A Time Window Approach to Simultaneous Scheduling of Machines and Material Handling System in an FMS

Ümit Bilge, Gündüz Ulusoy,

A TIME WINDOW APPROACH TO SIMULTANEOUS SCHEDULING OF MACHINES AND MATERIAL HANDLING SYSTEM IN AN FMS

ÜMIT BILGE and GÜNDÜZ ULUSOY

Boğaziçi University, Istanbul, Turkey

(Received January 1992; revisions received March, September 1993; accepted January 1994)

This paper exploits the interactions between the machine scheduling and the scheduling of the material handling system in an FMS by addressing them simultaneously. The material transfer between machines is done by a number of identical automated guided vehicles (AGVs) which are not allowed to return to the load/unload station after each delivery. This operating policy introduces an additional complexity to the problem because it results in sequence-dependent travel times for the deadheading trips between successive loaded trips of the AGVs. The problem is formulated as a nonlinear mixed integer programming model. Its objective is makespan minimization. The formulation consists of constraint sets of a machine scheduling subproblem and a vehicle scheduling subproblem which interact through a set of time window constraints for the material handling trip starting times. An iterative procedure is developed where, at each iteration, a new machine schedule is generated by a heuristic procedure, the operation completion times obtained from this schedule are used to construct time windows for the trips, and a feasible solution is searched for the second subproblem, which is handled as a sliding time window problem. The procedure is numerically tested on 90 example problems.

In this study, a flexible manufacturing system (FMS) in which material transfer between machines is performed by a number of identical automated guided vehicles (AGVs) is considered, and the problem of simultaneous scheduling of machines and AGVs is addressed. The importance of the material handling system for the efficiency of the overall system has been emphasized by several researchers (Raman, Talbot and Rachamadgu 1986, Greenwood 1988, Bozer 1989, Han and Mcginnis 1989, Kouvelis 1992). In the majority of the related literature, the subject of scheduling the material handling system has generally been set out either as a comparison of various vehicle dispatching rules in reaction to a prespecified schedule and on a particular layout (Egbelu and Tanchoco 1984, Bartholdi and Platzman 1989) or in relation with the design phase (Maxwell and Muckstadt 1982, Grosseschallau and Kusiak 1985, Chang, Sullivan and Wilson 1986, Mahadevan and Narendran 1990, Malmborg 1990, Gobal and Kasilingam 1991). Obviously, an increase in the performance of the FMS would be expected as a result of the coordination of the machine and the material handling system during the machine scheduling phase. Various aspects of this premise have been studied by several researchers. Sabuncuoğlu and Hommertzheim (1992a) investigated the performances of machine and AGV scheduling rules against mean flow-time criterion using a simulation model. Wu and Wysk (1989), Ro and Kim (1990), and Sabuncuoğlu and Hommertzheim (1992b) developed on-line scheduling and control rules for machines and AGVs. The case of a special material handling transporter in a real-time environment was treated by Han and Mcginnis. A deterministic off-line scheduling model formulated as an integer programming problem and a solution procedure based on concepts of project scheduling under resource constraints were presented by Raman, Talbot and Rachamadagu.

1. PROBLEM STATEMENT

The scheduling problem under study can be defined as follows: Given the FMS described, determine the starting and completion times of operations for each job and the trips between workstations together with the vehicle assignment according to the objective of minimizing the makespan, $C_{\max}$.

Makespan is one of the more frequently used criteria among those based upon completion times. It implies that the cost of a schedule depends on the duration for which the whole system is allocated to a set of jobs. Another class of performance criterion is based upon due dates. Due date based performance criteria, such as mean tardiness and maximum tardiness, have been employed relatively more in the last years, mainly due to the changing marketing environment. The possibility of using the approach presented here with due date based criteria is commented upon in the Conclusion.

The environment within which the FMS under consideration operates can be described as follows. We assume that all the design and setup issues within the hierarchy of OR/MS problems in an FMS as suggested by Stecke (1985), have already been resolved. The types and number of machines are known. There is sufficient input/output buffer space at each machine. Machine loading,

Subject classifications: Production/scheduling, flexible manufacturing: simultaneous scheduling of machines and AGVs. Transportation, vehicle scheduling: AGV scheduling using sliding time windows.
Area of review: Manufacturing, Operations and Scheduling.

0030-364X/95/4306-1058 $01.25
© 1995 INFORMS

i.e., the allocation of tools to machines and the assignment of operations to machines, is made. Pallets and other necessary equipment are allocated to parts. The set of part types to be produced during the planning period and the routing of each part type are available before making scheduling decisions. The routing for a part type can be selected based on considerations of technological feasibility and processing efficiency, or by formulating the setup phase problems in a manner that can also handle the routing decisions (Kiran and Tansel 1986). Operations are nonpreemptive. Ready-times of all jobs are known. Initially, partially processed parts might be available at machines waiting for further processing, and they can be treated as jobs having zero ready times and their routing consists of the remaining operations.

The load/unload (L/U) station serves as a distribution center for parts not yet processed and as a collection center for parts finished. All vehicles start from the L/U station initially. There is sufficient input/output buffer space at the (L/U) station.

Trips follow the shortest path between two points and occur either between two machines or between a machine and the L/U station. Preemption of trips is not allowed. The trips are called loaded or deadheading (empty) trips depending on whether a part is carried or no part is carried during that trip, respectively. The durations for the deadheading trips are sequence dependent and are not known until the vehicle route is specified.

Processing, setup, loading, unloading, and travel times are available and deterministic. Such issues as traffic control, congestion, machine failure or downtime, scraps, rework, and vehicle dispatches for battery changer are ignored here and left as issues to be considered during real-time control.

The basic distinction between the FMS described above and that considered by Raman, Talbot and Rachamadagu lies in the operating policy regarding the AGVs. According to their operating policy, AGVs always return to the L/U station upon completion of their task, so that the durations of the deadheading trips which follow each task are known. Thus, an AGV trip can be considered as an operation between two successive machining operations for any given job, and concepts of project scheduling under resource constraints can be utilized. The concept of sequence-dependent travel times is also encountered in Han and Mcginnis but their problem setting is different, as referred to before.

2. MIXED INTEGER PROGRAMMING FORMULATION

In this section, the combined machine and material handling system scheduling problem is formulated as a nonlinear mixed integer programming model. The notation for the mixed integer programming formulation (MIP) is:

J: the set of jobs available for processing;

n_j: the number of operations of job j;

n: $\sum_{j\in J} n_j$, the number of operations;

I: $\{1, 2, \ldots, n\}$, the index set of operations;

I_j: $\{N_j + 1, N_j + 2, \ldots, N_j + n_j\}$ is the set of indices in I associated with job j in J, where N_j is the total number of operations of the jobs indexed before j, and $N_1 = 0$;

$\bar{I}_i$: $I - \{h: h \geq i, i, h \in I_j\}$, the index set of operations excluding operation i and succeeding operations of the same job;

$\underline{I}_h$: $I - \{i: i \leq h, i, h \in I_j\}$, the index set of operations excluding operation h and preceeding operations of the same job;

K: the number of vehicles;

p_i: the processing time of operation i;

a_j: the ready time of job j;

t_i: the travel time of the loaded trip i including loading and unloading times;

τ_{hi}: travel time of the deadheading trip starting at the machine performing operation h and ending at the machine performing operation i;

c_i: the completion time of operation i;

T_i: the completion time of loaded trip i;

$$q_{rs}: \begin{cases} 1, & \text{if } c_r \text{ is less than } c_s, \text{ where } r \text{ and } s \text{ are operations of different jobs,} \\ 0, & \text{otherwise;} \end{cases}$$

$$x_{hi}: \begin{cases} 1, & \text{if a vehicle is assigned for the deadheading trip between trip } h \text{ and trip } i, \\ 0, & \text{otherwise;} \end{cases}$$

$$x_{oi}: \begin{cases} 1, & \text{if a vehicle starts from the L/U station to accomplish trip } i \text{ as its first assignment,} \\ 0, & \text{otherwise;} \end{cases}$$

$$x_{ho}: \begin{cases} 1, & \text{if a vehicle returns to the L/U station after completing trip } h \text{ as its last assignment,} \\ 0, & \text{otherwise;} \end{cases}$$

Since the routing for each job is known, for each operation index in I the corresponding machine index is also known and therefore, the machine index is not used explicitly in the formulation. Also, there is a one-to-one correspondence between operations and the loaded trips. For each operation i, there is a corresponding loaded trip i whose destination is the machine where operation i is assigned and its origin is either the machine where the operation preceeding i is assigned or the L/U station. In the MIP formulation given next, the objective to be considered is the makespan minimization via minimizing the maximum of the completion times of the last operations for all jobs. The second constraint sets (2a, b) are the precedence constraints for the operations. The third set of constraints, where H is a large positive number, is in either-or form and it assures that no two operations that are assigned to the same machine can be processed simultaneously. If two operations r and s belonging to different jobs require the same machine, then τ_{rs} will be zero by definition and the constraint set will be effective; otherwise, it will hold trivially. The fourth and fifth constraint sets assure that all operations are loaded and unloaded once, respectively. The sixth constraint limits each vehicle to enter the system at most one time. The seventh constraint is to keep the number of vehicles in the system consistent. The eighth constraint set states that operation i can start only after the trip to load it is completed. The next three constraint sets, (9) and (10a, b), are related to the starting times of the trips. Together they state that trip i cannot start before the maximum of the completion time of the previous operation and the deadheading trip to the previous operation. The nonnegativity and the zero-one constraints remain.

MIP Formulation

Minimize Z
subject to

$$Z \geq c_{N_j+n_j} \quad \text{for all } j \in J \tag{1}$$

$$c_i - c_{i-1} \geq p_i + t_i \quad \text{for all } i,\ i-1 \in I_j,\ j \in J \tag{2a}$$

$$c_{N_j+1} \geq p_{N_j+1} + t_{N_j+1} \quad \text{for all } j \in J \tag{2b}$$

$$\left.\begin{array}{l}(1 + H\tau_{rs})c_r \geq c_s + p_r - Hq_{rs} \\ (1 + H\tau_{rs})c_s \geq c_r + p_s - H(1 - q_{rs})\end{array}\right\}$$

$$\text{for all } r \in I_j,\ s \in I_k \text{ where } j,\ k \in J,\ j < k \tag{3}$$

$$x_{oi} + \sum_{h \in \bar{I}_i} x_{hi} = 1 \quad \text{for all } i \in I \tag{4}$$

$$x_{ho} + \sum_{i \in \underline{I}_h} x_{hi} = 1 \quad \text{for all } h \in I \tag{5}$$

$$\sum_{i \in I} x_{oi} \leq K \tag{6}$$

$$\sum_{i \in I} x_{oi} - \sum_{h \in I} x_{ho} = 0 \tag{7}$$

$$T_i \leq c_i - p_i \quad \text{for all } i \in I \tag{8}$$

$$T_i - t_i \geq c_{i-1} \quad \text{for all } i,\ i-1 \in I_j,\ j \in J \tag{9}$$

$$T_i - t_i \geq x_{oi}\tau_{o,i-1} + \sum_{h \in \bar{I}_i} x_{hi}(T_h + \tau_{h,i-1}) \tag{10a}$$

$$\text{for all } i,\ i-1 \in I_j,\ j \in J$$

$$T_{N_j+1} - t_{N_j+1} \geq \sum_{h \in \bar{I}_{N_j+1}} x_{h,N_j+1}(T_h + \tau_{ho}) \tag{10b}$$

$$\text{for all } j \in J$$

$$c_i \geq 0 \quad \text{for all } i \in I$$

$$x_{hi} = 0,\ 1 \quad \text{for all } h \in \bar{I}_i,\ i \in I$$

$$x_{oi},\ x_{ho} = 0,\ 1 \quad \text{for all } i,\ h \in I$$

$$q_{rs} = 0,\ 1 \quad \text{for all } r \in I_j,\ s \in I_k \text{ where } j,\ k \in J,\ j < k.$$

However, this formulation is intractable due to its size and nonlinearity, thus no solution procedure will be sought for it. Since **MIP** is an integer programming formulation of the simultaneous scheduling of machines and vehicles, it is expected to include the constraint sets of both a machine scheduling problem and a vehicle scheduling problem. Indeed, the constraint sets (1), (2), and (3) together with the related nonnegativity constraints and the objective function represent a machine scheduling problem of the form $n/m/G/C_{\max}$, a well known NP-hard problem (Lenstra and Rinnooy Kan 1978). On the other hand, the constraint sets (4), (5), (6), and (7) constitute the essential constraints of a generic vehicle scheduling problem (VSP) which is an NP-hard problem (Orloff 1976). These two subproblems interact through two constraint sets, (8) and (9), which, when written together define a time window around T_i:

$$c_{i-1} + t_i \leq T_i \leq c_i - p_i. \tag{11}$$

Or, equivalently, a time window can be constructed for the trip starting times, ST_i,

$$c_{i-1} \leq ST_i \leq c_i - p_i - t_i. \tag{12}$$

In a large number of vehicle scheduling problems, collections and deliveries are required to be made within given time windows: e.g., school busing, scheduling of urban transit, passenger trains, and airline fleets. A survey of time window constrained routing and scheduling problems can be found in Solomon and Desrosiers (1988). Ferland and Fortin (1989) have studied the problem in the school busing context and added another level of flexibility by allowing the time windows of trips to slide. This is called *vehicle scheduling with sliding time windows*. The time windows given by (12) also slide because given a machine schedule with a particular makespan, $C_{\max}$, the operation completion times c_i can slide between the earliest completion times EC_i and latest completion times LC_i dictated by $C_{\max}$.

3. THE ITERATIVE SOLUTION PROCEDURE

In the proposed solution procedure, the machine scheduling and the vehicle scheduling subproblems discussed before are embedded in an iterative solution procedure. A heuristic algorithm generates machine schedules and another one called the sliding time window heuristic (**STW**) finds a feasible solution to the VSP given the machine schedule. The **STW** heuristic checks whether it is possible to accomplish all handling trips with the given number of vehicles within the time windows resulting from the machine scheduling subproblem. If this is not the case, instead of stopping without a feasible solution for this particular operation sequence on machines, the original makespan is increased, new time windows are constructed, and the heuristic is resolved. These two algorithms are linked by an iterative structure which facilitates the search for a good solution. This iterative structure and the generation of machine schedules will be explained. The **STW** heuristic will be described in Section 4.

3.1. The Iterative Structure

Initially, the set of machine schedules, SS is empty. An upperbound, UB, is determined. This will be updated each time the **STW** heuristic results in a better solution, and will be used to reduce the number of machine schedules in the set SS. Initially, UB can be the makespan of a known solution to the original problem, or a large value.

A quick and good upper bound is obtained by a single-pass heuristic procedure that will be described in subsection 5.2. However, it is not easy to find a tight lower bound, because either the problem generated for this purpose is too difficult, or it ignores such important features that the resulting lower bound is of no use. An initial machine schedule is generated and the **STW** heuristic algorithm is applied. If the resulting makespan (after incrementing, if necessary) is less than the current *UB*, then a better solution has been found and this becomes the new *UB*. Then, new machine schedules who are neighbors of the seed on hand are generated and included in *SS*. If there are any schedules in *SS*, whose makespan are greater than or equal to the already found solution, *UB*, than these are excluded from *SS*, and thus from further consideration. The seed schedule is also removed from *SS*. After such eliminations, if *SS* is still nonempty, then the next iteration starts with a new seed schedule chosen from *SS*. The procedure is repeated until one of the two stopping criteria is satisfied, either the maximum iteration number is reached, or *SS* becomes null.

3.2. Machine Schedule Generation

In machine schedule generation two algorithms are used. The first is based on Giffler and Thompson's (1960) active schedule algorithm, and the second is a variation of it that produces nondelay schedules (French 1982). The only modification made for the application here is that the processing times are redefined to include the loaded trip times. In both algorithms, one operation is scheduled at each iteration that determines the machine and then the operation to be assigned to it. For the active scheduling case, the machine is determined by the operation with the earliest finishing time. Among the operations which utilize this machine for their next operation and have an earliest starting time strictly less than the earliest finishing time just found, one is assigned based on some dispatching rule. For the nondelay case, the machine is determined by the earliest starting time. If there is more than one such operation utilizing the same machine, then one is assigned based on some dispatching rule. Otherwise, the operation that determines the machine is assigned to it. In all cases, ties are broken arbitrarily. In this application, three dispatching rules have been used for both algorithms: most work remaining (MWKR), least work remaining (LWKR), and shortest processing time (SPT).

3.2.1. Neighborhood of a Seed Machine Schedule

Given a seed schedule, to change the sequence of an operation on a machine, a constraint will be introduced on its earliest starting time, which prevents that operation from appearing at the particular position that it used to appear in the seed. The resulting schedule is called a neighboring schedule of the seed schedule. If Γ_m denotes the set of operations to be processed on machine m, then

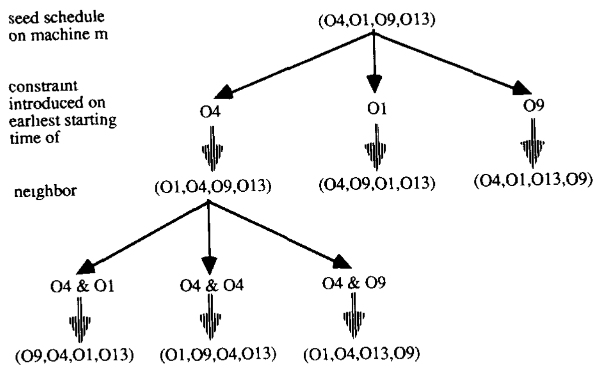

Figure 1. Illustration of neighborhood generation (O_i = operation number).

for each machine m, $|\Gamma_m| - 1$ choices are made, therefore, $|\Gamma_m| - 1$ neighbors can be generated assuming that there are no predecedence relationships among operations performed on the same machine. The complete neighborhood for a given seed schedule consists of $\sum_{m=1}^{M} |\Gamma_m| - M$ such schedules, each generated by introducing a constraint on an earliest starting time before reapplying the algorithm. Here, M is the number of machines. When a schedule generated in this fashion becomes the seed, its neighbors will be created by using the intersection of the seed's constraint with the new constraints. This is illustrated in Figure 1.

3.2.2. Selection of the New Seed Machine Schedule

Unfortunately, it was not possible to develop a gradient rule that guarantees a better solution to the original machine and vehicle scheduling problem when choosing the new seed from the set of machine schedules, *SS*. Note that a machine schedule with a minimum makespan does not guarantee a better solution because the slack times between operations created by a sequence with a longer makespan may fit better to the required handling times in between. Nevertheless, one rule (RULE 1) used is to select the machine schedule with the minimum makespan in *SS*. Ties are broken arbitrarily.

In the second rule, the aim is to make use of some information gained during the **STW** phase as a feedback. The number of times the sequence of operations on a particular machine causes conflicts in time windows is counted. These numbers are called "machine conflicts" and they are assigned as weights to the neighboring schedules of the seed. All the neighbors that are generated by constraining the earliest starting time of operations belonging to the same machine receive the same weight: the machine conflict number of that particular machine. Then, the machine schedule with the highest weight among the schedules in *SS* is chosen as the new seed. Ties are broken on the basis of a minimum makespan. This rule (RULE 2) is called the maximum machine conflict rule.

4. A HEURISTIC FOR VEHICLE SCHEDULING WITH SLIDING TIME WINDOWS

The general outline of the procedure and the opportunity cost concept that will be described next are due to Ferland and Fortin, who studied the problem in a school busing context. Their approach consists of identifying pairs of tasks that offer good opportunity costs for reducing the overall cost, and that find ways of modifying the starting times within their time windows to allow them to be linked. The general approach has been adapted to our application, however, conditions which allow two loaded trips to be linked and starting time modification procedures are specific to our problem. This is inevitable because the structure of time windows and their interrelationships are quite different from those in school busing context. Recall that our problem has the additional complexity that the time windows are not independent from each other. For instance, if a completion time c_i is increased to slide its corresponding time window, then the completion times for the next operation of the same job and for the next operation assigned to the same machine with operation i may be influenced. Furthermore, this influence may be reflected to other operations downstream, causing the corresponding time windows to slide. For this reason, the interrelationships are complex and the specification of the adjustments is not straightforward.

4.1. Outline of the STW Algorithm

The procedure starts with an initial set of starting times feasible for the time windows in **STWP** (i.e., $ST_i = EC_i - p_i - t_i$). When variables ST_i are fixed in this way **STWP** reduces to a minimum cost flow problem which will be called the fixed problem (**FP**), defined on a network where an arc (i, j) is included if $ST_i + t_i + t_{ij} \leq ST_j$.

Problem FP

Minimize

$$\sum_{(i,j)\in A} f_{ij} x_{ij}$$

subject to

$$\sum_{j\in P_i} x_{ij} = 1 \quad i = 1, \ldots, n,$$

$$\sum_{i\in B_j} x_{ij} = 1 \quad j = 1, \ldots, n,$$

$$\sum_{i\in P} (x_{oi} - x_{io}) = 0$$

$$0 \leq x_{ij} \leq 1 \quad (i, j) \in A,$$

where,

P = the set nodes where each node represents a loaded trip;
$A = \{(i, j)\colon ST_i + t_i + t_{ij} \leq ST_j\} U\{(o, i), (i, o)\colon 1 \leq i \leq n\}$, the set of admissible deadheading arcs including the arcs entering and leaving the L/U station $\{o\}$;
$P_i = \{j\colon (i, j) \in A\}$, the set of successors of trip i;
$B_j = \{i\colon (i, j) \in A\}$, the set of predecessors of trip j;
t_{ij} = the duration of the deadheading arc (i, j),

$$f_{ij} = \begin{cases} H & \text{if } i = o \text{ or } j = o \text{ with } H \gg t_{ij} \\ t_{ij} & \text{otherwise.} \end{cases}$$

The optimal solution to **FP** is found and the resulting vehicle routes are identified. If the number of routes in the optimal solution is less than or equal to the number of vehicles available, then the given set of starting times is feasible for the original problem and we have a vehicle schedule and routing compatible with the current machine schedule. If, on the other hand, this is not the case, then the starting times should be modified such that a new arc becomes admissible and is included in the network. The procedure will be repeated, each time solving a new **FP** created in this way until the number of routes in the optimal solution is less than or equal to the number of vehicles available.

The way to identify the arc (i, j) to be introduced in each iteration and the ways to modify the starting times to allow trips i and j to be linked will be described in the next section. If the procedure stops without being able to find any pairs (i, j) that can be linked (the heuristic cannot find a solution which is feasible in terms of the number of vehicles within the makespan given by the scheduling subproblem), the original makespan is increased, new time windows are constructed, and the procedure is continued with the new **STWP**. This will be described in subsection 4.3.

4.2. Introduction of a New Arc

FP can be solved by the simplex algorithm which will also generate simplex multipliers associated with the constraints of the problem. If π_i denotes the dual variable associated with node $i \in P$, then the opportunity cost of linking two trips $(i, j) \notin A$ will be defined (similar in form to the relative cost of an arc $(i, j) \in A$) as

$$r_{ij} = t_{ij} - \pi_i + \pi_j. \tag{13}$$

The opportunity cost of introducing arc (i, j) in A will be a measure of the influence on the value of the objective function of modifying ST_i or ST_j to induce the possibility of linking trip j to trip i. Hence, it is interesting to analyze the opportunity of linking task j to i if $r_{ij} < 0$. Consequently, the following condition is stated.

Condition A. A pair $(i, j) \notin A$ is considered for linking if

1. The opportunity cost of linking j to i is negative, and
2. i and j belong to different vehicle routes.

To guarantee a monotonic decrease in the objective function value during iterations of the procedure, the basic optimal solution on hand should remain a basic feasible solution for the new **FP** where the nonbasic variable x_{ij} has a negative relative cost. If $r_{ij} < 0$ and arc (i, j)

can be introduced into **A** in such a way that the vehicle routes obtained remain feasible for the next **FP**, then the objective function might decrease (or remain the same due to degeneracy).

Note that this adds another interrelationship to the time windows in addition to the two that have been stated earlier: the sequence of trips on the vehicle routes on hand. Therefore, modifications in the starting times to link two trips (i, j) should be such that the slidden time windows and the starting times will preserve the routes in the optimal solution for the problem on hand, as well as obey the time-window constraints in the **STWP**.

Assume that the trips i and j belong to different vehicle routes in the optimal solution to **FP** and j cannot be linked to trip i according to the ST_i and ST_j on hand; i.e., $ST_i + t_i + t_{ij} > ST_j$. Denote by S_i and S_j the routes including i and j, respectively. If trip j is to be linked to trip i, then trip i should be completed earlier at $(ST_i + t_i - \Delta_i^-)$, and trip j should start later at $(ST_j + \Delta_j^+)$ such that

$$\Delta_i^- + \Delta_j^+ \geq (ST_i + t_i + t_{ij}) - ST_j. \tag{14}$$

If S_i and S_j are to be preserved, then Δ_i^- and Δ_j^+ depend on the adjustments allowed by other trips before trip i in S_i and after trip j in S_j. Then, Δ_i^- and Δ_i^+, the maximal adjustments for ST_i, can be computed recursively for the tasks in the sequences as follows.

$$\Delta_i^- = \begin{cases} ST_i - (c_{i-1} - \Delta c_{i-1}^-) & \text{if } i \text{ is the first trip of } S_i, \\ ST_i - \max\{c_{i-1} - \Delta c_{i-1}^-, ST_\mu + t_\mu + t_{\mu i} - \Delta_\mu^-\} \\ \quad \text{if } \mu \text{ is the immediate predecessor of } i \text{ in } S_i. \end{cases} \tag{15}$$

$$\Delta_j^+ = \begin{cases} c_j - p_j - t_j + \Delta c_j^+ - ST_j & \text{if } j \text{ is the last trip of } S_j, \\ \min\{c_j - p_j - t_j + \Delta c_j^+, \mathrm{ST}_\beta + \Delta^+\beta - t_{j\beta} - t_j\} - ST_j \\ \quad \text{if } \beta \text{ is the immediate successor of } j \text{ in } S_j. \end{cases} \tag{16}$$

To compute these, Δc_{i-1}^- and Δc_j^+, the amount of time c_{i-1} can be decreased and the amount of time c_j can be increased while obeying the constraints that arise from the time windows of other trips that are influenced due to the three interdependencies, which should be computed first. They also need to be computed recursively. These recursive computation schemes are derived and described in Bilge and will be skipped here for the sake of simplicity.

Note that, when computing Δ_i^- by (15) all the influenced time windows slide to the left (backward), and for Δ_i^+ computed through (16) all the influenced time windows slide to the right (forward). It is clear that if there is any interference among the two groups of influenced time windows, then the computed modifications will no longer be valid. Therefore, another condition is necessary to prevent this situation.

Let us denote by BS_i the set including trip i and trips in S_i completed before trip i, and by AS_j the set including trip j and the trips in S_j completed after trip j. The trips i and j form a candidate pair to be linked only if adjustments in one set do not raise modifications in the other one. That will be true if the following condition holds.

Condition B. Here (i, j) forms a candidate pair to be linked if:

1. AS_j does not contain any job predecessors of k, for any k in BS_i;
2. Any r in AS_j (and its job successors) does not precede in machine sequence any k in BS_i (and its job predecessors).

In summary, any pairs of trips i and j are candidates to be linked if both Conditions A and B are satisfied, and they can actually be linked if Δ_i^- and Δ_i^+ computed using (15) and (16) satisfy (14). The proof can be found in Bilge.

However, Δ_i^- and Δ_i^+ are maximal adjustments and one would prefer to make minimal adjustments when modifying the starting times and sliding the time windows, to cause minimal distortions in criteria such as flow times and tardiness, even though they are not considered explicitly in this algorithm. Therefore, one has to recalculate the adjustments for (i, j) such that (14) now holds as an equality.

4.3. Incrementing the Makespan

The set of admissible arcs (AS) for the **STWP** are given by

$$EC_{i-1} + t_i + t_{ij} \leq LC_j - p_j - t_j. \tag{17}$$

If the algorithm fails to find any pairs of trips i and j which can be linked in this set, then, in order to find a vehicle schedule feasible with respect to the given sequence of operations on the machines, the admissible arc set should be expanded by increasing the makespan. An arc $(i, j) \notin AS$ which satisfies Conditions A and B can be included in AS if the makespan is increased by LB_{ij1}, the amount of infeasibility in (17).

$$LB_{ij1} = EC_{i-1} + t_i + t_{ij} - LC_j + p_j + t_j.$$

This still does not guarantee that the newly included arc(s) can be linked. In fact, LB_{ij1} is a loose lower bound because it does not consider preserving the routes. To define a better lower bound, the earliest time EH_i and the latest time LH_i, at which a vehicle can begin to execute trip i so that the vehicle route S_i will be preserved, should be determined. Given a vehicle route S_i, EH_i and LH_i for each trip i on S_i can be computed by using a forward pass, then a backward pass in the same manner as EC_i and LC_i are computed, but this time working on the vehicle routes instead of machine sequences. If trip i (and operation i) belongs to job h and $J_{h(i)}$ gives the serial precedence relationship among trips that belong to job h, then

$$EH_i = \begin{cases} 0 \quad \text{if } i \text{ is the first trip of } J_{h(i)} \text{ and } S_i, \\ \max\{EC_{i-1}, (EH_{i-1} + t_{i-1} + p_{i-1}), (EH_\mu + t_\mu + t_{\mu i})\} \\ \quad \text{if } i \text{ has a predecessor } (i-1) \text{ on } J_{h(i)}, \text{ and } \mu \text{ on } S_i. \end{cases} \tag{18}$$

$$LH_i = \begin{cases} (LC_i - p_i - t_i) \quad \text{if is the last trip in } J_{h(i)} \text{ and } S_i, \\ \min\{(LC_i - p_i - t_i), (LH_{i+1} - p_i - t_i)(LH_\beta - t_{i\beta} - t_i)\} \\ \quad \text{if } i \text{ has a successor } (i+1) \text{ on } J_{h(i)} \text{ and } \beta \text{ on } S_i. \end{cases} \tag{19}$$

A candidate pair (i, j) that satisfies Conditions A and B has the potential to be linked without destroying the routes on hand, if

$$EH_i + t_i + t_{ij} \leqslant LH_j. \tag{20}$$

Such pairs will be called feasible candidates. Otherwise, the minimum amount of increase in the makespan that is required for this pair (i, j) to become a feasible candidate is given by

$$LB_{ij2} = EH_i + t_i + t_{ij} - LH_j. \tag{21}$$

In the algorithm, all candidates that are infeasible to the current makespan (i.e., those which do not satisfy (20)) are kept in a list in ascending order of their lower bounds, LB. Then, if it turns out that all feasible candidates fail to satisfy (14), instead of stopping without a solution with the desired number of vehicles, it is preferable to search for a vehicle routing solution by increasing the makespan in an iterative manner. Initially, the makespan is increased by the lower bound of the first arc on the infeasible arc list, and all the arcs with the same lower bound are tested in order of their opportunity costs to check whether a link is possible. If so, the algorithm continues with this makespan. Otherwise, the makespan is increased once again using the lower bound of the first arc of the list after discarding the already tested arcs. In this manner, the makespan is increased until a solution is found or the makespan exceeds a given upper bound (i.e., the makespan of a previously obtained solution).

4.4. STW Heuristic Algorithm

STEP 1. Set the starting times at initial values, ST, which are feasible to the **STWP**.

STEP 2. Solve the associated **FP**. If the number of routes in the optimal solution is less than or equal to the number of vehicles available then report the solution and stop.

STEP 3. Prepare the candidate and infeasible arc list: For all pairs (i, j) which satisfy Conditions A and B, if $EH_i + t_i + t_{ij} \leqslant LH_j$, then put (i, j) in the candidate list in ascending order of opportunity; else, if makespan + $LB_{ij2} \leqslant UB$, then put (i, j) in the infeasible arcs list in ascending order of LB.

STEP 4. Check candidate arcs: Take the first candidate pair (i, j). Compute Δ_i^- and Δ_j^+. If a link is possible ((14) is satisfied), then make minimal adjustments in ST and c, and go to Step 2. Otherwise, repeat this step until all candidates are tested.

STEP 5. Increase the makespan: Take the first infeasible arc (i, j). Increase the makespan by LB_{ij2}. Compute Δ_i^- and Δ_j^+. If a link is possible, adjust ST and c, and go to Step 2. Otherwise, repeat this step until all infeasible arcs are tested.

STEP 6. A solution with makespan $\leqslant UB$ cannot be obtained with this machine sequence. Stop.

5. NUMERICAL STUDY

5.1. Example Problems

The testing of the iterative algorithm is performed in two ways: First, the behavior of the different schedule generation and seed schedule selection rules are compared and the efficiency of the algorithm against a single-pass heuristic is tested. For this purpose, 82 problems are generated. The problems are designed at different levels of the ratio of travel times to processing times because a major parameter influencing the interaction of the vehicle and machine schedules is the relative magnitudes of the processing times with respect to the travel times. Second, eight new problems are generated and solved in addition to eight existing problems to explore the interaction between the layout configuration and the process routes and its impact on the performance of the algorithm. The generation of example problems is based to a large extent on Bilge.

The ratio of travel times to processing times has been the main concern when generating the example problems. By employing basic layout types encountered in the literature (e.g., Kusiak 1985) such as loop or ladder or their combinations, four different layouts are generated (Figure 2) each with four machines. The travel times, i.e., deadheading trip durations, on each layout are given in Appendix A. The loaded trip times are obtained by adding loading and unloading times to travel times. Ten different job sets with different processing sequences, and process times are generated and presented in Appendix B. Different combinations of these ten job sets and four layouts are used to generate 82 example problems. In all these problems there are two vehicles. Table I consists of problems whose $\bar{t}_{ij}/\bar{p}_i$ ratios are lower than 0.25, and those with higher $\bar{t}_{ij}/\bar{p}_i$ ratios are represented in Table II. A code is used to designate the example problems which are given in the first column. The digits that follow EX indicate the job set and the layout. In Table II another digit is appended to the code.

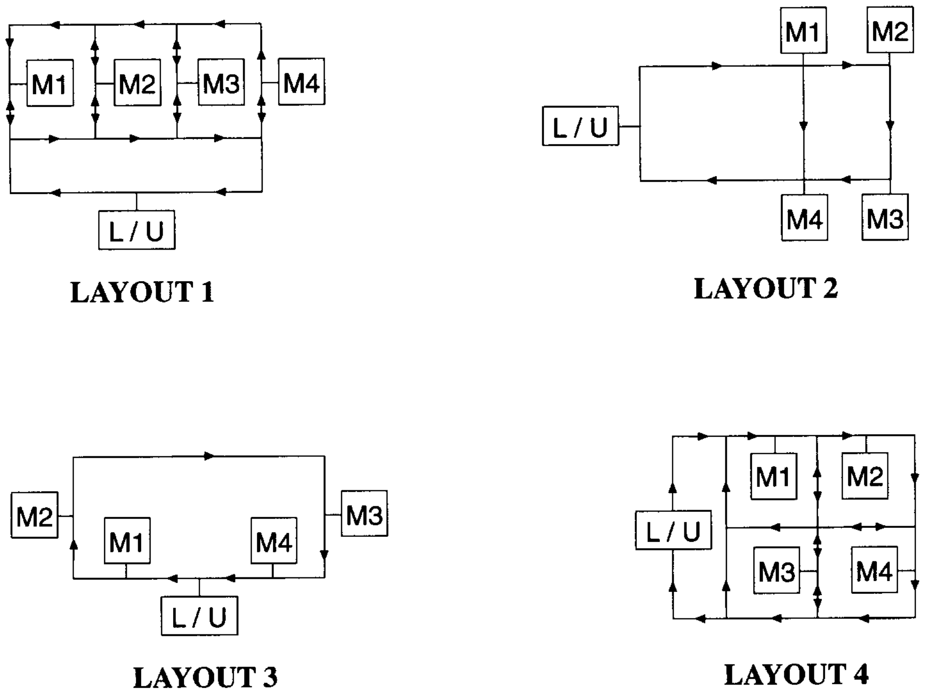

Figure 2. The layout configurations used in generating the example problems.

Here, having a 0 or 1 as the last digit implies that the process times are doubled or tripled, respectively, where in both cases travel times are halved.

5.2. A Single-Pass Heuristic Procedure

To provide a basis for comparison of efficiency, a single-pass heuristic procedure is developed, which can also be employed to obtain an upper bound at the beginning of the iterative solution procedure. This procedure is a variation of Giffler and Thompson's active schedule algorithm where vehicle availability is taken into account when calculating the earliest starting times of operations. The vehicle that can arrive first at the point of use is selected.

Three active schedule generating and three nondelay schedule generating versions of this single-pass heuristic are formed by employing the three dispatching rules mentioned before, namely **MWKR**, **LWKR**, and **SPT**. The example problems are solved using each of these rules. The best solutions are recorded for later use in making comparisons against the results of the proposed iterative technique and are reported in Tables I and II. Detailed results of this experiment can be found in Bilge. We observe that among active schedule generating versions, the **MWKR** rule performs better, while among nondelay schedule versions the **SPT** rule produces better results most of the time. Hence, with the iterative solution procedure, only these two schedule generation strategies will be used, that is, active with **MWKR** (Algorithm 1), and nondelay with **SPT** (Algorithm 2).

5.3. Numerical Study With the Proposed Algorithm

The results for Algorithms 1 and 2, under the seed selection rule of "maximum machine conflict" (Rule 2) are presented in Tables I and II. Algorithm 2 is superior 84% of the time. Then, for each problem, another strategy is tested by choosing the schedule generation method, which performed better, and changing the seed selection rule to "minimum makespan" (Rule 1). The results, which are displayed on the sixth column of Tables I and II, show that Rule 1 performs at least as well as Rule 2 with only a few exceptional cases (84%). This implies that the use of Rule 2, which is more complicated than Rule 1, is not justified.

In the seventh column of Table I, the percentage of improvement achieved by the best iterative solution over the best single-pass heuristic result is presented. For problems in Table I the average percentage of improvement is 4.2%. The problems in Table II can be grouped with respect to three $\bar{t}_{ij}/\bar{p}_i$ ratio levels: problems with ratios below 0.60, those with ratios between 0.60 and 0.75, and those with ratios higher than 0.75. For these three groups of problems the average percentage of improvement are 4.7%, 14.2%, and 16.7%, respectively. Thus, there is a substantial benefit in using the iterative technique, especially as the ratio of average travel time to average process time increases.

In all these problems the maximum iteration number allowed is 15. For most of the problems (especially in Table II), the procedure stops reaching this maximum iteration number, and the best solution is found in earlier iterations. The iteration number at which the best iterative solution is obtained is reported in the last column of Tables I and II. We observe that 62% of the solutions are found in less than or equal to three iterations, 84% in less than or equal to seven iterations, and the average number of iterations required is 4.2. This shows that a

Table I
Summary of the Results for Example Problems With $\bar{t}_{ij}/\bar{p}_i < 0.25$

Problem	$\bar{t}_{ij}/\bar{p}_i$	Best Single-pass Heuristic (BSH)	Rule 2 Maximum Machine Conflict		Rule 1 Minimum Makespan	% Improvement for Best Iterative Solution Over BSH	Iteration Number for Best Iterative Solution
			ALG1	ALG2			
EX110	0.15	126	152	126	**126**	0.0	1
EX210	0.15	153	168	148	**148**	3.3	2
EX310	0.15	157	168	150	**150**	4.5	3
EX410	0.15	123	130	121	**121**	1.6	4
EX510	0.21	115	108	102	**102**	11.3	3
EX610	0.16	194	216	**186**	190	4.1	2
EX710	0.19	146	151	138	**137**	6.2	6
EX810	0.14	292	320	292	**292**	0.0	2
EX910	0.15	186	185	**176**	179	5.4	4
EX1010	0.14	260	284	238	**238**	8.5	2
EX120	0.12	127	150	123	**123**	3.1	1
EX220	0.12	148	159	143	**143**	3.0	2
EX320	0.12	150	165	148	**148**	1.3	2
EX420	0.12	116	125	116	**116**	0.0	1
EX520	0.17	100	100	100	**100**	0.0	1
EX620	0.12	189	199	**183**	185	3.2	4
EX720	0.15	141	152	136	**136**	3.5	2
EX820	0.11	287	317	287	**287**	0.0	2
EX920	0.12	181	183	174	**174**	3.9	2
EX1020	0.11	257	276	236	**236**	8.2	2
EX130	0.13	122	149	122	**122**	0.0	1
EX230	0.13	149	150	146	**146**	2.0	2
EX330	0.13	153	166	149	**149**	2.6	2
EX430	0.13	133	126	116	**116**	12.8	2
EX530	0.18	99	99	99	**99**	0.0	1
EX630	0.14	190	200	**184**	186	3.2	4
EX730	0.17	144	153	137	**137**	4.8	2
EX830	0.13	288	318	288	**288**	0.0	2
EX930	0.13	182	181	176	**176**	3.3	2
EX1030	0.12	258	277	237	**237**	8.2	2
EX140	0.18	132	154	124	**124**	6.1	1
EX241	0.13	224	223	217	**217**	3.1	2
EX340	0.18	159	**151**	154	154	5.0	2
EX341	0.12	229	247	222	**222**	3.1	2
EX441	0.19	197	197	179	**179**	9.1	2
EX541	0.18	168	168	154	**154**	8.3	2
EX640	0.19	195	223	189	**185**	5.1	3
EX740	0.24	148	156	**138**	142	6.8	10
EX741	0.16	214	230	203	**203**	3.3	11
EX840	0.18	293	325	293	**293**	0.0	2
EX940	0.19	183	186	**177**	181	3.3	7
EX1040	0.17	263	285	240	**240**	8.7	3

relatively small maximum iteration number can be selected as the stopping criterion.

The iterative technique yields improvement over the single-pass heuristic in two ways:

1. Its schedule generation scheme, which resembles, to some extent, a branch-and-bound scheme, may bring out good machine schedules, both in terms of makespan and shuffled operations, to create suitable slack times in between for handling.
2. The single-pass heuristic can be regarded as a real-time schedule generation scheme which uses a dispatching rule coupled with a selection rule without a lookahead nature. Therefore, the vehicle assignment at any iteration affects the subsequent decisions and may inflate the makespan. On the other hand, the STW heuristic takes into account the combinatorial nature of the problem, anticipating all of the flow requirements created by a particular machine schedule and making vehicle assignments accordingly.

When the travel times are low relative to process times, both of these effects will be diminished. Because of the long process times, the machine schedules generated will

Table II
Summary of the Results for Example Problems With $\bar{t}_{ij}/\bar{p}_t > 0.25$

Problem	$\bar{t}_{ij}/\bar{p}_t$	Best Single-pass Heuristic (BSH)	Rule 2 Maximum Machine Conflict		Rule 1 Minimum Makespan	% Improvement for Best Iterative Solution Over BSH	Iteration Number for Best Iterative Solution
			ALG1	ALG2			
EX11	0.59	108	101	96	**96**	11.1	2
EX21	0.61	108	114	106	**105**	2.8	6
EX31	0.59	125	105	107	**105**	16.0	5
EX41	0.91	143	125	118	**118**	17.5	12
EX51	0.85	111	**89**	90	90	18.9	3
EX61	0.78	128	137	127	**120**	6.3	10
EX71	0.78	133	**119**	123	122	10.5	13
EX81	0.58	165	189	**161**	164	2.4	8
EX91	0.61	130	125	123	**120**	7.7	7
EX101	0.55	156	169	**153**	155	1.9	3
EX12	0.47	85	83	82	**82**	3.5	7
EX22	0.49	80	86	80	**80**	0.0	8
EX32	0.47	90	88	88	**88**	2.2	7
EX42	0.73	103	94	93	**93**	9.7	1
EX52	0.68	88	69	72	**69**	21.6	2
EX62	0.54	102	113	100	**100**	1.9	2
EX72	0.62	90	90	91	**90**	0.0	3
EX82	0.46	151	158	151	**151**	0.0	2
EX92	0.49	112	110	107	**104**	7.1	14
EX102	0.44	143	149	**139**	144	2.8	3
EX13	0.52	94	84	86	**84**	10.6	2
EX23	0.54	90	92	90	**86**	4.4	14
EX33	0.51	94	86	86	**86**	7.5	7
EX43	0.80	108	95	100	**95**	12.0	2
EX53	0.74	86	77	**76**	83	11.6	4
EX63	0.54	112	116	105	**104**	7.1	10
EX73	0.68	104	98	100	**91**	12.5	7
EX83	0.50	153	164	153	**153**	0.0	2
EX93	0.53	118	113	113	**110**	6.8	8
EX103	0.49	148	155	150	**143**	3.4	12
EX14	0.74	144	109	108	**108**	25.0	2
EX24	0.77	131	124	117	**116**	11.5	7
EX34	0.74	130	116	118	**116**	10.8	3
EX44	1.14	163	134	126	**126**	22.7	3
EX54	1.06	129	99	103	**99**	23.3	1
EX64	0.78	154	138	129	**120**	22.0	6
EX74	0.97	159	**136**	150	138	14.5	2
EX84	0.72	193	205	172	**163**	15.5	5
EX94	0.76	164	133	129	**125**	23.8	11
EX104	0.69	173	182	174	**171**	1.2	5

be more likely to have suitable slack times to absorb the low deadheading times required in between; thus, the $C'_{\max}$ at the end of the STW heuristic will probably be close to the original $C_{\max}$ of the machine schedule. This will create tight upper bounds for the schedule generation scheme, where any neighboring machine schedules with $C_{\max} \geqslant UB$ are discarded. As a consequence, the number of machine schedules generated during the procedure will be less. Actually, in nearly all of the problems in Table I, the procedure stops due to the second stopping criterion; that is, the set of machine schedules becomes empty. In Table II, on the other hand, nearly all problems reached the maximum number of iterations allowed.

Second, with relatively low travel times it will be less often that vehicle availability becomes a constraint during the single-pass heuristic procedure, so the adverse effect of the shortsighted vehicle selection will be less important. Furthermore, any improvement obtained through the use of the STW heuristic in this respect will be lower in magnitude, because the magnitudes of travel times are low.

This explains why it is beneficial to use the iterative technique instead of a real-time scheme such as the single-pass heuristic, and how the performance improves as the ratio of the average travel time to the average process time gets higher.

The problems in Tables I and II are sorted according to their layouts. Looking closer, one can observe that, while high improvements are achieved on layouts 1 and 4, improvements obtained on layouts 2 and 3 are less. Furthermore, the improvement levels attained by job sets 1, 4, and 5 are higher than those of job sets 2 and 3. A study of the characteristics of these layouts and job sets reveals that layouts 1 and 4 are more complex than layouts 2 and 3. Furthermore, in the process routes of job sets 2 and 3, jobs visit machines in the order they appear on the layouts, whereas job sets 1 and 4 contain process routes where returning to a previsited or passed-by machine is required.

These observations need to be clarified and supported through statistical analysis. For this purpose, layouts 2 and 3 and job sets 2 and 3 are selected. Those job sets and layouts make smart combinations with respect to the flow of materials. The experimental design consists of 16 example problems, eight of which have been generated already. The layouts are modified by interchanging the locations of two of the machines to suppress the smoothness of the flow. In this way, problem pairs, for which the $\bar{t}_{ij}/\bar{p}_i$ ratios are the same but material flow characteristics are different, are obtained. In layout 2A, machines M1 and M4, and in layout 3A, machines M2 and M4 are exchanged. The resulting problems are EX22A, EX23A, EX33A, and EX32A. Furthermore, to investigate the same effect on problems of low $\bar{t}_{ij}/\bar{p}_i$ ratios, another set of examples is generated, where the travel times of layouts 2A and 3A are halved, and the process times of job sets 2 and 3 are doubled. These problems are called EX20A, EX230A, EX30A, and EX320A. All problems are solved using Algorithm 2 and Rule 1, as well as the single-pass heuristic; the results are displayed in Table III.

A paired t-test reveals that different flow patterns significantly affect the performance of the algorithm. With smooth flow patterns where the process routes and the layout configurations suit each other well, the criticality of the vehicle selection is reduced, and therefore the effects of the shortsighted approach are less important. However, as the flow pattern becomes more complex, the performance of the single-pass heuristic declines, and more benefits are expected from the iterative procedure. Even when the travel times are low compared to process times, if the jobs to be processed create an inconvenient material flow on the layout, the use of the iterative technique is strongly recommended.

6. CONCLUSION

The purpose of this study is to make AGV scheduling an integral part of the scheduling activity, actively participating in the specification of the off-line schedule, rather than just reacting to it. The iterative algorithm created anticipates the complete set of flow requirements for a given machine schedule and makes vehicle assignments accordingly, as opposed to a real-time dispatching scheme that uses no information other than the move request queue.

Table III
The Effect of Material Flow Pattern on the Performance of the Procedure[a]

Problem	$\bar{t}_{ij}/\bar{p}_i$	Best Single-Pass Heuristic (BSH)	Iterative Procedure	% Improvement Over BSH
EX22	0.49	80	80	0
EX22A		99	97	2.0
EX23	0.54	90	86	4.4
EX23A		117	99	15.4
EX33	0.51	94	86	8.5
EX33A		136	112	17.6
EX32	0.47	90	88	2.2
EX32A		116	102	12.1
EX220	0.12	148	143	3.4
EX220A		153	147	3.9
EX230	0.13	149	146	2.0
EX230A		156	146	6.4
EX330	0.13	153	149	2.6
EX330A		168	152	9.5
EX320	0.12	153	148	3.3
EX320A		168	149	11.3

[a] $t_{\text{stat}} = 4.5$, Prob($t \leq 3.5$) = 0.995 with seven degrees of freedom; null hypothesis ($\mu_A - \mu_B = 0$) is rejected.

The impact of the relative magnitudes of processing and travel times, and the complexity of the material flow pattern defined by the interaction between the process routes and the layout configuration are analyzed. The results presented in subsection 5.3 demonstrate the significant benefits of using the proposed iterative algorithm instead of the single-pass heuristic, which can be regarded as a real-time schedule generation scheme that uses a dispatching rule coupled with a shortsighted vehicle selection rule. The iterative algorithm promises improvement in scheduling especially in environments where cycle times are short and travel times are comparable, or where the layout and the process routes do not suit each other.

The **STW** heuristic algorithm used in this study is a tool on its own, whose applicability to other problems with similar characteristics should be investigated. A few such potential areas of application may be:

- Resource constrained project scheduling when resources have sequence-dependent setup times,
- Scheduling on parallel machines with sequence-dependent setups, where operations have ready times and due dates,
- Dial-a-ride problems where requested trip start times are given as time intervals.

It is also possible to adapt the **STW** approach to performance criteria other than the makespan, such as the minimization of maximum tardiness. In this case, one would need to redefine the time windows and the opportunity cost structure. The time windows for the operation completion times would be stated by operational due dates obtained from job due dates relaxed by the maximum tardiness value given by the machine schedule. In machine schedule generation, on the other hand, due date related priority rules can be employed in conjunction with the active or nondelay schedule generators. Definition of new dominant subsets of schedules for due date related criteria for the jobshop problem is worthy of additional research.

The proposed algorithm can be used in a rolling time-horizon basis for dynamic scheduling if all previous assignments regarding machines and AGVs are relinquished at the beginning of each run. Otherwise, its applicability to dynamic scheduling needs more research.

APPENDIX A

Travel Time Data Used in Example Problems

Layout 1	L/U	M1	M2	M3	M4
L/U	0	6	8	10	12
M1	12	0	6	8	10
M2	10	6	0	6	8
M3	8	8	6	0	6
M4	6	10	8	6	0

Layout 2	L/U	M1	M2	M3	M4
L/U	0	4	6	8	6
M1	6	0	2	4	2
M2	8	12	0	2	4
M3	6	10	12	0	2
M4	4	8	10	12	0

Layout 3	L/U	M1	M2	M3	M4
L/U	0	2	4	10	12
M1	12	0	2	8	10
M2	10	12	0	6	8
M3	4	6	8	0	2
M4	2	4	6	12	0

Layout 4	L/U	M1	M2	M3	M4
L/U	0	4	8	10	14
M1	18	0	4	6	10
M2	20	14	0	8	6
M3	12	8	6	0	6
M4	14	14	12	6	0

APPENDIX B

Data for the Job Sets Used in Example Problems

Job Set 1

Job 1: M1(8); M2(16); M4(12)
Job 2: M1(20); M3(10); M2(18)
Job 3: M3(12); M4(8); M1(15)
Job 4: M4(14); M2(18)
Job 5: M3(10); M1(15)

Job Set 2

Job 1: M1(10); M4(18)
Job 2: M2(10); M4(18)
Job 3: M1(10); M3(20);
Job 4: M2(10); M3(15); M4(12)
Job 5: M1(10); M2(15); M4(12)
Job 6: M1(10); M2(15); M3(12)

Job Set 3

Job 1: M1(16); M3(15)
Job 2: M2(18); M4(15)
Job 3: M1(20); M2(10)
Job 4: M3(15); M4(10)
Job 5: M1(8); M2(10); M3(15); M4(17)
Job 6: M2(10); M3(15); M4(8); M1(15)

Job Set 4

Job 1: M4(11); M1(10); M2(7)
Job 2: M3(12); M2(10); M4(8)
Job 3: M2(7); M3(10); M1(9); M3(8)
Job 4: M2(7); M4(8); M1(12); M2(6)
Job 5: M1(9); M2(7); M4(8); M2(10); M3(8)

Job Set 5

Job 1: M1(6); M2(12); M4(9)
Job 2: M1(18); M3(6); M2(15)
Job 3: M3(9); M4(3); M1(12)
Job 4: M4(6); M2(15)
Job 5: M3(3); M1(9)

Job Set 6

Job 1: M1(9); M2(11); M4(7)
Job 2: M1(19); M2(20); M4(13)
Job 3: M2(14); M3(20); M4(9)
Job 4: M2(14); M3(20); M4(9)
Job 5: M1(11); M3(16); M4(8)
Job 6: M1(10); M3(12); M4(10)

Job Set 7

Job 1: M1(6); M4(6)
Job 2: M2(11); M4(9)
Job 3: M2(9); M4(7)
Job 4: M3(16); M4(7)
Job 5: M1(9); M3(18)
Job 6: M2(13); M3(19); M4(6)
Job 7: M1(10); M2(9); M3(13)
Job 8: M1(11); M2(9); M4(8)

Job Set 8

Job 1: M2(12); M3(21); M4(11)
Job 2: M2(12); M3(21); M4(11)
Job 3: M2(12); M3(21); M4(11)
Job 4: M2(12); M3(21); M4(11)
Job 5: M1(10); M2(14); M3(18); M4(9)
Job 6: M1(10); M2(14); M3(18); M4(9)

Job Set 9

Job 1: M3(9); M1(12); M2(9); M4(6)
Job 2: M3(16); M2(11); M4(9)

Job 3: M1(21); M2(18); M4(7)
Job 4: M2(20); M3(22); M4(11)
Job 5: M3(14); M1(16); M2(13); M4(9)

Job Set 10

Job 1: M1(11); M3(19); M2(16); M4(13)
Job 2: M2(21); M3(16); M4(14)
Job 3: M3(8); M2(10); M1(14); M4(9)
Job 4: M2(13); M3(20); M4(10)
Job 5: M1(9); M3(16); M4(18)
Job 6: M2(19); M1(21); M3(11); M4(15)

REFERENCES

Bartholdi, J. J., and L. K. Platzman. 1989. Decentralized Control of AGVs on a Simple Loop. *IIE Trans.* **21,** 76–81.

Bilge, Ü. 1991. Simultaneous Scheduling of Machines and the Material Handling System in a Flexible Manufac-
turing System. Ph.D. Dissertation. Department of Industrial Engineering, Boğaziçi University, Istanbul, Turkey.

Bozer, Y. A. 1989. Guided Vehicle Systems: Information/Control System Implication of Alternative Design and Operation Strategies. In *Advanced Information Technologies for Industrial Material Flow Systems*, S. Y. Nof and C. L. Moodie (eds.). NATO ASI Series, Springer-Verlag, Berlin, 417–436.

Chang, Y., R. S. Sullivan and J. R. Wilson. 1986. Using SLAM to Design the Material Handling System of an FMS. *Int. J. Prod. Res.* **24,** 15–26.

Egbelu, P. J., and J. M. A. Tanchoco. 1984. Characterization of Automated Guided Vehicle Dispatching Rules. *Int. J. Prod. Res.* **22,** 359–374.

Ferland, J. A., and L. Fortin. 1989. Vehicles Scheduling With Sliding Time Windows. *Eur. J. Opnl. Res.* **38,** 213–226.

French, S. 1982. *Sequencing and Scheduling*. Ellis Horwood Limited, England.

Gobal, S. L., and R. G. Kasilingam. 1991. A Simulation Model for Estimating Vehicle Requirements in Automated Guided Vehicle Systems. *Comp. and Ind. Eng.* **21,** 623–627.

Giffler, B., and G. L. Thompson. 1960. Algorithms for Solving Production Scheduling Problems. *Opns. Res.* **8,** 478–503.

Greenwood, N. R. 1988. *Implementing Flexible Manufacturing Systems*. Macmillan, London.

Grosseschallau, W., and A. Kusiak. 1985. An Expert System for Design of Automated Material Handling Systems (INSIMAS). *Material Flow* **2,** 157–166.

Han, M., and L. F. Mcginnis. 1989. Control of Material Handling Transporter in Automated Manufacturing. *IIE Trans.* **21,** 184–190.

Kiran, A. S., and B. C. Tansel. 1986. Mathematical Programming Models for FMSs. Working Paper 86-01, Department of Industrial and Systems Engineering, University of Southern California, Los Angeles.

Kusiak, A. 1985. Material Handling in Flexible Manufacturing Systems. *Material Flow* **2,** 79–95.

Kouvelis, P. 1992. Design and Planning Problems in Flexible Manufacturing Systems: a Critical Review. *J. Intell. Manuf.* **3,** 75–99.

Lenstra, J. K., and A. H. G. Rinnooy Kan. 1978. Complexity of Scheduling Under Precedence Constraints. *Opns. Res.* **26,** 22–35.

Mahadevan, B., and T. T. Narendran. 1990. Design of an Automated Guided Vehicle-Based Material Handling System for an FMS. *Int. J. Prod. Res.* **28,** 1611–1622.

Malmborg, C. J. 1990. A Model for the Design of Zone Control Automated Guided Vehicle Systems. *Int. J. Prod. Res.* **28,** 1741–1758.

Maxwell, W. L., and J. A. Muckstadt. 1982. Design of Automated Guided Vehicles. *IIE Trans.* **14,** 114–124.

Orloff, C. S. 1976. Route Constrained Fleet Scheduling. *Trans. Sci.* **10,** 149–168.

Raman, N., F. B. Talbot and R. V. Rachamadgu. 1986. Simultaneous Scheduling of Machines and Material Handling Devices in Automated Manufacturing. In *Proceedings of the Second ORSA/TIMS Conference on Flexible Manufacturing Systems*, K. E. Stecke and R. Suri (eds.). Elsevier, Amsterdam, 455–466.

Ro, I., and J. Kim. 1990. Multi-Criteria Operational Control Rules in Flexible Manufacturing Systems. *Int. J. Prod. Res.* **28,** 47–63.

Sabuncuoğlu, I., and D. L. Hommertzheim. 1992a. Experimental Investigation of FMS Machine and AGV Scheduling Rules Against the Mean Flow-time Criterion. *Int. J. Prod. Res.* **30,** 1617–1635.

Sabuncuoğlu, I., and D. L. Hommertzheim. 1992b. Dynamic Dispatching Algorithm for Scheduling Machines and Automated Guided Vehicles in a Flexible Manufacturing System. *Int. J. Prod. Res.* **30,** 1059–1079.

Solomon, M. M., and J. Desrosiers. 1988. Time Window Constrained Routing and Scheduling Problems. *Trans. Sci.* **22,** 1–13.

Stecke, K. E. 1985. Design, Planning, Scheduling, and Control Problems of Flexible Manufacturing Systems. *Anns. Opns. Res.* **3,** 3–12.

Wu, S. D., and R. A. Wysk. 1989. An Application of Discrete-Event Simulation to On-Line Control and Scheduling in Flexible Manufacturing. *Int. J. Prod. Res.* **27,** 1603–1623.