SURVEY, DEVELOPMENT, AND APPLICATIONS OF ALGORITHMS FOR SEQUENCING PACED ASSEMBLY LINES

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

Ahmet Bolat
Department of Mechanical Engineering
King Saud University
Riyadh, Saudi Arabia

Technical Report 88-13

December 1988
Revised December 1989

SURVEY, DEVELOPMENT, AND APPLICATION OF ALGORITHMS FOR SEQUENCING PACED ASSEMBLY LINES[1]

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

Ahmet Bolat
Department of Mechanical Engineering
King Saud University
Riyadh, Saudi Arabia

December 1988
Revised December 1989

[1]This research was supported in part by a research contract from a U.S. automobile manufacturer to the University of Michigan.

SURVEY, DEVELOPMENT, AND APPLICATION OF ALGORITHMS FOR SEQUENCING PACED ASSEMBLY LINES

ABSTRACT

This paper focuses on sequencing issues for paced assembly lines on which many different models of a product are assembled. We provide a critical review of the literature, emphasizing objectives and algorithms designed for these problems. We then propose a new approach that can be generalized to solve a class of problems with one of several different objectives. For two commonly-used objectives, the procedure is evaluated by comparing it with three existing procedures and with lower bounds on the optimal objective value. The results indicate that this new, flexible, procedure performs very well.

SURVEY, DEVELOPMENT, AND APPLICATION OF ALGORITHMS FOR SEQUENCING PACED ASSEMBLY LINES

1. INTRODUCTION

Research on sequencing jobs for paced assembly lines spans three decades, and a variety of different procedures, most of them heuristic, have been developed. Paced assembly lines have some special characteristics that are important to point out explicitly. First, there is one job in each position on a constant speed conveyor. Normally, these positions are equally spaced, so that one new job is introduced to the line at equal time intervals. This is referred to as *fixed rate launching*, and this is the case that we consider here. If the conveyor permits unequal spacing between jobs, it is possible to use variable rate launching.

Second, there are no offline buffers between stations. Moreover, except on rare occasions, jobs are not removed from the assembly line until they reach the end of the line. Thus, each station must receive the same sequence of jobs.

Third, each station corresponds to a particular (continuous) portion of the assembly line in which an individual or crew is assigned to work. Without loss of generality, we assume that the conveyor flows from left to right. The stations are called left (right) closed if the operators are not permitted to perform work beyond a specified left (right) boundary. Otherwise, they are referred to as open to the left (right). We consider the case where stations are closed to the left and right. Most models assume that the stations may overlap as long this does not cause undue interference between operators in adjacent stations.

This paper provides a critical review of the literature with an emphasis on approaches that, at least in concept, can accommodate the scheduling of many different models of a product that are intermixed on the same assembly line. Such problems arise frequently in the automobile industry, where each body style may have millions of distinct models because of customer choices regarding colors and optional equipment. (The reader

is referred to Schonberger (1982) for a discussion of the strategic and quality-related advantages of the simultaneous assembly of many different models on the same line.) We then propose a new approach that can be generalized to solve a class of problems with one of several different objectives. For two commonly-used objectives, we evaluate the new procedure by comparing it with three existing procedures and with lower bounds on the optimal objective values.

The remainder of the paper is organized as follows. In section 2, we present a review and critique of the literature. The problems that we address and our approach to them are described in section 3. Section 4 contains an experimental comparison of our approach with several existing procedures and some concluding remarks.

2. LITERATURE REVIEW AND CRITIQUE

Before presenting a literature review, it is useful to describe some of the objective criteria that have been used in the literature. Thomopoulos (1967) describes several factors (or labor "inefficiencies") that might be of concern: idleness, work deficiency, utility work, and work congestion. Idleness occurs when an operator has completed all the work in his station and must wait for the next job to arrive. (This can occur only in stations that are closed to the left.) Work deficiency occurs in open-to-the-left stations when the operator crosses the left boundary of the station to find a job on which to work.

Utility work occurs when an operator in a closed-to-the right station cannot complete the necessary work before the job leaves the station, and another (utility) worker is assigned to complete the work. This is accomplished by assisting the operator at the station, or by performing the additional work at a downstream repair station. Work congestion occurs in open-to-the right stations when the operators need to cross the right boundary of the station in order to complete work on a job.

Macaskill (1973) defines these inefficiencies somewhat differently. He assumes that the station has specified boundaries, but that beyond these boundaries, there are

upstream and downstream allowances which represent areas in which work deficiency and work congestion, respectively, occur. The operator is not permitted to move beyond the allowance regions, and thus, the station is assumed to be closed to the left and right. As a consequence, unlike in Thomopoulos's definitions, work deficiency and congestion can occur in closed stations.

Some researchers also have included setup costs as part of the objective. These costs might be incurred when a sequence switches from a one model (or group of models) to another model (or group of models). For example, in the automobile industry, setup costs are incurred in the painting process when the color is changed.

Most of the other objectives in the literature are functions of the four inefficiencies described above and of setup costs. The range of objective criteria reflects the variety of perceptions that exist about what factors are controllable, and various assumptions about operating policies. We describe these assumptions in more detail below.

Wester and Kilbridge (1964) pose a bicriterion problem with minimization of idle time as the primary criterion and minimization of the amount of work congestion as the secondary consideration. They assume that for each model, the total work content can be distributed equally among the stations (i.e., the assembly line is perfectly balanced for each model). This assumption reduces the problem to a single-station problem, since the assumption of perfect balance makes the stations identical. However, because the models differ, a scheduling problem still remains. They assume that the cycle time, which is the time between consecutive job launchings, is a decision variable. By deriving a set of constraints that ensure zero idle time, they derive the optimal cycle time and sequence of jobs. The amount of work congestion is controlled (although not necessarily minimized) through a greedy procedure in which the model with the smallest amount of work congestion is chosen when the congestion is positive. The solution procedure implicitly places an infinite cost on idle time and a finite positive cost on work congestion. Since the cycle time can be chosen, the production rate of the system cannot be prespecified.

Thomopoulos (1967) presents an approach which might be described as a greedy procedure to minimize the total cost of labor inefficiencies. For each position in the sequence, the inefficiency cost incurred by placing each model in that position is computed, and the model with the lowest cost is chosen. The procedure starts with the first position in the sequence, then sequentially considers each of the following positions. For one problem instance with six different models, the procedure performs well in comparison to five hundred randomly generated sequences.

In the same article, Thomopoulos then uses an example to show that inefficiency costs can be reduced by partitioning the jobs into groups such that the jobs of each model type are spread evenly among the groups. The greedy sequencing procedure is applied to each group. He also demonstrates by way of an example that the total inefficiency cost is a convex function of the size of the groups, and that the total cost increases exponentially as the groups become very small.

Macaskill (1973) describes a modification of Thomopoulos's method in which a job with low inefficiency cost is followed by a job with relatively high work content as long as that job does not incur any utility work costs. This modification prevents high work content jobs from being concentrated in the latter part of the schedule. Implicit in the modification is the assumption that utility work is the most expensive inefficiency and should be avoided whenever possible.

Dar-El and Cother (1975) address the problem of sequencing so as to minimize the overall length of the assembly line while ensuring that there is no interference (i.e, utility work, idle time, congestion, etc.). They propose a heuristic which begins with a lower bound on the length of each station (which is equal to the maximum processing time of all models at that station) and increases the station length whenever an infeasibility occurs. The sequencing procedure is based upon the objective of spreading out the jobs within each model type as smoothly as possible. At any point in the sequence, the rank of a model is computed as the difference between the number of jobs of that model that *should* have been

sequenced thus far (if all models were spread out perfectly) and the number of jobs of that model *actually* sequenced. The models are considered in descending order of this ranking, and feasibility with respect to the constraint of no interference is checked. The first model that passes the feasibility check is selected for the given position in the sequence. If no model passes the feasibility check, the station lengths are increased by equal amounts and the process repeats.

Dar-El and Cother indicate that this approach avoids the issues of defining station limits and penalties. Implicitly, their procedure places an infinite cost on all inefficiencies, and assumes that station lengths can be specified. For the case of fixed station lengths, they indicate that the problem should be one of minimizing idle time subject to the constraint of no work congestion or utility work. However, their procedure still attempts to spread out each model as evenly as possible.

Dar-El and Cucuy (1977) present a procedure to minimize the length of the assembly line while ensuring no idle time. (Note that idle time implicitly has an infinite cost.) As in the Wester and Kilbridge paper, they assume that the total assembly time for each model can be allocated equally to all stations, so this reduces to a single-station problem. The procedure involves generating all feasible subsequences of models that have the *regenerative* property. In their paper, a subsequence is called regenerative if the location of the operator within the station is exactly the same at the beginning and end of the subsequence. (Regeneration is defined somewhat differently in other papers.)

All possible regenerative subsequences with a selected starting point are generated. These subsequences can be scheduled in any arbitrary order and the overall sequence remains feasible. If one or more feasible regenerative subsequences is found, the number of subsequences of each type needed to satisfy production requirements is chosen by solving an integer program (IP). This process is repeated for each possible regeneration point. If a solution to the integer program cannot be found for any regeneration point, the station length is increased and the process repeats.

Note that this procedure restricts the solution to a series of regenerative subsequences. This simplifies the problem but also has some disadvantages. The additional constraint imposed on the problem may lead to suboptimal solutions for several reasons. There may be relatively few regenerative subsequences, which makes it difficult to find a feasible solution to the IP. In addition, some of the regenerative subsequences may be quite long, which not only makes them difficult to generate, but also contributes to difficult-to-solve IPs. Finally, the optimal solution may not contain a regenerative subsequence, so this procedure may increase the station length unnecessarily.

Okamura and Yamashina (1979) propose a heuristic method to minimize the maximum distance that a worker must go from the origin of the station to complete work on all jobs. The heuristic involves moving jobs from one location to another in the sequence, or interchanging two jobs, to reduce the maximum distance. Candidate jobs to be moved or interchanged are selected from the regeneration cycle which has the maximum distance from the origin. Here, a regeneration is defined as an instant when the operator returns to the *origin* of his station and finds no work remaining to be done on jobs in the station. In the case of interchanges, the other candidate job may be selected from a regeneration cycle with the smallest maximum distance from the origin.

They suggest a procedure which uses many different initial sequences. For each initial sequence, an improvement routine is applied in which jobs are moved until no improvement occurs, followed by an interchange of jobs until no improvement occurs. The best of the several sequences is the solution. Okamura and Yamashina present empirical results for problems with up to 100 jobs which suggest that the heuristic performs almost as well as a branch and bound procedure with a CPU time trap of two seconds.

The Okamura and Yamashina procedure described above applies to a single-station problem. For multiple stations, they propose an objective which is a weighted sum of the maximum distances for the individual stations. Unfortunately, CPU times for the multi-

station heuristic increase rapidly with the number of stations, and the estimated probability that the heuristic finds the optimal solution declines rapidly as well. The authors do not report on the quality of solutions for the multi-station problem.

Monden (1983) describes a "goal-chasing" procedure in which the sequencing objective is to smooth out the rate of use of each of several component parts as much as possible. For each candidate job, the squared difference between the target number of jobs requiring a part (which assumes perfectly smooth usage) and the actual number of jobs containing the part is computed for each part, and summed over all parts. The job that minimizes the sum of squares is selected for the next position in the sequence. Monden also presents a simplified version of the procedure which uses the sum of the differences rather than the sum of the squared differences. The simplification applies to situations in which each model uses an equal number of parts as well as an equal number of each type of part. He indicates that in practice, Toyota incorporates weights on important subassemblies into the formulas, as well as constraints on facilities, in selecting the job sequence.

Coffman et al. (1985) discuss a procedure that was developed for the purpose of resequencing jobs after the original sequence has been scrambled by an upstream repair process. This problem differs from the others described earlier because the jobs arrive dynamically; thus, the problem is not static. The objective is to minimize the number of violations of *spacing constraints*. Spacing constraints specify rules that no more than k out of n consecutive jobs may have a particular customer-selected option, and there may be more than one constraint associated with a particular option. For example, one can simultaneously specify that no more than two out of every five consecutive jobs may have an option *and* that two jobs with the option should not be sequenced consecutively.

Their proposed procedure is a greedy heuristic that assigns each arriving job to the first available position in the sequence for which the job satisfies all the spacing constraints. (The actual problem is further complicated by the form of the resequencing buffer. It may be physically impossible to position a job so that it can assume the first available and

feasible position in the sequence. Thus, a job actually would be placed in the first available, feasible, and accessible position in the sequence.) One drawback of the procedure is that it does not guarantee that all positions in the sequence will be filled. Yet, in practice, the jobs are drawn out in ascending order, with no gaps in the sequence. One result of this is that spacing constraints indeed are violated, and the nature and extent of the violations are not easy to predict in advance.

Burns and Daganzo (1985) view the sequencing problem as one one minimizing the setup costs incurred when the operators and/or machines on the assembly line switch from one type of operation to another (e.g., from one job with an option to one job without, or from one color to another). They show that there is an optimal grouping hierarchy in which jobs are first grouped to minimize the total setup cost on the operation with the highest setup cost. Then, within each of these groups, the jobs are further grouped to minimize the total setup cost of the operation with the next highest setup cost, and so forth.

They also consider a scenario in which there is one operation with a setup cost and one (possibly different) operation with a "capacity cost." The capacity cost is the cost of providing sufficient capacity, whether assembly workers or machine, or both, to ensure that the spacing constraint is not (or rarely) violated. For the case in which the option (operation) choices are independent, Burns and Daganzo suggest that the capacity cost is convex increasing in the number of jobs between setups. Thus, since the total setup cost is convex decreasing in the number of jobs between setups, the best solution is some finite number of jobs between setups.

We comment here that the tradeoff between setup costs and capacity cost is especially important when the choice of the option associated with the setup cost is positively correlated with the choice of the option associated with the capacity cost. For example, assume that there is a setup cost for changing paint colors, and a capacity cost for the installation of an engine on an automobile. Now suppose that customers choosing a turbo-charged engine tend to prefer cars painted red. Long subsequences of red cars will

reduce setup costs, but will increase capacity costs because of the concentration of turbo-charged engines, which have a larger than average work content. On the other hand, short subsequence of red cars will have a larger total setup cost but relatively smaller capacity costs.

Bird (1986) analyzes the mean time until a single spacing constraint is violated by using a Markov chain model with an absorbing state. As in Coffman (1984) the spacing constraint states that no more than k out of the next n jobs may have the specified option. He assumes that jobs are sequenced randomly; thus, for each position in the sequence there is a specified probability that the job in that position will have the option under consideration. In addition, he assumes that the assembly process is such that a job with the option is processed with probability k/n, and a job without the option is processed with probability 1 - k/n.

The system is modeled as if there were a finite storage space for n jobs from which jobs are selected for the next position in the sequence. Bird assumes that all jobs are accessible and that the storage space is always full. He defines the state of the system as the number of jobs in the storage area without the option. He also assumes that a spacing constraint is violated as soon as the finite storage space contains only jobs with the option. With this state definition and the assumption about when the spacing constraint is violated, the absorbing state is state zero. For the stated assumptions, Bird shows that the mean time to violation of a set of constraints is achieved by maximizing the ratio k/n for each option.

We note that the spacing constraint actually is not violated until the storage space has n jobs with the option *and k out of the last n jobs in the sequence are jobs with the option*. If fewer than k out of the last n jobs in the sequence have the option, it is still possible to select one job from storage. Thus, it appears that a more complex state definition is necessary to model the actual time to violation of a constraint. In order to model the state transitions and time to violation accurately, one not only needs to know how many of the last n jobs have the option, but also the type of job (with or without the

option) in the first position within that subsequence. This means that, effectively, one must store whether or not each of the last n jobs has the option. Bird's model can be generalized to account for this, but the resulting state space becomes quite large. (There are $2^n + 1$ states.) Moreover, with this new state definition, it would not be reasonable to model the assembly process as randomly completing a job with or without the option, since the presence or absence of the option in the last n jobs is known.

Yano and Rachamadugu (1987) present a heuristic algorithm for a multiple station problem with the goal of minimizing total utility work. Each station may offer two types of operations, one for a "basic" job and one for a job with a particular option. The algorithm is a greedy procedure which selects the best job for each position in the sequence, but differs from the other approaches in that the criterion is the utility work for the job under consideration *plus a lower bound on utility work for the remainder of the sequence*. The lower bound is determined by using regenerative sequencing to determine an optimal sequence for each station (or option) separately, then summing the objective values over all stations.

Parrello, Kabat and Wos (1988) describe an expert system to solve a problem in which the goal is to satisfy several constraints of the "k out of n" form and of the form "at least n spaces between two jobs with an option" as closely as possible. Penalties are assigned according to the severity of the impact of the violation of a constraint. The penalty for a job is the sum of the penalties incurred at all relevant stations. They initially suggest an algorithm in which one position at a time (starting from the beginning of the sequence) is filled by the job that produces the smallest total penalty. They point out, however, that without additional considerations, such a procedure results in *cherry picking*, that is, selecting "easy" jobs early in the sequence and leaving difficult jobs until the end of the sequence. For this reason, they suggest that ties (with respect to total penalty) be broken by considering the difficulty of the job. Each option is assigned a difficulty factor and the difficulty factor of the job is the sum of the applicable difficulty factors.

Details of the difficulties of developing and testing the expert system described above are discussed by Parrello (1988). He explains that even finding the best job for each position was difficult, and that the team eventually resorted to brute-force approaches such as enumeration to solve small problems to provide a basis for generating good sequences. They discovered that spreading "bad" (difficult) jobs out evenly and then sequencing the remaining jobs using the approach given above appeared to produce good solutions. Ultimately, the system was never used because the client decided to use a cost formula which could not be modeled with the software that had been developed. The cost penalty was a (symmetric) quadratic function of the distance between two jobs with the same option, with the minimum of the cost function occurring at the desired spacing.

Penalties for too-far-apart spacing effectively serve to reduce the amount of cherry picking in a procedure that sequences one job at a time. One interesting characteristic of a symmetric quadratic cost formula, however, is that it penalizes too-close-together spacing in the same way as too-far-apart spacing. It is not clear whether this reflects the company's actual cost function or whether it is used as a proxy for a more complicated objective function.

Miltenburg (1989) considers the problem of finding a "level" schedule which minimizes the variation in the usage rates of parts when there is a limited number of models, and the demand rate for each is constant. He considers four different metrics for "levelness" which are similar to those of Monden (1983) and shows that the sequence closest to a level schedule cannot be guaranteed to be feasible because it can lead to negative production (i.e., destruction of jobs). Miltenburg develops several different procedures to find solutions, all of which are myopic in the sense that they look ahead at most two jobs. For problems with a large number of jobs, the proposed solution procedures are based on the his assumption that the optimal sequence is cyclic, that is, the sequence consists of a short sequence which is repeated indefinitely.

Analysis of Existing Procedures

The Wester and Kilbridge, Thomopoulos, Dar-El and Cother, and Dar-El and Cucuy articles focus on the issue of eliminating idle time and permit flexibility with regard to the station length(s) and/or cycle time. On the other hand, the later papers take the cycle time and station lengths as given, and concentrate on smoothing out the flow of work. Thus, the two sets of articles are based upon different assumptions about what factors are controllable.

All of the articles directly or indirectly address issues related to efficient use of labor. It should be pointed out that for any set of N jobs, avoidable idle time will be reflected in the amount of utility work . Thus, the problem of minimizing total utility work is the same as the problem of maximizing labor utilization or equivalently, minimizing idle time. (This can be shown formally, but we will not do so here.) From this, it should be clear that if a linear objective is used, it is not necessary to penalize both utility work and idle time. The primary differences among the papers with linear objectives lie in the constraints on idle time. Some models have constraints that there be no idle time whereas others simply attempt to minimize it, but do not constrain it to be zero.

With work deficiency and work congestion, scheduling issues become more complicated because increasing work deficiency normally causes a decrease in work congestion. However, the more recent articles, many of which are based upon actual applications, do not distinguish between idle time and work deficiency, nor do they distinguish between work congestion and utility work. It appears that it is difficult to make such distinctions in realistic settings. On the other hand, it is not well understood how inclusion of these factors influences the "quality" of schedules, so perhaps more research on this issue is needed.

The only attempt at implementing non-linear penalties (Parrello) appears to have been unsuccessful. One might argue that these non-linear penalties could be used to

capture the differential effects of work congestion and utility work (and similarly, work deficiency and idle time). Here, also, further research is needed to achieve workable approaches.

Other Possible Procedures

In theory, most of problems described above, and the problems that we address, can be solved optimally by dynamic programming. However, only small problems can be solved optimally. For example, Yano and Rachamadugu (1987) describe such a procedure for the objective of minimizing total utility work, and indicate that the computational complexity of the procedure increases as the square of the number of jobs and exponentially with the number of stations. Furthermore, it has a very high proportionality constant which depends on the problem data, and the size of the state space appears to be unmanageable even for relatively small problems.

Another possible approach is Branch and Bound (implicit enumeration), which has been used extensively in the scheduling literature. Within this framework, a depth-first search (i.e., last in, first out) node selection strategy is often used in order to generate feasible solutions quickly, thereby reducing computing times. Unfortunately, in most of the problems described above, many different sequences are possible, and in the problems that we address, every sequence is feasible (although not necessarily good). Moreover, unless tight lower bounding procedures are available, it is unlikely that many alternatives would be eliminated by a feasible solution generated early in the search.

Heuristic implementations of Branch and Bound procedures also can be applied to many of these problems. Examples include filtered beam search, where only a few "best" nodes are retained at any point in time (see, for example, Ow and Morton 1988), and truncated Branch and Bound, where the search is terminated after a specified number of nodes have been investigated or a CPU time limit has been exceeded.

In the next section, we describe a procedure which is based upon a fairly flexible linear objective function. Although it is a sequential procedure (as are many of the others), it has a look-ahead feature that provides for significant improvements in solution quality over similar procedures without look-ahead. We selected this approach after eliminating from consideration both dynamic programming (because of computing time requirements), and branch and bound approaches. In investigations with many small problems, we found that our sequential procedure obtained solutions that were nearly as good as optimal solutions obtained from Branch and Bound (see Bolat 1988). Thus, we decided to focus on sequential procedures which offer the combination of simplicity, flexibility, and computational efficiency.

3. A FLEXIBLE HEURISTIC PROCEDURE WITH LOOK-AHEAD

The development of this procedure was motivated by the management of a major automobile manufacturer, who believed that the company's existing sequencing algorithm could be improved. The company's algorithm might be viewed as a generalization of the simplified version of Monden's procedure (with a linear objective). We cannot disclose further details of the algorithm because of confidentiality considerations.

We had several goals in developing a new procedure. First, it had to be relatively efficient from a computational standpoint because approximately 1000 vehicles must be sequenced every day for each of several automobile assembly facilities. The existing procedure required approximately 30 minutes to sequence 1000 jobs on a minicomputer, and this was viewed as an upper limit on the amount of time available to sequence one day's production for one facility. The computer was used for functions other than sequencing during approximately two shifts, and sequences for approximately ten assembly facilities had to be generated during the third shift, on each of five or six nights per week. Although 30 minutes may appear to be a substantial amount of computing time, much of it was required for updating the list of orders available for scheduling, for transfer

of the sequence to other computer programs and other computers to support various functions, and routine input and output. Basic data management and data transfer are time-consuming in this context because of the massive amount of information required to specify each order.

Second, it had to alleviate the major perceived weaknesses of the existing algorithm: (i) inadequate consideration of the relative importance of each option with regard to workload smoothing; and (ii) a moderate amount of cherry picking. Since the existing algorithm was similar to Monden's, it tended to give highest priority to jobs that included particular options for which the cumulative usage lagged far behind the respective targets. The resulting schedule tended to overload stations connected with these options and virtually ignore the other stations. The cherry picking was a consequence of the procedure being myopic; the only look-ahead within the procedure was implicit in the target usage rates, and this was frequently insufficient.

Third, the procedure had to be applicable to all of the assembly facilities, since it was considered impractical to use different algorithms for different facilities. Some assembly plants produced several different body types with a large number of different options, while other plants had a much narrower range of combinations. At some plants, the labor content of the most complex vehicle was several times that of the least complex, while at others, the variation in labor content was not significant. Thus, the desired amount of "smoothing" differed considerably among the plants. Yet, each needed a schedule produced by the same algorithm. We next explain how our procedure evolved into its current form.

With these ideas in mind, we first sought to develop a reasonable objective function starting from first principles. We also made a conscious decision to keep the objective function simple. One key advantage of simple objective functions is that the effects of modifications are relatively predictable. In addition, for practical reasons, we wanted an objective function that would be easily explained to and readily accepted by management.

Through discussions with managers at corporate headquarters and at assembly facilities, we discovered that the primary concern was to ensure that assembly workers had enough time to complete their tasks. Insufficient time resulted in quality problems, which, in some cases, were difficult to rectify afterward. Interestingly, we also were informed that too much idle time between jobs also causes quality problems because it tends to disrupt the rhythm of the assembly worker.

On the basis of these discussions and conversations with other individuals in various companies who were responsible for actual assembly line sequencing applications, we decided to use the minimization of total utility work as our primary objective. As noted earlier, minimizing utility work also contributes to minimizing unnecessary idle time. Thus, this objective addresses quality problems of both types described above. We note that while utility work is a linear objective, it tends to produce "smoother" schedules than objectives such as "minimizing the number of jobs violating spacing constraints," or "minimizing the number of spacing constraints violated." The reason for this is that utility work incorporates penalties that are linear in the *extent* of the violations.

We assume that each station is closed to the left and right, but that each station can be defined to include upstream and downstream allowances that may extend beyond the normal boundaries of the station. For example, the station with its upstream and downstream allowances might be specified as the reach of the power tool required for the operation. Stations may overlap, and this is consistent with the manner in which assembly workers share work spaces in practice.

Our work is a generalization of the research by Yano and Rachamadugu (1987) and Bolat and Yano (1988). The work of Yano and Rachamadugu was described earlier. The result most critical to our work is that for the objective of minimizing utility work, it is optimal for an operator to complete as much of job t as possible before starting on job t+1. Under this policy, the scheduling problem simplifies to one of determining a sequence. Bolat and Yano use this fact to analyze the relationship between utility work and spacing

constraints in a situation where a single station can perform one of two operations (basic or optional). They show that an objective function that appropriately penalizes violations of spacing constraints is almost equivalent to one which minimizes utility work. (It is equivalent except in certain special cases where there are violations of spacing constraints, but utility work is zero.) They also develop an optimal sequencing procedure for the approximate objective function in the context of a single station.

This led us to the realization that there is an entire class of linear objective functions in which a different weight can be placed upon each spacing consideration. The fact that spacing issues and utility work, both of which have received considerable attention in the research literature and in practice, can be reflected by the same type of objective also made us more confident that this class of linear objectives would be very useful. Furthermore, since single-station lower bounds can be constructed quite easily, the look-ahead capability afforded by these bounds can be incorporated into the procedure.

Our work differs from that of Yano and Rachamadugu in that we assume only the existence of spacing rules and relative weights associated with the violation of each rule. (We later explain how the value of the objective function is affected by the number and extent of these violations.) In general, these rules and weights are much easier to obtain than the detailed processing time and station length information required by the Yano-Rachamadugu procedure. Indeed, processing times tend to change over time because of learning curve effects and engineering changes, among other factors. Also, station lengths, especially upstream and downstream allowances from the actual physical station, are sometimes difficult to specify accurately. In most cases, management familiar with the relevant operations on the assembly line can specify both spacing rules and the relative weights for violations of these rules on the basis of experience.

Because of the flexible form of our objective function, in addition to utility work, we are able to consider another objective which is similar to that used by Coffman, et al. (1985) in instances where the system is not heavily overloaded. Next, we describe the

objective function that we use and explain why appropriate weights lead to a good approximation of utility work. (A more formal argument is presented in Bolat and Yano.) We then explain the similarity of the second objective to that used by Coffman et al. We subsequently state the sequencing algorithm for arbitrary weights.

The Objective Function

To facilitate the exposition, we introduce the concept of a *window*, which is defined as a subsequence of n_j consecutive jobs if the spacing constraint at station j is "no more than k_j out of n_j consecutive jobs can have the option." Without loss of generality, we take the cycle time to be one time unit, and assume that the processing time of a basic job is less than or equal to the cycle time, while a job with the option has a processing time greater than the cycle time. Let N represent the number of jobs to be sequenced. Then for each station, there is one window *ending* with each position p, $p = k_j+1,..., N + n_j - k_j - 1$. (We assume, without loss of generality, that there are dummy basic jobs prior to position 1 and after position N. Consequently, windows ending before $p = k_j + 1$ or after $p = N + n_j + k_j - 1$ cannot have more than k_j jobs with the option, and therefore need not be considered.) Also let

$m_j = n_j - k_j$,

q_{jw} = number of jobs with the option in window w at station j,

v_{jw} = number of excess jobs with the option in window w at station j (also called "unit violations")

$= (q_{jw} - k_j)^+$,

θ_j = weight associated with station j,

b_j = processing time for a basic job at station j,

o_j = processing time for a job with the option at station j, and

L_j = sojourn time of a job at station j (physical length of station j divided by the cycle time)

= number of jobs in station j at any time.

The general form of the objective function is:

$$z = \Sigma_j \theta_j \Sigma_w v_{jw}. \quad (1)$$

Observe that v_{jw} reflects the extent to which the k_j out of n_j constraint is violated in window w. These "unit violations" are summed over all windows and multiplied by the appropriate weight for station j. These weighted sums are again summed over all stations. For the special case in which the true objective is to minimize total utility work, we define

$$\theta_j = (o_j - b_j)/n_j. \quad (2)$$

The rationale for this formula is as follows. Suppose the assembly worker at station j starts at the origin (leftmost point) of the station at time zero The maximum number of jobs with option j that can be completed in sequence without incurring any utility work is the largest value of k_j for which

$$o_j k_j \leq k_j + L_j - 1. \quad (3)$$

The left hand side of the inequality is the amount of time required to process k_j jobs with the option and the right hand side is the time at which the last such job leaves the station. If (3) is satisfied as an equality, the worker uses the full width of the station to complete the jobs with the option, and is standing at the right boundary of the station when the k_jth job is completed.

We now select the value of n_j (or equivalently m_j) in such a way that the system regenerates (i.e., the worker returns to the leftmost point of the station), possibly with zero idle time when these n_j jobs are completed. For this to be true, we must have

$$o_j k_j + b_j m_j \leq k_j + m_j, \quad (4)$$

and m_j is defined as the smallest value of m_j for which (4) is satisfied. If (4) is satisfied as an equality, there is no idle time (i.e., the worker is fully utilized) and the assembly worker returns to the origin of the station exactly at the point in time when the n_j +1st job enters the station. Figure 1 depicts the position of the worker within the station over time for a case in which (3) and (4) are satisfied as equalities. In the figure, $k_j = 2$ and $m_j = 3$.

Figure 1

Since we have assumed the worst case (all k_j jobs with the option in series), any subsequence of n_j jobs satisfying the k_j out of n_j spacing constraint will incur no utility work if the worker begins at a regeneration point. (The proof is omitted because the result is intuitively evident.)

In the remainder of the paper, we assume that k_j is the largest integer satisfying (3) and m_j is the smallest integer satisfying (4) for the given k_j. (If (3) and (4) are not satisfied as equalities, the correspondence between utility work and the objective function in (1) with θ_j defined as in (2) is weaker, but the difference per job between the two objectives is bounded by a value that asymptotically approaches zero as N --> ∞. See Bolat and Yano for details.) Suppose that we begin with a sequence in which the pattern of k_j jobs with the option followed by m_j basic jobs is repeated until there are N jobs in the sequence. If the actual number of jobs with the option in the set to be scheduled is less than the number of jobs with the option in this initial sequence, we can modify the initial sequence by replacing an appropriate number of jobs with the option by basic jobs. The resulting sequence will clearly have no utility work and will not violate any spacing constraints.

Let us now observe what happens when the number of jobs with the option in the set to be scheduled is greater than that in the repeating schedule. Consider increasing the number of jobs with the option by one. To do so, we must replace one basic job by a job with the option. With the possible exception of jobs near the end of the sequence, this

modification will lead to $o_j - b_j$ time units of utility work if the workers are fully utilized (i.e., if (3) and (4) are satisfied as equalities). Also observe that the job so modified appears in n_j windows and thus increases $\Sigma_w v_{jw}$ by n_j (again, with the possible exception of jobs near the end of the sequence). This process can be repeated with the same result. Consequently, utility work at station j is closely approximated by

$$[(o_j - b_j)/n_j] \Sigma_w v_{jw}$$

and we have $\theta_j = (o_j - b_j)/n_j$.

We now turn to a discussion of why the objective function with $\theta_j = 1$ for all j is similar to that used in Coffman et al. (1985). We earlier made the observation that if we begin with a repeating schedule and replace one basic job by a job with the option, there would be n_j windows, each with one unit violation. Observe that n_j spacing constraints are also violated. If we now modify another basic job very close in the sequence (e.g., adjacent) to the one changed earlier, this job will add one unit violation to each of n_j windows. However, *less* than n_j *additional* spacing constraints will be violated because some of the constraints related to the newly changed job were already violated. On the other hand, if the system is not heavily overloaded, and if the sequencing algorithm tends to smooth out the workload at each station as much as possible, it is unlikely that such a heavy workload will be concentrated in such a short subsequence. If this is true, our objective function with $\theta_j = 1$ for all j will be very similar to that of Coffman et al. More formally, if we can guarantee that jobs violating spacing constraints are at least n_j spaces apart for all j, the two measures are identical. In most realistic situations, ensuring this for each station separately is not difficult. The question is whether it is possible to guarantee this for all stations simultaneously. In any case, it should be clear from this discussion why our objective function will tend to produce smoother sequences than the objective used by Coffman et al.

Complete formulations of the two problems appear in Appendices A and B, respectively. The formulation in Appendix A uses an exact expression for utility work. The formulation in Appendix B is for the problem of minimizing weighted unit violations. Choosing θ_j according to (2) leads to an approximate expression for utility work, as explained earlier. We now give an algorithm for the latter formulation.

The Algorithm

The proposed algorithm is a sequential heuristic procedure in which a job is selected for each position in the sequence, starting from the first position and considering each subsequent position in turn. It can be viewed as a heuristic branch and bound procedure in which only the best node is retained at each level of the branch and bound tree. The best node is the one with the smallest value of the sum of utility work in the partial sequence and an approximate lower bound on the remainder of the sequence. The latter is obtained by assuming that the system (i.e., every station) regenerates at the completion of the job currently being assigned and finding an optimal sequence for each station separately, using the objective of weighted unit violations, for the remaining jobs . The corresponding "costs" are summed to produce an approximate lower bound. This lower bound ignores the constraint that each station must receive the same sequence. One attractive feature of this lower bound is that it is not necessary to *find* these optimal sequences; only the corresponding objective values are needed. Algebraic expressions for objective values are derived in Bolat and Yano as functions of the number of jobs of each type remaining to be sequenced and the spacing constraint parameters. These expressions are presented in Appendix C.

In order to facilitate the exposition, let us define:

$$a_{ij} = \begin{cases} 1 \text{ if job i has the option installed at station j,} \\ 0 \text{ otherwise;} \end{cases}$$

y_{ip} = 1 if job i is assigned to position p,
0 otherwise;

f_{jp} = time at which work terminates on job p at station j

= $\max[s_{jp} + b_j + (o_j - b_j) \Sigma_i a_{ij}y_{ip}, p - 1 + L_j]$, $p \geq 1$

s_{jp} = time at which work starts at station j for the job in position p

= $\min [f_{j,p-1}, p - 1]$, $p \geq 2$

= 0, $p = 1$

u_{jp} = utility work at station j for the job in position p

= $[b_j + (o_j - b_j) \Sigma_i a_{ij}y_{ip} - (f_{jp} - s_{jp})]$

U_p = total utility work for the job in position p (depends on y_{ip})

= $\Sigma_j u_{jp}$

$LB(j,R,r_j)$ = approximate value of lower bound on total utility work for the remaining R jobs if there are r_j jobs with the option installed at station j still unassigned

N_j = number of jobs with the option which is installed at station j

A = set of jobs remaining to be assigned.

Observe that for each station j, u_{jp} depends only upon the value of $\Sigma_i a_{ij}y_{ip}$, and not on the detailed components of the sum. The value of the sum is 1 if a job with the option installed at station j is assigned to position p, and zero otherwise. Thus, we can simply compute the two possible values of u_{jp}, then select the appropriate values when computing the value of U_p that results from the assignment of a particular job to position p.

A formal statement of the algorithm follows.

SEQUENCING ALGORITHM

1. $r_j = N_j$ for all j.

   A = {1,...,N}.

2. For p = 1 to N

   For i = 1 to N, i ε A

   Tentatively set y_{ip} = 1and $y_{vp} = 0$ for $v \neq i$.

   Compute $C_i = U_p + \Sigma_j \; LB(j,N-p,r_j-a_{ij})$

   Next i

   Set $y_{ip} = 1$ for i with minimum C_i value, zero otherwise.

   Remove i such that $y_{ip} = 1$ from A.

   For j = 1 to J

   r_j <-- $r_j - \Sigma_i \; a_{ij}y_{ip}$

   Next j

   Next p

3. Terminate.

The computational complexity of this algorithm is proportional to the square of the number of jobs, since N + (N-1) + ...+1 values of C_i must be computed.

4. EXPERIMENTAL RESULTS AND CONCLUSIONS

We obtained data on options for twenty sets of 1000 jobs from the automobile manufacturer, along with related processing time data. Twelve stations were considered critical. In order to assess how the algorithm performs on other data sets, we modified the problem parameters (i.e., processing times and station lengths) so as to produce two other data sets. The second data set has its primary "bottlenecks" at stations that differ from the primary bottlenecks in the first data set. The third data set differs from the first in that it is more difficult to sequence because the spacing constraints are tighter.

In order to evaluate the proposed algorithm, we also implemented a version of this automobile manufacturer's existing algorithm, a version of the algorithm being used at another automobile manufacturer (which is very similar to one of the procedures described earlier in the literature review), and the Yano and Rachamadugu algorithm. We considered two different objective functions. The first objective had weights which would be appropriate for approximately minimizing total utility work. The weights were in the range of 0.5 to 0.95. Since the true objective is to minimize total utility work, we report actual values of utility work, not values of the approximate objective. The second objective had all weights equal to 1.0. One of the primary reasons for using this objective is that such weights are consistent with one of the objectives of the second automobile manufacturer.

Percentage reductions in total utility work over solutions from existing procedures are reported in Tables 1 through 3, and the same statistics for total unit violations are reported in Tables 4 through 6. For comparison, we also report the percentage reduction over the objective value of the best of 200 randomly generated sequences. The computing time required to generate and evaluate the 200 random sequences is approximately equal to the computing time required for either the proposed procedure or the Yano and Rachamadugu algorithm. Thus, it represents a practical benchmark.

TABLES 1 THROUGH 6

It is evident from the results that the proposed procedure and the Yano and Rachamadugu procedure perform quite well relative to the algorithms of the two automobile manufacturers. In order to evaluate how the procedure performs relative to the optimal solution, we also obtained lower bounds on the respective objective values. The lower bounds are derived by finding the optimal sequence for each station independently, then summing the corresponding objective values. Since the bound totally ignores the constraint that all stations must receive the same sequence, it is quite loose.

We considered a variety of different ways to obtain tighter bounds, but they are extremely difficult to find for several reasons. First, for small problems with the objective of minimizing utility work, we found that the bounds obtained from linear programming relaxations (which permit fractional assignments of jobs to positions), were very close to the bounds that we obtain with our procedure (see Bolat 1988). Thus, although it was impractical to solve the linear relaxations of the 1000-job problems (there are over one million variables), we did not have any evidence to suggest that the resulting bounds would be better. Second, we have observed that the reported lower bound for total unit violations is generally tighter than a bound from a linear program (see Bolat 1988). Third, although the problem can be solved optimally by dynamic programming for either objective function, the computational complexity of the procedure increases rapidly with the size of the problem (number of jobs and number of stations). Thus, it is difficult to extend our bounding idea by partitioning the stations into sets of two or more stations, and computing a lower bound for each set. Finally, even lower bounds obtained from a branch-and-bound procedure with a time trap are unlikely to be tight because the number of jobs is so large that only a very small portion of the tree can be considered.

Notably, despite the looseness of the bounds, for the objective of minimizing total unit violations, the proposed algorithm is within 10 percent of the lower bound on average (see Table 7). Results for the objective of minimizing total utility work (see Table 8) are qualitatively similar. Details appear in Table 8. From these results, it appears that our procedure also performs well in an absolute sense.

TABLES 7 AND 8

The primary difference between the proposed algorithm (or the Yano and Rachamadugu algorithm) and the two existing procedures is the look-ahead capability provided by the lower bound for the remaining jobs. This suggests that even very simple, easy-to-implement, look-ahead procedures can provide significant savings.

The proposed algorithm performs quite well for the objective of minimizing total unit violations. Indeed, it is unlikely that much further improvement can be obtained from a tighter, more sophisticated lower bound. The algorithm has the advantage of being quite flexible; it can incorporate any spacing rules and place any desired weight on each. Also, since unit violations can be computed using available formulas, it is not necessary to construct sequences for the various stations in order to determine lower bounds. Thus, the algorithm can be implemented quite easily without special customization. This is especially helpful when the same procedure is used for multiple facilities (such as the circumstances that motivated this study), or when product characteristics change frequently.

Further research may be useful in improving lower bounds for the objective of minimizing total utility work, not only to serve as a benchmark, but also for use within the heuristic procedure.

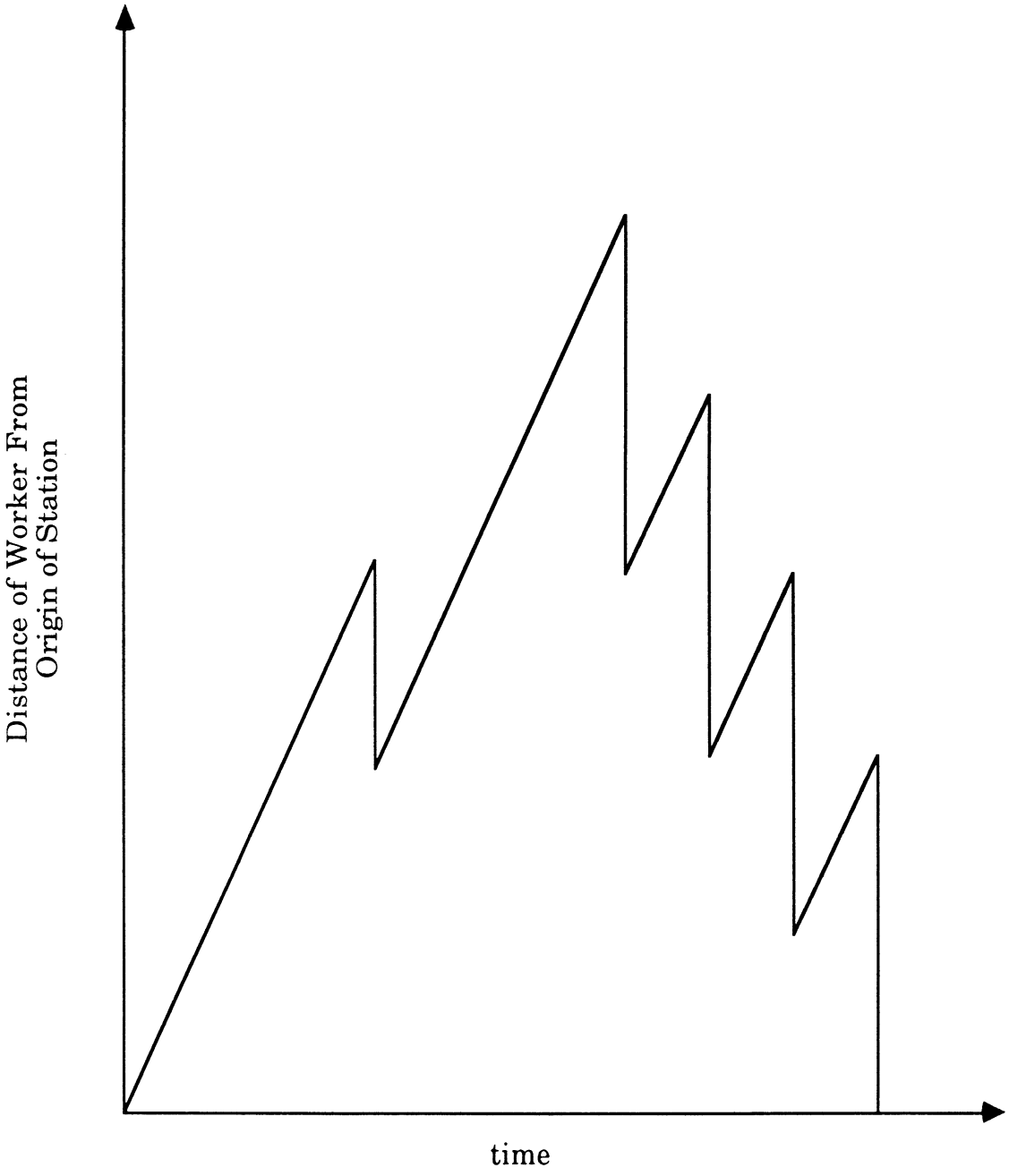

Figure 1

Position of Worker Within Station
When k=2 and m=3

TABLE 1

Percent Improvement of New Procedure over Existing Procedures:

Total Utility Work for Data Set #1 (20 problems)

Sequencing Procedure	Avg.	Std. Dev.	Min.	Max.
Best of 200 Random Sequences	58.7	15.7	19.8	81.4
Company A	49.0	15.2	21.8	75.6
Company B	32.6	17.4	3.3	72.0
Y & R*	-3.7	7.3	-18.1	5.3

* Yano and Rachamadugu Algorithm

TABLE 2

Percent Improvement of New Procedure over Existing Procedures:

Total Utility Work for Data Set #2 (20 problems)

Sequencing Procedure	Avg.	Std. Dev.	Min.	Max.
Best of 200 Random Sequences	41.0	12.0	16.6	56.5
Company A	30.7	11.3	11.3	45.9
Company B	10.2	7.9	0	24.4
Y & R*	-2.9	2.7	-10.5	0.9

* Yano and Rachamadugu Algorithm

TABLE 3

Percent Improvement of New Procedure over Existing Procedures:

Total Utility Work for Data Set #3 (20 problems)

Sequencing Procedure	Avg.	Std. Dev.	Min.	Max.
Best of 200 Random Sequences	34.4	11.3	10.3	52.7
Company A	30.0	9.8	14.2	48.9
Company B	16.2	9.6	0.4	37.0
Y & R*	-5.7	2.6	-11.4	-1.3

* Yano and Rachamadugu Algorithm

TABLE 4

Percent Improvement of New Procedure over Existing Procedures:

Total Unit Violations for Data Set #1 (20 problems)

Sequencing Procedure	Avg.	Std. Dev.	Min.	Max.
Best of 200 Random Sequences	51.0	13.9	15.8	71.0
Company A	40.3	13.2	14.2	62.7
Company B	23.0	12.9	2.7	48.2
Y & R*	3.5	5.5	-12.6	13.3

* Yano and Rachamadugu Algorithm

TABLE 5

Percent Improvement of New Procedure over Existing Procedures:

Total Unit Violations for Data Set #2 (20 problems)

Sequencing Procedure	Avg.	Std. Dev.	Min.	Max.
Best of 200 Random Sequences	50.7	15.5	15.4	70.5
Company A	39.7	15.0	10.9	61.4
Company B	13.6	10.2	-2.9	30.0
Y & R*	0.9	3.3	-6.4	6.0

* Yano and Rachamadugu Algorithm

TABLE 6

Percent Improvement of New Procedure over Existing Procedures:

Total Unit Violations for Data Set #3 (20 problems)

Sequencing Procedure	Avg.	Std. Dev.	Min.	Max.
Best of 200 Random Sequences	38.5	12.1	12.9	58.4
Company A	33.0	10.4	15.3	52.5
Company B	17.6	9.7	0.4	37.0
Y & R*	2.8	2.9	-0.1	12.6

* Yano and Rachamadugu Algorithm

TABLE 7

Ratio of Lower Bound to Objective Value of Solution:

Total Unit Violations

	Data Set		
	#1	#2	#3
Average	91.6	86.6	86.7
Standard Deviation	6.1	5.4	4.6
Minimum	76.2	77.4	78.6
Maximum	99.0	95.1	94.7

TABLE 8

Ratio of Lower Bound to Objective Value of Solution:

Total Utility Work

	Data Set		
	#1	#2	#3
Average	42.2	90.8	74.1
Standard Deviation	18.9	3.8	5.8
Minimum	17.3	82.4	65.1
Maximum	69.7	96.7	87.8

REFERENCES

Bird, C.G. (1986), "Sequencing Vehicles for Assembly Under Precedence Constraints," Paper presented at the ORSA/TIMS Conference in Los Angeles, California.

Bolat, A. (1988), "Generalized Mixed Model Assembly Line Sequencing Problem," Unpublished Ph.D. Dissertation, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, Michigan.

Bolat, A. and Yano, C.A. (1988), "Scheduling Algorithms to Minimize Utility Work at a Single Station on a Paced Assembly Line," Technical Report 88-7, Department of Industrial and Operations Engineering, The University of Michigan, Ann Arbor, Michigan.

Burns, L.D., and Daganzo, C.F. (1985), "Assembly Line Sequencing Principles," Research Publication GMR-5127, General Motors Research Laboratories, Warren, Michigan.

Coffman, P.E., Jr., Hoffman, S.E., and Weiner, S.A. (1985), "An O.R. View of Assembly Plant Modeling," Paper presented at the TIMS/ORSA Conference in Boston, Massachusetts.

Dar-El, E.M. and Cother, R.F. (1975), "Assembly Line Sequencing for Model-Mix," *Inter. J. Prod. Res.* 13 (5), 463-477.

Dar-El, E.M. and Cucuy, S. (1977), "Optimal Mixed-Model Sequencing for Balanced Assembly Lines," OMEGA 5 (3), 333-342

Macaskill, J.L.C. (1973), "Computer Simulation for Mixed-Model Production Lines," *Management Science* 20 (3), 341-348.

Miltenburg, J. (1989), "Level Schedules for Mixed-Model Assembly Lines in Just-In-Time Production Systems," *Management Science* 35 (2), 192-207.

Monden, Y. (1983), Toyota Production System, Industrial Engineering and Management Press, IIE, Atlanta, Georgia.

Okamura, K. and Yamashina, H. (1979), "A Heuristic Algorithm for the Assembly Line Model-Mix Sequencing Problem to Minimize the Risk of Stopping the Conveyor," *Inter. J. Prod. Res.* 17 (3), 233-247.

Ow, P.S. and T.E. Morton (1988), "Filtered Beam Search in Scheduling," *Inter. J. Prod. Res.* 26 (1), 35-62.

Parrello, B. (1988), "Car Wars: The (Almost) Birth of an Expert System," *AI Expert*, 60-64.

Parrello, B., Kabat, W.C., and Wos, L. (1988), "Job-Shop Scheduling using Automated Reasoning: A Case Study of the Car-Sequencing Problem," *Journal of Automated Reasoning* 2, 1-42.

Schonberger, R.J. (1982), Japanese Manufacturing Techniques: Nine Hidden Lessons in Simplicity, The Free Press, New York, New York.

Thomopoulos, N. T. (1967), "Line Balancing-Sequencing for Mixed Model Assembly," *Management Science* 14 (2), 59-75.

Wester, L. and Kilbridge, M. (1964), "The Assembly Lines Mixed Model Sequencing Problem," Proceedings of the Third International Conference on Operations Research, 1963

Yano, C.A. and Rachamadugu, R. (1987), "Sequencing to Minimize Work Overload in Assembly Lines with Product Options," Technical Report 87-22, Department of Industrial and Operations Engineering, The University of Michigan, Ann Arbor, Michigan.

APPENDIX A

Formulation for Utility Work Minimization

Minimizing utility work is equivalent to maximizing the total work completed. The formulation below has the latter objective. In addition to the notation defined earlier, we define

τ_{jp} = time allocated at station j to the job assigned to position p.

One possible formulation is:

$$\text{Max} \quad \Sigma_j \; \Sigma_p \; \tau_{jp}$$

s.t.

$$s_{j1} = 0 \qquad \text{for all j} \qquad \text{(A1)}$$

$$s_{jp} \geq s_{j,p-1} + \tau_{j,p-1} \qquad \text{for all j and p} \qquad \text{(A2)}$$

$$s_{jp} \geq p - 1 \qquad \text{for all j and p} \qquad \text{(A3)}$$

$$s_{jp} + \tau_{jp} \leq p - 1 + L_j \qquad \text{for all j and p} \qquad \text{(A4)}$$

$$\tau_{jp} \leq b_j + (o_j - b_j) \Sigma_i a_{ij} y_{ip} \qquad \text{for all j and p} \qquad \text{(A5)}$$

$$\Sigma_i y_{ip} = 1 \qquad \text{for all p} \qquad \text{(A6)}$$

$$\Sigma_p y_{ip} = 1 \qquad \text{for all i} \qquad \text{(A7)}$$

$$y_{ip} = 0 \text{ or } 1 \qquad \text{for all i and p} \qquad \text{(A8)}$$

$$\tau_{jp} \geq 0 \qquad \text{for all j and p} \qquad \text{(A9)}$$

The constraints ensure that (i) the first job starts at time zero (without loss of generality), (ii) all other jobs start after work has terminated on their predecessors, (iii)

work on a job starts no earlier than its arrival at the station, (iv) work is terminated no later than when a job leaves the station, (v) at each station, the amount of work performed on a job is less than or equal to its processing time, (vi) each job is assigned to exactly one position, (vii) each position is allocated exactly once, (viii) the assignments are binary, (ix) the time allocations are non-negative.

Appendix B

Formulation for Minimization of Weighted Unit Violations

The problem can be formulated as follows:

$$\text{Min} \quad \Sigma_j \theta_j \sum_{w=1}^{N-m_j-1} \left(\sum_{p=w}^{w+n_j-1} \Sigma_i \, a_{ij} y_{ip} - m_j \right)^+$$

subject to

$$\Sigma_i \, y_{ip} = 1 \qquad \text{for all } p$$

$$\Sigma_p \, y_{ip} = 1 \qquad \text{for all } i$$

$$y_{ip} = 0 \text{ or } 1 \qquad \text{for all } i \text{ and } p$$

The objective is to minimize total weighted unit violations. The term in parentheses is the number of unit violations in the window of n_j consecutive positions starting with position w. The second sum reflects the totaling of these unit violations over all windows of length $m_j + 1$ or greater. The external sum weights these unit violations and sums them over all stations. The assignment and binary constraints on the decision variables ensure that the sequence is feasible.

Observe that this formulation is much simpler than the one for utility work (in Appendix A). The primary difference is that the detailed timing decisions do not appear in this formulation.

Appendix C

Expressions for Unit Violations

The formulas below are based upon the observation that if one can sequence k_j jobs with the option followed by $m_j = n_j - k_j$ jobs without the option, and repeat this pattern until the total number of jobs has been sequenced, such a sequence will have no unit violations since there will be:

1. exactly k_j jobs with the option in any subsequence of n_j consecutive jobs, and
2. no more than k_j jobs with the option in any subsequence of fewer than n_j consecutive jobs (e.g., in the short windows at the beginning and end of the sequence).

Thus, the formulas for unit violations are derived by determining how many basic jobs must be replaced by jobs with the option in order to match the original option mix and making these replacements so that unit violations are minimized. The reader is referred to Bolat and Yano (1988) for detailed derivations.

For station j, let

G_j = maximum number of repeating cycles of k_j jobs with the option followed by m_j basic jobs

$= \lfloor N/n_j \rfloor$

r_j = number of jobs in last fractional cycle

$= N - G_j n_j$

H_j = actual number of jobs with option j

H_j^* = maximum number of jobs with the option that can be sequenced with no unit violations

$= G_j k_j + \min (k_j, r_j)$

X_j = number of excess jobs with the option

$= (H_j - H_j^*)^+$

With these definitions, the total unit violations in an optimal sequence is:

0	if $H_j \leq H_j^*$,
$X_j n_j$	if $H_j > H_j^*$ and $r_j = k_j$,
$X_j (X_j + r_j)$	if $H_j > H_j^*$, $r_j < k_j$, and $X_j < \min (k_j - r_j, m_j)$,
$X_j n_j - m_j (k_j - r_j)$	if $H_j > H_j^*$, $r_j < k_j$, and $X_j \geq \min (k_j - r_j, m_j)$,
$X_j n_j - X_j (r_j - X_j)$	if $H_j > H_j^*$, $r_j > k_j$, and $X_j < \min (k_j - r_j, k_j)$,
$X_j n_j - k_j (r_j - k_j)$	if $H_j > H_j^*$, $r_j > k_j$, and $X_j \geq \min (k_j - r_j, k_j)$.

In the algorithm, the values of N and X_j are updated as the sequence is constructed to reflect the number of remaining jobs and the current number of excess jobs with the option, respectively.