimize the time devoted to assembly at a work station given a limit on the number work stations available.

1. INTRODUCTION

One of the problems inherent in organizing for mass production is how to group work tasks to be performed into "work packages" or work stations so as to achieve a desired level of performance. When the number of work stations or production employees is fixed, the objective is to allocate the work to be performed to the work stations in such a way that the maximum time required for assembly at any given station (i.e., the cycle time) is minimal across all feasible station balances. The optimal solution to this problem, called the Type II problem (Mastor, 1970), maximizes the output rate or the number of units that are produced for the given line length. A Type I problem, on the other hand, is characterized by a forecast of demand or a production plan dictating the output or the production rate. The objective in this problem is to detemine the minimum number of work stations required to meet the specified production requirements. A line with fewer stations translates into lower labor costs and reduced space requirements, and hence, a more cost effective production plan. In this paper, we examine in detail the latter of these two problem types, and offer guidelines for applying our results to either one.

There have been two previous investigations of the performance of various heuristic decision rules for solving the Type II assembly line balancing (ALB) problem for the case in which heuristic performance is assessed relative to the <u>best heuristic</u> result obtained. Mastor (1966) evaluated the performance of ten heuristic decision rules on twenty and forty task assembly problems for three different order strengths (ratio of the number of ordering relations among the tasks to be grouped into work stations to the total number of possible orderings), and for line lengths (number of work stations) ranging

from four to fifteen for the forty task problems and from three to twelve for the twenty task problems. Since the majority of the line balancing heuristics investigated by Mastor were designed to solve the Type I line balancing problem, Mastor obtained minimal cycle time balances by iteratively employing each of the evaluated techniques, increasing the cycle time in one percent increments above the lower bound cycle time until a balance was achieved for the specified number of work stations. Mastor's general conclusion is that best results are obtained by the Held et al. (1963) dynamic programming technique, although the range of experimental conditions over which this procedure was evaluated was less than for most of the other procedures. (This was due to excessive computation times being required on certain categories of problems using the dynamic programming approach. The difficulty reported by Mastor in solving certain problem types using dynamic programming is consistent with our experience with this approach, as described later. Also, Magazine and Wee (1981b) observed this same result when attempting to obtain optimal solutions to the line balancing problem using a dynamic programming procedure.) The results obtained with the Held et al. dynamic programming technique were followed closely by Arcus's COMSOAL (1963) in Mastor's investigation. Further, the range of experimental conditions over which this procedure was evaluated was not curtailed as with the Held et al. dynamic programming procedure.

Dar-El (Mansoor) (1975) investigated twelve heuristic decision rules, also for Type II problems. Dar-El used fifty, one-hundred and five, and one-hundred and forty task assembly problems, varied across three task precedence configurations, and four ratios of work elements per station (WEST ratios) in his evaluation. Dar-El's general conclusion is that MALB, a technique he developed (1973), which is based upon a backtracking extension to the Ranked

Positional Weight heuristic of Helgeson and Birnie (1961), gives consistently superior results to the Arcus or to the other techniques investigated. (A procedure we have found effective, a backtracking approach due to Hoffmann (1963), was not included in Dar-El's investigation.)

In this paper, we report on a computational experiment designed to assess the efficacy of heuristic procedures for solving the Type I assembly line balancing problem. Heuristic decision rules that have been found effective elsewhere are included in this evaluation, as are some that are reported on for the first time. Four data sets are used to assess heuristic performance. The first data set consists of 120 computer generated problems each of which is solved for 11 cycle times, yielding 1,320 balances. The second data set consists of 12 problems which have appeared in the open literature. These 12 problems are solved for a variety of cycle times, yielding 64 balances in total. The third and fourth data sets each consist of 110 "difficult" balances. These latter problems have been generated in order to assess the performance of several of the heuristic decision rules evaluated under a "worst case" scenario.

We first compare the various approaches examined to the minimum heuristic result obtained for each problem. This measure provides us with a relative measure of heuristic performance, and thus is similar to previous investigations. Then, because many of the heuristic procedures evaluated are able to determine and verify optimal solutions within the specified time limit for a subset of the problems examined, we also compare heuristic performance to the known optimal solution for this subset of problems. In so doing, we obtain an estimate of absolute performance for each of the heuristic decision rules that previous investigations have not provided.

In Section 2, each of the heuristic decision rules investigated is described. Section 3 then describes the test problems used to assess heuristic performance. In Section 4 we present the computational results. Because the results reported in Section 4 are somewhat at variance with other reported results, in Section 5 we suggest why the variability in results has occurred. We also offer guidelines for applying the heuristic decision rules to either type of line balancing problem. Concluding remarks are given in Section 6.

2. HEURISTIC LINE BALANCING TECHNIQUES INVESTIGATED

In this section, each of the heuristic decision rules investigated is described. These decision rules vary from simple list processing prioritizing schemes to optimal-seeking procedures which have had the amount of time to devote to each search limited. To aid in our description of these procedures, we have placed each decision rule into one of four categories. These categories allow us to capture salient features of each approach, and to emphasize the similarities and differences among them. The first category consists of those decision rules which implement a list processing prioritizing scheme for task assignment based upon a single attribute of each assembly task. We term this category "Single-Pass Decision Rules" (Dar-El (1975)). The second category consists of decision rules which are a composite of the Single-Pass Decision Rules, or which otherwise produce multiple single-pass solutions for a given problem, selecting that solution for implementation which results in the fewest number of work stations. We include in this category the Arcus COMSOAL (1966) procedure found effective by Mastor, as well as a composite rule which selects the best of the solutions determined by our application of the single-pass decision rules included in Category I. This second category is

termed "Composite Decision Rules." Two of the procedures evaluated consist of backtracking approaches which attempt to improve upon a solution or station assignment previously obtained. These procedures are contained in a third category labeled "Backtracking Decision Rules." Procedures in this category include the Precedence Matrix, Enumeration method of Hoffmann (1963), and MALB (1973). Finally, six procedures are included which are optimizing approaches, but which have been programmed with a time limit set to restrict the time permitted for each solution. The procedures included in this category are two versions of Branch and Bound (Magazine and Wee, 1981b), two variations of Integer Programming (Talbot and Patterson, 1984), Dynamic Programming (Schrage and Baker, 1978), and MUST (Dar-El and Rubinovitch, 1979). This last category is titled, "Optimal Seeking Decision Rules."

2.1 Single-Pass Decision Rules

In this category, thirteen single-pass, single attribute, priority dispatch scheduling rules such as Maximum Ranked Positional Weight (Helgeson and Birnie, 1961), Maximum Number of Immediate Followers (Tonge, 1961), Maximum Task Time First (Moodie and Young, 1965), etc., are included. The heuristic decision rules in this set include those which have been found effective elsewhere, plus two "benchmark" heuristic decision rules. Several techniques are additionally included which have been recently developed to compete against the set of established rules.

Each of the decision rules included in this first category consists of a simple, computationally efficient, list-processing procedure that assigns tasks to work stations according to a task's computed priority. Operationally, in the implementation of each of these procedures, a task is first assigned a numerical priority specified by the logic of the heuristic decision rule.

Then, tasks that are both precedence and cycle-time feasible (i.e., all predecessors have been assigned to a work station and the task time is not larger than the remaining time available at the work station), are placed on an available list. The task on the available list with the highest priority is assigned first. The available list is then updated to reflect the possible addition of tasks that are now precedence feasible, and the amount of time available to be assigned to tasks in the work station is reduced by the task time of the assigned task. This process continues for a station until no more tasks can be assigned to it. The assignment process then continues to the next station, and so on, until all tasks have been assigned to some work station. When the final task has been assigned, a complete balance has been obtained.

The thirteen decision rules included in our investigation are delineated in Table 1. Also included in Table 1 is the basis for determining task priority. A random rule is included in this set in order to provide a comparison with the more logically appealing decision rules. Rules which are being investigated for the first time include: MINLB, MINUB, MINSLK, MAXAVGRPW, MIN(UB/TFOL), MAX(DUR/UB), and MAX(TFOL/SLK). This set thus includes the majority of those single-pass, heuristic decision rules which have been found effective elsewhere, as well as includes several new approaches for solving this problem.

2.2 Composite Decision Rules

2.2.1 The Best of Thirteen Rule (COMPOSITE-13)

The first composite decision rule evaluated selects the best solution available from Category I decision rules. Because a given decision rule does not always discriminate among tasks on an available list (for example, two or more tasks available for assignment to a work station could tie on their task

TABLE 1

SINGLE-PASS DECISION RULES

Rule	Notation	Basis for Determining Task Priority
1. Maximum Ranked Positional Weight	MAXRPW	$RPW_i = t_i + \sum_{j \varepsilon S_i} t_j$
2. Maximum Total Number of Follower Tasks	MAXTFOL	NS_i
3. Maximum Task Time	MAXDUR	t_i
4. Maximum Number of Immediate Follower Tasks	MAXIFOL	NIS_i
5. Minimum Slack	MINTSLK	$UB_i - LB_i$
6. Random Task Assignment	RANDOM	random (uniform)
7. Minimum Lower Bound	MINLB	$LB_i = [(t_i + \sum_{j \varepsilon P_i} t_j)/C]^+$
8. Minimum Upper Bound	MINUB	$UB_i = N+1-[(t_i + \sum_{j \varepsilon S_i} t_j)/C]^+$
9. Minimum Task Number	MINTSKNO	task number, i
10. Maximum Average Ranked Positional Weight	MAXAVGRPW	$RPW_i/(NS_i+1)$
11. Minimum Upper Bound Divided by the Total Number of Followers	MIN(UB/TFOL)	$UB_i/(NS_i+1)$
12. Maximum Task Time Divided by Task Upper Bound	MAX(DUR/UB)	t_i/UB_i
13. Maximum Total Task Followers Divided by Task Slack	MAX(TFOL/SLK)	$NS_i/Slack_i$

C	Station cycle time.
N	The number of tasks to be balanced into work stations.
$NS_i(NP_i)$	The total number of tasks which succeed (precede) task i (i.e., the number of elements of $S_i(P_i)$).
$NIS_i(NIP_i)$	The number of tasks which must immediately succeed (precede) task i.
$S_i(P_i)$	The set of tasks which must succeed (precede) task i.
t_i	Assembly time required to complete task i.
$[X]^+$	The smallest integer greater than or equal to X.

times when the prioritizing scheme is based upon task duration), it is also possible to use one of the remaining twelve decision rules to break ties. Including the tie-breaker decision rules yields 156 (13 x 12) rule/tie-breaker rule combinations for solving each problem. Further, the possibility also exists to balance a given line beginning with the last task in the network and working forward, achieving what previous researchers of the line balancing problem have termed a "reverse" balance. Considering decision rule/tie-breaker rule combinations as well as balancing a line in reverse yields 312 possible decision rules for solving a given problem (13 x 12 x 2) using the thirteen heuristic decision rules described in Category I. The Best of Thirteen Rule selects the best of these 312 balances, implemented as described below.

Whenever a balance is determined by any decision rule that is equal to the smallest integer greater than or equal to the sum of the task times divided by the cycle time, the optimal solution (that is, the solution requiring the minimum number of work stations) has been determined. This is because it is not possible to obtain a balance requiring fewer work stations than given by this quantity, which we refer to as the LOWER BOUND (M) on the value of a solution.[1] In the implementation of the Best of Thirteen Decision Rule (as well as with all of the decision rules which follow), the procedure terminates the search for the minimum solution whenever a balance is obtained that is equal to this lower bound. The effect of this process is to eliminate unnecessary computations and, hence, to reduce the computation time required to solve many of the problems investigated.

---

[1]The quantity

$$M = [\sum_i t_i/C]^+$$

has been termed the "theoretical minimum" number of work stations by previous researchers investigating the line balancing problem. This is an unfortunate choice of words, as a given problem may have an optimal solution in which the number of work stations exceeds the "theoretical minimum."

2.2.2 Arcus' Biased Sampling Procedure (ARCUS)

The Arcus (1966) procedure uses a biased sampling approach to generate feasible sequences of tasks for assignment to a work station. A "fit list," consisting of those tasks which can be assigned to a work station is constructed, and weights governing the probability the task will be selected for assignment to the work station are assigned to each task. Tasks so assigned are removed from the fit list, and a new fit list, consisting of the tasks which can currently be assigned to a work station is constructed. The process continues until all tasks have been assigned to some work station. A given line is then balanced several times, resulting in different assignments based upon the probabilistic selection of tasks from the fit list. To be consistent with the work of Arcus (1963), Mastor (1970), and Dar-El (1975), Arcus' Rule IX (which is the product of five separate weights) is used in our evaluation. The procedure we use is programmed to determine eighty balances for each problem, and to then select the balance requiring the fewest number of work stations. However, if a balance is obtained during execution of this procedure that is equal to the lower bound M, then the procedure terminates, yielding M as the number of work stations required.

2.3 Backtracking Decision Rules

2.3.1 Hoffmann's Enumeration Procedure (HOFFMANN-0.0)

Hoffmann (1963) developed a heuristic method that assigns to each work station, in numerical order, a combination of tasks which minimizes station idle time. Starting with station one, a precedent feasible list of tasks is maintained from which the combination of tasks which will minimize station idle time is found via complete enumeration. These tasks are assigned to station one, and the process continues with station number two using an updated precedent feasible list. This procedure is repeated for each station

in numerical order, until all tasks have been assigned. Hoffmann uses a special zero-one precedence matrix and index vector to implement the enumeration process, which results in a very simple computer code, a description and FORTRAN listing of which is in his original article (1963). The Hoffmann methodology can be applied to tasks in decreasing order as well as in numerical order, which results in a second solution to a problem. In this investigation, all balances are solved in both the forward and reverse direction (where a forward balance did not determine the minimum required number of work stations, M), with the best solution being reported.

2.3.2 Hoffmann's Modified Enumeration Procedure (HOFFMANN-0.5 HOFFMANN-1.0 HOFFMANN-2.0)

Because the original Hoffmann approach considers stations in numerical order, it has a tendency to concentrate idle time in the later stations. In addition, a significant amount of computation time can be spent at those stations where the task precedence feasible list is long, and where a zero idle time solution doesn't exist or is difficult to determine. In the latter case, all feasible combinations of task assignments may have to be enumerated. In order to overcome these difficulties, Gehrlein and Patterson (1975) proposed a very slight modification to the original Hoffmann procedure: instead of determining the minimum idle time solution at each work station, determine one that is 'acceptably' close to minimum.

To define acceptably close, Gehrlein and Patterson use a measure of average idle time per station, which is derived in the following manner. The quantity (M x C) gives the <u>minimum</u> amount of time that will be devoted to producing one unit of product. This quantity bears the following relationship to the total work content ($\sum_i t_i$) for a given line:

$$(M \times C) \geq \sum_i t_i .$$

The difference between these two quantities $((M \times C) - \sum_i t_i)$ represents the minimum total line idle time that will necessarily be present in any balance. This quantity divided by the minimum number of work stations required, M, yields an estimate of the average amount of idle time present in a work station.

The Hoffmann procedure is thus modified by curtailing the search for the set of tasks resulting in the minimum amount of idle time for a given station whenever a subset of tasks σ is found which satisfies the following relation:

$$0 \leq C - \sum_\sigma t_i \leq \Theta \left([(C \times M) - \sum_i t_i]/M\right)$$

where the parameter Θ is varied in an attempt to obtain a balance requiring the fewest number of work stations. Gehrlein and Patterson (1975 and 1978) have shown that this slight modification to Hoffmann's original procedure can have a significant impact on reducing the amount of time devoted to each search, while simultaneously smoothing the idle time present among work stations for certain classes of line balancing problems. Further, this improvement in computation time is often achieved with no increase in the number of work stations required.

Based upon previous experience in solving line balancing problems with this approach, we allow Θ to assume the values 0.5, 1.0, and 2.0, and denote these modified Hoffmann procedures by HOFFMANN-0.5, HOFFMANN-1.0, and HOFFMANN-2.0, respectively.

2.3.3 Dar-El's Line Balancing Heuristic (MALB)

Dar-El developed MALB (1973) as a heuristic variant of his earlier optimal-seeking iterative procedure (1964). His optimal seeking procedure is based upon the Rank Positional Weight heuristic method of Helgeson and Birnie

(1961), enhanced with a backtracking algorithm that generates all feasible sequences of task assignments. Dar-El (1973) found that the computation time of his optimal seeking method restricted its applicability, so he sought to develop a heuristic approach that would retain most of the power of the optimal approach, without incurring its computational expense. MALB is the result of these efforts. To limit the number of sequences generated, and hence, to reduce computation time, MALB incorporates four types of heuristics that control the amount of backtracking permitted. In a comparative study (1975), Dar-El found that MALB dominated both ARCUS and the best of the ten single-pass rules he evaluated. On the basis of these results, MALB was included in the present investigation.

MALB, however, was originally written to solve the Type II problem. Hence, it was necessary to modify it slightly here. Like most of the methods developed to solve Type II problems, MALB, in effect, solves a series of Type I problems. In the Type I problem the cycle time is fixed, and the minimum number of stations is sought. The Type II problem fixes the number of stations, and seeks to determine the minimum cycle time. Operationally however, MALB, and most of the other procedures, solves the Type II problem by fixing the cycle time at the theoretical minimum time (the largest task time), and determines the minimum number of stations less than a specified 'goal' number of stations, γ. If no solution is found, then the cycle time is increased one time unit, and the process is repeated. The first feasible solution found with the number of work stations less than or equal to γ yields the desired balance. (In general, if the procedure is optimal-seeking, then the first feasible solution is optimal. Otherwise, as with MALB, it is heuristic.)

For the present investigation, instead of permitting the cycle time to increase when no feasible solution is determined, the goal γ is increased by

one work station, and another attempt is made to find a feasible solution. The first feasible solution found is the minimum (heuristic) number of stations. Specifically, then, the only modification made to MALB (other than changing input and output formats) was to defeat the logic in the procedure that permits the cycle time to increase. The cycle time is fixed at the desired experimental level. Initially, γ is set equal to M. If no feasible solution is determined, the value of γ is increased by one station, and the process is repeated until a feasible solution is determined.

2.4 Optimal-Seeking Decision Rules

The methods described in this section are included in our evaluation in order to gain insight into the benefit obtained by using optimal-seeking algorithms given a computational time constraint roughly comparable to the time required to solve a problem using the more sophisticated heuristic procedures. Clearly, given enough computation time, any optimal-seeking procedure will dominate a heuristic one. Our purpose is not to verify this fact, but to give an indication of the quality of solutions obtained for a given amount of computation time expended. All of the solution routines listed here have thus been run with an internal CPU time trap of 3.0 seconds per problem (balance). The reported solution is the minimum one obtained within the 3.0 second limit.

2.4.1 Branch and Bound Methods (MAG-1 and MAG-2)

Magazine and Wee (1981b) report excellent results for the Type I line balancing problem with their branch and bound procedure. With their method, each node in the solution tree corresponds to a feasible set of task assignments to a particular work station, where all nodes at the same depth in the tree refer to the same station number. Starting with node zero (the null set), descendent nodes from a node of depth (d) are generated, which are

maximal feasible assignments of tasks to station (d+1). The branching direction, fathoming criteria, and growth rate of the tree are controlled via a number of heuristics and dominance tests incorporated into their procedure.

Two decision rules of special significance used in their procedures are IUFFD (Immediate Update First Fit Decreasing), which we refer to as MAXDUR, and IUBRPW (Immediate Update Backward Recursive Positional Weight), a variation of MAXRPW. Although their branch and bound procedure incorporates a breadth-first (versus depth-first) branching strategy, an upper bound on the solution is found at each node using one of these two heuristics for the unassigned tasks. This permits their procedure to terminate prior to verification of optimality with a given heuristic solution to a problem. To facilitate the use of this feature, they allow the user to select one of these heuristics and to specify a time limit for implementation of their approach. This is the time limit capability used in our experiments. The references to MAG-1 and MAG-2 correspond, respectively, to the use of the heuristics IUFFD and IUBRPW.

2.4.2 Integer Programming (ALBCUT and ALBHOFF)

Two variations of Talbot and Patterson's (1984) integer programming procedure are included in our evaluation. The basic algorithm is a depth-first, implicit enumeration, backtracking procedure to which various search, fathoming, and backtracking decision rules are applied. The first variation included (ALBCUT) contains network cuts (1984), where search and backtracking are controlled with the heuristic decision rule MINUB (with ties broken using MAXDUR). The second variation (ALBHOFF) also uses MINUB and MAXDUR for search and backtracking, but does not use any cut associated fathoming rules. Also, prior to enumeration, the problem is first solved with HOFFMANN-0.5, which

provides an initial heuristic starting solution and computational upper bounds.

2.4.3 Dynamic Programming (DYNAMIC)

Schrage and Baker (1978) have proposed an efficient method for implementing the dynamic programming approach of Held et. al. (1963) through improved procedures for generating feasible subsets, and for labeling. Magazine and Wee (1981b) programmed and tested the Schrage and Baker approach for solving the Type I line balancing problem. Magazine and Wee concluded that their branch and bound solution procedure is preferred to dynamic programming for solving these types of line balancing problems, both with regard to computation time and computer storage required. For completeness, however, we have included the dynamic programming approach in our evaluation.

2.4.4 Multiple Solutions Technique (MUST)

MUST, by Dar-El and Rubinovitch (1979), employs exhaustive enumeration to generate all solutions, or some subset of them, for solving the Type II line balancing problem. As with MALB, the Type II problem is solved as a sequence of Type I problems. The cycle time is fixed, and a "hurdle" number of work stations, γ, is specified. If a feasible number of work stations cannot be found given γ, the cycle time is increased one time unit, and the process is repeated. Starting with station number one, MUST generates all feasible task assignments. Redundancies are eliminated, and the remaining feasible task assignments (subsets) for this station are saved. At station two, and subsequent stations taken in numerical order, these subsets are extended by adding all feasible task assignments at a work station. Redundancies are then eliminated, and the extended subsets are saved. This process is repeated until all tasks have been assigned (all subsets are fully extended).

This exhaustive enumeration procedure results in an exponential growth in the number of subsets saved. In order to make this process manageable, MUST saves sets as a sequence of bits, and also uses circular storage buffers. Even with these storage-saving features however, MUST, unlike the other methods evaluated, is programmed to use external storage. To control the growth in the number of subsets saved, MUST contains three user-specified parameters. These can be set to permit the generation of either multiple optimal solutions, or multiple heuristic solutions.

As a result of experiments by Dar-El and Rubinovitch (1979) which demonstrated that MUST dominates MALB, MUST was included in our investigation. It was modified to solve the Type I problem in the same manner as was MALB, by permitting the hurdle number of work stations γ to increase by one station if a feasible solution could not be found, rather than (as with the Type II approach) increasing the cycle time by one time unit in each iteration.

3. DESCRIPTION OF PROBLEMS SOLVED

In order to generalize the results of our investigation, both randomly generated problems and problems taken from the open literature are solved. The characteristics used for problem generation are based upon an examination of actual assembly line balancing problems, and are experimentally varied over a wide range of representative values. There are four data sets in total: three comprising randomly generated problems, and a fourth data set consisting of problems available in the open literature.

3.1 Main Experimental Data Set

This data set contains 120 unique assembly networks, each of which is solved for 11 cycle times. Details of problem characteristics and generation schemes for these problems are described below.

3.1.1 Problem Size

Sixty 50-task and sixty 100-task assembly networks are randomly generated. These line sizes are sufficiently large to permit generalization of our results to actual line balancing problems, and are comparable in size to those used by Dar-El (1975), and are significantly larger than those examined by Mastor (1970).

3.1.2 Distribution of Task Times

An investigation of actual line balancing problems appearing in the open literature suggests parameters for generating task times. These times are derived from the binomial distribution ($n=30$, and $\pi=0.25$), with zero duration tasks having their task times increased to one time unit.

3.1.3 Distribution of Cycle Times

When the magnitude of the cycle time approaches the maximum task time in an assembly line balancing problem, the alternatives available for scheduling tasks into work stations decrease, as do the opportunities for the heuristic scheduling rule to exercise its intended logic in assigning tasks to a work station. Alternatively, as the cycle time increases relative to the maximum task time, the number of alternatives for assigning tasks to work stations increases, as does the discretionary "power" of the investigated decision rule. In order to investigate the efficacy of the heuristic decision rules evaluated over a wide range of problem types, each problem generated is solved for the cycle time equal to the maximum task time. The cycle time is then increased in 10 percent increments of the maximum task time to the closest integer equal to 10 percent, 20 percent, 30 percent, etc., above the minimum possible cycle time until eleven balances are obtained for each problem.

3.1.4 Strength of the Precedence-Ordering Relations Among Tasks

The <u>strength</u> of the precedence-ordering relations among tasks affects the number of alternative production lines that may be established. This in turn

influences the discretionary power of the heuristic decision rule and, hence, its effectiveness. Network density, a characteristic which measures the strength of this relation, has been found to be an important factor in influencing heuristic performance in previous investigations of the line balancing problem.

To define density, let W be an N x N 0-1 matrix that represents a precedence-ordering relation P. For a given element ω_{ij} of W, let ω_{ij} = 1 if $X_i \varepsilon P_j$ (i.e., if task i precedes task j) and ω_{ij} = 0 if task i does not precede task j, or $X_i \notin P_j$. Let d be the total number of relations contained in P, or

$$d = \sum_{i=1}^{N} \sum_{j=1}^{N} \omega_{ij}$$

The maximum number of relations that can be included in P is N(N-1)/2. The density of the assembly network is defined to be the ratio:

$$D = 2d/[N(N-1)],$$

or the ratio of the number of relations that exist to the number which could exist. The measure 1-D is the F-ratio used by Dar-El (1975), and the measure D is referred to as <u>order strength</u> by Mastor (1970).

Values of D close to 1 indicate a highly interconnected network, and fewer alternatives available for assigning tasks to a work station. Values of D close to 0 indicate relatively fewer precedence relationships, and more opportunities for assigning tasks to a work station. In our evaluation, values of D equal to 0.2, 0.3, 0.5, and 0.8 are used for problem generation.

3.1.5 Partial-Order Generating Methods

It is possible to test not only for the effect of network density on heuristic performance, but also for the effect of the method used to generate precedence-ordering relations. That is, several methods for generating a

precedence-ordering relation on a set of tasks can be devised, each resulting in the identical problem density. However, various factors of the ordering relation structure can be different for different generating methods. Three methods of generating precedence-ordering relations are investigated, and their effects on problem solution are assessed.

All three methods of generating precedence-ordering relations work in the same general way. We begin to describe how each method works by recognizing that precedence-ordering relations are partial ordering relations. A binary relation P on a set X is a partial order if it is irreflexive ($x_i P x_i$ for no x_i in X), asymmetric ($x_i P x_j$ requires not $x_j P x_i$ for all x_i and x_j in X), and transitive (if $x_i P x_j$ and $x_i P x_k$ then $x_i P x_k$ for all x_i, x_j, and x_k in X). Given a linear ordering relation $L = x_1 x_2 \ldots x_N$, we wish to generate a random partial order P that is contained in L, or $P \subsetneq L$. That is, if $x_i P x_j$, then $i < j$. We do not lose any generality by requiring this, because any partial order can be transformed so that if $x_i P x_j$ then $i < j$ simply by interchanging the subscripts attached to the x's.

The first step in each generating process is to generate a suborder P_0. Suborders are binary relations that are irreflexive and asymmetric, but not necessarily transitive. P_0 is then changed to a partial order by taking the transitive closure of P_0, denoted by P_0^t. This indicates that all necessary P relations are added so that P_0^t is transitive. For example, if the suborder P_0 has $x_i P_0 x_j$ and $x_j P_0 x_k$, it is not necessarily true that $x_i P_0 x_k$, since suborders are not always transitive. In forming P_0^t, all relations, such as $x_i P_0 x_k$, are appended so that P_0^t is transitive. Since P_0^t is a transitive suborder, it is a partial order, and it is used as the generated partial order in network synthesis (Fishburn & Gehrlein (1974)).

We now describe the three partial order generating methods used in our investigation:

Method 1: Let σ be a randomly selected permutation on $\{1, 2, \ldots, N\}$. For each j from 2 through N select a y_i at random from $\{x_{\sigma(1)}, x_{\sigma(2)}, \ldots, x_{\sigma(j-1)}\}$. Let $P_0 = L \cap \{[y_2, x_{\sigma(2)}], [x_{\sigma(2)}, y_2], \ldots, [y_N, x_{\sigma(N)}], [x_{\sigma(N)}, y_N]\}$ so that P_0 contains N-1 order pairs, and form the transitive closure P_0^t. Then P_0^t is made more dense by adding q additional pairs from $L-P_0^t$. The transitive closure is taken after each pair is added and P results after the final transitive closure is taken.

Method 2: Form P_0 by taking each (i,j) pair with $i < j$ and enter $x_i P_0 x_j$ with probability p. The same p is used when each (i,j) pair is considered for entry into P_0. Then $P = P_0^t$.

Method 3: Form P_0 by taking each (i,j) pair with $i < j$ and enter $x_i P_0 x_j$ with probability $\left(\frac{(j-i)}{(N-1)}\right)k$ for some positive k. The same k is used when each (i,j) pair is considered for entry into P_0. Then $P = P_0^t$.

By examinating three distinct methods for generating the partial orders necessary to construct each network, we can assess the sensitivity of our results to the network generation scheme used. Many examples of simulation analysis can be given where the results reported are dependent on the procedure used to generate random partial orderings. (For example, see Fishburn & Gehrlein (1974) or Patterson (1976).) Hence, an examination of their effect is prerequisite to making general statements concerning the efficacy of the heuristic procedures investigated. (For a more comprehensive discussion of how the desired density measures are determined by using partial order generating methods, see Gehrlein (1980).)

3.1.6 Main Experimental Problems Solved

For each line size (50 and 100 tasks), each density (0.2, 0.3, 0.5, and 0.8), and each partial-order generating method (Method 1, Method 2, and Method 3), five line balancing problems are constructed. The density of the partial orders generated by Methods 1, 2, and 3 is controlled by the values of q, p, and k used, respectively, in each method. Each of the generated problems is then solved for cycle times equal to the largest task time, the largest task time plus 10 percent of the largest task time (rounded to the nearest integer), the largest task time plus 20 percent of the largest time (rounded to the nearest integer), etc., until the cycle time equals twice the largest task time. This yields 1,320 balances in total [(2)(4)(3)(5)(11)] in the Main Experimental Data Set.

3.2 Literature Problems

A number of problems, ranging in size from 7 assembly tasks to 111 assembly tasks, were taken from the open literature to be used in this evaluation: Arcus (1963), Bowman (1960), Dar-El (1964), Heskia (1968), Jackson (1956), Jaeschke (1964), Kilbridge and Wester (1961), Merten (1967), Sawyer (1970), and Tonge (1961 and 1969). These problems also serve as the basis for determining the network generator parameters used for the main experimental data set. The cycle times used to assess heuristic performance in evaluating the procedures described are the same as those used by Talbot and Patterson (1984) in their evaluation of an integer programming algorithm for obtaining optimal solutions to this line balancing problem, with the exception of the Bowman and the Dar-El problems, which they did not include. For these latter two problems, the same cycle times as reported by the original authors were used (C = 20 for the Bowman problem, and C = 48, 62, and 94 for the Dar-El problem). For a more complete description of this data, see Talbot et. al.

(1985). Sixty-four balances were obtained using this literature data set over the range of cycle times considered.

3.3 "Difficult" Problem Set One

Five 50-task and five 100-task assembly line networks with task times uniformly distributed between 1 and 10,000, and with network density equal to 0.2 were randomly generated and solved for 11 trial cycle times equal to 10,000, 11,000, ..., 20,000. These balances were included in the experiment because with few precedence relations in the network and a very wide range of task times, the enumeration procedures evaluated can be expected to exhibit large increases in computation times. There are two arguments to support this notion: With an enumeration procedure, there will be fewer opportunities to terminate the search procedure at a given station before full enumeration is complete, because there are fewer sets of sums of task times that equal, or nearly equal, the cycle time. Also, with fewer precedence relationships, many more assignments are possible, resulting in an anticipated increase in the computation time required to solve a given problem.

3.4 "Difficult" Problem Set Two

Preliminary experimental results using "Difficult" Problem Set One, led to the development of "Difficult" Problem Set Two. This set contains the same 110 networks as in Set One, except that all the odd task times are increased by one time unit to make them <u>even numbers</u>. The eleven cycle times include the largest task time and added increments of 10% of this value (to the nearest integer) up to twice the largest task time, where all even cycle times are increased by one time unit to make them <u>odd numbers</u>. With all problems having even task times and odd cycle times, <u>no</u> zero idle time station assignments of tasks or zero balance delay solutions are possible, thus

forcing enumeration methods such as HOFFMANN-0.0 to completely enumerate all feasible task assignments at every work station.

We emphasize that these latter two data sets were not generated to represent actual problem types such as those found in examining industrial line balancing problems. Rather, they were generated to assess "worst case" performance for several of the techniques investigated, especially those employing enumerative and backtracking approaches.

4. COMPARATIVE RESULTS

4.1 Experimental Conditions

An attempt was made to solve each of the problems in all four data sets using the 26 solution procedures described in Section 3. A time limit of 3.0 CPU seconds per balance for each solution procedure was implemented as described. This is an internal time trap that excludes input time, but includes minimal computer output time per balance. All methods (with the exception of MUST) are programmed in FORTRAN and were run under the same operating system and compiler (IBM FORTRAN H OPT=2) and on the same computer, an AMDAHL 470/V8. MUST is programmed in PL1, and was run on the same AMDAHL 470/V8 using the same operating system after being compiled by the IBM PL1 optimization compiler (OPT=2). All methods, including MUST, were programmed to read from the same data set in an identical manner, and to write summary results to a common output data base. A concerted effort was made to run each program under similar operating conditions.

With the exception of MALB, HOFFMANN-0.0, MUST and DYNAMIC, the methods were able to solve all problems within the time limit specified per problem. Unfortunately, neither MUST nor DYNAMIC were able to solve more than just a few of the smaller problems within this time limit. It became very clear that

their performance would be even worse on the larger problems, and it would be a misuse of a constrained computer budget to have them attempt the larger problems. Hence, they were dropped from further experimentation, leaving 24 solution methods to be evaluated.

Before proceeding with the main results, an indication of the results obtained using MUST and DYNAMIC will be given to justify excluding them from further consideration. (The following are representative of our results--many more tests were run than are described here.)

As mentioned in Section 2.4.4, MUST contains three user-specified parameters that control the growth of the solution space, which in turn affect computation time and solution quality. We first input the recommended parameter values as given in Table 2 of Dar-El and Rubinovitch (1979). Using these parameter values, MUST was able to solve only 27 of the 64 literature balances (the largest solved consisting of 21 tasks) within the 3.0 second time limit specified. Even with the limit doubled to 6.0 seconds per balance, only 29 of the 64 problems were solved, the largest consisting of 28-tasks. Within the 6.0 second limit, MUST was not able to solve any of the problems in either of the "Difficult" Data Sets One or Two, or any of the first 11 50-task problems in the Main Experimental Data Set.

In an effort to obtain more solutions using MUST, even at the possible expense of lower quality solutions, the parameter values were all set equal to one, which would have the anticipated effect of reducing computation time and storage requirements for the algorithm. With these parameter values, MUST was able to solve 40 of the literature balances (the largest being a 30-task problem), two of the 110 balances (both 50-task) in the "Difficult" Set One, and none of the problems in the "Difficult" Set Two, within the 3.0 second time limit. A number of other tests were conducted with similar results. It

should be noted that these timing results are consistent with those of Dar-El and Rubinovitch, who report an average of 125.4 seconds per balance on an IBM 370/168 for a set of 50 to 140-task problems, for which they also report MALB required an average of 22.4 seconds on the much slower IBM 7044.

The dynamic programming procedure was able to solve only the smallest 27 of the 64 literature balances within the 3.0 second time limit per balance, the largest problem being Mitchell's 21-task problem (C=39). Attempts to solve larger problems (e.g., 28-task and 45-task) also failed with a time limit of 6.0 seconds per balance. These findings are completely consistent with Magazine and Wee's results (1981b), and reinforced our decision to exclude this procedure from further evaluation.

4.2 Main Experimental Data Set Results

Table 2 contains summary results for all heuristic decision rules retained, and for each data set. The first column under each data set is a relative measure of heuristic performance, the average percent above the best solution found by any method. The second column under each data set is the average percent above optimum, an absolute measure of heuristic performance. Whereas Table 2 indicates average heuristic performance, Table 3 shows the worst single result using each method. Specifically, the first column under each data set lists for each method the maximum percent above best heuristic solution determined for a single balance, and the second column lists the maximum percent above optimum. (All methods determined each balance within the 3.0 second time limit specified, except as noted in Table 2.)

In order to more clearly illustrate the relationships depicted in Table 2 for the Main Experimental Data Set, Figures 1A and 1B were prepared. Figure 1A is a histogram of first column data with the heuristics grouped by category, and ranked within category. Figure 1B is a similar histogram of second

TABLE 2

Average Percent Results For All Data Sets

DATA SET:	MAIN EXPERIMENTAL DATA SET		LITERATURE		DIFFICULT SET ONE		DIFFICULT SET TWO	
CRITERIA:	Above Best (1) Heuristic	Above (2) Optimum	Above Best (3) Heuristic	Above (4) Optimum	Above Best (5) Heuristic	Above (6) Optimum	Above Best (7) Heuristic	Above (8) Optimum
SOLUTION METHOD								
1 MAXRPW	5.16	5.77	4.62	4.85	4.76	4.35	4.43	4.33
2 MAXTFOL	4.89	5.55	4.27	4.31	4.70	4.46	4.22	4.02
3 MAXDUR	4.71	5.20	4.81	4.70	4.99	5.22	4.72	4.87
4 MAXIFOL	6.00	6.24	3.96	3.94	4.79	4.37	4.59	4.62
5 MINTSLK	6.50	6.81	5.16	5.34	5.43	5.20	5.09	4.75
6 RANDOM	9.33	9.48	10.26	10.71	9.11	8.32	8.96	8.22
7 MINLB	6.03	6.45	5.81	6.03	5.15	5.19	4.94	4.73
8 MINUB	4.51	5.11	3.43	3.34	3.91	3.75	3.40	3.43
9 MINTSKNO	6.63	7.21	8.96	9.02	7.08	6.95	7.04	6.75
10 MAXAVGRPW	6.35	6.76	5.66	5.84	5.15	4.93	5.09	5.09
11 MIN(UB/TFOL)	5.00	5.61	4.27	4.31	4.82	4.51	4.30	4.07
12 MAX(DUR/UB)	4.50	5.14	4.40	4.52	4.87	4.88	4.59	4.75
13 MAX(TFOL/SLK)	5.05	5.65	4.71	4.83	4.60	4.22	4.26	4.03
14 COMPOSITE-13	2.34	2.97	0.46	0.23	1.90	2.47	1.65	1.72
15 ARCUS	2.73	3.35	2.78	2.94	2.06	2.00	1.75	1.80
16 MALB	2.15	1.78	0.31	0.36	2.80	2.40	2.78	2.52
17 HOFFMANN-0.0	0.46	0.04	1.34	1.48	1.75	0.65	1.42	0.44
18 HOFFMANN-0.5	0.47	0.04	3.29	3.44	1.58	0.15	1.61	0.33
19 HOFFMANN-1.0	0.50	0.09	3.29	3.44	1.61	0.39	1.56	0.20
20 HOFFMANN-2.0	0.65	0.36	3.41	3.44	2.41	1.68	1.96	1.34
21 MAG-1	1.05	0.92	0.00	0.00	0.89	0.26	1.09	0.31
22 MAG-2	2.39	2.21	0.52	0.61	2.16	1.34	2.27	1.18
23 ALBCUT	3.68	4.13	0.60	0.26	3.52	2.92	3.22	3.23
24 ALBHOFF	0.42	0.02	0.73	0.45	1.13	0.01	1.10	0.13
Number of Balances	1,320[a]	635[a]	64	55	110[b]	50[b]	110[c]	45[c]

a. The results for MALB are based upon 872 balances (out of 1,320) and 431 (out of 635) that it solved.
b. The results for MALB are based upon 79 (out of 110) and 30 (out of 50) balances.
c. The results for MALB are based upon 80 (out of 110) and 24 (out of 45) balances. The results for HOFFMANN-0.0 are based upon 69 (out of 110) and 25 (out of 45) balances, respectively.

TABLE 3

Maximum Percent Results For All Data Sets

DATA SET:	MAIN EXPERIMENTAL DATA SET		LITERATURE		DIFFICULT SET ONE		DIFFICULT SET TWO	
CRITERIA:	Above Best (1) Heuristic	Above (2) Optimum	Above Best (3) Heuristic	Above (4) Optimum	Above Best (5) Heuristic	Above (6) Optimum	Above Best (7) Heuristic	Above (8) Optimum
SOLUTION METHOD								
1 MAXRPW	17.4	15.0	50.0	50.0	18.2	15.8	21.1	21.1
2 MAXTFOL	17.4	15.8	33.3	33.3	21.1	21.1	21.1	21.1
3 MAXDUR	13.6	13.6	33.3	33.3	15.0	15.0	13.3	13.3
4 MAXIFOL	17.4	17.4	33.3	33.3	21.1	21.1	21.1	21.1
5 MINTSLK	17.6	16.7	33.3	33.3	18.2	15.0	14.3	11.1
6 RANDOM	25.0	25.0	50.0	50.0	22.7	20.0	17.6	17.6
7 MINLB	20.0	20.0	33.3	33.3	20.0	20.0	15.8	15.8
8 MINUB	14.3	14.3	33.3	33.3	15.0	15.0	15.8	15.8
9 MINTSKNO	17.6	17.6	50.0	50.0	18.2	15.8	16.7	15.8
10 MAXAVGRPW	17.9	15.8	33.3	33.3	18.2	13.3	15.8	11.8
11 MIN(UB/TFOL)	17.4	15.8	33.3	33.3	21.1	21.1	21.1	21.1
12 MAX(DUR/UB)	15.0	15.0	33.3	33.3	15.8	15.0	15.8	13.3
13 MAX(TFOL/SLK)	17.4	15.0	33.3	33.3	19.0	15.8	19.0	15.8
14 COMPOSITE-13	10.0	7.7	16.7	12.5	7.7	7.7	7.7	7.7
15 ARCUS	10.5	10.5	33.3	33.3	9.1	7.1	8.7	7.1
16 MALB	13.6	10.5	20.0	20.0	15.8	15.8	15.8	15.8
17 HOFFMANN-0.0	10.0	7.1	25.0	25.0	9.5	5.9	7.1	5.9
18 HOFFMANN-0.5	10.0	7.1	33.3	33.3	10.5	4.8	7.7	5.6
19 HOFFMANN-1.0	10.0	7.1	33.3	33.3	10.5	5.0	10.0	5.9
20 HOFFMANN-2.0	10.0	7.1	33.3	33.3	10.5	7.7	9.1	6.7
21 MAG-1	11.6	11.6	0.0	0.0	14.3	6.9	14.3	10.5
22 MAG-2	13.5	11.4	33.3	33.3	14.3	11.8	14.3	10.5
23 ALBCUT	12.5	12.5	16.7	14.3	15.0	15.0	15.8	15.8
24 ALBHOFF	8.3	7.1	25.0	25.0	9.5	2.9	6.7	3.0
Number of Balances	1,320	635	64	55	110	50	110	45

NOTE: The footnotes to Table 2 also apply to Table 3.

FIGURE 1A

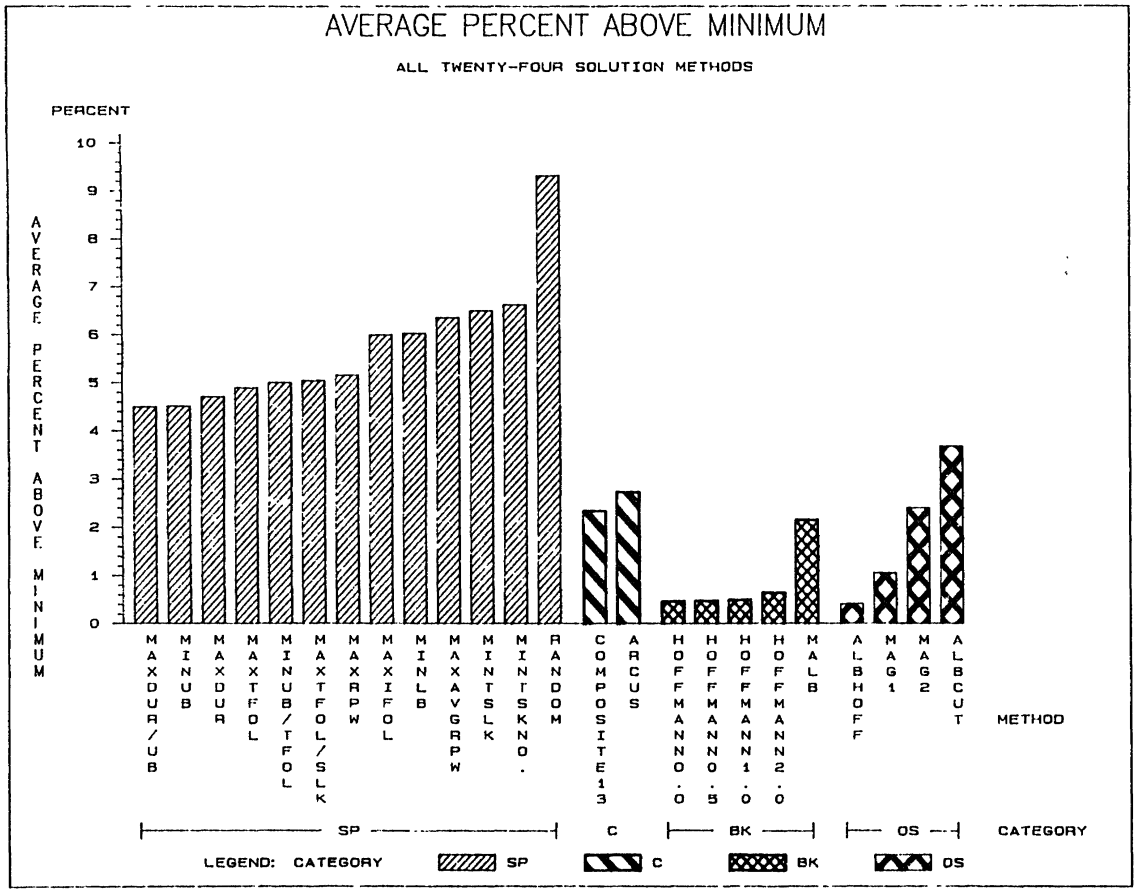
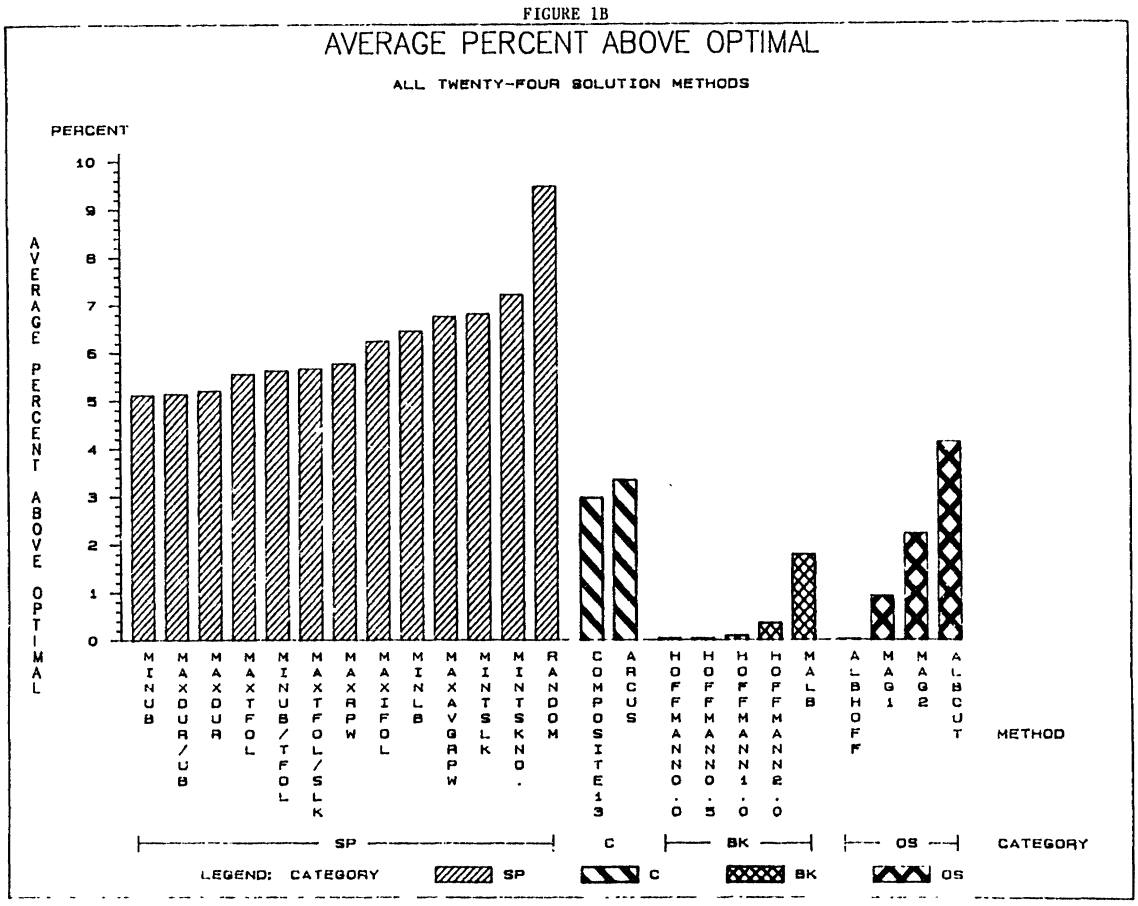


FIGURE 1B



column data. As indicated in Figure 1 (A or B), the enumeration procedure of Hoffmann, or a variant of it, performs best on these problems. The original Hoffmann procedure, HOFFMANN-0.0, yields an average increase of 0.46 percent above the best heuristic solution result obtained, and an average increase of 0.04 percent above the optimal number of work stations. The Gehrlein and Patterson modification to improve computation time and idle time distribution characteristics yields only slightly larger results, as seen with HOFFMANN-0.5, -1.0, and -2.0. ALBHOFF, which begins with the HOFFMANN-0.5 solution and systematically attempts to improve upon it, has the best overall results of 0.42 percent above the best heuristic result obtained, and 0.02 percent above optimal solution on this group of test problems. ALBHOFF also found and verified the optimal solution more frequently than did any of the other optimizing methods (632 out of 635 optimal balances examined) and has the best "worst case" performance for a single balance (8.3 percent above best heuristic, and 7.1 percent above optimal, respectively).

The next best results obtained are 1.06 percent above best heuristic and 0.92 percent above optimum for MAG-1. The performance of MAG-2, at more than twice the average percentage levels of MAG-1, and the much larger differences between ALBHOFF and ALBCUT, indicate the importance of the incorporated heuristics in implementing each of the optimization approaches. The primary difference between MAG-1 and MAG-2, for example, is in the use of IUFFD versus IUBRPW influencing the search strategy. Similarly, the overriding reason for the difference in performance between ALBHOFF and ALBCUT lies in the use of different initializing heuristics with each approach (HOFFMANN-0.5 versus MINUB). Although each of these methods will <u>eventually</u> determine the optimal solution given sufficient time and computer storage, these results suggest that the computational effort required in using these procedures as

heuristics is significantly influenced by the choice of the internal heuristic procedure employed.

For the 872 balances it obtained, MALB has an average percent increase above the best heuristic result of 2.15, and an average percent increase above optimum of 1.78. These percentages are not directly comparable to the percentages for all of the other methods which are based upon the larger set of 1,320 balances. In an effort to correct for this difference, all methods were reranked on just the 872 balances that MALB obtained. This resulted in no changes in the rankings for the percent above optimum criterion. For percent above best heuristic, the ranking of MALB did not change. However, there were two pairwise exchanges: MINUB and MAX(DUR/UB) exchanged rankings, and COMPOSITE-13 and MAG-2 exchanged rankings.

A more detailed examination of MALB's results indicate that both problem size and cycle time affect computational performance. Of the 448 balances not obtained, 308 are from 100-task and 140 are from 50-task problems. More striking, however, is the impact the cycle time increment has on heuristic performance. The most difficult problems for MALB, in terms of computation time, are those with cycle times in close proximity to the largest task time. A plot of balances <u>not</u> solved against cycle time increment looks surprisingly exponential, with 112 not solved when the cycle time equals the largest task time, 96 not solved for the first increment above the lowest cycle time, etc., decreasing to 14 not solved at twice the largest task time. In terms of solution quality, however, our results with MALB are generally consistent with Dar-El's (1975) findings that MALB yields better average results than ARCUS or a composite of the single-pass heuristics he tested.

Predictably, the composite methods perform better than the single-pass heuristics in terms of average and worst case performance. Somewhat

surprising is that over all methods considered, COMPOSITE-13 has the second best worst case results, even though a number of other methods have better average results.

Among the single-pass methods examined, two new heuristics, MINUB and MAXDUR/UB, slightly outperformed MAXDUR as the best of the group. The poorest average and worst case results come from the benchmark rule RANDOM, as one designing heuristics would hope. (This is not always the case, however. Davis and Patterson (1975) found for the resource-constrained project scheduling problem, that a random rule was often better than several apparently reasonable heuristics.) Although no attempt was made to number the tasks in a structured fashion during problem generation, solving the problems by assigning tasks in numerical order using MINTSKNO, our second benchmark rule, did produce slightly better solutions than when an explicit effort was made to randomly assign tasks with the RANDOM rule. This indicates the need to explicitly apply a random rule in this type of experiment, rather than rely on an 'arbitrary' surrogate.

Not indicated in the tables and figures is that while certain rules produced superior results on average, or in terms of worst case results, no one procedure consistently dominated any other. All procedures, including RANDOM, are able to produce minimum, and often unique, solutions to certain problems, albeit is rare for some of them.

In order to more systematically analyze the results for the Main Experimental Data Set, a randomized complete block, full factorial analysis of variance was performed.[1] Assessed are the effects of the number of tasks (T),

---

[1]MALB was omitted from this analysis because of the overall size of the experiment and the consequent effect unequal cell sizes had on the analysis. The SAS statistical software system (PROC ANOVA) on an Amdahl 470/V8 computer was used to analyze these results.

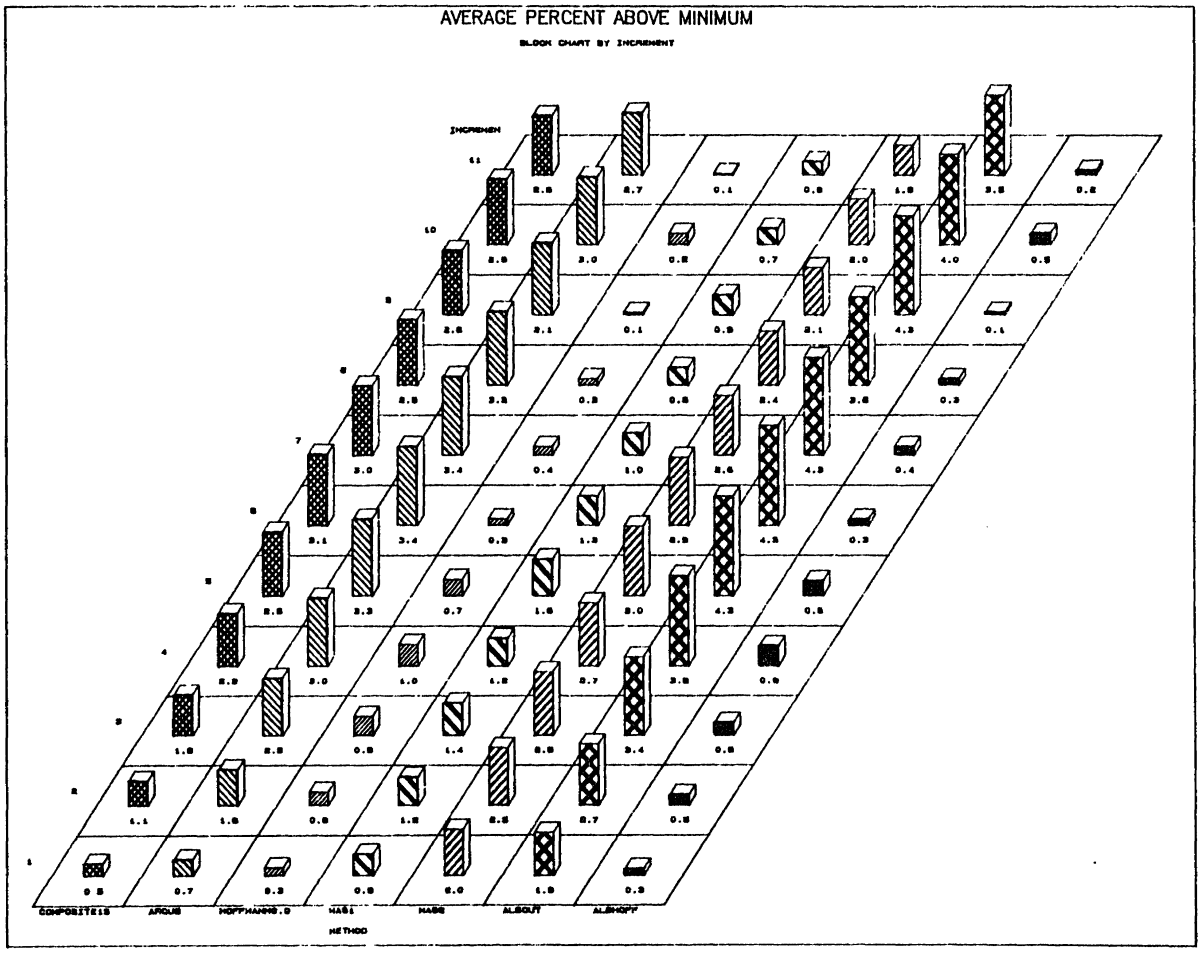
network density (D), method of generating partial orders (G), number of increments above the lower bound cycle time [i.e., $MAX_i(t_i)$] (I), and heuristic solution technique (H) on line balancing performance. The response variable measured is the Percent Increase Over the Minimum Heuristic Result obtained. Those variables found to be significant in this analysis include (in order of significance): the heuristic solution technique (H) examined ($r^2 = 0.47$), and the number of increments (I) above the lower bound cycle time ($r^2 = 0.30$). Variables found relatively insignificant in contributing to heuristic performmance include network density ($r^2 = 0.015$), number of tasks in the line balancing problem ($r^2 = 0.0006$), and the method used to generate random partial orders ($r^2 = 0.002$). The only significant interaction effect identified is the interaction between the heuristic solution technique employed and the number of increments above the lower bound cycle time (H*I).[1] Thus, it appears as though heuristic performance is not contingent in a statistical sense upon these latter measures (T, D, G) of problem structure.

In order to illustrate the (H*I) interaction, we include Figure 2, which is a three-dimensional histogram of the Percent Increase Above the Best Heuristic Solution for seven of the methods examined and for all eleven cycle times (C = maximum task time at Increment 1. At Increment 2, C = 110% of maximum task time, etc.). What is clear from this chart is that the average performance of each rule varies as a function of the cycle time, and in some cases by a considerable amount (by a factor of six for COMPOSITE-13). For each method, the distribution of Average Percent Above the Best Heuristic

---

[1]MAG-1 did generally (1) decline in performance when the number of tasks in an assembly network increased from fifty to one-hundred, and did generally (2) improve in performance over the other techniques examined when the density of a network increased (especially to 0.8). These results, however, were not found to be statistically significant.

FIGURE 2



Result Obtained is low at Increment 1, then gradually rises, and then decreases (eventually to zero for all methods when $C = \Sigma t_i$, although this latter situation is not specifically plotted in Figure 2). In an effort to better understand this relationship, the distributions of the average <u>number</u> of stations (instead of average <u>percent</u>) above best heuristic solution were also plotted to control for the fact that as C gets larger, the number of stations in a solution generally gets smaller. These distributions are very similar in appearance to those shown in Figure 2, revealing the same general types of information. In general, with both plots, one would expect there to be much less variability among the methods when the cycle time is equal to or is close to the maximum task time. This is because there are fewer opportunities to effect alternative task assignments. As the cycle time increases

beyond the maximum task time, however, there become more opportunities to effect different assignments as there are more combinations of tasks that can be feasibly assigned into each work station. This phenomenon continues until the cycle time approaches the work content of the product, in which case all methods will assign each of the tasks to the one work station required, giving zero variability in heuristic performance.

One has to exercise a great deal of caution in interpreting Figure 2. It is tempting to conclude, for example, that the best performance overall is achieved by the heuristic decision rules when the cycle time is equal to the maximum task time. What is minimal when the cycle time is equal to the maximum task time is the variance in the performance of the procedures. There are simply fewer opportunities using these procedures to effect alternate balances at this cycle time. And, in fact, it is at the lower cycle times that the optimization approaches used in this evaluation (as heuristic procedures) have the greatest amount of difficulty in determining and verifying the optimality of the solutions obtained. Specifically, for only one balance out of 120 at the first increment was an optimal solution found and verified using the four approaches described, whereas 112 out of 120 optimal balances were found and verified when the cycle time was 200% (increment 11) of the maximum task time. The entire range of values for the eleven increments are (1, 6, 18, 25, 43, 68, 69, 87, 103, 103, 112). Hence, without additional verified optimal solutions at the lower cycle times, it is simply not possible to claim that all of the methods studied perform better in an absolute sense at the lower cycle time increments.

Tukey's studentized range test (HSD) was used to compare the results obtained for the performance measure Percent Increase Above Minimum Heuristic Result Obtained. An alpha value of 0.05 was used for reporting significant

differences. It is encouraging that using this procedure, the RANDOM rule which produced the highest mean value overall was placed into a group by itself, and the five variants of the HOFFMANN procedure which produced the lowest overall sample mean values were also grouped together. These results were identical when the performance measure became Percent Increase Over Optimal. Employment of Duncan's multiple range test and Scheffe's comparison procedure produced similar results on this group of test problems.

4.2 Literature Data Set

As indicated in Tables 2 and 3, MAG-1 performed relatively best on the Literature Data Set by determining minimum balance solutions to all problems, and verified optimum solutions for 55 balances. The second best results were determined by MALB and COMPOSITE-13, for the measure Percent Increase Above Best Heuristic and Above Optimum, respectively. COMPOSITE-13, MALB, MAG-2, ALBCUT, and ALBHOFF each found 54 of the 55 verified optimum solutions.

Although it is instructive to compare methods with a common set of test problems appearing in the open literature, some caution should be exercised in extrapolating percentage results from this data set. First, it is a very small sample. More important however, is the use of Percentage Increase Above Minimum Heuristic Results criterion. Over half of the balances obtained using this data have solutions consisting of less than ten stations (ranging from two stations up to 29), with only two balances containing more than 20 stations. With such small numbers in the denominator using this criterion, even a one station difference between methods often yields a large percentage increase. In contrast, this does not seem to be such a problem with the Main Experimental Data Set where solutions range from 13 to 90 stations, with over 90% distributed approximately uniformly in the 15 to 60 station range.

4.3 "Difficult" Data Set One

This data set was generated with low network density and high variability in task times to especially challenge the backtracking and optimal-seeking methods which have the best performance on the other two data sets. As can be seen from Tables 2 and 3, however, MAG-1 and the Hoffmann-based codes yield the best results on this data set, overall.

4.4 "Difficult" Data Set Two

This 'pathological' data set with all even task times and odd cycle times was also created to tax backtracking and optimal-seeking methods, but specifically HOFFMANN-0.0, which was expected to be unable to solve problems in this data set within the time limit specified. This expectation is based upon the notion that HOFFMANN-0.0 uses exhaustive enumeration of task assignments unless a zero idle time solution is first found. As is shown in Tables 2 and 3, HOFFMANN-0.0 was able to solve 69 of the 110 balances. However, along with MAG-1, the other Hoffmann-based procedures yield the five best average results reported.

4.5 Computer Time and Computer Storage Requirements

Table 4 indicates the actual computer time and storage used by the 24 methods. The times quoted are the average seconds of CPU time required by each method to solve, or attempt to solve, all balances in each data set. For each balance, the minimum of the actual CPU time and 3.0 seconds is used. These values are summed over all balances in the data set and divided by the number of balances in the data set to obtain the times appearing in Table 4. The storage values are given in kilobytes of primary storage required.

Although the values in Table 4 are accurate measurements, one should interpret them cautiously in evaluating the relative efficacy of each solution

TABLE 4

Computation Time and Amount of Computer

Storage Required to Solve Line Balancing Problems

Solution Procedure	Average CPU Time to Solve Each Balance in Each Data Set[a]				Primary Storage Required[b]
	Main Experimental Data Set 1,320 Balances	Literature Data Set 64 Balances	Difficult Data Set One 110 Balances	Difficult Data Set Two 110 Balances	
Each Single-Pass Decision Rule[c]	0.004	0.002	0.005	0.005	8
COMPOSITE-13	1.37	0.57	1.52	1.52	91
ARCUS	0.49	0.05	2.14	2.19	63
HOFFMANN-0.0	0.17	0.53	1.48	8.36[e]	251
HOFFMANN-0.5	0.18	0.35	0.28	0.29	251
HOFFMANN-1.0	0.18	0.29	0.22	0.24	251
HOFFMANN-2.0	0.18	0.29	0.19	0.20	251
MALB	1.85	0.07	1.57	1.60	175
MAG-1	1.48	0.03	1.15	1.18	1,711
MAG-2	1.87	0.03	1.64	1.82	1,711
ALBCUT	2.76	0.52	2.40	2.52	248
ALBHOFF[d]	1.67	0.91	1.76	1.91	172

[a]AMDAHL 470/V8 CPU time, in seconds with a 3.0 second time limit per balance. FORTRAN-H Compiler used, with OPT=2.
[b]In kilobytes.
[c]One computer program was used to find individual single-pass results and COMPOSITE-13 results. The storage value is an estimate, however, of what would be required to code any one single-pass decision rule.
[d]Half-word integers are used in the HOFFMANN-0.5 subroutine in ALBHOFF. This has the effect of increasing the computation time and decreasing the storage over the standard HOFFMANN-0.5 procedure.
[e]The location of the internal timer in this code permitted some balances for which no solution was found to be attempted for more than 3.0 seconds in this data set.

procedure. First of all, through clever programming one could reduce both storage and time requirements for any of the methods examined. Second, the methods vary in the predictability of their use of time and storage, which affects the potential expense and ease of their use. For example, it is not possible to specify precisely, and in advance, the amount of computer storage that will be required to solve a given problem with the MAG-1 and MAG-2 branch and bound procedures. (This is a generic problem with branch and bound codes, and has been observed elsewhere (Patterson, 1984).) Thus, the array dimensions, and hence storage values, had to be determined by trial and error (repetitively increasing dimensions until overflows disappeared). The two codes dropped from the study, DYNAMIC and MUST also have this drawback, although MUST has been programmed to directly use external storage, which

reduces the need to a priori specify storage precisely. None of the other 22 procedures evaluated have this drawback relative to storage needs. However, all methods other than the single-pass and composite methods (and the optimum codes with time traps) have difficult-to-predict time requirements for an arbitrary balance.

Within a given data set, it is fairly safe to compare the relative computation times of the methods. Across sets it is more risky, although in terms of number of tasks in each balance, the Main Experimental Data Set, and the "Difficult" Sets One and Two, are comparable. In creating the latter two data sets, it was expected that some of the methods, specifically the backtracking methods, would perform worse in terms of computation time. This is certainly true of HOFFMANN-0.0 on "Difficult" Data Set Two. In "Difficult" Data Set One, HOFFMANN-0.0 is about nine times slower than it is in the Main Experimental Data Set. However, relative to all other methods on "Difficult" Data Set Two, its time performance was average. The only other method that exhibits substantially increased time per balance is ARCUS.

5.0 A Comparison of These Results with Previous Investigations

There have been two previous major investigations of heuristics for solving the Type II line balancing problem [Mastor (1970) and Dar-El (1975)], but none, until the current investigation, for the Type I problem. Hence, it is difficult to make direct comparisons among these three evaluations. However, as indicated earlier, most methods for solving the Type II problem are in reality methods for solving the Type I problem that are repetitively applied to the same network with an increasing cycle time. This is certainly true for all of the methods included in our evaluation. Hence, we feel that the procedures examined will give the same relative performance for the Type II problem as they have for the Type I problem. In fact, by solving each

network using 11 increments of cycle time, we have in one sense mimicked the process needed to solve a Type II problem.

The major methods included in both our evaluation and Mastor's (with Mastor's terminology given in parentheses) are MINTSKNO (Lexicographical), MAXRPW (Helgeson), MAXIFOL (Immediate Follower), RANDOM (Random), HOFFMANN-0.0 (Hoffmann), and ARCUS (Arcus). Mastor also included a version of the Held, et. al. (1963), dynamic programming procedure. In general, the poorest results obtained by Mastor were for the Lexicographical rule, followed by the single-pass decision rules. The best performance was obtained by Held et. al., followed by Arcus and Hoffmann, although the Held et. al. and Hoffmann methods were tested over a subset of his data because both methods required excessive computation times. Dar-El included MALB, ARCUS, and the best of ten single-pass heuristics (10-SP) in his comparative evaluation. The ten single-pass methods are combinations of four procedures included in our investigation, MAXRPW, MAXTFOL, MAXDUR, and MAXIFOL. Dar-El excluded Hoffmann and Held et. al. from his investigation on the basis of Mastor's observation that these methods exhibit high computation times. Overall, Dar-El found that MALB consistently gave superior results to ARCUS, and that generally, ARCUS outperformed 10-SP.

All three investigations (Mastor's, Dar-El's and ours) are consistent in finding that the more complicated procedures give better solutions than the single-pass methods. Our results comparing MALB and ARCUS are not quite as conclusive as are Dar-El's. However, they are generally in agreement. More noteworthy are the differences among the results, the most important of which is the difference in the performance of HOFFMANN-0.0 in our study and the Hoffmann in Mastor's investigation. In FORTRAN, many decimal fractions are imprecisely represented in the computer. When these numbers are arithmetically

manipulated (e.g., normalized) and compared, this imprecision can give logically erroneous results. Specifically, by treating task times and cycle times as decimal fractions, the test in Mastor's version of the Hoffmann procedure to determine if a station's idle time is equal to zero is very rarely met thus forcing complete enumeration at each work station using this approach. This is in fact the situation we have artificially created in "Difficult" Data Set Two with all even task times and all odd cycle times. If one feels that these types of problems are likely to occur in balancing an industrial assembly line, then he or she should probably use one of the modified versions of the Hoffmann procedure (we prefer HOFFMANN-0.5). It has been our experience, however, that these instances arise rarely in practice, and one has to go to the extreme of normalizing the task times or else constructing a data set such as given in our "Difficult" Data Set Two in order to observe the HOFFMANN-0.0 procedure experiencing difficulty in deriving a satisfactory solution to a line balancing problem.

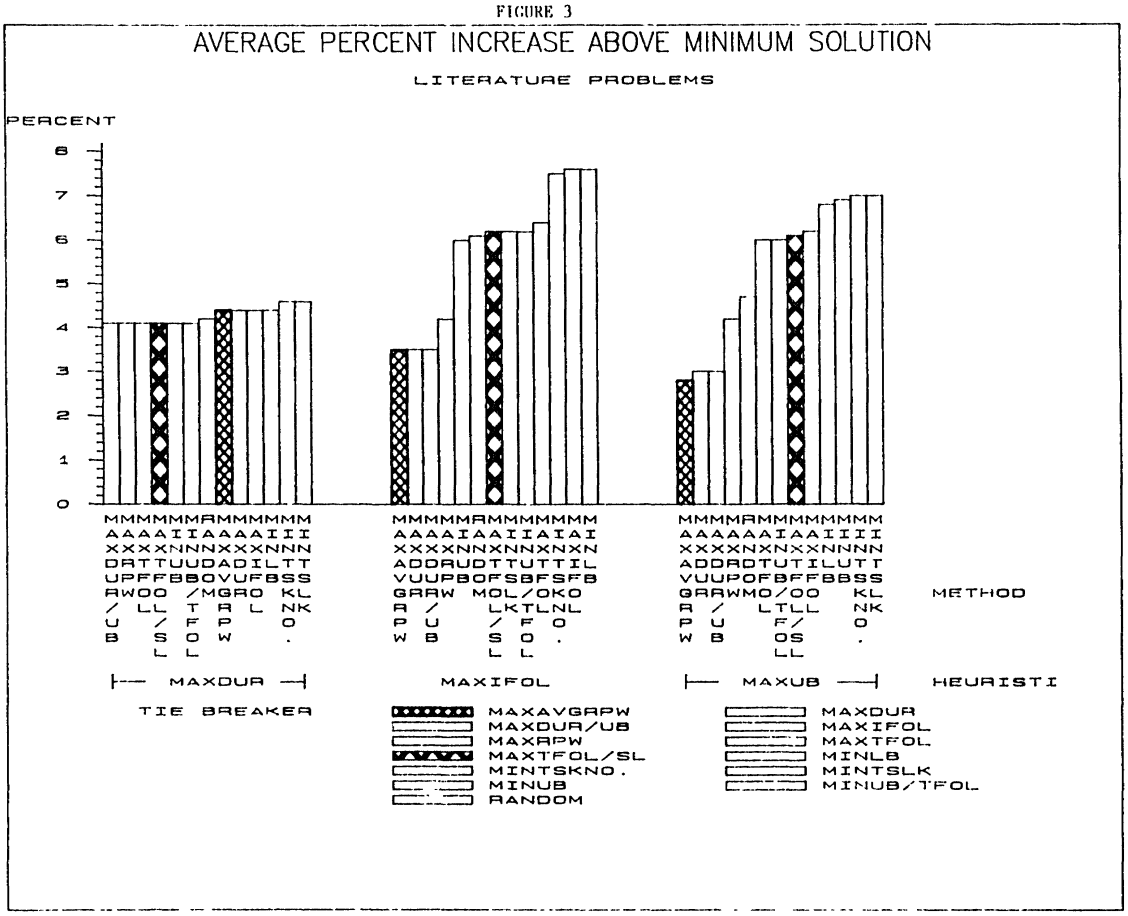
The less significant differences in the individual performance of single-pass decision rules among these three investigations may be explained by the lack of explicit (or at least stated) tie-breaking rules used by Mastor and Dar-El. We have observed that the effect of the tie-breaker rule can become quite significant, and that the tie-breaker's performance is highly dependent upon the main scheduling heuristic used. This is largely because of the heuristic discretion the tie-breaker is or is not able to exercise, a concept addressed previously for the project scheduling problem (Patterson, 1976). For example, when MINTSLK is the main scheduling rule employed, there are usually numerous opportunities for the tie-breaker to exercise its influence (discretion) because several tasks available for scheduling at a given station will possess equal amounts of slack. In this instance, the tie-breaker rule

becomes the dominating heuristic because of the inability of the main heuristic to discriminate among the competing tasks for assignment to a work station. The use of a heuristic such as RANDOM, on the other hand, produces relatively fewer opportunities to break ties since it is unlikely the random numbers assigned for task priority will be equal. To illustrate this point, in Figure 3 we give the Average Percent Increase Above the Minimum Heuristic Result Obtained using the single-pass scheduling heuristics of MAXDUR, MAXIFOL, and MAXUB with the other single-pass decision rules serving as tie-breakers for the literature problems data set. Apparent in this figure is that the performance of MAXDUR varies little regardless of the tie-breaker used because it leaves little discretion to the tie-breaker. However, the performance of MAXIFOL and MAXUB are strongly influenced by the tie-breaker employed because of the large number of ties these rules produce. Figure 3 also illustrates that a tie-breaker's effectiveness is a function of the main rule it is paired with. Two rules, MAXAVGRPW and MAXTFOL/SL, have been highlighted in Figure 3 to illustrate this point. If one were to evaluate these tie-breakers using a main scheduling heuristic such as MAXDUR, then one might conclude that MAXAVGRPW isn't a very effective tie-breaker. However, when paired with MAXIFOL and MAXUB, this tie-breaker actually delivers the best results. Differences in reported results previously may have been due simply to the manner in which ties are broken, or in the order in which tasks are labeled numerically as data input.

6.0 Conclusions and Recommendations

Twenty-six heuristic decision rules are evaluated in this paper, which are designed to assign tasks into work stations such that given a cycle time, or production rate, the number of stations is minimized. The procedures vary from simple list processing prioritizing schemes, which

FIGURE 3



consider a single attribute of each work task, to optimal-seeking procedures which have had the amount of time permitted for search limited. These methods are tested on four data sets, one large set consisting of 1,320 balances, one set of problems found in the open literature, and two specially constructed sets of problems.

A randomized complete block, full factorial analysis of variance was performed to assess the effects of number of tasks, network density, method of generating precedence relationships, and cycle time on heuristic performance. The results indicate that heuristic performance is not significantly affected by network structure, but that performance does vary with the magnitude of the cycle time.

In general, our evaluation demonstrates that methods exist which will give optimal or near-optimal solutions to large line balancing problems with very little computational effort. Of the pure heuristic methods, variants of Hoffmann's Precedence Matrix approach (1963), followed by Dar-El's MALB (1973), appear to be among the best procedures. Of the optimal-seeking methods, Magazine and Wee's branch and bound procedure (1981b) using the heuristic IUFFD (MAXDUR), and Talbot and Patterson's implicit enumeration procedure (1984) initialized with the Hoffmann procedure yield the best overall results.

On the basis of our analysis and experience in using the computer implementations of each of these procedures, we make the following general recommendations. If one is constrained in terms of computer storage and computational speed (for example as with a personal computer), then solving the problem with HOFFMANN-0.5 first, and then improving upon it with an always-feasible optimal-seeking approach such as ALBHOFF (with or without the additional cutting based rules) and using a time trap, would seem to return the best results for the effort expended. If computer storage is not a limiting factor, then MAG-1 (used with a time trap) is likely to give among the best results. To reduce the risk of not determining a good solution, multiple methods should be attempted. This is especially true for balances with cycle times close to the largest task time, which appear to be the most difficult for which to find and verify an optimal solution for a problem.

ACKNOWLEDGEMENTS

The authors wish to extend their appreciation to the following individuals who assisted in the effort required to complete this evaluation: Professor Michael Magazine and Dr. Thomas Wee provided us with copies of their branch and bound computer program, their version of MALB, and their version of Schrage and Baker's dynamic programming procedure. They additionally responded to a number of questions we had about the operation of these algorithms. Professor Ezey Dar-El provided us with a copy of MUST. Ms. Kay Miller and Mr. Ronald Howren of the University of Missouri-Columbia provided much help in coding, running programs, and preparing final results for analysis. Finally, the anonymous referees gave us a number of constructive comments on earlier drafts of this paper.

REFERENCES

[1] Arcus, A. L. "An Analysis of a Computer Method of Sequencing Assembly Line Operations." Ph.D. dissertation, University of California, Berkeley, 1963.

[2] _____. "COMSOAL: A Computer Method of Sequencing Operations for Assembly Lines." In Readings in Production and Operations Management, edited by E. S. Buffa. New York: John Wiley and Sons, 1966.

[3] Bowman, E. H. "Assembly-Line Balancing by Linear Programming." Operations Research 8 (May-June 1960), 385-389.

[4] Dar-El, E. M. (Mansoor). "Assembly Line Balancing--An Improvement on the Ranked Positional Weight Technique." Journal of Industrial Engineering 15, 2 (March-April 1964), 73-77.

[5] _____. "MALB--A Heuristic Technique for Balancing Large Single-Model Assembly Lines." AIIE Transactions 5, 4 (December 1973), 343-56.

[6] _____. "Solving Large Single-Model Assembly Line Balancing Problems--A Comparative Study." AIIE Transactions 7, 3 (September 1975), 302-310.

[7] _____, and Rubinovitch, Y. "MUST--A Multiple Solutions Technique for Balancing Single Model Assembly Lines." Management Science 25, 11 (November 1979), 1105-1115.

[8] Davis, E. W. and James H. Patterson. "Resource-Based Project Scheduling: Which Rules Perform Best?" Project Management Quarterly, 6, 4 (December 1975), 25-31.

[9] Fishburn, P. C., and Gehrlein, W. V. "Alternate Methods of Constructing Strict Weak Orders from Interval Orders." Psychometrika 39, 12 (1974), 501-16.

[10] Gehrlein, William V. "Generating Random Partial Ordering Relations." Proceedings, Northeast American Institute for Decision Sciences, Philadelphia, PA (April 1980), 119-22.

[11] _____, and James H. Patterson. "Balancing Single Model Assembly Lines: Comments on a Paper by E. M. Dar-El (Mansoor)," AIIE Transactions 10, 1 (March 1978), 109-12.

[12] _____. "Sequencing for Assembly Lines with Integer Task Times." Management Science 21, 9 (May 1975), 1064-70.

[13] Held, M., R. M. Karp, and R. Sharesian. "Assembly-Line Balancing--Dynamic Programming with Precedence Constraints." Operations Research 11, 3 (May-June 1963), 442-60.

[14] Helgeson, W. P., and D. P. Birnie. "Assembly Line Balancing Using the Ranked Positional Weight Technique." Journal of Industrial Engineering 12, 6 (November-December 1961), 394-98.

[15] Heskia, Heskiaoff. "An Heuristic Method for Balancing Assembly Lines." Western Electric Engineer 12, 3 (October 1968), 9-16.

[16] Hoffmann, Thomas R. "Assembly Line Balancing with a Precedence Matrix." Management Science 9, 4 (July 1963), 551-62.

[17] Jackson, J. R. "A Computing Procedure for a Line Balancing Problem." Management Science 2, 3 (April 1956), 261-72.

[18] Jaeschke, G. "Eineallgemaine Methods Zur Losung Kombinatorisher Probleme." Ablaufund planungforschung 5 (1964), 133-53.

[19] Kilbridge, M. D., and L. Wester. "A Heuristic Method of Assembly Line Balancing." Journal of Industrial Engineering 12, 4 (July-August 1961), 292-98.

[20] Magazine, M. S., and T. S. Wee. "Fast Algorithms for the Assembly Line Balancing Problem." Working Paper 149, University of Waterloo, Department of Management Science (June 1981a).

[21] _____ and _____. "An Efficient Branch and Bound Algorithm for an Assembly Line Balancing Problem--Part I: Minimize the Number of Work Stations." Working Paper 150, University of Waterloo, Department of Management Science (June 1981b).

[22] _____ and _____. "An Efficient Branch and Bound Algorithm for an Assembly Line Balancing Problem--Part II: Maximize the Production Rate." Working Paper 151, University of Waterloo, Department of Management Science (June 1981c).

[23] Mastor, Anthony A. "An Experimental Investigation and Comparative Evaluation of Production Line Balancing Techniques." Ph.D. dissertation, University of California, Los Angeles, 1966.

[24] _____. "An Experimental Investigation and Comparative Evaluation of Production Line Balancing Techniques." Management Science 16, 22 (July 1970), 728-45.

[25] Merten, P. "Assembly Line Balancing by Partial Enumeration." Ablaufund Planungforschung 8 (1967), 429-33.

[26] Moodie, C. L., and H. H. Young. "A Heuristic Method of Assembly Line Balancing for Assumptions of Constant or Variable Work Element Tirmes." Journal of Industrial Engineering 16, 1 (January-February 1965).

[27] Patterson, James H. "Project Scheduling: The Effects of Problem Structure on Heuristic Performance." Naval Research Logistics Quarterly 23, 1 (March 1976), 95-123.

[28] _____. "A Comparison of Exact Approaches for Solving the Multiple Constrained Resource, Project Scheduling Problem." Management Science 30, 7 (July 1984), 854-67.

[29] Sawyer, J. F. H. Line Balancing. Washington, D.C., Machinery and Allied Products Institute, 1970.

[30] Schrage, L., and K. R. Baker. "Dynamic Programming Solution of Sequencing Problems with Precedence Constraints." Operations Research 26 (May-June 1978), 444-49.

[31] Talbot, F. Brian, and James H. Patterson. "An Integer Programming Algorithm with Network Cuts for Solving the Assembly Line Balancing Problem." Management Science 30, 1 (January 1984), 85-99.

[32] _____, _____, and William V. Gehrlein. "A Comparative Evaluation of Heuristic Line Balancing Techniques." Working Paper No. 415 (Revised), The University of Michigan Graduate School of Business, Ann Arbor, Michigan, 1985.

[33] Tonge, F. M. A Heuristic Program of Assembly Line Balancing, Englewood Cliffs, NJ: Prentice-Hall, 1961.

[34] _____. "Summary of a Heuristic Line Balancing Procedure." Management Science 7, 1 (October 1969), 21-42.

Appendix

Listing of Literature Data Set

Note: The format of the problems listed is as follows:

1. Problem title, number of tasks (N), cycle times.

2 - N+1. Task time, immediate successor tasks.

One blank record separates consecutive problems.

```
 1    MERTEN'S 7-ELEMENT      7    6    7    8   10   15   18
 2        1  2  4
 3        5  3  5
 4        4
 5        3  7
 6        5  6
 7        6
 8        5
 9
10    BOWMAN'S PROBLEM        9   20
11       11  2
12       17  3  4
13        9  5  6
14        5  6
15        8  7
16       12  8
17       10  9
18        3  9
19        0
20
21    JAESCHKE'S 9-ELEMENT    9    6    7    8   10   18
22        5  2  3
23        3  4
24        4  4
25        5  5  6  7
26        4  8
27        5  9
28        1  9
29        4  9
30        6
31
32    JACKSON'S 11-ELEMENT   11    7    9   10   13   14   21
33        6  2  3  4  5
34        2  6
35        5  7
36        7  7
37        1  7
38        2  8
39        3  9
40        6 10
41        5 11
42        5 11
43        4
44
45    DAR-EL'S PROBLEM       11   48   62   94
46        4  4
47       38  4  5
48       45 11
49       12  6
50       10  7
51        8  8
52       12  9
53       10 10
54        2 10
55       10 11
56       34
57
58    MITCHELL'S 21-ELEMEN   21   14   15   21   26   35   39
59        4  2  3
60        3 21
61        9  4
62        5  5 21
63        9  6  7
64        4  8
65        8  8 14
66        7  9
67        5 10 11 12 13
68        1 15
69        3 15
70        1 15
71        5 17 18
72        3 19
73        5 16 18
74        3 17
75       13 20
76        5 19
77        2
78        3
79        7
80
81    HESKIAOFF'S 28-ELEM.   28  138  205  216  256  324  342
82       70  3  4  5 21 22 23  8 19 24 26
83       59  6 17
84       33 28
85        6 28
86        1 28
87       27  7
88       17 18
89       62  9
90       31 10
91       53 11 12
92       21 15
93       19 13
94      108 14 16
95       52 28
96        5 28
97        8 28
98       97 28
99        8 28
100      47 20
101      67 28
102      17 28
103       8 28
104       3 28
105      21 25
106     107 28
107       3 27
108       2 28
109      72
110
111   SAWYER'S 30-ELEMENT    30   25   27   30   36   41   54   75
112       8  4  5
113       7 11 12
114      19 16 17
115      10  7
116       2  6
```

```
117        6  7
118       14  8
119       10  9
120        1 26
121        4 24
122       14
123       15 13
124        5 14
125       12 15 20
126        9 22
127       10 20
128        2 18
129       10 19
130       18
131       16 21 24
132       21 22
133       14 23
134       16 27
135        7 25
136       17 26
137        9 27
138       25 28 29
139        7
140       14 30
141        2
142
143    KILBRIDGE & WESTER      45   57   79   92  110  138  184
144        9  3  7  0
145        9  4  8  0
146       10  5  0
147       10  6  0
148       17  9  0
149       17 10  0
150       13  9 14  0
151       13 10 14  0
152       20 41  0
153       20 41  0
154       10 13  0
155       11 13 37  0
156        6 14 15  0
157       22 29 30 31 32 25 17  0
158       11 16 18 23 24  0
159       19 19  0
160       12 26 27  0
161        3 19  0
162        7 20 33  0
163        4 21 00
164       55 22 00
165       14 28 00
166       27 33  0
167       29 33  0
168       26 26  0
169        6 38  0
170        5 28 33  0
171       24 38  0
172        4 41  0
173        5 41  0
174        7 41  0
175        4 41  0
176       15 34 35 36  0
177        3 38  0
178        7 40  0
179        9 38  0
180        4 43  0
181        7 40  0
182        5 41  0
183        4 41  0
184       21 42  0
185       12 44 45  0
186        6  0
187        5  0
188        5  0
189
190    TONGE'S 70-ELEMENT      70  176  364  410  468  527
191       17  2 41 69 70
192       66  3
193       54  4 68
194       52  6  7
195        6  6 24 30
196       88  8
197       21  8
198      128 12
199       68 10
200       70 11
201       85 12
202       21 13 14
203      134 23
204      135 23
205       94 16
206       90 17 18
207       50 19
208      143 19
209       19 20 22 57
210       54 21
211       50 23
212       40 23
213       73 25 31 33
214       12 25
215      152 26 27 28 29
216       42 35
217       45 35
218       74 35
219       26 35
220       11 31
221       31 32
222       50 35
223      102 34
224       46 35
225       35 36 51 48 44 53 56 60 62 61
226       40 37
227        2 38
228        1 39
229        3 40
230       13 42
231       16 42
232       25 43
```

```
233        21 50
234        43 45
235        30 46
236        83 47
237        89 50
238        56 49
239        59 50
240        43
241        11 52
242        26 54
243        44 54
244       121 55
245        38
246        68
247        22 58
248         7 59
249        16 60
250        32
251        25 65
252        27 63
253       156 64
254        28 65 66 67
255        15
256        26
257        18
258        72
259        23
260        27
261
262     ARCUS' 83-ELEMENT      83 5048 5853 6842 7571 8412 889810816
263      1673  2
264       985  3  4  5
265      1836  6
266       973  6  7
267      1700  8
268      2881  9 10
269      2231 11
270      1040 77 78
271      1793 12
272      1250 13 14 25
273       700 15
274       464 16
275       500 17 18 20
276      1133 19
277       577 20 39
278       483 77 78
279       880 21 22 28
280       667 23
281       600 24
282       233 26
283       408 27
284       847 27
285       767 74
286       850 28 29
287       780 32
288       912 74
289       748 69
290      1863 32
291       714 30
292      1004 31
293       713 39
294       642 33 34 35 36
295       629 37
296      1234 77 78
297      1143 77 78
298      1266 38 39
299       792 40
300      1251 41
301      1310 42 43 44 75
302       663 77 78
303       494 45
304      1288 77 78
305       792 77 78
306       578 46
307       594 47
308       578 48
309       622 49
310       578 50
311       564 69
312       578 51
313       578 52
314       578 53
315       578 54
316       578 55
317       578 56
318       578 57
319       578 58
320       578 59
321       578 60
322       578 61
323       578 62
324       578 63
325       578 64
326       578 65
327       578 66
328       578 67
329       578 68
330       578 74 75
331       467 70 71
332       887 72
333       396 73
334      1296 73
335      1100 74 75
336      2543 76
337       764 76
338       357 77 78
339       701 79
340      1164 79
341       286 80 81
342      2100 82
343       450 83
344      1300 83
345      3691
346
347     ARCUS' 111-ELEMENT     111 5755 884710027107431137817067
348      1960  2
```

349	1715	3								
350	735	4								
351	1715	5	6	7	8	9	10			
352	490	39								
353	1225	39								
354	169	83								
355	2252	71								
356	1225	32								
357	2319	11	12							
358	1715	13	14	15	16	17	18	19	20	21
359	980	13	14	15	16	17	18	19	20	21
360	735	71								
361	2281	22	23	24	25					
362	2750	26								
363	77	27								
364	89	28								
365	51	29								
366	364	30								
367	405	91								
368	3060	111								
369	125	31	83							
370	3429	32	33							
371	43	69	70							
372	3430	34								
373	1960	82								
374	29	35								
375	27	36								
376	15	37								
377	121	38								
378	1715	39								
379	2127	41								
380	1470	111								
381	4037	42								
382	68	43								
383	62	44	91							
384	42	45	91							
385	364	46	91							
386	4998	40								
387	1470	111								
388	2963	69	70							
389	5689	47								
390	68	48	49	91						
391	18	50								
392	10	51								
393	81	52								
394	5200	54	55	56	57	58	59	60		
395	39	53								
396	67	91								
397	27	111								
398	15	111								
399	121	111								
400	58	111								
401	1715	69	70							
402	125	61	62	63						
403	4010	63	64							
404	1470	65	91							
405	1470	66	91							
406	2303	67	91							
407	1960	68	91							
408	2205	69	70							
409	4018	71								
410	2744	111								
411	2999	72								
412	735	111								
413	735	111								
414	735	111								
415	735	111								
416	545	77	78							
417	3386	73								
418	3234	91								
419	2205	74	91							
420	2206	75								
421	490	76								
422	825	77	78	91	79					
423	3528	80	81	82						
424	3568	83								
425	1200	84								
426	618	85								
427	1470	86	91							
428	1715	87	91							
429	735	111								
430	1960	91								
431	2889	88	89	91						
432	618	111								
433	490	111								
434	735	90								
435	490	105								
436	921	105								
437	326	111								
438	5390	92	93	94						
439	243	95								
440	371	95								
441	58	95								
442	5059	96	97	98	99	100	104			
443	1225	101								
444	769	102								
445	768	103								
446	1670	111								
447	1670	111								
448	490	105								
449	202	106	107							
450	203	107	108							
451	202	111								
452	2744	111								
453	162	109								
454	324	111								
455	162	110								
456	121	111								
457	162	111								
458	91									
459										
460										