Les Cahiers du GERAD ISSN: 0711–2440

A Scheduling and Conflict-Free Routing Problem Solved with a Hybrid Constraint Programming/Mixed Integer Programming Approach

A. Insa Corréa, A. Langevin, L.-M. Rousseau

G–2005–05

January 2005

A Scheduling and Conflict-Free Routing Problem Solved with a Hybrid Constraint Programming/Mixed Integer Programming Approach

Ayoub Insa Corréa

André Langevin*

Louis-Martin Rousseau

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

* *and GERAD*

January 2005

Les Cahiers du GERAD

G–2005–05

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 modeled with Constraint Programming and the sub problem (conflict-free routing) with Mixed Integer Programming. Logic cuts are generated and used to prune optimal scheduling solutions whose routing plan exhibits conflicts. The hybrid method presented herein allowed to solve instances with up to six AGVs.

Key Words: Constraint programming, mathematical programming, hybrid model, logical Benders decomposition, material handling system, automated guided vehicles, vehicle routing and scheduling.

Résumé

Dans cet article, nous proposons une méthode hybride pour résoudre un problème de répartition et routage sans conflit d'une flotte de véhicules automatiques dans un atelier flexible. Notre problème est un problème simultané d'affectation et routage sans conflit de véhicules automatiques. Notre approche consiste en une méthode hybride dans laquelle le problème maître (ordonnancement) est modélisé en programmation par contraintes tandis que le sous problème (routage sans conflit) est modélisé en programmation linéaire en nombres entiers. Des coupes logiques sont générées pour éliminer les solutions optimales pour la partie ordonnancement mais qui présentent des conflits de routage. La méthode hybride présentée dans cet article permet de résoudre des problèmes contenant jusqu'à 6 véhicules automatiques.

1 Introduction

This study focuses on the routing and scheduling 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 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 Co and Tanchoco (1991), King and Wilson (1991) and Ganesharajah et al. (1998). For a recent review on AGVs scheduling and routing problems and issues, the reader is referred to the survey of Qiu et al. (2002). 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 routing as with categories (1a) and (1b). The method presented in this article belongs to the third category (1c) and address 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 with a review of relevant works. Section 3 presents a hybrid Constraint Programming (CP)/Mixed Integer Programming (MIP) approach. Section 4 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). Each order determines a sequence of operations on the various cells of the FMS. The production scheduling, i.e., setting the earliest starting time of each operation for each pallet of each order, is done *a priori*. The unit load is one pallet. Hence, each material handling request is composed of the pick-up and the delivery of a 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 position) 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.* 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 delivery tasks subject to the following constraints:

- For each pick-up or delivery task, the lower bound of its associate time windows 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.
- All nodes may be visited several times by the AGVs either to perform a task or for a simple traversal.

A number of authors have addressed the conflict-free routing problem with a static job set, i.e., with all jobs known *a priori*. Lee et al. (1998) 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). Qiu and Hsu (2001) present a method consisting of scheduling and routing a batch of AGVs concurrently on a bi-directional linear path layout. Regarding the scheduling part of their approach, jobs are handled one batch at a time. The presence of linear track enables to find conflict free routes. The major shortcoming of their method is unrealistic assumptions; for example, the set of AGVs is partitioned in two groups and they are routed to run on both parallel lanes simultaneously in opposite directions to achieve a high degree of synchronism. Rajotia et al. (1998) 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. (1993) 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. (1999) 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. (1993) 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. (1996) 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. (2003) 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 A Hybrid CP/MIP Approach

A hybrid CP/MIP decomposition method has been developed for the problem of simultaneously scheduling all the pick-up and delivery tasks and routing without conflicts the AGVs. The basic goal of this decomposition is to benefit from the strengths of CP for scheduling and of MIP for routing in this particular context. CP is particularly useful since it can easily handle non linear constraints. The approach is inspired by the logic-based Benders decomposition. The reader is referred to Hooker and Ottosson (2003) and Milano (2004) for comprehensive reviews on hybrid methods. Hooker (2000) 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. (2002) 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 calling center. Eremin and Wallace (2001) 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 (2001) proposes a hybrid framework that encapsulates Benders decomposition as a special case. Jain and Grossmann (2001) 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 (2004) 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. In the same line of research, Hooker (2004) uses a hybrid MIP/CP de-

composition 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). This approach gives good results even though it is less effective when there are more than 30 tasks to schedule. In the decomposition method developed herein, the master problem, solved by CP, determines both the assignment of the transportation tasks to the vehicles and the scheduling (i.e., the expected time) of the pick-up and the deliveries, based on the shortest path routes. The CP master problem provides a very good lower bound to the original model objective function. It contains essentially non linear constraints. For each solution of the master problem, the MIP sub problem checks whether there is a feasible solution (collision-free routes) and, if not, generate logic cuts sent back to the master problem. 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). This approach is very similar to the logic Benders decomposition proposed by Hooker and Ottosson (2003) since the master problem generates an optimal but potentially infeasible solution which is then validated by a subproblem. When this solution is infeasible, the subproblem generates a set of feasibility cuts that are added to the master problem. However our approach does not take advantage of duality as it is usually the case in logic or classic Benders decomposition. The method is depicted in Figure 1. The CP and the MIP models are described next.

3.1 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. The total amount of delays is measured by summing the difference between the planned start time 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 don't 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.

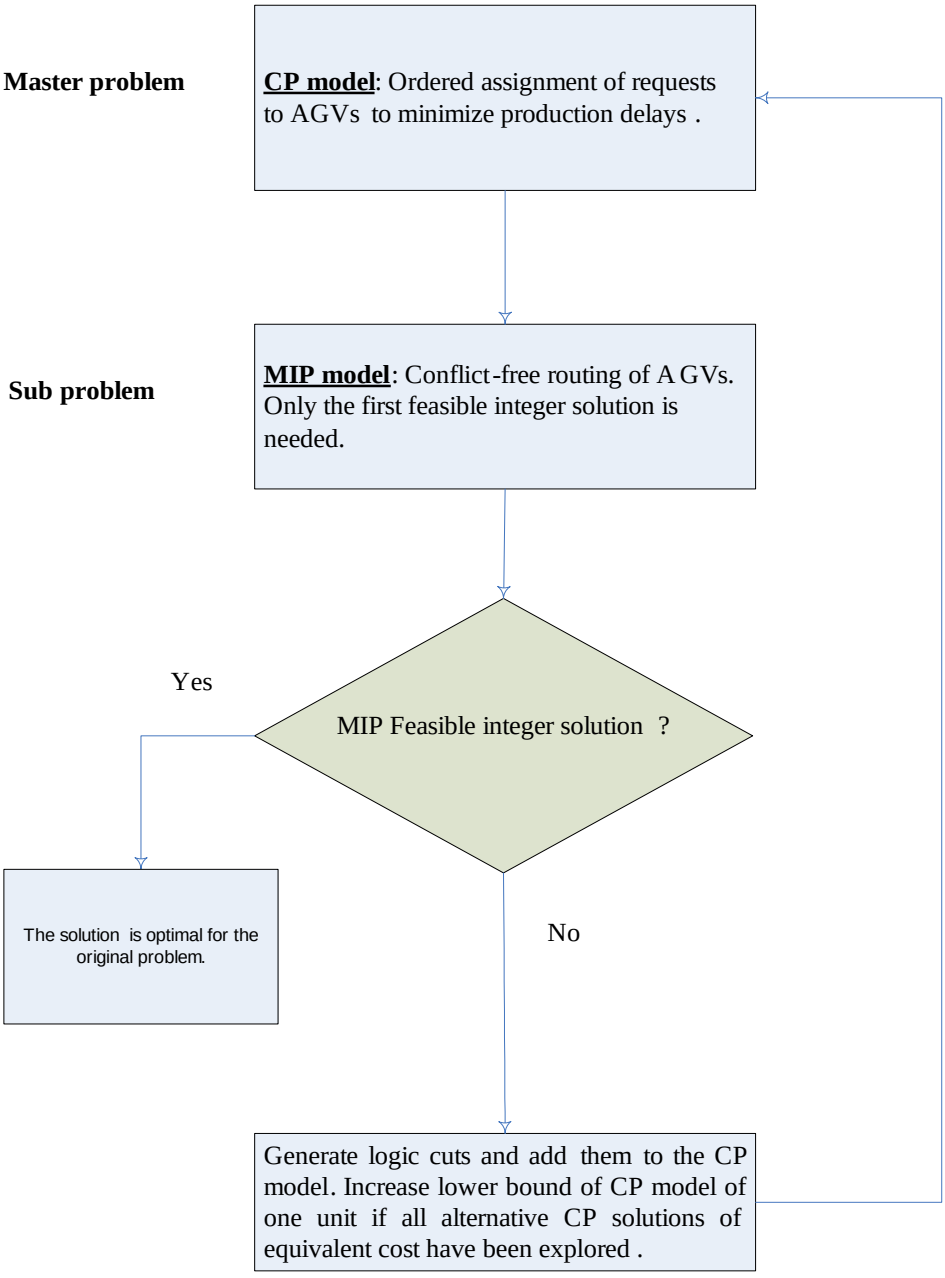

Figure 1: The hybrid method

Sets and parameters:

Dummy	set of dummy starting requests. Each of them is in fact the starting node of a vehicle corresponding to the last delivery node of a vehicle in the previous planning horizon;
n^v	the starting node of AGV v;
W^p	set of pick-up tasks;
W^d	set of delivery tasks;
W	set of pick-up and delivery tasks;
D$(\cdot\ ,\cdot)$	length of the shortest path between the nodes of two tasks;
n_i	the node for task i;
$\|V\|$	number of vehicles available;
V	set of AGVs;
R	set of requests. Each request contains two fields: the pick-up task and the associated delivery task;
$\|R\|$	number of requests to plan;
I	set containing R and the dummy start requests;
O	set containing R and the dummy end requests;
e$(\cdot)$	duration of the processing at a workstation;
P	set of couples of tasks linked by a precedence relationship (the first task is to be performed before the other);
E_j	earliest starting time of task j.

The model uses the three following finite domain variables:

$A_r = k$	if request r is assigned to vehicle k;
$S_u = v$	if request v is the successor of request u on the same vehicle;
T_j	is the start time of task j.

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 variables). In constraint programming, a boolean expression given in parenthesis represents an *indicator* variable equal to 1 if the event described is true and 0 otherwise (for instance $(X = a)$ takes value 1 when if variable X takes value a and 0 if it does not). It is thus possible to build constraints on indicators.

Minimize

$$\sum_{j \in W^d} (ST[j] - EST[j])$$

The constraints used in the model are the following (for concision, we have not included the starting and ending conditions which are straightforward):

$$\sum_{s\ in\ I} (S_r = s) = 1 \qquad \forall\, r\ in\ Dummy \tag{1}$$

$$A_o = A_{S_o} \qquad \forall\, o\ in\ O \tag{2}$$

$$T_r^p + 1 + D(r^p, r^d) \leq T_{r^d} \qquad \forall\, r \in R \tag{3}$$

$$S_{r_1} = r_2 \Rightarrow T_{r_1^d} + 1 + P(r_1^d, r_2^p) \leq T_{r_2^p} \qquad \forall r_1, r_2 \in R \tag{4}$$

$$\begin{array}{l} Not(T_{u.before} \leq T_i \leq T_{u.after}) \qquad \forall\, u\, in\, P, i \in W: \\ (u.before \neq i) \wedge (u.after \neq i) \wedge (n_i = n_{u.after} = n_{u.before}) \end{array} \tag{5}$$

$$\begin{array}{l} T_{u.before} + 1 \leq T_{u.after} \\ \qquad \forall\, u \in P : (u.before \in W^p) \wedge (u.after \in W^d) \end{array} \tag{6}$$

$$\begin{array}{l} T_{u.before} + 1 + e_{u.before} \leq T_{u.after} \\ \qquad \forall\, u \in P : (\, u.before \in W^d\,) \wedge (\, u.after\, in\, W^p\,) \end{array} \tag{7}$$

$$\begin{array}{l} (\, T_i \geq T_j + 1\,) \vee (\, T_i + 1 \leq T_j\,) \\ \qquad \forall\, i\,,\, j \in W : (\, i \neq j\,) \wedge (\, n_i = n_j\,) \end{array} \tag{8}$$

Constraints (1) ensure that the successor of each starting dummy request is either a transportation request or a dummy end request (in this case no transportation requests are assigned to that vehicle for the whole horizon). This set of constraints can also be modeled with a Global Cardinality Constraint (Regin, 1996) for better performance. 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 pick-up node to the delivery node of the request. Constraints (4) ensure that if one request is the successor of another request on the same vehicle, the AGV must have enough time to make the trip from the delivery node of the first request to the pick-up node of the second request. Constraints (5) enforce that no other task can be processed on a node between the processing at that node of two tasks linked by a priority constraint. Constraints (6) state that at least one period (duration of the pick-up) must elapse between the starting times of a pick-up and a 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.

3.2 The MIP model

The formulation of this part (finding a set of conflict-free routes) may seem counterintuitive to the CP community as they might see it as a Constraint Satisfaction Problem (CSP). A CSP formulation has been indeed tried but without much success. 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 Figure 2 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

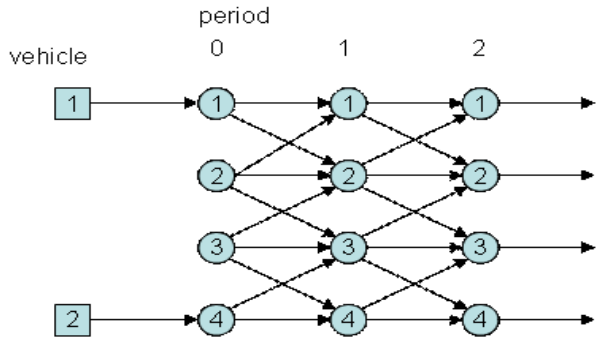

Figure 2: Time-space network

of an AGV between to 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.

As we only search for a feasible solution, any objective function can be used for the MIP. We used a constant for the objective function.

Sets and parameters:

V	set of AGVs;
Nodes	set of nodes;
Periods	set of periods;
Arcs	set of arcs;
ArcsPlus	set of all arcs (including waiting arcs);
ArcsFrom[.]	ArcsFrom[i] is the set of arcs coming from node i;
ArcsTo[.]	ArcsTo[i] is the set of arcs entering in node i;
n^v	the starting node of AGV v;
Opp[.]	array giving the opposite of each arc;
Wait[.]	array of waiting arcs;
Task	array of records describing each a task with two fields (node, earliesttime);
M	length of the horizon (number of periods);
A [·], T [·]	data obtained from the CP model;
R	set of requests. Each request is in fact a record with a pick-up and a delivery fields;
U	set of pick-up and delivery tasks.

The MIP model uses the following variables:

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

The constraints are the following:

$$\sum_{a \in ArcsFrom[n^k]} X^o_{k,a} = 1 \qquad \forall\, k \in V \tag{9}$$

$$\sum_{a \in ArcsPlus} X^t_{k,a} = 1 \qquad \forall\, t \in Periods, k \in V \tag{10}$$

$$\sum_{a \in ArcsFrom[i]} X^t_{k,a} - \sum_{a \in ArcsTo[i]} X^t_{k,a} = 0 \qquad \forall\, i \in Nodes, k \in V, t \in [1..(M-1)] \tag{11}$$

$$\sum_{k \in V} X^t_{k,a} + \sum_{k \in V} X^t_{k,Opp[a]} \leq 1 \qquad \forall\, t \in Periods, a \in Arcs \tag{12}$$

$$X^{T_{r^p}}_{A_r,\, Wait[n_{r^p}]} = 1 \qquad \forall\, r \in R \tag{13}$$

$$X^{T_{r^d}}_{A_r,\, Wait[n_{r^d}]} = 1 \qquad \forall\, r \in R \tag{14}$$

$$\sum_{k \in V, a \in ArcsTo[i]} X^t_{k,a} \leq 1 + \left(\sum_{r \in U:(t = T_r - 1) \wedge (i = n_r)} 1 \right) \times \left(\sum_{r \in U:(t = T_r) \wedge (i = n_r)} 1 \right) \qquad \forall\, i \in Nodes, t \in Periods : t \geq 1 \tag{15}$$

Constraints (9) ensure that each AGV is located at its initial position at the beginning of the horizon while constraints (10) ensure that each AGV is located at a unique position at each period. Constraints (11) are flow conservation constraints. Constraints (12) are collision avoidance constraints. Constraints (13) state that every pick-up task must be done by the right vehicle at the right time while constraints (14) are the equivalent of constraints (13) for delivery tasks. Constraints (15) 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 (15)). If the above MIP has a feasible solution then optimality for the global problem is obtained. If not, then cuts are added to the master problem which is re-solved.

Logic cuts

The logic cuts generated when a feasible routing cannot be found are based on the starting times of deliveries, the assignments of requests to AGVs and the sequence of tasks on the

same AGV. If a conflict occurs between two AGVs, at least one of them does not have enough time to fulfill its next task. Now let's describe a logic cut generated with regard to a fixed level of production delay. In addition to the parameters and sets defined in the CP model, the following ones are used:

PrevT[j]	previous starting time of task j;
PrevA[r] = k	if the request r was previously assigned to vehicle k;
PrevS[r] = v	if request v was previously the successor of request r on the same vehicle.

A logic cut consists of trying the following four options:

- 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 conjunction of constraints can be written as

$$\begin{array}{l} \sum_{r \in O} (\ A_r \ = \ PrevA[r]) \ = \ |R| \ + \ |V| \\ \text{and} \\ \sum_{r \in R} ((T_{r^d} - T_{r^p}) \geq (PrevT[\ r^d\ ] - PrevT[\ r^p\ ])) \geq 1 \end{array} \tag{16}$$

- 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

$$\begin{array}{l} \sum_{r \in O} (\ A_r \ = \ PrevA[r]) \ = \ |R| \ + \ |V| \\ \text{and} \\ \sum_{r \in R} (T_{r^d} \ = \ PrevST[\ r^d\ ]\ ) \ \leq \ |R| \ - \ 1 \end{array} \tag{17}$$

- 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

$$\begin{array}{l} \sum_{r \in R} (\ S_r \ = \ PrevS[\ r\ ]\ ) \ = \ |R| \ + \ |V| \\ \text{and} \\ \sum_{r \in O} (\ A_r \ = \ PrevA[r]\ ) \ \leq |R| \ + \ |V| \ - \ 1 \end{array} \tag{18}$$

- Generate a different solution by changing the sequences of tasks and the assignments of requests to AGVs.

$$\begin{array}{l} \sum_{r \in R} (S_r \; = \; PrevS[\, r \,]) \; = \; |R| \; + \; |V| \\ \text{and} \\ \sum_{r \in O} (A_r \; = \; PrevA[r]) \; \leq |R| \; + \; |V| \; - \; 1 \end{array} \tag{19}$$

Defining good logic cuts is a challenging task since the problem at hand is highly combinatorial i.e., 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.

4 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.

4.1 The instances

The FMS used for the experiments is presented in Figure 3. The guide path was divided into segments of 7.5 m. Assuming that the AGVs move at a constant speed of 0.5 meter per second, these segments are therefore traveled in 15 seconds, which corresponds to one time unit. The instances we used are built from the dataset of Desaulniers et al. (2003). They constructed 12 different request sets from several orders of an order book. The details of the order book can be found in Langevin et al. (1996) and Drolet (1991). A planned production schedule based on average material handling times provides the processing 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 requires 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.

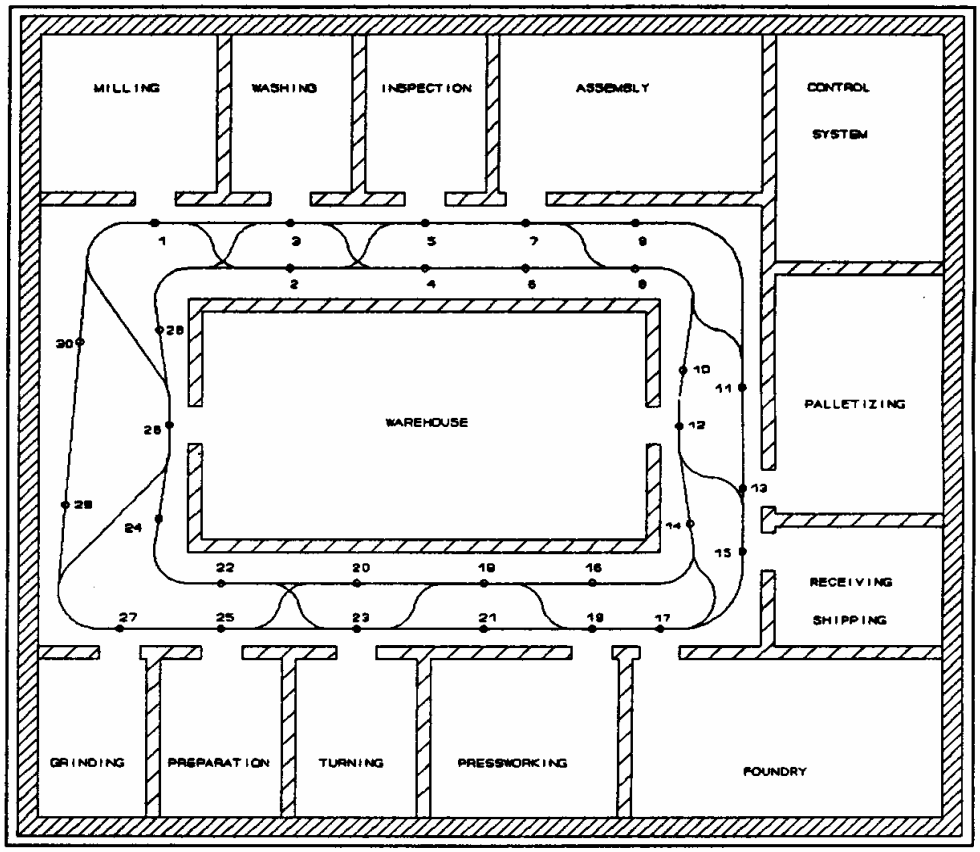

Figure 3: The FMS layout with the AGV guide path

Table 1 presents the characteristics of the fifteen 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 seconds, which corresponds to 37.5 minutes. 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 Desaulniers et al., 2003) 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.

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

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. E.g. DX $\rightarrow$ DY means that task DX precedes task DY.

4.2 Results

Experiments were performed using OPLScript (2002) on a Pentium 4, 1.5 GHz, 512 Mo (RAM) PC. We set arbitrarily the limit for the CPU time at 12 minutes which seems reasonable for a 37.5 minutes horizon. The results are presented in Table 3 to Table 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 minutes but within the horizon of 37.5 minutes (such solutions could be used 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

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		

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 minutes. 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

Table 3: 2 AGVs problem set

	First CP Model				
Instance	**Solution time**	**Choice points**	**Global solution time**	**NbCuts**	**Production delay**
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	.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

a higher number of AGVs (5 AGVs in Table 6 and 6 AGVs in Table 7) does change this picture very much. In both situations, congestion problems prevent two instances to be solved in the given 12 minutes time horizon. Furthermore, production delays are now only slightly reduced (problem 11 in Table 6 and problem 9 in Table 7). The increase 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 since we need to track the position of every AGV at every period. 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 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.

Table 4: 3 AGVs problem set

	First CP Model				
Instance	**Solution time**	**Choice points**	**Global solution time**	**NbCuts**	**Production delay**
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

Table 5: 4 AGVs problem set

	First CP Model				
Instance	**Solution time**	**Choice points**	**Global solution time**	**NbCuts**	**Production delay**
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

	First CP Model				
Instance	**Solution time**	**Choice points**	**Global solution time**	**NbCuts**	**Production delay**
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

Table 7: 6 AGVs problem set

	First CP Model				
Instance	**Solution time**	**Choice points**	**Global solution time**	**NbCuts**	**Production delay**
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

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 Ilog OPL Studio (2002). 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 go against the heuristics. 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 augments 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 A 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. The search strategy described in next section 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 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 result.

4.3 Additional features

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

- **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:
  $N_t^i = 1$ if there is a conflict on node i at period t, 0 otherwise.
  $A_t^a = 1$ if there is a conflict on arc a at period t, 0 otherwise.
  The following non constant objective function is then to minimize (the sum of conflicts on arcs and nodes):

$$\sum_{i \in Nodes\,,\, t \in Periods} N_t^i \; + \sum_{a \in Arcs,\, t \in Periods} A_t^a \tag{20}$$

These conflict detection variables are present in the following two families of constraints which respectively replace constraints 12 and 15:

$$\sum_{k \in V\,,\, a \in ArcsTo[\,i\,]} X_{k,a}^t \;\leq\; 1 \;+\; N_t^i \;+ \left(\sum_{r \in U\,:\,(t = T_r - 1\,) \wedge (i = n_r)} 1 \right) \times \left(\sum_{r \in U\,:\,(t = T_r\,) \wedge (\,i = n_r\,)} 1 \right) \qquad \forall\, i \;\in\; Nodes\,,\; t \;\in\; Periods \;:\; t \;\geq\; 1 \tag{21}$$

$$\sum_{k \in V} X_{k,\,a}^t \;+\; \sum_{k \in V} X_{k,\,Opp[\,a\,]}^t \;\leq\; 1 \;+\; A_t^a \qquad \forall\, t \;\in\; Periods\,,\; a \;\in\; Arcs \tag{22}$$

However, these would not give enough insight about the origin of a conflict and the way 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 (Figure 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

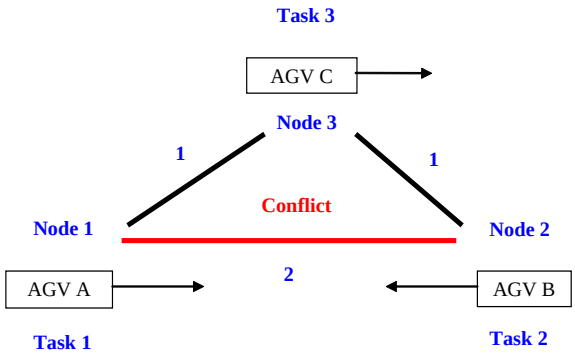

Figure 4: Why conflict detection variables are not effective

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 guaranty that the precedence and consecutiveness constraints would be satisfied *a posteriori*. If precedence constraints could also be modeled in the routing model it was not the case of consecutiveness constraints since they are not linear.

5 Conclusion

In this paper, we used an original 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 allows us to design a specific search strategy. The second sub problem is basically a CSP problem but modeled in a counter intuitive way as a MIP problem to benefit from its network substructure. The two main reasons for choosing a logic-based Benders 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, thirteen requests on a rolling horizon of 37.5 minutes. However the CP part needs to be improved to tackle problems with more AGVs, requests on a longer horizon. Our method can also be used to determine 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 work 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

Benoist, T., Gaudin, E., Rottembourg, B., Constraint Programming Contribution to Benders Decomposition : A Case Study. LNCS 2470. 8th International Conference, CP 2002 2470. 09/2002. Pascal Van Hentenryck. Springer.

Co, C.G., Tanchoco, J.M.A. , " A Review of Research on AGVs Vehicle Management," Engineering Costs and Production Economics, vol. 21, pp. 35–42, 1991.

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, vol. 15, pp. 309–331, 2003.

Eremin, A., Wallace, M. Hybrid Benders Decomposition Algorithms in Constraint Logic Programming. Lecture Notes in Computer Science. CP 2001. LNCS 2239. 2001.

Ganesharajah,T., Hall, N.G. , Sriskandarajah, C., "Design and Operational Issues in AGV - Served Manufacturing Systems," Annals of Operations Research, vol. 76, pp. 109–154, 1998.

Hooker, J. N. "'Logic-Based Methods for Optimisation'". 2000. New York, Wiley.

Hooker, J. N. "A Hybrid Method for Planning and Scheduling"'. Wallace, M. CP 2004. 2004. Springer-Verlag Berlin Heidelberg. LNCS 3258.

Hooker, J.N., Ottosson, G. , "Logic-based Benders Decomposition," Mathematical Programming, vol. 96, pp. 33–61, 2003.

Ilog OPL Studio, "'Ilog OPL Studio 3.6 Language Manual"', 2002.

Jain, V., Grossmann, I., "Algorithms for Hybrid MILP / CP Models for a class of Optimization Problems," Informs Journal on Computing, vol. 13, pp. 258–276, 2001.

King, R.E., Wilson, C. , " A review of Automated Guided Vehicle System Design and Scheduling," Production Planning and Control, vol. 2, pp. 44–51,1991.

Krishnamurthy, N.N., Batta, R., Karwan, M.H., "Developing Conflict-free Routes for Automated Guided Vehicles in a Flexible Manufacturing System," Operations Research, vol. 41, pp. 1077–1090, 1993.

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, vol. 8, pp. 246–262, 1996.

Lee, J.H., Lee, B.H., Choi, M.H., "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, vol. 28, pp. 347–358, 1998.

Maravelias, C. T., Grossmann, I. E. Using MILP and CP for the Scheduling of Batch Chemical Processes. Régin, J-C. and Rueher, M. Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. First International Conference, CPAIOR. Lecture Notes in Computer Science. LNCS 3011. 2004. Springer.

Milano, M. Constraint and Integer Programming : Toward a Unified Methodology. 33-53. 2004. Kluwer Academic Publishers. OPERATIONS RESEARCH/COMPUTER SCIENCE INTERFACES SERIES. Ramesh Sharda and Stefan Vob.

Oboth, C., Batta, R. , Karwan, M., "Dynamic Conflict-Free Routing of Automated Guided Vehicles," International Journal of Production Research, vol. 37, pp. 2003–2030, 1999.

Qiu, L., Hsu, W.-J., "A Bi-Directional Path Layout for Conflict-Free Routing of Agvs," International Journal of Production Research, vol. 39, pp. 2177–2195, 2001.

Qiu, L., Hsu, W.-J. , Wang, H. , "Scheduling and Routing Algorithms for AGVs: A survey," International Journal of Production Research, vol. 40, pp. 745–760, 2002.

Rajotia, S. ,Shanker, K. , Batra, J.L. , "A Semi-Dynamic Window Constrained Routing Strategy in an AGV System," International Journal of Production Research, vol. 36, pp. 35–50, 1998.

Regin, J. C. Generalized Arc Consistency for Global Cardinality Constraint. Proceedings of AAAI/IAAI. 1., pp. 209–215. 1996. AAAI Press/The MIT Press.

Thorsteinsson, E. S. 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 2239, pp. 16–30. 2001.