Les Cahiers du GERAD ISSN: 0711–2440

Dispatching and Conflict-Free Routing of Automated Guided Vehicles: A Hybrid Approach Combining Constraint Programming and Mixed Integer Programming

A.I. Corréa, A. Langevin
L.M. Rousseau

G–2004–21

March 2004

Dispatching and Conflict-Free Routing of Automated Guided Vehicles: A Hybrid Approach Combining Constraint Programming and Mixed Integer Programming

Ayoub Insa Corréa, André Langevin*
Louis-Martin Rousseau

Département de Mathématiques et de Génie Industriel
École Polytechnique de Montréal and GERAD
C.P 6079, Succ. Centre-ville
Montréal, Canada, H3C 2A7
**and GERAD*

{iacorrea, andre.langevin, louis-martin.rousseau}@polymtl.ca

March, 2004

Les Cahiers du GERAD
G–2004–21

Abstract

This paper reports on the on-going development of a hybrid approach for dispatching and conflict-free routing of automated guided vehicles used for material handling in manufacturing. The approach combines Constraint Programming for scheduling and Mixed Integer Programming for routing without conflict. The objective of this work is to provide a reliable method for solving instances with a large number of vehicles. The proposed approach can also be used heuristically to obtain very good solution quickly.

Keywords: Automated guided vehicles, hybrid model, constraint programming, material handling systems, routing.

Résumé

Cet article court porte sur le développement en cours d'une approche hybride sur un problème de répartition et routage de chariots automatiques dans un atelier de manufacture flexible. Notre approche combine la programmation par contraintes pour l'ordonnancement et la programmation linéaire en nombres entiers pour le routage sans conflits. L'objectif de ces travaux est de fournir une méthode fiable pour résoudre des instances comportant un grand nombre de véhicules. L'approche proposée peut être utilisée de façon heuristique pour obtenir rapidement des solutions de bonne qualité.

Mots-clé : chariots automatiques, modèle hybride, programmation par contraintes, système de manutention, routage.

1 Introduction

This study focuses on automated guided vehicles (Agvs) in a flexible manufacturing system (FMS). An Agv is a material handling equipment that travels on a network of 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 the pick-ups and deliveries are made. The guide path is composed of aisle segments with intersection nodes. The vehicles are assumed to travel at a constant speed and can stop only at the ends of the guide path segments. The guide path network is bidirectional and the vehicles can travel forward or backward. The unit load is one pallet. The number of available vehicles and the duration of loading and unloading at the P/D stations are known. As many vehicles travel on the guide path simultaneously, collisions must be avoided. There are two types of 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 on a segment in opposite directions. 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. Then, the production is scheduled. This production scheduling sets the starting time for each order. Pallets of products are moved between the cells by the Agvs. Hence, each material handling request is composed of a pickup and a delivery with their associated earliest times. At each period, the position of each vehicle must be known. Time is in fifteen second periods. A production delay is incurred when a load is picked up or delivered after its planned earliest time. The problem is thus defined as follows: *Given a number of Agvs 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.*

2 Literature review

For a recent general review on Agvs 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) algorithms for general path topology, (2) path layout optimization and (3) algorithms for specific path topologies. In our study, we work on algorithms for general path topology. Methods of this type can be divided in three categories: (1) static methods, where an entire path remains occupied until a vehicle completes its route; (2) time-window based methods, where a path segment may be used by different vehicles during different time-windows; and (3) dynamic methods, where the utilization of any segment of path is dynamically determined during routing rather than before routing as with categories (1) and (2). Our method belongs to the third category and we focus on bidirectional networks and conflict-free routing problems with an optimization approach. Furthermore we have a static job set, i.e., all jobs are known *a priori.* Krishnamurthy *et al.* (1993) proposed first an optimization approach to solve a conflict-free routing problem. Their objective

was to minimize the makespan. They assumed that the assignment of tasks to Agvs is given and they solved the routing problem by column generation. Their method generated very good solutions in spite of the fact that it was not optimal (column generation was performed at the root node of the search tree only). Langevin *et al.* (1996) proposed a dynamic programming based method to solve exactly instances with two vehicles. They solved the combined problem of dispatching and conflict-free routing. Desaulniers *et al.* (2003) proposed an exact method that enabled them to solve instances with up to four vehicles. They used slightly the same data set as Langevin *et al.*. Their approach combined 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 and the system considers at most four Agvs.

3 A Constraint Programming / Mixed Integer Programming approach

The decisions for dispatching and conflict-free routing of automated guided vehicles can be decomposed into two parts: first, the assignment of requests to vehicles with the associated schedule, then, the simultaneous routing of every vehicle. The hybrid approach presented herein combines a Constraint Programming (CP) model for the assignment of requests to the vehicles with their actual pick-up or delivery times (in order to minimize the delays) and a Mixed Integer Programming (MIP) model for the conflict-free routing. The two models are imbedded in an iterative procedure as shown in Figure 1. For each assignment and schedule found by the CP model, the MIP model tries to find conflict-free routes satisfying the schedule. CP is used to deal with the first part because it is very efficient for scheduling and, in the present case, it allows identifying easily all optimal solutions. Here optimal solutions that are equivalent in terms of value but represent different assignment might yield very different routing solution. The routing part is addressed with MIP since it can be modeled with a time-space network with some interesting sub-structures that allow fast solutions.

The method can be described in three steps:

- Step 1: find an optimal solution $x*$ (i.e., an assignment of requests to vehicles) to the CP model. Let $z*$ be the optimal objective function value (the total delay).
- Step 2: use $x*$ in the MIP model to find a conflict-free routing. If there exists any, the optimal solution to the entire model is found. Otherwise (no feasible solution found), go to step 3.
- Step 3: find another optimal solution to the CP model different from $x*$ but with the same objective function value. If there exists any, return to step 2. If no feasible

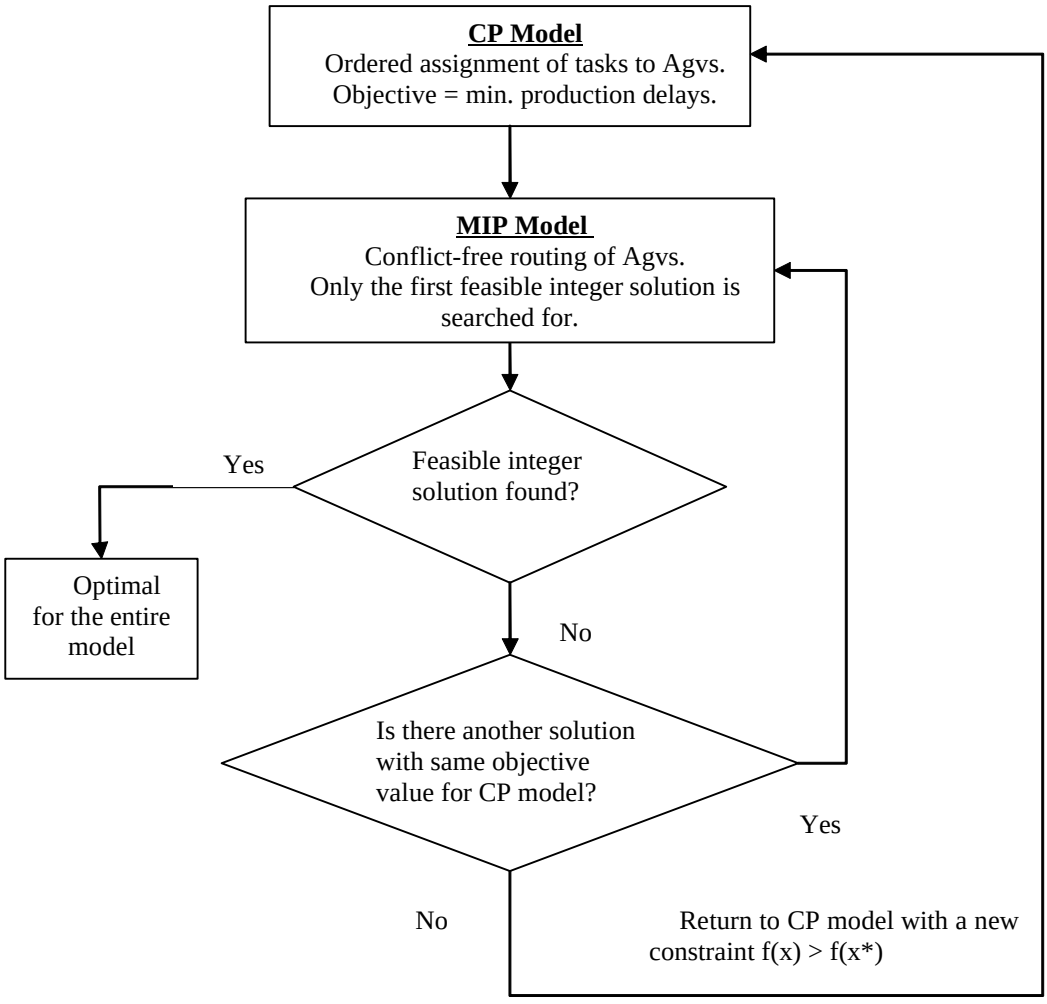
CP Model
Ordered assignment of tasks to Agvs.
Objective = min. production delays.
MIP Model
Conflict-free routing of Agvs.
Only the first feasible integer solution is searched for.
Feasible integer solution found?
Yes
Optimal for the entire model
No
Is there another solution with same objective value for CP model?
Yes
No
Return to CP model with a new constraint f(x) > f(x*)

Figure 1

solution has been found with any of the optimal solutions of the CP model, go to step 1 and add a lower bound to the objective function ($f(x) > z^*$) before solving anew the CP model. This lower bound is set to $z*$ and is always updated when returning to step 1.

3.1 The CP model

The model answers the question "Which vehicle is processing what material handling task and when?" by yielding an ordered assignment of tasks to Agvs. The total amount of delays is measured by summing the difference between the actual start time and the earliest start time of all deliveries. 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) are an approximation (a lower bound) of the actual delays.

Sets and parameters used:

DummyStartTasks*:* set of dummy starting tasks. 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.

Start[k]: starting node of Agv k.

Pickups: set of pick-up tasks.

SP [·,·]**:** length of the shortest path between a couple of nodes

Node (p): node for task p. It is used here to alleviate the notation.

nbRequests: number of requests to perform.

nbChar: number of vehicles available.

Requests: set of requests. Each request contains two fields: the pick-up task and the associated delivery task.

DummyStartRequests: set of dummy starting requests.

Inrequest: set of dummy start requests and real requests.

Pick [·]**:** pick-up field of a request.

Del [·]**:** delivery field of a request.

Each task (dummy or not) is defined by three fields: the node where the task is to be performed, the processing time at this work station and the earliest starting time of the task.

Duration [·]**:** duration of a task.

Priorities: set of couples of tasks linked by a precedence relationship (the first task is to be performed before the other).

Tasks: set of all tasks with a (mandatory) successor.

This model uses the three following variables:

- **Alloc**[i] = k if task i is performed by vehicle k. The index lower than 1 represent dummy requests.
- **Succ**[u] = v if request v is the successor of request u on the same vehicle.
- **Starttime**[j] is the start time of task j.

For each vehicle a couple of dummy tasks are created, a starting task and an end task. The starting task has the following characteristics: its node is the starting node of the Agv, its duration and earliest starting time are set to zero. We define the set **Tasks** by the set of dummy start tasks and real pickups and deliveries tasks. A request consists of a pickup and a delivery tasks. The constraints used in the model are the following:

(**1**) $\forall$ **k** $\in$ **Char**

$$\mathbf{Alloc}[\mathbf{1}-\mathbf{k}] = \mathbf{Alloc}[\mathbf{nbRequests}+\mathbf{k}] = \mathbf{k}$$

(**2**) $\forall$**r** $\in$ **DummyStartRequests**

$$\sum_{\mathbf{s\ in\ Inrequest}} (\mathbf{Succ}[\mathbf{r}] = \mathbf{s}) = \mathbf{1}$$

(**3**) $\forall$ **o** $\in$ **Tasks**

$$\mathbf{Alloc}[\mathbf{o}] = \mathbf{Alloc}[\mathbf{Succ}[\mathbf{o}]]$$

(**4**) $\forall$ **d** $\in$ **DummyStartTasks**

$$\mathbf{Starttime}[\mathbf{d}] = \mathbf{0}$$

(**5**) $\forall$ **k** $\in$ **Char**, $\forall$ **d** $\in$ **DummyStartRequests**, $\forall$ **r** $\in$ **Requests**

$$(\mathbf{Alloc}[\mathbf{d}] = \mathbf{k}) \wedge (\mathbf{Succ}[\mathbf{d}] = \mathbf{r}) \Rightarrow \mathbf{Starttime}[\mathbf{pick}[\mathbf{r}]] \geq \mathbf{SP}[\mathbf{Start}[\mathbf{k}], \mathbf{node}(\mathbf{pick}[\mathbf{r}])]$$

(**6**) **Alldifferent**(**Succ**)

(**7**) $\forall$ **r** $\in$ **Requests**

$$\mathbf{Startime}[\mathbf{pick}[\mathbf{r}]] + \mathbf{1} + \mathbf{SP}[\mathbf{node}(\mathbf{pick}[\mathbf{r}]), \mathbf{node}(\mathbf{del}[\mathbf{r}])] \leq \mathbf{Starttime}[\mathbf{del}[\mathbf{r}]]$$

(**8**) $\forall$ **r1**, **r2** $\in$ **Requests**

$$\mathbf{Succ}[\mathbf{del}[\mathbf{r1}]] = \mathbf{pick}[\mathbf{r2}] \Rightarrow (\mathbf{Starttime}[\mathbf{del}[\mathbf{r1}]] + \mathbf{1} + \mathbf{SP}[\mathbf{node}(\mathbf{del}[\mathbf{r1}]), \mathbf{node}(\mathbf{pick}[\mathbf{r2}])] \leq \mathbf{Starttime}[\mathbf{pick}[\mathbf{r2}]])$$

(**9**) $\forall$ **u** $\in$ **Priorities**

$$\mathbf{Starttime}[\mathbf{before}[\mathbf{u}]] + \mathbf{duration}(\mathbf{before}[\mathbf{u}]) \leq \mathbf{Starttime}[\mathbf{after}[\mathbf{u}]]$$

(**10**) $\forall$ **i**, **j** $\in$ **Tasks** : (**i** $\neq$ **j**) **and node(i) = node(j)**

$$(\mathbf{Starttime}[\mathbf{i}] \geq \mathbf{Starttime}[\mathbf{j}] + \mathbf{1}) \vee (\mathbf{Starttime}[\mathbf{i}] + \mathbf{1} \leq \mathbf{Starttime}[\mathbf{j}])$$

Constraints (1) ensure that a dummy starting task and its dummy end task are performed by the same Agv. Constraints (2) ensure that the successor of a dummy start request is either a real request or a dummy end request (in this case the vehicle is idle during the entire horizon but can move to avoid collisions). Constraints (3) ensure that every request and its successor must be performed by the same Agv. Constraints (4) ensure that, at the beginning of the horizon (period zero), each vehicle is located at its starting node. Constraints (5) specify that each vehicle must have enough time to reach its first pick-up node. Constraints (6) imply that the successor of each request is unique. Constraints (7) 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 (8) 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. They link the tasks that must be processed at the same nodes so that there is no overlapping. Constraints (9) enforce that, for every couple of tasks linked by precedence relationship, the first task must start and be processed before the beginning of the second task. Constraints (10) ensure that for each couple of tasks that must be performed on the same node, one must start one period after the beginning of the other.

3.2 The MIP model

For a given schedule obtained from the CP model, the MIP model allows to find whether there exists a feasible routing without conflict. This could be seen as a Constraint Satisfaction Problem since we only search for a feasible routing without conflict. However, the inherent network structure of the routing problem allows using a MIP model where only the first feasible integer solution is searched for, thus preventing a potentially time consuming search for the optimal solution of the MIP. The MIP corresponds to a time-space network which defines the position of every vehicle at anytime (see Figure 2). The original guide path network is composed of segments of length 1, 2 and 3. This network has been transformed into a directed network where all arcs are of length 1 by incorporating dummy nodes on segments of length 2 or 3. At every time period, there is a node for each intersection node (including the dummy nodes) of the guide paths. An arc is defined between two nodes of two successive time periods if the corresponding intersection nodes are adjacent on the guide path layout. Each vehicle enters the network at a given node at period 0. The time-space network model has the following characteristics:

- One unit of flow is equivalent to one vehicle present in the system.
- The total amount of entering flow at each period is equal to the total number of Agvs (busy or idle).
- At most one unit of flow can enter in a node (no collision can occur at a node).
- There is flow conservation at each node.
- An arc whose origin and destination are the same node at two successive periods corresponds to waiting at that node.

- A vehicle can move without having a task to perform, just for avoiding conflicts.

See Figure 2 for time-space network description.

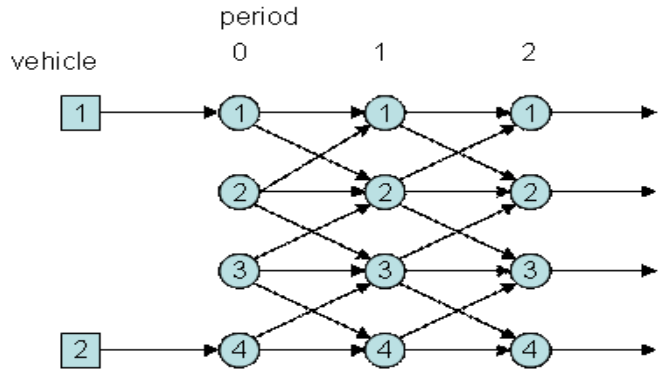

Figure 2: Description of the time-space network (MIP)

Several versions of MIPs are presently under investigation. Here, we present one that has given interesting results up to now.

Sets and parameters of the MIP model:

Char: the set of agvs

Nodes: the set of nodes

Periods: the set of periods.

ArcsPlus: the set of all arcs (including those with dummy nodes), represented as an interval of integers.

M is the length of the horizon (number of periods).

The variables **Alloc** [·] and **Starttime** [·] obtained from the CP model are used as input.

Segment[a] is a record having two fields. The first field (**Segment[a].orig**) is the origin of the arc whereas the second field (**Segment[a].dest**) is the destination of a.

The variables of the MIP model:

Y [k, t, p] = 1 if vehicle k ∈ Char is on node p ∈ Nodes at period t ∈ Periods.

Z [k, t, a] = 1 if vehicle k ∈ Char starts visiting arc a ∈ ArcsPlus at period t ∈[0 ... M-1].

The MIP model is defined as follows:

$$\textbf{Min} \sum_{\mathbf{k}\in\mathbf{Char}} \sum_{\mathbf{t}\in\mathbf{Periods}} \sum_{\mathbf{p}\in\mathbf{Nodes}} \boldsymbol{Y}[k,t,p]$$

s.t

(**1**) $\forall\, \mathbf{k} \in \mathbf{Char}$

$\mathbf{Y}[\mathbf{k}, \mathbf{Starttime}[\mathbf{2}-\mathbf{k}], \mathbf{node}[\mathbf{Task}[\mathbf{2}-\mathbf{k}]]] = \mathbf{1}$

(**2**) $\forall\, \mathbf{r} \in \mathbf{Requests}$

$\mathbf{Y}[\mathbf{Alloc}[\mathbf{r}], \mathbf{Starttime}[\mathbf{pick}[\mathbf{r}]], \mathbf{node}[\mathbf{Task}[\mathbf{r}]]] = \mathbf{1};$

(**3**) $\forall\, \mathbf{r} \in \mathbf{Requests}$

$\mathbf{Y}[\mathbf{Alloc}[\mathbf{r}], \mathbf{Starttime}[\mathbf{del}[\mathbf{r}]], \mathbf{node}[\mathbf{Task}[\mathbf{r}]]] = \mathbf{1};$

(**4**) $\forall\, \mathbf{r} \in \mathbf{Requests}$

$\mathbf{Y}[\mathbf{Alloc}[\mathbf{r}], \mathbf{Starttime}[\mathbf{pick}[\mathbf{r}]] + \mathbf{1}, \mathbf{node}[\mathbf{Task}[\mathbf{r}]]] = \mathbf{1};$

(**5**) $\forall\, \mathbf{r} \in \mathbf{Requests}$

$\mathbf{Y}[\mathbf{Alloc}[\mathbf{r}], \mathbf{Starttime}[\mathbf{del}[\mathbf{r}]] + \mathbf{1}, \mathbf{node}[\mathbf{Task}[\mathbf{r}]]] = \mathbf{1};$

(**6**) $\forall\, \mathbf{t}$ **in Periods,** $\forall\, \mathbf{k}$ **in Char**

$$\sum_{\mathbf{p}\ \mathbf{in}\ \mathbf{Nodes}} \boldsymbol{Y}[k,t,p] = \mathbf{1};$$

(**7**) $\forall\, \mathbf{k} \in \mathbf{Char}, \forall\, \mathbf{t} \in [\mathbf{0..M-1}], \forall\, \mathbf{a} \in \mathbf{Arcs}$

$\mathbf{Y}[\mathbf{k}, \mathbf{t}, \mathbf{Segment}[\mathbf{a}].\mathbf{orig}] + \mathbf{Y}[\mathbf{k}, \mathbf{t}+\mathbf{1}, \mathbf{Segment}[\mathbf{a}].\mathbf{dest}] - \mathbf{Z}[\mathbf{k}, \mathbf{t}, \mathbf{a}] \Leftarrow \mathbf{1};$

(**8**) $\forall\, \mathbf{k} \in \mathbf{Char}, \forall\, \mathbf{t} \in [\mathbf{0..M-1}], \forall\, \mathbf{a} \in \mathbf{Arcs}$

$\mathbf{Z}[\mathbf{k}, \mathbf{t}, \mathbf{a}] \Leftarrow \mathbf{Y}[\mathbf{k}, \mathbf{t}, \mathbf{Segment}[\mathbf{a}].\mathbf{orig}];$

(**9**) $\forall\, \mathbf{k} \in \mathbf{Char}, \forall\, \mathbf{t} \in [\mathbf{0..M-1}], \forall\, \mathbf{a} \in \mathbf{Arcs}$

$\mathbf{Z}[\mathbf{k}, \mathbf{t}, \mathbf{a}] \Leftarrow \mathbf{Y}[\mathbf{k}, \mathbf{t}+\mathbf{1}, \mathbf{Segment}[\mathbf{a}].\mathbf{dest}];$

(**10**) $\forall\, \mathbf{t} \in [\mathbf{0..M-1}], \forall\, \mathbf{k} \in \mathbf{Char}$

$$\sum_{\mathbf{a}\in\mathbf{ArcsPlus}} \mathbf{Z}[\mathbf{k}, \mathbf{t}, \mathbf{a}] = \mathbf{1};$$

(**11**) $\forall\, \mathbf{t} \in \mathbf{Periods}, \forall\, \mathbf{p} \in \mathbf{Nodes}$

$$\sum_{\mathbf{p}\in\mathbf{Nodes}} \mathbf{Y}[\mathbf{k}, \mathbf{t}, \mathbf{p}] \leq \mathbf{1} + \left(\begin{array}{l} \sum \mathbf{1} \\ \mathbf{r} \in \mathbf{Realtasks}: \\ \mathbf{t} = \mathbf{Starttime}[\mathbf{r}] \wedge \mathbf{p} = \mathbf{node}(\mathbf{Task}[\mathbf{r}]) \end{array} \right) * \left(\begin{array}{l} \sum \mathbf{1} \\ \mathbf{r} \in \mathbf{RealTasks}: \\ \mathbf{t} = \mathbf{Starttime}[\mathbf{r}] - \mathbf{1} \wedge \mathbf{p} = \mathbf{node}(\mathbf{Task}[\mathbf{r}]) \end{array} \right);$$

(**12**) $\forall\, \mathbf{t} \in [\mathbf{0..M-1}], \forall\, \mathbf{a} \in \mathbf{ArcPlus}$

$$\sum_{\mathbf{k}\in\mathbf{Char}} \mathbf{Z}[\mathbf{k}, \mathbf{t}, \mathbf{a}] + \sum_{\substack{\mathbf{k}\in\mathbf{Char}, \\ \mathbf{b}\in\mathbf{Opp}[\mathbf{a}]}} \mathbf{Z}[\mathbf{k}, \mathbf{t}, \mathbf{b}] \leq \mathbf{1};$$

Constraints (1) specify that every vehicle must be present at its starting node at period 0. Constraints (2-3) enforce the presence of vehicles at their task node in due time. Constraints (4-5) ensure that every vehicle stays at least one period at its task node to

load or unload. Constraints (6) ensure that every vehicle has a unique position at each period. Constraints (7) imply that if a vehicle starts visiting the origin of an arc at period t, it will visit the destination at period t+1. Constraints (8) enforce that if a vehicle is on a node at period t, it means that it has started visiting an arc (waiting arc or not) at period t-1. Constraints (9) enforce that if a vehicle is on a node at period t+1, it means that it has started visiting an incoming arc (waiting arc or not) at period t. Constraints (10) ensure that every vehicle starts running on a unique arc (real or waiting arc) at each period. Constraints (11) forbid the presence of two vehicles on the same node except the case where one vehicle is finishing its task while another is starting its task on a work station. In a certain sense, these are anti-collision constraints on nodes. Constraints (10) are anti-collision constraints on arcs: no two vehicles can travel at the same time on the same arc in opposite directions.

4 Preliminary Results

The method has been implemented in OPL Script. We compared our method to the approach of Desaulniers *et al.* and we not only gain on flexibility by using CP but we were able to solve formerly unsolved cases (their algorithm failed in two instances). We also solved some new cases with five and six Agvs (the maximum number of Agvs in Desaulniers *et al.* was four). The number of Agvs used was limited to six since it didn't make sense to increase the number of Agvs with regards to the number of requests. However, larger applications like container terminal operations use dozens of agvs and no optimization automated solutions exist. Presently, the size of the MIP model for the routing part is very large. It depends largely on the size of the horizon. Then larger time horizons will likely be more difficult to handle. We need to test our method on problems of larger number of tasks or Agvs with the idea of rolling horizons.

Our method took more computing time than that of Desaulniers *et al.* even though the computation times found are below the limit of ten minutes that we set. Our tests were done on a Pentium 4, 2.5 GHz. Desaulniers em et al. did their tests on a SUNFIRE 4800 workstation (900 MHz).

Our approach can be transformed into a heuristic version by limiting the time of each scheduling solution to 30 seconds. Experiments are planned to see if this technique can yield quickly very good solutions.

5 Conclusion

This article reports on the development of a flexible hybrid algorithm based on the decomposition into CP and MIP components. We were able to solve some formerly unsolved cases of a complex AGV dispatching and routing problem. Tests on problems with a greater number of tasks or Agvs are needed to fully evaluate the effectiveness of the proposed method. It would be interesting to analyze the impact of the number and the diversity

of precedence relationships between tasks. This research is an ongoing project as we are now working on refining the presented models and the iterative loop that guides them. As future work, an adaptation of our approach to a mining context should be very interesting due to the high level of congestion present in these problems.

References

- Desaulniers, G., Langevin, A. and Riopel, D. 2003. Dispatching and conflict-free routing of automated guided vehicles: an exact approach. *International Journal of Flexible Manufacturing Systems. To appear.*
- Krishnamurthy, N.N., Batta, R. and Karwan, M.H. 1993. Developing conflict-free routes for automated guided vehicles. *Operations Research*, 41(6), 1077–1090.
- Langevin, A., Lauzon, D. and Riopel, D. 1996. Dispatching, routing and scheduling of two automated guided vehicles in a flexible manufacturing system. *International Journal of Flexible manufacturing Systems*, 8, 246–262.
- Qiu, L., Hsu, W-J., Huang, S-Y. and Wang, H. 2002. Scheduling and routing algorithms for Agvs: A survey. *International Journal of Production Research*, 40(3), 745–760.