Available online at www.sciencedirect.com
ScienceDirect
Computers & Operations Research 34 (2007) 1688–1707
computers & operations research
www.elsevier.com/locate/cor

Scheduling and routing of automated guided vehicles: A hybrid approach

Ayoub Insa Corréa, André Langevin, Louis-Martin Rousseau

Département de mathématiques et de génie industriel, Ecole Polytechnique de Montréal, C.P. 6079, Succ. Centre-ville, Montréal (Québec), Canada H3C 3A7

Available online 3 October 2005

Abstract

We propose a hybrid method designed to solve a problem of dispatching and conflict free routing of automated guided vehicles (AGVs) in a flexible manufacturing system (FMS). This problem consists in the simultaneous assignment, scheduling and conflict free routing of the vehicles. Our approach consists in a decomposition method where the master problem (scheduling) is modelled with constraint programming and the subproblem (conflict free routing) with mixed integer programming. Logic cuts are generated by the sub problems and used in the master problem to prune optimal scheduling solutions whose routing plan exhibits conflicts. The hybrid method presented herein allowed to solve instances with up to six AGVs.

Keywords: Constraint programming; Mathematical programming; Hybrid model; Logical Benders decomposition; Material handling system; Automated guided vehicles; Vehicle routing and scheduling

1. Introduction

This study focuses on the simultaneous scheduling and routing of automated guided vehicles (AGVs) in a flexible manufacturing system (FMS). An AGV is a material handling equipment that travels on a network of guide paths. The FMS is composed of various cells, also called working stations, each with a specific function such as milling, washing, or assembly. Each cell is connected to the guide path network by a pick-up/delivery (P/D) station where pallets are transferred from/to the AGVs. Pallets of products are moved between the cells by the AGVs. The guide path is composed of aisle segments on which the vehicles are assumed to travel at a constant speed. The vehicles can travel forward or backward. As many vehicles travel on the guide path simultaneously, collisions must be avoided. AGV systems are

0305-0548/$ - see front matter © 2005 Published by Elsevier Ltd.
doi:10.1016/j.cor.2005.07.004

implemented in various industrial contexts: container terminals, part transportation in heavy industry, flexible manufacturing systems. For a general review on AGV problems, the reader is referred to [1–3]. For a recent review on AGVs scheduling and routing problems and issues, the reader is referred to the survey of Qiu et al. [4]. These authors identified three types of algorithms for AGVs problems: (1) for general path topology, (2) for path optimization and (3) for specific path topologies. Methods of the first type can be divided in three categories: (1a) static methods, where an entire path remains occupied until a vehicle completes its route; (1b) time-window based methods, where a path segment may be used by different vehicles during different time-windows; and (1c) dynamic methods, where the utilization of any segment of path is dynamically determined during routing rather than before as with categories (1a) and (1b). The method presented in this article belongs to the third category (1c) and addresses the conflict free routing problem with an optimization approach. The plan of the article is as follows. Section 2 presents a description of the problem. Section 3 reviews the relevant works. Section 4 presents the hybrid constraint programming (CP)/mixed integer programming (MIP) approach. Section 5 describes in detail the experimentation. The conclusion follows.

2. Problem description

Every day, a list of orders is given, each order corresponding to a specific product to manufacture (here, product means one or many units of the same product). The product units are carried on pallets by the AGVs and the unit load is one pallet. Each order determines a sequence of operations on the various cells (machines) of the FMS. Fig. 1 presents the FMS with the AGV guide path used in the experimentation. The production scheduling, i.e., setting the earliest starting time of each machine operation for each pallet of each order, is done a priori using the P.E.R.T. method. Hence, each material handling request is composed of the pick-up and the delivery of a specific pallet with the corresponding earliest times. The guide path network is bi-directional. The vehicles can stop only at the ends (intersection nodes) of the guide path segments. There are two types of possible collisions: the first type may appear when two vehicles are moving toward the same node. The second type of collision occurs when two vehicles are traveling head-to-head on a segment. A production delay is incurred when a load is delivered after its planned earliest time. The problem is thus defined as follows:

Given a number of AGVs (and their starting positions) and a set of transportation requests, find the assignment of the requests to the vehicles and conflict free routes for the vehicles in order to minimize the sum of the production delays.

A solution of the problem determines:

- The number of AGVs required to perform the set of tasks.
- The assignment of requests to the AGVs.
- The position of each AGV at each period in the FMS.
- The schedule of each pick-up or delivery task.
- The revised schedule for each machine given the transportation schedule.

Our problem may be seen as a vehicle routing problem with time windows (VRPTW) in which the objective is to minimize the sum of deviations from the lower bound value of the time windows of the

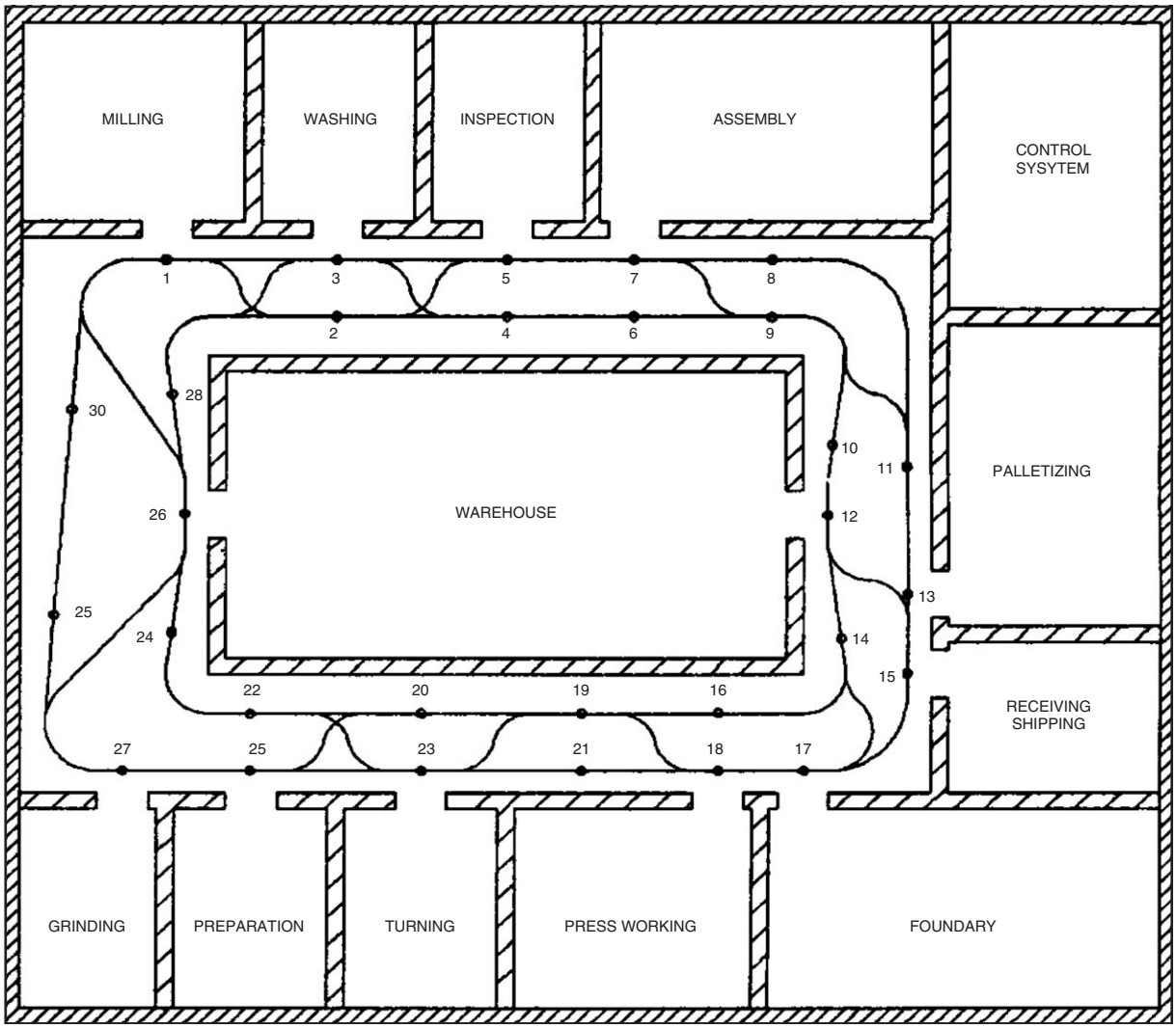

Fig. 1. The FMS layout with the AGV guide path.

delivery tasks subject to the following constraints:

- For each pick-up or delivery task, the lower bound of its associate time window is equal to its earliest processing time while the upper bound is the length of the horizon.
- There exist anti-collision constraints on nodes and arcs.
- There exist two types of precedence constraints between some specified pick-up and delivery tasks. The first type corresponds to a pick-up task that must immediately precede a delivery task on the same node to satisfy the production sequence. The second type corresponds to a delivery that must precede a pick-up on the same node. A pallet has to be delivered and processed at a work station before being available for a pick-up.

- All nodes may be visited several times by the AGVs either to perform a task or for a simple traversal.

3. Literature review

This section is divided in three parts. In the first part, a review on AGVs scheduling and routing is presented. In the second part, a brief description of the CP paradigm is intended for the OR researchers not well acquainted with CP. The last part reviews the CP/MIP hybrid approaches.

3.1. AGVs scheduling and routing

A number of authors have addressed the conflict free routing problem with a static transportation requests set, i.e., with all requests known a priori. Lee et al. [5] present a two-staged traffic control scheme to solve a conflict free routing problem. Their heuristic method consists of generating off-line k-shortest paths in the first stage before the on-line traffic controller picks a conflict free shortest path whenever a dispatch command for an AGV is issued (second stage). Rajotia et al. [6] propose a semi-dynamic time window constrained routing strategy. They use the notions of reserved and free time windows to manage the motion of vehicles. Krishnamurthy et al. [7] propose an optimization approach. Their objective is to minimize the makespan. They assume that the assignment of tasks to AGVs is given and they solve the routing problem by column generation. Their method generates very good solutions in spite of the fact that it is not optimal (column generation is performed at the root node of the search tree only). Oboth et al. [8] present a heuristic method to solve the dispatching and routing problems but not simultaneously. Scheduling is performed first and a sequential path generation heuristic (SPG) is used to generate conflict free routes. The SPG is inspired from Krishnamurthy et al. [7] static version of the AGV routing problem and applied to a dynamic environment while relaxing some of the limiting assumptions like equal and constant speeds of AGVs. When conflict is encountered, no feed back is sent to the scheduling module. The AGV being routed has to be delayed if an alternate route cannot be generated. The authors use rules for positioning idle AGVs instead of letting the system manage them. Langevin et al. [9] propose a dynamic programming based method to solve exactly instances with two vehicles. They solve the combined problem of dispatching and conflict free routing. Desaulniers et al. [10] propose an exact method that enables to solve instances with up to four vehicles. Their approach combines a greedy search heuristic (to find a feasible solution and set bound on delays), column generation and a branch and cut procedure. Their method presents however some limits since its efficiency depends highly on the performance of the starting heuristic. If no feasible solution is found by the search heuristic, then no optimal solution can be found. The search heuristic performs poorly when the level of congestion increases.

3.2. The CP paradigm

Constraint programming, which has been very successful in solving hard combinatorial problems in the field of scheduling, planning, and transportation [11–13] the subject of several textbooks [14–17]. Traditionally a constraint programming model is composed of a set of variables (X), a set of domains (D), and a set of constraints (C) specifying which assignments of values in D to variable X are legal. The

efficiency of the constraint programming paradigm lies in powerful constraint propagation algorithms which remove from the domain of the variables the values which will generate infeasible solutions. If constraint propagation is not sufficient to find a feasible solution then a branching process is performed to further narrow the domains; a feasible solution is found when each domain contains only one value. The branching process is necessary in most difficult combinatorial problems. Typically, at each node of the search tree the following steps are taken: first a variable not yet fixed is selected and a remaining value of its domain is assigned to it. Then constraint propagation occurs. If during propagation the domain of a variable is emptied then the solver has detected an inconsistency in the previously taken decisions and the whole search process backtracks, typically by choosing another value for the variable. When constraint propagation terminates while there are still some unfixed variable, then the solver creates a new search node and goes on with the procedure just detailed. The branching strategy is thus defined by *variable* and *value* selection policies. This is the most simple and frequently used branching strategy but more intricate policies are often used. For instance one could split the domain of a variable into to two sets of similar size or even use two opposing constraints to define two branching directions. It is fairly simple to extend this method to solve combinatorial optimization problems, that is to identify the feasible solution which minimizes (or maximizes) a given objective function. Once a feasible solution as been identified, the set C is extended to contain a new constraint specifying that future feasible solutions should have a strictly better cost than the cost of the solution just identified. The solver thus keeps searching for better solutions until it can prove that the last found is optimal.

3.3. CP/MIP hybrid approaches

Hooker and Ottosson [18] and Milano [19] present comprehensive reviews on hybrid methods. We focus here on the more relevant works. Hooker [20] gives insights on how CP can be integrated to Benders decomposition. A number of researchers have integrated CP in the classical Benders decomposition for MIP (the master problem or the sub problem is formulated in CP and logic – *no goods* – cuts are used). Benoist et al. [21] formulate their master problem as a CP model. Their master problem is reduced to a global constraint whereas the sub problems use linear duality. The global constraint uses the network structure of the original problem and consists of two types of equations defining the feasible flow problem. This method has been efficiently applied to a workforce scheduling problem in a telecommunications company call center. Eremin and Wallace [22] also present a hybrid decomposition method in which the master problem is solved with CP. The major interest of their approach is that the user only needs to specify the variables of each sub problem. This enables the automatic derivation of the dual form of each sub problem and an automatic extraction of the Benders decomposition. It can help researchers focus on CP or mathematical programming (not both fields) for quickly designing prototypes of models.

In other papers, the master problem is solved with MIP and sub problems are formulated and solved with CP. Thorsteinsson [23] proposes a hybrid framework that encapsulates Benders decomposition as a special case. Jain and Grossmann [24] use a MIP/CP decomposition method in which the master MIP model is a relaxation of the original model and feasibility sub problems are solved with CP. They proposed also a LP/CP-based branch-and-bound algorithm to solve their hybrid models. Their application example is a scheduling problem of dissimilar parallel machines. In the same line of research, the paper of Maravelias and Grossmann [25] presents a conceptual similarity with our decomposition. Their master MIP is a relaxation of the original problem. Then given a relaxed solution, the CP sub problem checks whether there exists a feasible solution and generates integer cuts. This method is used to solve a batch

chemical process problem. As an extension of the previous approach, Hooker [26] uses a hybrid MIP/CP decomposition method to solve a class of planning and scheduling problems for manufacturing and supply chain management. This method, named logic-based Benders decomposition, exploits the strengths of MIP in the assignment portion and CP to tackle the scheduling part. Tasks are assigned to facilities by using MIP and then scheduled subject to release dates and deadlines using CP. The cuts used are based on either the information on the sets of tasks assigned to facilities (in case of cost minimization) or the information on the makespan (in case of makespan minimization).

4. A hybrid CP/MIP approach

The development of our approach stems from the limitation of a previous mathematical programming approach [10] as discussed in Section 3.1. The basic motivation for this CP/MIP decomposition is to benefit from the strengths of CP for scheduling (as it can handle nonlinear constraint) and of MIP for routing in this particular context. The approach is inspired by the logic-based Benders decomposition [18] although it does not take advantage of duality. Section 4.1 presents the decomposition framework, while in Sections 4.2 and 4.3, the master and the sub problem models are respectively described. Section 4.4 discusses the logic cuts generated from the sub problems.

4.1. The decomposition framework

In the decomposition method developed herein, the CP master problem determines both the assignment of the transportation requests to the vehicles and the schedule (i.e., the expected times) of the pick-up and the deliveries based on the shortest path routes (neglecting the possible route conflicts). For each solution of the master problem, the MIP sub problem tries to find collision free routes satisfying the schedule obtained from the master problem. When there are no solution (i.e., it is not possible to find conflict free routes that satisfy the schedule), logic cuts are generated and sent back to the master problem. The method is depicted in Fig. 2.

The logic cuts (described in detail in Section 4.3) are constraints added to the model to eliminate the CP solutions that have no feasible routing solution (both the incumbent and as many as possible other CP solutions with the same objective value (total delay)). One cut consists of a disjunction of three options: with the first two options, the same assignment of requests to the AGVs is kept but either the transportation time or the starting time of at least one delivery is increased by one unit, the third option consists in making a change to the assignment of requests to the AGVs (at least one request must be reassigned).

The CP master problem provides a very good lower bound to the original scheduling and routing problem and it contains essentially nonlinear constraints. The sub problem has a very strong minimum cost flow problem structure and thus can be solved efficiently by the network simplex (together with B&B).

4.2. The CP model

The model determines which vehicle will be processing what material handling request at what time by generating an ordered assignment of tasks to AGVs while minimizing the total number of delivery delays. The total amount of delays is measured by summing the difference between the planned start time

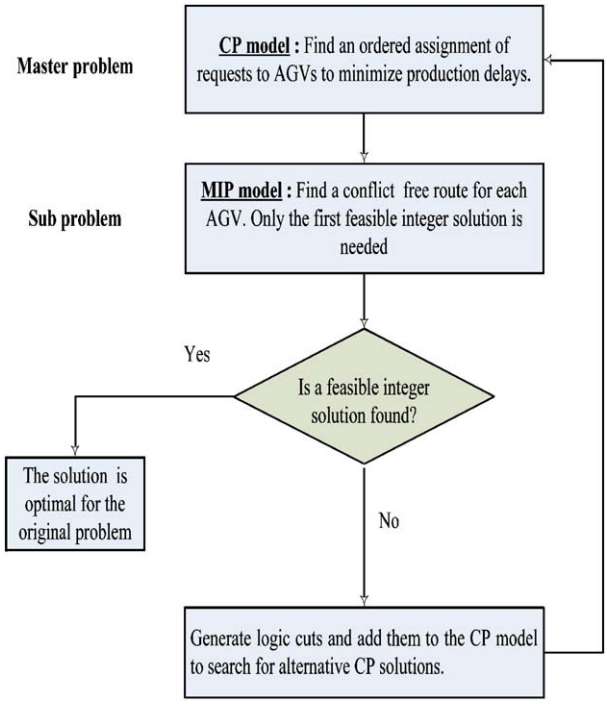

Fig. 2. The hybrid method.

and the earliest start time of each delivery. In this model, the distance (time) matrix is obtained by using shortest paths between nodes. Thus, the delays calculated (which do not take into account the possible conflicts) provide a lower bound of the actual delays. A transportation request consists of a pick-up and a delivery tasks. For modeling purposes, dummy start and end requests (and tasks) are associated with each AGV. If the successor of a dummy start request is the dummy end request corresponding to the same vehicle, it means that the AGV is not used during the whole horizon. In addition, we assume that a dummy start task corresponds to a delivery task of the preceding horizon. Hence the starting (beginning of the horizon) and final (end of the horizon) positions can be seen as dummy task nodes for each AGV for the current horizon and the next one. We define the set W as the set of all tasks including the dummy start ones.

Sets and parameters of the CP model

W	set of pick-up and delivery tasks
W^p	set of pick-up tasks
W^d	set of delivery tasks
V	set of AGVs
R	set of requests: each is a pair of pick-up (r^p) and delivery (r^{d})
T	the set of time periods
I	set containing R and the dummy start requests
O	set containing R and the dummy end requests

P	set of couples of tasks linked by a precedence relationship (p^1, p^2)
E	the set of all earliest start time
$e(\cdot)$	duration of the processing at a workstation
$D(\cdot,\cdot)$	length of the shortest path between the nodes of two tasks
n_i	the node for task i
n^v	the starting node of AGV v

Variables of the CP model

$v_r \in V$	variable representing the AGV assigned to request r
$s_r \in O$	variable representing the request performed immediately after r on the same vehicle
$t_r \in T$	variable representing the start time of request r

The objective function consists in minimizing the sum of the differences between the given earliest starting times and the actual starting time of the deliveries (the t_r) variables. The constraints used in the model are the following (for concision, we have not included the starting and ending conditions which are straightforward):

$$\text{Min} \quad \sum_{j \in W^d} (t_j - E_j) \tag{1}$$

$$\text{s.t.} \quad v_o = v_{s_o} \quad \forall o \in O, \tag{2}$$

$$t_{r^p} + 1 + D(r^p, r^d) \leqslant t_{r^d} \quad \forall r \in R, \tag{3}$$

$$s_{r_1} = r_2 \Rightarrow t_{r_1^d} + 1 + D(r_1^d, r_2^p) \leqslant t_{r_2^p} \quad \forall r_1, r_2 \in R, \tag{4}$$

$$\neg(t_{p^1} \leqslant t_i \leqslant t_{p^2}) \quad \forall p \in P,\ \ i \in W : (p^1, p^2 \neq i \wedge (n_i = n_{p^1} = n_{p^2}), \tag{5}$$

$$t_{p^1} + 1 \leqslant t_{p^2} \quad \forall p \in P : (p^1 \in W^p) \wedge (p^2 \in W^d), \tag{6}$$

$$t_{p^1} + 1 + e_{p^1} \leqslant t_{p^2} \quad \forall p \in P : (p^1 \in W^d) \wedge (p^2 \in W^p), \tag{7}$$

$$(t_i \geqslant t_j + 1) \vee (t_i + 1 \leqslant t_j) \quad \forall i, j \in W : (i \neq j) \wedge (n_i = n_j). \tag{8}$$

Constraints (2) ensure that each request and its successor are assigned to the same AGV. Constraints (3) specify that each vehicle processing a request must have enough time to go from the request pick-up node to the related delivery node. Constraints (4) ensure that if one request is the successor of another request on the same vehicle, the AGV must have enough time for dead-heading. Constraints (5) enforce that if two tasks linked by a priority constraint share the same node, they must be processed consecutively at this node. Constraints (6) state that at least one period (duration of the pick-up) must elapse between the starting times of pick-up and delivery tasks linked by a priority constraint. Furthermore, no other task can be performed on a node between the execution of two tasks linked by a priority constraint (more details are given in Section 4.1). Constraints (7) ensure that, if a delivery precedes a pick-up, there is at least one period (duration of the delivery) plus the processing time of the delivered product between their starting times. Constraints (8) ensure that two different tasks cannot start at the same node at the same time.

Search strategy. To help the CP model, we implemented some selection heuristics used in combination to the pre-implemented slice-based search (SBS), a technique similar to limited discrepancy search (LDS) (see [27]). In fact SBS determines the shape of the branching tree while our heuristics specify how to move inside the tree. The basic idea behind SBS (or LDS) is to minimize the number of decisions that

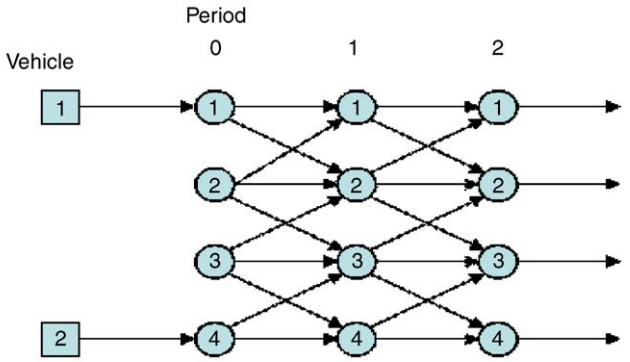

Fig. 3. Time-space network.

do not respect the heuristic first choices. To implement this search, one needs a selection heuristic that ranks all branches of a node. Then at each node of the search tree, a *discrepancy* is counted each time a branch, not ranked first, is taken. The overall process thus starts by traversing the search tree without allowing any discrepancy, and then it iteratively increases the number of discrepancies tolerated and traverses the search space again. The description of our heuristic procedure is as follows: variables are chosen according to a first-fail strategy, i.e., the variable with the smallest domain is instantiated first. Regarding the order of values for the instantiation of the variables, the following strategy was used: for the v variables, similar requests (same pick-up node and same delivery node) are assigned as much as possible to the same AGV, the closest AGV to the pick-up node being chosen first. The t and s variables are instantiated to values in increasing order.

4.3. The MIP model

Since an AGV may visit an arc or a node more than once during the horizon, a time-space graph was used to model the routing of the AGVs. The time-space graph is illustrated in Fig. 3 and can be described as follows: a node is defined for each period (of one time unit) and each endpoint of the guide path segments of the FMS. Each node of a given period is linked by arcs to the corresponding adjacent nodes of the next period. Those arcs correspond to the movement of an AGV between two adjacent endpoints of the segments of the guide path. Each node of a given period is also linked by an arc to the corresponding same node of the next period (i.e., a horizontal arc). This corresponds to waiting one unit of time at that node. There are no arcs between nodes of a same period. All arcs of the graph have a length of one time unit. For each AGV, there is a starting node at period 0. The problem corresponds then to a multi-commodity network flow model where each commodity represents one vehicle. The master problem solution imposes that each flow visits specific nodes at specific times.

Sets and parameters of the MIP model

V	set of AGVs
N	set of nodes
T	set of all time periods
T^+	set of all time periods but the first

A	set of arcs
A^+	set of all arcs (including waiting arcs)
$A^f[.]$	the set of arcs coming from node i
$A^t[.]$	the set of arcs entering in node i
$A^w[.]$	the waiting arc associated to each node
$A^o[.]$	the opposite arc of each arc
n^v	the starting node of AGV v
W	array of records describing each a task with two fields (node, earliesttime)
v, t	data obtained from the CP model
R	set of requests: each is a pair of pick-up (r^p) and delivery (r^d)
$R^+[., t]$	for every pair of (node i, period t), set of pick-up and delivery tasks r whose service node are i and which are performed at time t ($(t = t_r - 1) \wedge (i = n_r)$)

Variables of the MIP model

$X^t_{k,a}$	Boolean variable = 1 if AGV k starts traversing arc a at period t and 0 otherwise.

Since we only search for a feasible solution the objective function is irrelevant, we thus used a constant objective function. The model is the following:

$$\text{Min} \quad 1 \tag{9}$$

$$\text{s.t.} \quad \sum_{a \in A^f[n^k]} X^o_{k,a} = 1 \quad \forall k \in V, \tag{10}$$

$$\sum_{a \in A^+} X^t_{k,a} = 1 \quad \forall t \in T, \ \ k \in V, \tag{11}$$

$$\sum_{a \in A^f[i]} X^t_{k,a} + \sum_{a in A^t[i]} X^t_{k,a} = 0 \quad \forall i \in N, \ \ k \in V, \ \ t \in [1 \ldots (M-1)], \tag{12}$$

$$\sum_{k \in V} X^t_{k,a} + \sum_{k \in V} X^t_{k,A^o[a]} \leqslant 1 \quad \forall t \in T, \ \ a \in A, \tag{13}$$

$$X^{T_{rp}}_{V_r, A^w[n_{rp}]} = 1 \quad \forall r \in R, \tag{14}$$

$$X^{T_{rd}}_{V_r, A^w[n_{rd}]} = 1 \quad \forall r \in R, \tag{15}$$

$$\sum_{k \in V, a \in A^t[i]} X^t_{k,a} \leqslant 1 + \left(\sum_{r \in R^+[t-1,i]} 1 \right) \times \left(\sum_{r \in R^+[t,i]} 1 \right) \quad \forall i \in N, \ \ t \in T^+. \tag{16}$$

Constraints (10) ensure that each AGV is located at its initial position at the beginning of the horizon while constraints (11) ensure that each AGV is located at a unique position at each period. Constraints (12) are flow conservation constraints. Constraints (13) are collision avoidance constraints. Constraints (14) state that every pick-up task must be done by the right vehicle at the right time while constraints (15) are the equivalent of constraints (14) for delivery tasks. Constraints (16) are node capacity constraints stating that there can be only one AGV on a node at any time except the case where one AGV leaves a node task while another one is just entering in the same node task (that's why the product of the sums is added to the right hand side of constraints (16)). If the above MIP has a feasible solution then optimality for the global problem is obtained. Otherwise, cuts are added to the master problem which is re-solved.

4.4. Logic cuts

As there are many different solutions to the scheduling model with the same production delay, the objective is to define cuts which forbid not only the current infeasible (i.e., conflicting) solution but as many other solutions exhibiting the same undesired properties. The logic cuts, generated when no feasible routing can be found, are based on the starting times of deliveries and the assignments of requests to AGVs. The intuition behind the presented cuts is that when a conflict occurs between two AGVs, at least one of them does not have enough time to fulfill its next task.

In addition to the parameters and sets defined in the CP model, the following ones are used:

t_j^* the current optimal starting time of task j
v_j^* the current optimal vehicle assigned to r
s_j^* the current optimal successor of request r

The cuts generated in this approach are essentially feasibility cuts. When a conflict arise in the routing model, it means that the duration of at least one trip (material handling or deadhead) is too short. To correct this situation it is thus necessary to either change at least one starting time of a task (pick-up/delivery) or change the assignments of transportation requests to AGVs. A logic cut consists of a disjunction of the three following constraints:

- Keep the assignment of requests to AGVs and increase the time of routing between the pick-up and the delivery of a same request. This corresponds to extending at least by one period the time window between a pick-up and a delivery. This constraint can be written as

$$\left(\sum_{r\in O}(v_r = v_r^*) = |R| + |V| \wedge \sum_{r\in R}((t_{r^d} - t_{r^p}) \geqslant (t_{r^d}^* - t_{r^p}^*)) \geqslant 1\right).$$

- OR keep the assignment of requests to AGVs and the time of routing between the pick-up and the delivery of each request but increase the starting time of some tasks. This corresponds to delaying at least one request, namely

$$\bigvee\left(\sum_{r\in O}(v_r = v_r^*) = |R| + |V| \wedge \sum_{r\in R}(t_{r^d} = S_{r^d}^*) \leqslant |R| - 1\right).$$

- OR change the assignments of requests to AGVs.

$$\bigvee\left(\sum_{r\in O}(v_r = v_r^*) \leqslant |R| + |V| - 1\right).$$

5. Experimentation

This section presents the computational experiments. We first describe the instances used for the tests and then we report the results. Finally, we discuss some additional features of the method that were explored.

Table 1
Description of the problem sets

Problem set	Category	Number of requests	Number of precedence relationship
1	Compact	8	7
2	Compact	10	9
3	Compact	8	7
4	Compact	9	7
5	Compact	10	4
6	Spread out	8	1
7	Spread out	10	2
8	Spread out	8	3
9	Spread out	8	5
10	Spread out	8	9
11	Spread out	8	9
12	Spread out	7	8
13	Mixed	12	9
14	Mixed	12	6
15	Mixed	13	7

5.1. The instances

The FMS used for the experiments is presented in Fig. 1. The guide path was divided into segments of 7.5 m. Assuming that the AGVs move at a constant speed of 0.5 m/s, these segments are therefore traveled in 15 s, which corresponds to one time unit. The instances we used are built from the data set of Desaulniers et al. [10] using 12 different request sets from several orders of an order book. The details of the order book can be found in [9,28]. A planned production schedule based on average material handling times provides the processing times at the work station, the earliest processing start time, and the processing time, as well as the precedence relationships to satisfy for each pair of operation and part type. The number of transportation requests in those 12 sets varies from 7 to 10. There are two categories of sets, differing by the spatial density of the requests. The first category, denoted *compact*, corresponds to the case where several requests are planned in the same region of the FMS. This situation arises when an order requires the transportation of a large number of components between two or three neighbouring work stations, or when two orders require similar treatments. The second category, called *spread out*, corresponds to the case where the requests occur in different regions of the FMS. Such a request configuration naturally leads to an assignment of the AGVs by region, thus reducing the combinatorial aspect of the problem. We have added to these 12 requests sets three other sets with more requests. This third category of request sets, denoted *mixed*, is a mixture of the types *spread out* and *compact* and might reflect more complex situations which are closer to real world instances. Table 1 presents the characteristics of the 15 problem sets tested in this paper. For each of those 15 request sets, we solved the problem with 2, 3, 4, 5, and 6 AGVs, respectively, which represents 75 instances. We used a time horizon of 150 periods of 15 s, which corresponds to 37.5 min. This length of horizon was chosen in order to allow a feasible solution to all the instances. We implemented the following constraints (not present in [10]) in our model in order to make

it more realistic:

(1) When two tasks are linked by priority constraints on the same node, no other task can be performed at this node between their starting times. For example, when a delivery precedes a pick-up on a workstation, the product delivered must be processed during a certain amount of time. And when a product is being processed in a workstation, the corresponding node can be considered busy for any other potential pick-up/delivery task.
(2) When two tasks are linked by priority constraints on two different nodes, there is an order relation between the starting times but another task can be performed between the two times at either nodes.

Table 2 describes in detail three request sets, one per category. The last column gives the starting position of the AGVs in the instance with 6 AGVs. For each set, the material handling requests are listed in the second column. In the third (respectively, fourth) column, we have the pick-up nodes (respectively delivery nodes) and their associated earliest end of processing time (EEP) of the pallet at the pick up node (respectively, the earliest start of processing time (ESP) of the pallet at the delivery node). The fifth column presents the pallet processing time at delivery nodes. The sixth column enumerates the precedence relationship between the tasks. For example, DX $\rightarrow$ DY means that task DX precedes task DY.

5.2. Results

Experiments were performed using OPLScript 3.6 on a Pentium 4, 1.5 GHz, 512 Mo (RAM) PC. The CP model is solved with Ilog Solver 5.2 while the MIP model is solved with CPLEX 8.0. We set arbitrarily the limit for the CPU time at 12 min which seems reasonable for a 37.5 min horizon. The results are presented in Tables 3–7. In those tables,'$X : Y$' means that the instance considered is request set X with Y AGVs. NbCuts is the number of logical cuts generated. An asterisk (*) means that a solution to the original model is found after the limit of 12 min but within the horizon of 37.5 min (such solutions could be used for future fleet sizing). A hyphen (-) means that no CP solution was found in less than one hour. Looking at Tables 3–7, one can observe the following: first when the number of AGVs is not sufficient (Table 3, 2 AGVs), the production delays are significant. Furthermore, it seems that producing a valid schedule is difficult since 6 out of the 15 problems could not be solved within an hour. By increasing the number of AGVs (Table 4), it becomes possible to solve all instances in the given time horizon and the added flexibility allows to reduce the production delays of most instances. If we try to increase again the number of AGV to 4 (in Table 5), some instances start to experience congestion as problems 7 and 13 are now solved in about 20 min. The solution times reported for problem 7 show that it is easy to schedule but complex to route, which indicate that conflicts on arcs are probably difficult to avoid. Again the increase number of AGVs allows to decrease the production delays of several problems. Considering a higher number of AGVs (5 AGVs in Table 6 and 6 AGVs in Table 7) does not change this picture very much. In both situations, congestion problems prevent two instances to be solved in the given 12 min time horizon. Furthermore, production delays are now only slightly reduced (problem 11 in Table 6 and problem 9 in Table 7). The increased number of vehicles is thus of no help to accelerate production and even counter productive since they congest the network. For *compact* and *spread out* instances, around 90% of time solution is consumed by the MIP model. The drawback of using a time space graph is that the number of variables and constraints of the MIP formulations rapidly grow with the size of the instance. But this representation is necessary to track the position of every AGV at every period.

Table 2
Detailed description of request sets 3, 15, and 13 with 6 AGVs

Set number	Request number	Pickup (Node, EEP)	Delivery (Node, ESP)	Processing time	Precedence relationship	Initial Node
3	1	(15, 0)	(13, 5)	12	D1 → P5	AGV1:15
	2	(15, 12)	(13, 19)	12	D2 → P6	AGV2:14
	3	(15, 24)	(13, 33)	12	D3 → P7	AGV3:19
	4	(15, 36)	(13, 47)	12	D4 → P8	AGV4:27
	5	(13, 17)	(25, 28)	10	P5 → D2	AGV5:12
	6	(13, 31)	(25, 41)	10	P6 → D3	AGV6:8
	7	(13, 45)	(25, 55)	10	P7 → D4	
	8	(15, 24)	(13, 33)	12		
15	1	(15, 0)	(13, 5)	10	D1 → P3	AGV1:15
	2	(15, 10)	(13, 17)	10	D2 → P4	AGV2:14
	3	(13, 15)	(19, 21)	24	D1 → P3	AGV3:19
	4	(13, 37)	(19, 57)	24	D5 → P8	AGV4:27
	5	(13, 0)	(25, 18)	40	D3 → P9	AGV5:12
	6	(13, 60)	(25, 78)	40	P10 → D7	AGV6:25
	7	(25, 16)	(1, 29)	10	D10 → P11	
	8	(25, 79)	(1, 92)	10	D11 → P12	
	9	(19, 56)	(7, 64)	18		
	10	(1, 43)	(27, 55)	10		
	11	(27, 62)	(3, 86)	10		
	12	(3, 106)	(5, 110)	20		
	13	(3, 37)	(5, 41)	20		
13	1	(15, 0)	(13, 5)	10	D1 → P5	AGV1:15
	2	(15, 12)	(13, 27)	10	D2 → P6	AGV2:14
	3	(15, 24)	(13, 49)	10	D3 → P7	AGV3:19
	4	(15, 36)	(13, 71)	10	D4 → P8	AGV4:27
	5	(13, 25)	(19, 31)	14	D5 → P9	AGV5:12
	6	(13, 47)	(19, 57)	14	P6 → D3	AGV6:25
	7	(13, 69)	(19, 81)	14	P7 → D4	
	8	(13, 91)	(19, 107)	14	D10 → P11	
	9	(19, 55)	(7, 100)	18		
	10	(13, 0)	(25, 18)	18		
	11	(25, 0)	(1, 18)	10		
	12	(3, 30)	(5, 38)	10		

Looking at the results all in all, we notice that the implemented logic cuts, are almost never generated for compact or spread out requests sets; it seems that the CP is sufficient to eliminate the conflicts on task nodes by setting good slacks for material handling of each request and deadheads between requests.

Logic cuts are mostly generated on the mixed requests sets where they allow producing potentially conflict free schedules. Unfortunately the disjunctive nature of these cuts makes the model more difficult and its resolution more time consuming. This negative impact is however limited since the number of generated cuts is rather small.

The CP search strategy we used helped to avoid a number of conflicts (particularly for *compact* or *spread out* instances) early in the scheduling stage (CP model). For *compact* instances, no production

Table 3
2 AGVs problem set

Instance	First CP model			NbCuts	Production delay
	Solution time	Choice points	Global solution time		
1:2	0.51	2057	17.45	0	0
2:2	0.52	1696	16.53	0	4
3:2	0.48	2213	102.94	0	14
4:2	—	—	—	—	—
5:2	—	—	—	—	—
6:2	0.28	1816	58.26	0	2
7:2	4.62	17187	57.25	0	2
8:2	—	—	—	—	—
9:2	0.61	3628	13.65	0	12
10:2	—	—	—	—	—
11:2	0.50	2681	44.97	0	13
12:2	0.03	239	11.13	0	6
13:2	—	—	—	—	—
14:2	—	—	—	—	—
15:2	154.53	415873	212.47	1	60

Table 4
3 AGVs problem set

Instance	First CP model			NbCuts	Production delay
	Solution time	Choice points	Global solution time		
1:3	2.56	17038	84.79	0	0
2:3	0.45	2585	147.72	0	0
3:3	1.39	10331	199.48	0	0
4:3	7.11	43674	210.54	0	0
5:3	7.55	40898	140	0	1
6:3	0.19	1461	212.63	0	2
7:3	2.52	15638	229.35	0	2
8:3	0.25	1771	198.84	0	2
9:3	0.70	3786	234.06	0	6
10:3	0.67	5737	84.04	0	8
11:3	1.36	6689	88.35	0	10
12:3	0.11	848	45.86	0	6
13:3	298.74	1052988	534.79	20	26
14:3	101.30	530036	358.67	2	0
15:3	144.50	471880	282.16	4	25

delay occurs with more than 3 AGVs. For *spread out* and *mixed* instances, they are all solved with 3 AGVs but with a certain level of production delay. To decrease the production delay, the number of AGVs must be increased. But this has a major drawback: it increases congestion with unsolved instances as a final

Table 5
4 AGVs problem set

Instance	First CP model			NbCuts	Producton delay
	Solution time	Choice points	Global solution time		
1:4	5.62	33471	90.17	0	0
2:4	0.47	2698	195.19	0	0
3:4	2.58	18133	182.75	0	0
4:4	20.32	123212	424.41	0	0
5:4	4.47	20221	86.05	0	1
6:4	0.22	1616	203.00	0	2
7:4	4.62	28238	1117.05*	1	2
8:4	0.44	3286	105.25	0	2
9:4	0.44	2301	99.84	0	5
10:4	0.89	7347	158.04	0	8
11:4	0.98	4070	129.44	0	9
12:4	0.01	142	75.86	0	6
13:4	974.21	314	1290.94*	24	26
14:4	183.81	939828	602.11	24	0
15:4	273.10	1284629	632.27	0	17

Table 6
5 AGVs problem set

Instance	First CP model			NbCuts	Production delay
	Solution time	Choice points	Global solution time		
1:5	6.81	41772	140.02	0	0
2:5	0.58	2698	117.52	0	0
3:5	3.68	21899	257.95	0	0
4:5	33.00	203897	338.83	0	0
5:5	5.18	22146	109.09	0	1
6:5	0.44	3070	1011.13*	1	2
7:5	4.64	25293	336.83	0	2
8:5	0.32	2093	194.46	0	2
9:5	—	—	—	—	—
10:5	0.06	347	89.69	0	8
11:5	0.70	4647	199.41	0	8
12:5	0.01	123	68.47	0	6
13:5	325.583	1566728	395.85	0	26
14:5	10.16	53178	625.59	16	0
15:5	519.14	2073129	706.61	4	17

result. For *mixed* problems, the scheduling problem (modeled in CP) becomes more difficult to solve. The tables of results also show that the CP model needs to be enhanced since in almost half of the instances the number of choice points (branching tree nodes) is very large, due to the lack of a more efficient search

Table 7
6 AGVs problem set

Instance	First CP model			NbCuts	Production delay
	Solution time	Choice points	Global solution time		
1:6	7.96	45628	84.57	0	0
2:6	0.54	2698	268.11	0	0
3:6	4.05	23544	937.97*	1	0
4:6	45.79	2613777	138.20	0	0
5:6	5.73	23078	78.69	0	1
6:6	0.50	3129	833.00*	1	2
7:6	2.01	10903	108.55	0	2
8:6	0.36	2101	120.45	0	2
9:6	0.13	255	229.01	0	3
10:6	0.06	347	178.03	0	8
11:6	0.70	4553	93.59	0	8
12:6	0.02	123	123.50	0	6
13:6	72.31	282457	509.81	20	26
14:6	25.41	115965	455.27	20	0
15:6	525.93	2301091	605.09	0	17

Table 8
Some statistics on the first CP and final MIP models

NbAGV	First CP model		Final MIP model	
	NbVar	NbConstr	NbVar	NbConstr
2	75	338	42600	76400
3	83	374	63900	103100
4	89	428	85200	129800
5	96	470	106500	156400
6	101	496	127800	183100

strategy for all types of problems. We mention that in each instance solved, no AGV remains idle. This is why the solution of our problem can be seen as a way of finding a balance between two objectives: increasing the number of available AGVs to decrease production delays and avoiding congestion.

Table 8 presents the average number of variables and constraints for the first CP and final MIP models with a specified number of AGVs available in the FMS. In these two types of models, NbVar denotes the number of variables while NbConstr is the number of constraints. The MIP model is "stable" since we have almost the same number of variables and constraints in each table shown above.

5.3. Additional features

We report here some features that we explored, unfortunately without much success, in the hope of improving our method. We nevertheless present them as they could possibly be useful in another AGV context.

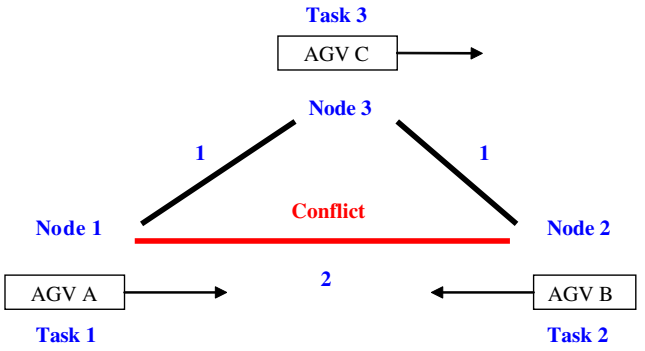

Fig. 4. Why conflict detection variables are not effective.

- *Search strategy based on space proximity*: This strategy assigns requests to AGVs according to the proximity of the associated pick-up node of each request to the starting node of each AGV. This strategy worked well only for problems with a *compact* set of requests (i.e., with many requests planned in the same region of the FMS).
- *Conflict detection variables*: We tried to use conflict detection variables in the MIP that would enable to send information (cuts) to the CP model. The following Boolean conflict detection variables were used:

$C^n_{it} = 1$ if there is a conflict on node i at period t, 0 otherwise.
$C^a_{at} = 1$ if there is a conflict on arc a at period t, 0 otherwise.

The following nonconstant objective function is then minimized (the sum of conflicts on arcs and nodes):

$$\sum_{i\in N, t\in T} C^n_{it} + \sum_{a\in A, t\in T} C^a_{at}. \tag{17}$$

These conflict detection variables are present in the following two families of constraints which, respectively, replace constraints 13 and 16:

$$\sum_{k\in V, a\in A^t[i]} X^t_{k,a} \leqslant 1 + \left(\sum_{r\in R^+[t-1,i]} 1\right) \times \left(\sum_{r\in R^+[t,i]} 1\right) + C^n_{it} \quad \forall i \in N,\ \ t \in T, \tag{18}$$

$$\sum_{k\in V} X^t_{k,a} + \sum_{k\in V} X^t_{k,A^o[a]} \leqslant 1 + C^a_{at} \quad \forall t \in T,\ \ a \in A. \tag{19}$$

However, these would not give enough insight about the origin of a conflict and how to repair it. When two tasks generate a conflict on a segment, it is not sufficient to simply try to remove one of the two tasks from the list of tasks assigned to a vehicle to resolve the conflict. There may exist a solution with the same assignments of tasks but with a third task delayed. In the example (Fig. 4), the distance between node 1 and node 3 and between node 3 and node 2 is 1. The arc between node 1 and node 2 is of length 2. Here, tasks 1 and 2 cannot be delayed because their starting time is equal to their maximum starting time. But task 3 can be delayed. A conflict detection variable would detect a conflict between

AGV A and AGV B even though an optimal solution may exist when delaying task 3 and allowing a detour for AGV A or AGV B.

- *Time windows for pick-ups*: In order to reduce the number of equivalent solutions to the scheduling problem, we attempted to transfer the starting time decision to the routing problem. This means that the scheduling model only defines feasible time windows for each task. These time windows are transferred to the routing problem so that they can be exploited to avoid conflicts. These time windows are intervals between the starting time and the associated maximum value of each pick-up. This strategy yielded interesting results (the presence of time windows helps the MIP model to be solved faster), but unfortunately it was no longer possible to guarantee that the precedence and consecutiveness constraints would be satisfied a posteriori. Precedence constraints could also be modeled in the routing model, but it was not the case of consecutiveness constraints which they are not linear.

6. Conclusion

In this paper, we used a decomposition method to solve a difficult combinatorial integrated scheduling and conflict free routing problem. This hybrid method consists in dividing the problem into two interrelated sub problems. The first sub problem is modeled with CP, a very strong tool to address scheduling problems that allowed us to design a specific search strategy. The second sub problem is basically a CSP problem modeled as a MIP problem to benefit from its network substructure. The two main reasons for choosing such a decomposition method are the need to instantiate assignment variables before routing and the existence of many non linear constraints in the scheduling part of the problem. We solved problems with up to six AGVs, 13 requests on a rolling horizon of 37.5 min. Our method can also be used to determine the size of the AGVs fleet. An interesting avenue of research consists in designing better logic cuts. This would be useful, as the cuts presented herein tend to be less effective when the level of production delay increases. The design of a better search strategy for the CP model could also allow to address problems with a longer horizon, more tasks or more AGVs. In such instances, the CP model tends to perform poorly. When the domain size significantly increase, an efficient search strategy becomes essential in identifying the good values in each domain.

References

[1] Co CG, Tanchoco JMA. A review of research on AGVs vehicle management. Engineering Costs and Production Economics 1991;21:35–42.

[2] King RE, Wilson C. A review of automated guided vehicle system design and scheduling. Production Planning and Control 1991;2:44–51.

[3] Ganesharajah T, Hall NG, Sriskandarajah C. Design and operational issues in AGV-served manufacturing systems. Annals of Operations Research 1998;76:109–54.

[4] Qiu L, Hsu W-J, Wang H. Scheduling and routing algorithms for AGVs: a survey. International Journal of Production Research 2002;40:745–60.

[5] Lee JH, Lee BH, Choi MH. A real-time traffic control scheme of multiple AGV systems for collision free minimum time motion: a routing table approach. IEEE Transactions on Systems, Man and Cybernetics—Part A: Systems and Humans 1998;28:347–58.

[6] Rajotia S, Shanker K, Batra JL. A Semi-dynamic window constrained routing strategy in an AGV system. International Journal of Production Research 1998;36:35–50.

[7] Krishnamurthy NN, Batta R, Karwan MH. Developing conflict-free routes for automated guided vehicles in a flexible manufacturing system. Operations Research 1993;41:1077–90.
[8] Oboth C, Batta R, Karwan M. Dynamic conflict-free routing of automated guided vehicles. International Journal of Production Research 1999;37:2003–30.
[9] Langevin A, Lauzon D, Riopel D. Dispatching, routing and scheduling of two automated guided vehicles in a flexible manufacturing system. International Journal of Flexible Manufacturing Systems 1996;8:246–62.
[10] Desaulniers G, Langevin A, Riopel D, Villeneuve B. Dispatching and conflict-free routing of automated guided vehicles: an exact approach. The International Journal of Flexible Manufacturing Systems 2003;15:309–31.
[11] Dincbas M, Van Hentenryck P, Simonis H. Solving large combinatorial problems in logic programming. Journal of Logic Programming 1990;8:75–93.
[12] Van Hentenryck P. Constraint satisfaction in logic programming. Cambridge, MA: MIT Press; 1989.
[13] Wallace M. Practical applications of constraint programming. Constraints 1996;1:139–68.
[14] Saraswat V, Van Hentenryck P. Principles and practice of constraint programming. Cambridge, MA: MIT Press; 1995.
[15] Mariott K, Stuckey PJ. Programming with constraints. MA: MIT Press; 1998.
[16] Frhwirth T, Abdennadher S. Essentials of constraint programming. Berlin: Springer; 2003.
[17] Apt K. Principles of constraint programming. Cambridge: Cambridge University Press; 2003.
[18] Hooker JN, Ottosson G. Logic-based Benders decomposition. Mathematical Programming 2003;96:33–61.
[19] Milano M. Constraint and integer programming: toward a unified methodology. In: Sharda R, editor. Operations research/computer science interfaces series. Dordrecht: Kluwer Academic Publishers; 2004. p. 33–53.
[20] Hooker JN. Logic-based methods for optimisation. New York: Wiley; 2000.
[21] Benoist T, Gaudin E, Rottembourg B. Constraint programming contribution to Benders decomposition: a case study. In: Van Hentenryck P, editor. Lecture notes in computer science, vol. 2470. Eigth international conference, CP 2002 2470. 09/2002. Berlin: Springer; 2002.
[22] Eremin A, Wallace M. Hybrid Benders decomposition algorithms in constraint logic programming. Lecture notes in computer science. CP 2001, vol. 2239. Berlin: Springer; 2001.
[23] Thorsteinsson ES. Branch-and-check: a hybrid framework integrating mixed programming and constraint logic programming. Principles and practice of constraint programming - CP 2001. Lecture notes in computer science, vol. 2239. 2001. p. 16–30.
[24] Jain V, Grossmann I. Algorithms for hybrid MILP/CP models for a class of optimization problems. Informs Journal on Computing 2001;13:258–76.
[25] Maravelias CT, Grossmann IE. Using MILP and CP for the scheduling of batch chemical processes. In: Régin J-C, Rueher M, editors. Integration of AI and OR techniques in constraint programming for combinatorial optimization problems. First international conference, CPAIOR. Lecture notes in computer science, vol. 3011. Berlin: Springer; 2004.
[26] Hooker JN. A hybrid method for planning and scheduling. In: Wallace M, editor. CP 2004, Lecture notes in computer science, vol. 3258. Berlin, Heidelberg: Springer; 2004.
[27] Ilog OPL Studio. Ilog OPL Studio 3.6 language manual; 2002.
[28] Drolet M. Ordonnancement d'un carnet de commandes pour un atelier flexible de sous traitance. Projet de fin d'études, 1991. Département de génie industriel, École Polytechnique de Montreal, Canada;1991.