The International Journal of Flexible Manufacturing Systems, 8 (1996): 247-262

Dispatching, Routing, and Scheduling of Two Automated Guided Vehicles in a Flexible Manufacturing System

ANDRÉ LANGEVIN
École Polytechnique de Montréal, Department of Applied Mathematics, P.O. Box 6079, Station Centre-Ville, Montréal, Québec, Canada H3C 3A7

DANIEL LAUZON
École Polytechnique de Montréal, Department of Applied Mathematics, P.O. Box 6079, Station Centre-Ville, Montréal, Québec, Canada H3C 3A7

DIANE RIOPEL
École Polytechnique de Montréal, Department of Industrial Engineering, P.O. Box 6079, Station Centre-Ville, Montréal, Québec, Canada H3C 3A7

Abstract. This article presents a new approach for planning the dispatching, conflict-free routing, and scheduling of automated guided vehicles in a flexible manufacturing system. The problem is solved optimally in an integrated manner, contrary to the traditional approach in which the problem is decomposed in three steps that are solved sequentially. The algorithm is based on dynamic programming and is solved on a rolling time horizon. Three dominance criteria are used to limit the size of the state space. The method finds the transportation plan minimizing the makespan (the completion time for all the tasks). Various results are discussed. A heuristic version of the algorithm is also proposed for an extension of the method to many vehicles.

Key Words: automated guided vehicles, flexible manufacturing, material handling, routing, scheduling.

1. Introduction

Automated guided vehicles (AGVs) are programmable material handling equipment used in a flexible manufacturing system (FMS). These vehicles transport tools and materials between different workstations. FMSs consist of several workstations performing the specific functions of machining, plus some service functions such as tooling, storage, etc. When production of each required part type is divided into lots, each production lot specifies the movement of tools, fixtures, and parts through the workstations. Different lots may specify different sets of workstations and thus different sequences in which the workstations must be visited.

The AGVs circulate on a network of guidepaths connecting the various workstations. AGV technology has introduced new challenges both in the planning and management of the material handling system such as fleet sizing, guidepath network and workstation layout design, lot sizing, and vehicle management (see Co and Tanchoco, 1991; Maxwell and Muckstadt, 1982). Vehicle management itself is composed of three main functions:

- dispatching, which consists of assigning transportation tasks to vehicles;

- routing, which consists of selecting the route that each vehicle will follow in order to accomplish its transportation tasks;
- scheduling, which consists of determining the times at which vehicles enter and leave each guidepath segment on their routes in order to avoid conflicts.

As there can be more than one vehicle in the system simultaneously, two or more vehicles may be competing for the same path at the same time. Thus, the routing and scheduling of the vehicles should take into account these possible conflicts.

The problem to solve consists of determining the dispatching, the routing, and the scheduling of the fleet of AGVs. The main objective of the vehicle controller is to satisfy the transportation demands in the shortest time and in a nonconflicting manner. The inputs to the computerized controller of the AGV system are the layout of the guidepath, the number of vehicles in the system and their characteristics (travel speed, loading, and unloading times), and the transportation requests with their corresponding due dates.

The traditional approach to these kind of problems has been to decompose the problem in three steps, and to optimally or heuristically solve them sequentially. Some authors have proposed methods to solve two combined steps. However, to our knowledge, it is the first time that the three steps, i.e., dispatching, routing, and scheduling, are integrated and solved optimally.

2. Literature review

Various strategies have been proposed for vehicle management, as evidenced by the increasing literature on the subject. We shall survey the more relevant articles here. Egbelu and Tanchoco (1984) propose a set of heuristic dispatching rules. These rules are divided into two classes: workstation-initiated and vehicle-initiated, depending on whether the system has idle vehicles (vehicle-initiated) or whether the system has transportation requests queued (workcenter-initiated). These heuristic rules are used to assign transportation requests to vehicles. The results demonstrate their effects on the performance of a 13-machine and 6-vehicle FMS with a unidirectional guidepath. Kusiak and Cyrus (1985) present a heuristic method to find an approximate solution to an integer linear program formulation of the vehicle scheduling problem with time window constraints on each movement. In their model, dispatching is done by a heuristic method that does not take into account traffic management.

Taghaboni and Tanchoco (1988) propose a control strategy based on dispatching rules in Egbelu and Tanchoco (1984), but they also plan vehicle routes which avoid conflicts. When a vehicle is dispatched to a new task, all preestablished routes for other vehicles are considered fixed, and a route is chosen for the new task so as to avoid any conflict with the preplanned routes of the other vehicles. The intersection conflict is solved on a first-come, first-served basis. Fujii, Sandoh, and Hohzaki (1988) develop a control model to minimize vehicle interference and idle time. Routes are obtained by solving an LP. For a given set of paths (one per vehicle), this LP produces a conflict-free schedule that minimizes delays. The method produces its routing and scheduling solution by solving this LP for different sets of paths. These paths are the k-shortest paths for a particular vehicle. The computational complexity of this approach makes it inapplicable in a real-time context.

Krishnamurthy, Batta, and Karwan (1991) present a method for obtaining conflict-free routes for AGVs via an LP model. They assume that the dispatching has already been done. The proposed model discretizes time and fixes the routes of all vehicles simultaneously. The LP is solved by a column generation heuristic. Palekar, Kapoor, and Huang (1990) introduce a method using a shortest path with time-windows algorithm to obtain conflict-free routes for an AGV system so as to avoid all conflicts with preplanned routes of other vehicles. The dispatching is done by a heuristic method. Kim and Tanchoco (1991) propose a similar approach using a time-windows-constrained shortest path algorithm.

Daniels (1991) proposes a branch-and-bound technique to obtain a conflict-free route for a new vehicle task. Branching is done on sets of possible paths. Here again, the dispatching of the tasks to the vehicles is done heuristically. Sabuncuoglu and Hommertzheim (1992) propose a dynamic dispatching algorithm for scheduling machines and AGVs. They define an FMS as two interrelated subsystems: a machining subsystem and a material handling subsystem. Both subsystems must be taken into account simultaneously in a real-time scheduling decision. A similar approach is proposed by Blazewicz et al. (1991). However, they only solve very small problems (three machines, nine operations, two vehicles). Sabuncuoglu and Hommertzheim (1993) investigate the scheduling problems of FMSs. They analyze the relative performances of machine and AGV scheduling rules against various due-date criteria. The rules are tested by using simulation. The objective is to model three important elements of FMSs (machines, AGVs, a finite buffer capacities) and their interactions.

In this article, we propose an optimal dynamic programming approach to determine both the dispatching of the transportation tasks, and the routing and scheduling of the vehicles. It constitutes a more general approach to the dispatching *and* routing problems. Routing conflicts are resolved using a shortest path with time-windows algorithm. Our method uses a state-space of partial transportation plans (dispatches and associated conflict-free routes) to obtain a solution to the dispatching/routing problem over a certain time horizon. This method is reiterated on a rolling time horizon for real-time operation. The travel speed of the vehicles is assumed constant and the vehicles may travel along a track segment in either direction.

3. The method

The production schedule is assumed to be known on a certain time horizon. This production schedule generates a set of transportation "requests" or tasks. A request consists of a pickup point with an earliest time and a drop-off point. The AGV system is represented by a network, i.e., a set of guidepath segments (the edges of the network) delimited by control points (the nodes) (see figure 1). A weight representing travel time is associated to each edge. We consider two bidirectional vehicles.

The objective is to obtain a transportation plan for all of the requests, that is, we seek a schedule and an itinerary for each vehicle. However, as both vehicles travel on the same network, possible conflicts, i.e., simultaneous occupations of a node or an edge, must be avoided. The method that is presented herein allows one to find the transportation plan that minimizes the makespan (the completion time for all tasks).

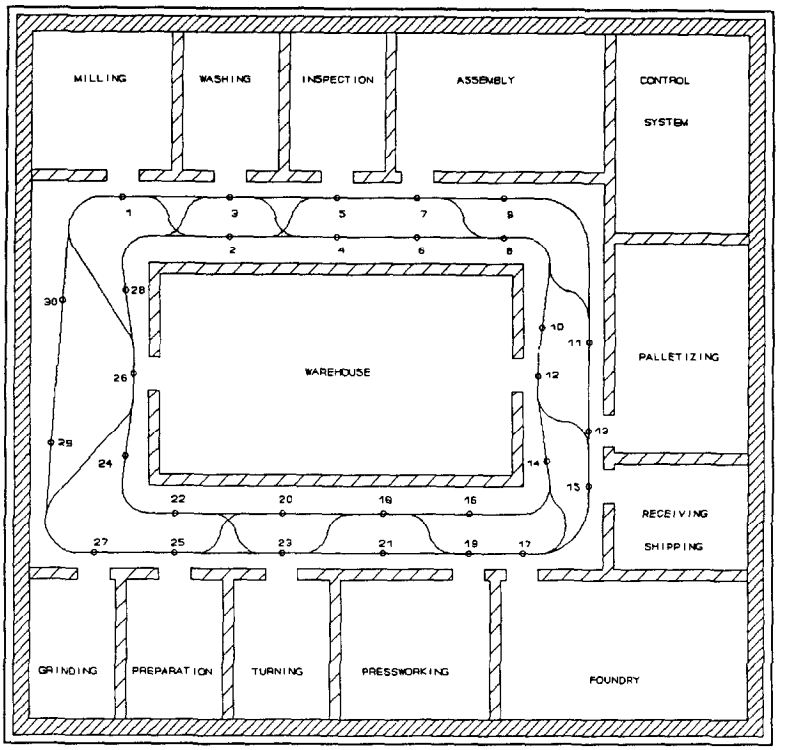

Figure 1. The FMS layout with the AGV network.

3.1. The method of solution

To solve the problem, a dynamic programming approach is used. In this subsection, an outline of the method is presented. The dynamic programming scheme is detailed in the next subsection.

Let us define a *partial transportation plan* as a schedule and a route for each vehicle satisfying a subset of the transportation tasks. The underlying idea is to iteratively construct partial transportation plans with more and more tasks until complete transportation plans are obtained. From a given partial transportation plan, the best way to add another task to each one of the two routes (one for each vehicle) is found, hence generating two new partial transportation plans, including one more task. Each time a task is added at the end of the itinerary of a vehicle, a shortest path problem is solved. Then possible conflicts between the two routes are detected. If no conflict is detected, then a new partial plan is generated. If a conflict is detected, one of the two routes is kept fixed, while the other is modified. The new route is obtained by solving a shortest path problem with time windows using the algorithm of Desrochers and Soumis (1988). The time windows on the nodes and edges of the network are determined by the occupation intervals for the fixed route. The best solution, i.e., the one with the smallest makespan, is kept as a new partial plan. So the method is based on an enumeration of the sets of transportation plans by means of dynamic programming.

3.2. The dynamic programming scheme

The states correspond to the partial transportation plans. A state S_i is defined by

$$S_i = (\phi_i^1, R_i^1, \phi_i^2, R_i^2),$$

where ϕ_i^1 (ϕ_i^2) is the *ordered* set of tasks accomplished by vehicle 1 (2) and R_i^1 (R_i^2) is its associated route. Let $Z(S_i)$ be the objective function value associated with the state. In our case, the objective function value is the completion time of the transportation operations. We define $C_1(\phi_i^1, R_i^1)$ as the completion time of the route R_i^1 for vehicle 1 accomplishing tasks ϕ_i^1. We define $C_2(\phi_i^2, R_i^2)$ analogously. Then, for a given state S_i,

$$Z(S_i) = \max\{C_1(\phi_i^1, R_i^1), C_2(\phi_i^2, R_i^2)\}. \tag{1}$$

The transition from a given state to an immediate successor state is done by adding one of the yet unscheduled requests to one of the two routes, following the method described in subsection 3.1. If S_j is a successor state of S_i obtained by adding to route 1 a task t_k with an origin node o_k and a destination node d_k, then:

$$S_j = (\phi_j^1, R_j^1, \phi_j^2, R_j^2)$$

with

$$\phi_j^1 = \phi_i^1 \cup t_k \text{ (t_k added at the end)}$$

$$R_j^1 = R_i^1 \cup \{o_k, d_k\} \text{ (added at the end)}$$

$$\phi_j^2 = \phi_i^2$$

$$R_j^2 = R_i^2$$

The value of this new state $Z(S_j)$ is calculated using (1). Adding a task to route 2 is analogous. The method starts from an initial state S_0, with a starting position for both vehicles and no request assigned to either one. We have

$$S_0 = (\phi_1^o, R_1^o, \phi_2^o, R_2^o), \text{ and } Z(S_0) = 0$$

with

$$\phi_1^o = \text{empty set}, R_1^o = \{\text{starting point of vehicle 1}\}$$

$$\phi_2^o = \text{empty set}, R_2^o = \{\text{starting point of vehicle 2}\}.$$

The successive states are then generated until all final states representing complete transportation plans are produced. The set of the generated states constitutes an enumeration tree. The order in which successive states are generated corresponds to a best first search: the state with the smallest value of Z is chosen among the already generated states for which their successors have not been generated. All of the first successors of this state are generated

and added to an *active* list. The final state with the best Z value constitutes the optimal solution to the scheduling and routing problem. It describes a conflict-free route and schedule for both vehicles.

The algorithm

The algorithm uses two lists of states: an *active list* and a *final state list*.

Step 1: Place initial state S_0 on the active list.
Step 2: Repeat until the active list is empty:
- select the S_i in the active list with the smallest $Z(S_i)$;
- for each task not yet assigned satisfying the precedence relations, generate two immediate succesor states of S_i by assigning the task to each vehicle;
- place each successor state on the active list or on a final state list if it is a final state;
- remove state S_i from the active list.

Step 3: Select the best final state.

Dominance tests

This algorithm generates an astronomical number of states. To make it usable, a procedure has been designed to eliminate some states (and their numerous successors) by way of dominance tests. A state S_1 is said to *dominate* another state S_2 if it can be shown that any successor of S_2 can be associated with a successor of S_1 which is equivalent or better. The dominated state S_2 can then be discarded.

Three types of dominance tests have been implemented. In each test, two states are compared.

1. *Redundancy*: This test allows the elimination of states that are identical: if two states have the same set of tasks, the same finishing time for both vehicles, and if the finishing position for each vehicle is the same in the two states, then the two states are considered identical, and the successors of only one of them have to be generated.
2. *Dominance 1*: Let S_a and S_b be two states. If S_a contains all of the tasks of S_b, i.e.,

$$\phi_1^a \cup \phi_2^a \supseteq \phi_1^b \cup \phi_2^b,$$

and if it is possible for vehicle k ($k = 1, 2$) to travel from the finishing positions of S_a to the finishing positions of S_b and arrive there before $C_k(\phi_k^b, R_k^b)$ ($k = 1, 2$), then S_a dominates S_b, and consequently S_b can be eliminated.
3. *Dominance 2*: Let M be an upper bound on the time needed to optimally travel from any origin to any destination (taking into account possible detours to avoid conflicts), and let S_a be a state and $\bar{\phi}_a$ be the set of tasks still to be assigned. If for each task in $\bar{\phi}_a$ the earliest pickup time is greater than $Z(S_a) + M$, then each state S_b, such that $\phi_1^a \cup \phi_2^a \supseteq \phi_1^b \cup \phi_2^b$ and $Z(S_b) > Z(S_a)$, is dominated by S_a. This test is similar to the dominance 1 test, except that the time to travel from the finishing position of S_a to the finishing position of S_b is not calculated, since no task can begin before M time, and hence both vehicles can travel to any point on time.

Real-time operation

The algorithm described produces the dispatching and routing of vehicles to accomplish a given set of tasks. To obtain a workable control strategy, this problem is resolved on a certain time horizon for a certain set of tasks. It is resolved anew when one of three events occurs:

- a breakdown on the track (modification of underlying network);
- a task is modified (added or deleted);
- a new plan is required because the planned time horizon has expired.

A new time horizon is thus considered, and a new transportation schedule (due dates) is elaborated. This way of considering more and more tasks defines what is usually called a *rolling* time horizon. The length of the horizon will depend on the specific characteristics of a given manufacturing system. Of course, as will be seen in the next section, the shorter the time horizon is, the more efficient the procedure is.

4. Experimentation

The FMS which we model in this experiment consists of 12 machines of which seven are production machines, while the five others are service stations, such as washing, preparation, etc. The FMS mills, turns, grinds, pressworks, and has foundry, inspection, and assembly operations. The layout of the FMS is shown on figure 1. The material handling system is an AGV system. The guidepath is schematized by a network of 30 nodes and 44 bidirectional edges. Thirteen nodes correspond to the input/output positions of the machines. The two AGVs travel at a constant speed of 0.5 meters per second, and the loading and unloading times are 20 seconds each. To validate our dynamic programming method, we used a six-part-type production release (Drolet, 1991). With the process sheet of each order (provided in table 1), production was planned, and the transportation schedules derived using the job-oriented heuristic developed by Hastings and Yeh (1990). The aim of this heuristic is to complete a set of jobs as soon as possible, but without scheduling the supporting jobs (e.g., tools and fixtures preparation) unnecessarily early. All the jobs are first scheduled forward. The scheduled start times of the final job are thus obtained. The supporting jobs are then rescheduled backward from that time. The result is a schedule in which the jobs are completed as soon as possible, but with no waiting time for the supporting jobs.

Results

For the tests, two typical eight-hour production days have been divided into 16 one-hour blocks. A dispatching and scheduling plan has been elaborated for each block. The corresponding set of transportation requests was quite demanding for the AGV system, especially at the beginning and end of the production days. The tests were conducted on a Sun Sparc Station (Sun 4) running at 10 mips. The number of tasks in a given block varies from eight

Table 1. Process sheets.

Order 1: 50 Parts	*Order 2*: 100 Parts	*Order 3*: 15 Parts	*Order 4*: 20 Parts	*Order 5*: 30 Parts	*Order 6*: 10 Parts
Parts:	*Parts*:	*Parts*:	*Parts*:	*Parts*:	*Parts*:
Receiving: 20 min (5)*	Receiving: 20 min (8)	Receiving: 20 min (3)	Receiving: 20 min (4)	Receiving: 20 min (3)	Receiving: 20 min (3)
Foundry: 878 min (13)	Palletizing: 40 min (8)	Palletizing: 24 min (8)	Palletizing: 30 min (10)	Palletizing: 30 min (10)	Foundry: 673 min (10)
Grinding: 65 min (13)	Pressworking: 48 min (8)	Preparation: 120 min (8)	Preparation: 15 min (10)	Preparation: 150 min (10)	Griding: 50 min (10)
Preparation: 195 min (13)	Assembly: 48 min (8)	Milling: 226 min (8)	Milling: 100 min (10)	Turning: 150 min (10)	Preparation: 100 min (10)
Turning: 250 min (13)	Palletizing: 20 min (4)	Washing: 40 min (8)	Washing: 50 min (10)	Washing: 50 min (10)	Milling: 400 min (10)
Washing: 65 min (13)	Shipping: 20 min	Inspection: 151 min (8)	Inspection: 50 min (10)	Inspection: 90 min (10)	Washing: 50 min (10)
Inspection: 375 min (13)	*Supplies*:	Palletizing: 15 min (8)	Milling: 300 min (10)	Milling: 110 min (10)	Inspection: 150 min (10)
Palletizing: 20 min (4)	Receiving: 20 min (1)	Shipping: 20 min	Washing: 50 min (10)	Grinding: 50 min (10)	Palletizing: 20 min (10)
Shipping: 20 min	Assembly: 48 min (1)	*Milling tools*:	Inspection: 200 min (10)	Washing: 50 min (10)	Shipping: 20 min
Turning tools:	Shipping	Preparation: 80 min (1)	Palletizing: 20 min (4)	Inspection: 90 min (10)	*Milling tools*:
Preparation: 60 min (1)		Milling: 226 min (1)	Shipping: 20 min	Assembly: 40 min (10)	Preparation: 90 min (1)
Turning: 250 min (1)		Preparation: 12 min	*Milling tools*:	Palletizing: 30 min (6)	Milling: 400 min (1)
Preparation: 10 min		*Milling fixture*:	Preparation: 105 min (1)	Shipping: 20 min	Preparation: 12 min
Inspection fixture:		Preparation: 30 min (1)	Milling: 400 min (1)	*Turning tools*:	*Milling fixture*:
Preparation: 30 min (1)		Milling: 226 min (1)	Preparation: 15 min	Preparation: 30 min (1)	Preparation: 30 min (1)
Inspection: 375 min (1)		Preparation: 10 min	*Milling fixtures*:	Turning: 150 min (1)	Milling: 400 min (1)
Preparation: 10 min		*Inspection fixture*:	Preparation: 30 min (1)	Preparation: 5 min	Preparation: 10 min
		Preparation: 30 min (1)	Milling: 100 min (1)	*Milling tools*:	*Inspection fixture*:
		Inspection: 151 min (1)	Preparation: 10 min	Preparation: 15 min (1)	Preparation: 30 min (1)
		Preparation: 10 min	Preparation: 30 min (1)	Milling: 110 min (1)	Inspection: 150 min (1)
			Milling: 300 min (1)	Preparation: 3 min	Preparation: 10 min
			Preparation: 10 min	*Milling fixture*:	
			Inspection fixtures:	Preparation: 30 min (1)	
			Preparation: 30 min (1)	Milling: 110 min (1)	
			Inspection: 50 min (1)	Preparation: 10 min	
			Preparation: 10 min	*Inspection fixture*:	
			Preparation: 30 min (1)	Preparation: 30 min (1)	
			Inspection: 200 min (1)	Inspection: 90 min (1)	
			Preparation: 15 min	Preparation: 10 min	
				Supplies:	
				Receiving: 20 min (1)	
				Assembly: 40 min (1)	
				Shipping	

*Number of transports required.

to 61 and is typically between 20 and 30. The number of states that were generated varies considerably (from ten to many thousands). The conflict testing routine is called two or three times per generated state on average, and a conflict is resolved in about half the cases.

The results shown in table 2 illustrate the efficiency of the dominance tests. For each of the 16 one-hour blocks (test 1.0, 1.1, ..., 1.7, 2.0, 2.1, ..., 2.7), three runs of the program have been done; in each run, the dominance 2 test was applied. In the first run, the redundancy test was added; in the second run, the dominance 1 test was added; in the third run, the dominance 2 *and* redundancy tests were both applied. For each block, three lines of results are presented in table 2, corresponding to the three runs. The columns correspond to the solution time in seconds, the total number of states generated, the maximum number of states at a time, and then, for each test, the number of times which the test was called and the number of states that were eliminated by the test.

It can be seen that the dominance 1 test is much more efficient than the redundancy test. Without the dominance 1 test (and with the redundancy test), the solution times vary up to almost 100 seconds. Using the dominance 1 test dramatically reduces the number of generated states and consequently the solution times. With this test, the hardest problem is solved in 3.27 seconds. Also, dominance 1 with redundancy is almost equivalent to dominance 1 only. The dominance 2 test is independent of the two other tests and does not affect their efficiency. The horizon taken is one hour with the transportation times under a minute. Reducing the time horizon (by half or more) would have a striking effect on the solution times, given the exponential nature of the process. This would allow the solution of every problem to be under a second.

As the running times were extremely quick, we then tested the method on more difficult problems. To do so, we compressed the earliest times of the requests by a factor of two and of three, that is, we divided the earliest times by two or three (see table 3).

The compressions increase the number of requests per time unit. The results of the compressions are presented in table 4. It can be seen that with the compression (i.e., with more challenging problems), the method is able to handle most of the instances. However, one can see that, in some instances, the number of states and the solution times explode, and as the program was limited to a maximum of 10,000 states, some instances were not solved. With compression 2, four instances out of 16 were not solved (due to the number of states exceeding the maximum allowed). With compression 3, 2 more instances were not solved.

In order to assess the difficulty of the problems, in table 5, we present the vehicle utilization. The effect of the compressions is to multiply the utilization by a factor approximately equal to the factor of compression.

To summarize the results of the test, we see that the method was validated on actual examples. The method is, however, sensitive to the number of requests to be scheduled. We planned on time horizons of one hour, 30 minutes, and 20 minutes, whereas the transportation times were under a minute. In a real-world implementation, shortening the time horizon to ten or 15 minutes could be extremely efficient. Whenever an event having a major impact on production occurs (having some of the machines starving, for instance), or lateness accumulates sufficiently to impede the normal course of the production plan or the transportation missions, a new production plan is easily obtained (via the Hasting and Yeh heuristic), and a new transportation schedule is derived within a few seconds. It should be noted that potential blocking of the AGVs is prevented by the way the dynamic programming algorithm generates the routing of each vehicle.

Table 2. Dominance tests.

Test No.	Time (cpu)	Number of States	Maximum Number at a Time	Redundancy		Dominance 2		Dominance 1	
				Calls	Elim.	Calls	Elim.	Calls	Elim.
1.0	70.02	9001	2888	2824	3543	11	8	—	—
	1.34	626	62	—	—	11	7	261	337
	1.30	621	61	259	56	11	7	259	278
1.1	30.75	5984	2108	2565	877	11	2589	—	—
	1.81	1011	106	—	—	11	115	448	424
	1.92	970	97	423	53	11	113	423	358
1.2	0.05	28	3	14	0	11	3	—	—
	0.05	28	3	—	—	11	3	14	0
	0.03	28	3	14	0	11	3	14	0
1.3	0.01	8	1	2	0	6	0	—	—
	0.01	8	1	—	—	6	0	2	0
	0.00	8	1	2	0	6	0	2	0
1.4	9.94	2535	1021	997	498	9	1036	—	—
	1.08	551	105	—	—	9	56	230	256
	1.13	551	102	230	40	9	51	230	221
1.5	1.31	822	231	372	221	7	227	—	—
	0.46	304	65	—	—	7	29	139	129
	0.48	302	63	138	42	7	28	138	87
1.6	0.01	8	1	2	0	6	0	—	—
	0.01	8	1	—	—	6	0	2	0
	0.00	8	1	2	0	6	0	2	0
1.7	97.17	9000	3622	3226	2226	14	4	—	—
	3.27	977	184	—	—	17	69	409	482
	3.61	988	178	414	101	17	57	414	399
2.0	0.60	577	182	260	69	12	231	—	—
	0.22	197	35	—	—	12	39	94	49
	0.23	197	35	94	0	12	39	94	49
2.1	10.04	4000	645	2406	686	16	49	—	—
	0.41	345	37	—	—	16	16	162	134
	0.41	345	37	163	13	16	16	163	121
2.2	0.04	38	4	19	0	14	5	—	—
	0.05	38	4	—	—	14	5	19	0
	0.05	38	4	19	0	14	5	19	0
2.3	0.01	12	1	2	0	10	0	—	—
	0.01	12	1	—	—	10	0	2	0
	0.00	12	1	2	0	10	0	2	0

Table 2. Continued.

Test No.	Time (cpu)	Number of States	Maximum Number at a Time	Redundancy		Dominance 2		Dominance 1	
				Calls	Elim.	Calls	Elim.	Calls	Elim.
2.4	0.01	12	1	2	0	10	0	—	—
	0.01	12	1	—	—	10	0	2	0
	0.01	12	1	2	0	10	0	2	0
2.5	0.02	22	3	8	1	11	2	—	—
	0.02	22	3	—	—	11	2	8	1
	0.03	22	3	8	1	11	2	8	0
2.6	0.01	12	1	3	0	9	0	—	—
	0.01	12	1	—	—	9	0	3	0
	0.01	12	1	3	0	9	0	3	0
2.7	0.01	17	2	7	0	9	1	—	—
	0.02	17	2	—	—	9	1	7	0
	0.01	17	2	7	0	9	1	7	0

Table 3. Compression by a factor of 3.

	Earliest Times	
Request	Uncompressed	Compressed
# 1	4.5	1.5
# 2	12.0	4.0
# 3	19.5	6.5

5. Extensions

In this section, we explore some extensions of our method. First, we consider the use of lateness measures in the comparison and selection of partial transportation plans. Then, in section 5.2, we devise a heuristic based on our method in order to handle more difficult problems, i.e., problems with more than two vehicles or with too many tasks for the optimal approach.

5.1. Lateness

A transportation task is said to be *late* if its assigned vehicle does not reach the origin of the transportation request by the earliest pickup time. This concept of lateness could be generalized to account for more sophisticated time windows associated with requests and would then encompass a scheduling problem class wider than our *earliest pickup time* scheme. For the complete transportation plans generated, we tabulate different lateness

Table 4. Effects of time compression.

Test No.	Compression Factor	Time	Number of States	Maximum Number of States at a Time	Test No.	Compression Factor	Time	Number of States	Maximum Number of States at a Time
1.0	1	1.34	626	62	2.0	1	0.22	197	35
	2	186.24	* 9002	3426		2	174.23	* 9002	4145
	3	206.43	* 9011	4610		3	57.80	* 9002	1634
1.1	1	1.81	1011	106	2.1	1	0.41	345	37
	2	81.82	* 9002	1701		2	22.12	4089	554
	3	79.49	* 9008	2360		3	59.80	* 9002	1901
1.2	1	0.05	28	3	2.2	1	0.05	38	4
	2	0.24	133	25		2	0.16	149	21
	3	0.53	330	41		3	27.78	* 9002	513
1.3	1	0.01	8	1	2.3	1	0.01	12	1
	2	0.01	10	2		2	0.01	14	2
	3	0.01	8	1		3	0.01	14	2
1.4	1	1.08	551	105	2.4	1	0.01	12	1
	2	28.58	5962	753		2	0.01	14	2
	3	100.73	* 9002	2337		3	0.01	14	2
1.5	1	0.46	304	65	2.5	1	0.02	22	3
	2	1.89	1059	158		2	0.07	47	11
	3	9.43	3537	317		3	0.08	61	11
1.6	1	0.01	8	1	2.6	1	0.01	12	1
	2	0.01	10	2		2	0.01	14	2
	3	0.01	9	2		3	0.02	18	4
1.7	1	3.27	977	184	2.7	1	0.02	17	2
	2	142.46	* 9000	2294		2	0.02	24	3
	3	91.87	* 9001	2471		3	0.03	35	6

*Before stopping.

statistics, such as total lateness for all of the transportation tasks in the plan, the number of late tasks, and the maximum lateness of any single task in a transportation plan (see table 6). As some of the generated transportation plans produce lateness, it seems that a natural extension can account for this lateness by using some measure in the comparison and selection of partial transportation plans. Hence, we use two approaches to incorporate these ideas into the algorithm.

The first approach consists of adding "hard" constraints to limit admissible states to a fixed upper bound of a lateness measure, either limiting the total accumulated lateness for a plan or limiting the maximum lateness of any single request in a plan. The second approach integrates a measure of lateness into the objective function:

$$Z' = Z^{\text{makespan}} + K \times \text{lateness},$$

where K typically has a large value ($K \gg 1$).

Table 5. Vehicle utilization.

	Compression					
	1		2		3	
Test	Vehicle 1	Vehicle 2	Vehicle 1	Vehicle 2	Vehicle 1	Vehicle 2
1.0	31.56	30.83				
1.1	60.95	58.08				
1.2	24.30	24.03	45.31	45.48	64.92	65.42
1.3	7.13	7.66	14.07	13.41	24.22	19.02
1.4	25.32	22.26	47.82	43.81		
1.5	19.08	15.88	32.50	38.72	50.74	51.93
1.6	8.06	8.54	15.87	14.94	25.35	21.17
1.7	32.22	31.73				
2.0	37.44	38.63				
2.1	41.91	42.73	75.65	79.92		
2.2	30.97	26.32	57.19	51.17		
2.3	11.82	12.16	24.89	22.26	35.33	32.81
2.4	11.82	12.16	24.89	22.26	35.33	32.81
2.5	16.15	15.64	29.91	33.01	50.92	43.92
2.6	14.68	14.29	30.81	27.49	44.27	37.81
2.7	19.26	11.96	37.53	21.56	47.06	38.11

The two approaches have a significant impact on the lateness characteristics of the solution. In most instances, both the total amount of lateness and the maximum lateness are substantially reduced. The first approach is also very effective computationally as it limits the number of admissible states to examine. However, for a given upper limit on the lateness measure, one cannot guarantee a priori the existence of a feasible solution. The second approach increases the computational complexity in time and number of generated states. With these tests, we conclude that, while adding to the complexity of the solution process, the second approach performs very well, and is a viable extension to our approach that could significantly improve the quality of the generated solution.

5.2. Heuristics based on the dynamic programming scheme

We now present a heuristic generalization of our method for cases where the number of states explode despite the elimination of states by the dominance tests described in section 3.2. The idea is to limit the number of states that are generated. Optimality is lost if all possible states are not explored. However, a clever exploration of the tree can lead to a very good, if not optimal, solution.

A first way to limit the number of states is to consider, in generating the successor states of a given state, only the tasks in a certain time interval from the finishing time of the state ($Z(S_i)$). The tasks are ordered according to time and only the first "n" tasks are generated for the successor states. By not considering the later tasks, there is little chance of losing optimality. By reducing the number "n" of considered tasks, the decrease in solution time can be important. This would allow the consideration of many AGVs.

Table 6. Lateness statistics.

Test No.	Compression Factor	Number of Late Tasks	Total Lateness (second)	Maximum Lateness (second)	Test No.	Compression Factor	Number of Late Tasks	Total Lateness (second)	Maximum Lateness (second)
1.0	1	7	397.00	158.24	2.0	1	13	379.92	92.14
	2					2			
	3					3			
1.1	1	28	1024.80	141.06	2.1	1	14	597.52	142.76
	2					2	25	1299.32	166.88
	3					3			
1.2	1	3	37.34	36.52	2.2	1	4	53.62	27.96
	2	5	234.40	70.58		2	8	197.28	42.96
	3	11	399.08	120.92		3			
1.3	1	0	0.00	0.00	2.3	1	0	0.00	0.00
	2	0	0.00	0.00		2	0	0.00	0.00
	3	0	0.00	0.00		3	0	0.00	0.00
1.4	1	8	375.96	139.80	2.4	1	0	0.00	0.00
	2	13	814.50	160.50		2	0	0.00	0.00
	3					3	0	0.00	0.00
1.5	1	7	317.90	103.06	2.5	1	1	29.62	29.62
	2	8	496.48	158.30		2	1	62.04	62.04
	3	9	727.62	168.30		3	4	247.68	138.58
1.6	1	0	0.00	0.00	2.6	1	0	0.00	0.00
	2	0	0.00	0.00		2	0	0.00	0.00
	3	0	0.00	0.00		3	1	4.26	4.26
1.7	1	8	534.60	236.50	2.7	1	1	47.92	47.92
	2					2	3	97.14	47.92
	3					3	3	173.86	105.68

Another way of limiting the number of states is to assign a given new task to only one vehicle (the best one according to our time measure). Hence, for each task, only one successor of a state is generated instead of two.

These two ways of limiting the number of states would permit an efficient use of the method when, in a real-time setting, the solution needs to be obtained quickly and when many incidents require frequent reoptimization.

6. Conclusions

This article presents a dynamic programming-based method of generating an optimal transportation plan, consisting of the dispatching, routing, and scheduling of two AGVs in a conflict-free manner so as to meet a production plan. The method is well-suited for a production plan having lots of variations over time. The algorithm was implemented and tested on realistic data provided by the Industrial Engineering Department of École Polytechnique

de Montréal. The corresponding set of transportation requests was quite demanding for the AGV system, especially at the beginning and end of the production days.

Our method produces, within acceptable time for real-time operations, conflict-free dispatching and routing so as to minimize the makespan. Additional tests were conducted to incorporate a measure of lateness into the objective, and again the method proved to be very efficient. Moreover, by considering concurrently the dispatching, routing, and scheduling, the method produces high-quality solutions.

We have also indicated how the method can be used efficiently in a heuristic manner to handle the dispatching, routing, and scheduling of more than two vehicles. We presented ways to limit the number of states that are generated.

Acknowledgments

This work was supported by the Natural Sciences and Engineering Research Council of Canada and the FCAR Program of the Québec Government. We would like to thank Ms. Michèle Drolet for her contribution to the production planning phase and Professor K.E. Stecke for her valuable suggestions. We also thank the associate editor and the anonymous referees for their comments which led to an improved version of the article.

References

Blazewicz, J., Eiselt, H.A., Finke, G., Laporte, G., and Weglarz, J., "Scheduling Tasks and Vehicles in a Flexible Manufacturing System," *International Journal of Flexible Manufacturing Systems*, Vol. 4, No. 1, pp. 5–16 (1991).

Co, C.G. and Tanchoco, J.M.A., "A Review of Research on AGVS Vehicle Management," *Engineering Costs and Production Economics*, Vol. 21, pp. 35–42 (1991).

Daniels, S.C., "Real Time Conflict Resolution in Automated Guided Vehicle Scheduling," Ph.D. dissertation, Pennsylvania State University, Department of Industrial Engineering (May 1991).

Desrochers, M. and Soumis, F., "A Generalized Permanent Labelling Algorithm for the Shortest Path with Time Windows," *INFOR*, Vol. 26, pp. 193–214 (1988).

Drolet, M., "Ordonnancement d'un carnet de commandes pour un atelier flexible de sous-traitance," projet de fin d'études, département de génie industriel, École Polytechnique de Montréal (April 1991).

Egbelu, P.J. and Tanchoco, J.M.A., "Characterization of Automatic Guided Vehicle Dispatching Rules," *International Journal of Production Research*, Vol. 22, No. 3, pp. 359–374 (March 1984).

Fujii, S., Sandoh, H., and Hohzaki, R., "Routing Control of Automated Guided Vehicles in FMS," in *Proceedings of the USA-Japan Symposium on Flexible Automation*, Minneapolis, MN, pp. 629–636 (July 1988).

Hasting, N.A.J. and Yeh, C.-H., "Job Oriented Production Scheduling," *European Journal of Operational Research*, Vol. 47, pp. 35–48 (1990).

Kim, Chang W. and Tanchoco, J.M.A., "Conflict-Free Shortest-Time Bidirectional AGV Routing," *International Journal of Production Research*, Vol. 29, No. 12, pp. 2377–2391 (December 1991).

Krishnamurthy, N.N., Batta, R., and Karwan, M.H., "Developing Conflict-Free Routes for Automated Guided Vehicles," *Operations Research*, Vol. 41, No. 6, pp. 1177–1190 (November/December 1993).

Kusiak, A. and Cyrus, P., "Routing and Scheduling of Automated Guided Vehicles," in *Proceedings of the 8th International Conference on Production Research and, 5th Conference of the Fraunhofer-Institute for Industrial Engineering*, Springer Verlag, pp. 247–251 (August 1985).

Maxwell, W.L. and Muckstadt, J.A., "Design of Automatic Guided Vehicle Systems," *IIE Transactions*, Vol. 14, No. 2, pp. 114–124 (June 1982).

Palekar, U.S., Kapoor, S.G., and Huang, J., *A Labelling Algorithm for the Navigation of Automated Guided Vehicles*, Working paper, Department of Mechanical and Industrial Engineering, University of Illinois at Urbana-Champaign (June 1990).

Sabuncuoglu, I. and Hommertzheim, D.L., "Dynamic Dispatching Algorithm for Scheduling Machines and Automated Guided Vehicles in a Flexible Manufacturing System," *International Journal of Production Research*, Vol. 30, No. 5, pp. 1059–1079 (May 1992).

Sabuncuoglu, I. and Hommertzheim, D.L., "Experimental Investigation of an FMS Due-Date Scheduling Problem: Evaluation of Machine and AGV Scheduling Rules," *International Journal of Flexible Manufacturing Systems*, Vol. 5, No. 4, pp. 301–323 (1993).

Taghaboni, F. and Tanchoco, J.M.A., "A LISP-Based Controller for Free-Ranging Automated Guided Vehicle Systems," *International Journal of Production Research*, Vol. 26, No. 2, pp. 173–188 (February 1988).