Deadlock Control Software for Tow Automated Guided Vehicles using Petri Nets

Carlos Rovetto[1], Elia Cano[1] and José-Manuel Colom[2]

[1] Dpt. of Computer Science and Systems Engineering (DIIS)
[2] Aragón Institute of Engineering Research (I3A)
University of Zaragoza, Spain
{carlos.rovetto,elia.cano}@utp.ac.pa, jm@unizar.es

Abstract. Factoring and warehouse distribution centers face numerous and interrelated challenges in their efforts to move products and materials through their facilities. New technologies in navigation and guidance allow true autonomy with more flexibility and resource efficiency. In this paper we investigate a complete design approach to obtain deadlock-free minimal adaptive routing algorithms for these systems. The approach is based in an abstract view of the system as a Resource Allocation System. The interconnection network and the routing algorithm elaborated by the designer, are the initial information used to obtain in an automatic way a Petri Net model. For this kind of routing algorithms, we prove that the obtained Petri Net belongs to the well-known class of S^4PR net systems, and therefore the rich set of analysis and synthesis results can be applied to enforce the liveness property of the routing algorithm.

Key words: *AGVs, Control Software, Resource Allocation Systems (RAS), Modular Models, Structural Analysis.*

1 Introduction

Nowadays, many factories and warehouse and distribution centers use *Automatic Guided Vehicles* (*AGVs*) for item transportation among workstations. The wheeled trailers are the most productive form of *AGV* for tugging and towing because they haul more *conveyor-loads* per trip than other *AGV* types. In this paper we consider a warehouse distribution center as a programmable system for conveyor-loads movement among workstations using tugger *AGV*. The problem to be investigated concerns the design of *Deadlock-Free* minimal adaptive routing algorithms for the guidance system of tuggers *AGVs*, travelling into an warehouse distribution center. We say that the routing algorithm is minimal because only routes of minimal length between two workstations are taken into account. Moreover, the routing algorithms we are considering are adaptive in the sense that the route of a conveyor-load is constructed segment by segment. The assignment of a segment to the route of a conveyor-load is done in a workstation when the first trailer try to leave the workstation towards its destination workstation.

From the methodological point of view, the design of deadlock-free minimal adaptive routing algorithms is a complex task, where the designer experience is required because deadlock states can appear. There exist several approaches

to cope with this problem [1–5]. They consider more general routing algorithms than those considered in this paper (including, for example, non-minimal routes). Because this generality, very few powerful analysis and synthesis results are available.

Our approach gives a full design cycle for *minimal adaptive routing algorithms* using Petri Nets as formal model that allows structural analysis of the liveness property of the model. Afterwards, if it is necessary, the initial routing algorithm is changed. From the point of view of software engineering, in the context of the control software for $AGVs$ systems, this paper intends to make contributions in the following directions: (a) The formalization of an *abstraction* process of the system to retain only the relevant characteristics in the study of deadlock problems in the routing software of $AGVs$. This abstraction is constructed around a minimal set of concepts – processes and resources. (b) The demonstration that for AGVs with minimal adaptive routing algorithms, the proposed abstraction process gives rise to models belonging to a well known class of Petri Nets named S^4PR, and so, we have many available results to cope with these systems. (c) *A modular methodology* to construct the models based on the specification of processes with resources that form the modules. The modules are composed by the fusion of common shared resources (segments) by different modules. This paper is organized as follows. In Section 2 an illustrative example is presented. In section 3 the proposed methodology is presented in detail. Section 4 presents the first step of the methodology consisting of the abstraction of the warehouse distribution center and the routing algorithm to retain only those aspects related to the appearing of deadlocks. Section 5 is devoted to the Petri Net model representing the Resource Allocation view of the system. This section also proves that the Petri Nets obtained for these routing algorithms belong to the class of the S^4PR nets. Section 6 presents the analysis and synthesis phases of the methodology that profit the theoretical results known for the class of S^4PR. Finally, section 7 presents some conclusions.

2 An Example

In this section, a simple example of a warehouse distribution center, will be presented. The specification of this example illustrates the typical situation in the transportation system of items. We start with a layout of the shipping areas defined by a set of workstations WS and a set of segments SG interconnecting the workstations. The connection pattern among workstations will be called the $framework$ of the warehouse distribution center. The example that we are considering is an unidirectional ring in clockwise fashion as underlying framework. There are four workstations WS=$\{w_0, w_1, w_2, w_3\}$ and they are interconnected by a set of segments SG=$\{sa_0, sa_1, sa_2, sl_1, sl_2, sl_3\}$. This warehouse distribution center is depicted in schematic way in Fig. 1.a. Observe that if a workstation has two output segments, a train can follow any of them. This decision is taken by the local *minimal adaptive routing algorithm* of the workstation. The other defining element of the warehouse distribution center is the behavior of the conveyors because a train can tow single or multiple trailers hence the length of the conveyors is variable. As the conveyors flow in pipeline fashion through

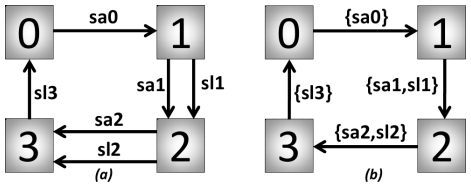

Fig. 1. a) Framework skeleton and its, b) Warehouse Graph.

the framework, these can have simultaneously allocated several segments of the framework. The first trailer of the AVG train is the **head** of the conveyors and reserve the segments to transit; the last trailer is the **tail** and release them. Traditionally, each segment supports only one conveyor at time to avoid collisions. In our example, each workstation executes, an instance of the following minimal adaptive routing algorithm parameterized by the identity of the workstation.

ALGORITHM 1 *Minimal Adaptive Routing Algorithm for workstation i.*
Input: The head trailer cl from the conveyor-load queue.
Local: $S_i \subseteq SG$, *output segments for workstation* i
$F \subseteq S_i$, *set of non-assigned output segments*
Output: The next segment to be used for cl
begin
if $(destination(cl) = i)$ **then** *store the conveyor-load cl in workstation i*
else
if ($sa_i \in F$) **then** *use* sa_i ; $F{:=}F\backslash\{sa_i\}$
else
if $((destination(cl) < i) \wedge (sl_i \in F))$ **then**
use sl_i ; $F{:=}F\backslash\{sl_i\}$
else *enqueue cl*
end if
end if
end if
end

That means the workstation knows its non-assigned output segments and the algorithm assigns, if it is possible, the output segment that the first trailer must follows in order to reach its destination. In other words, to reach a destination workstation, w_d, different to the current workstation w_i, the algorithm tries to assign the output segment sa_i if it is an output free segment of w_i. Otherwise, sl_i is assigned if this segment is an output free segment of w_i and the index d of the workstation w_d is less than the index i of w_i. This reservation is done by the head trailers. The intermediate trailers follow through the reserved segments and the tail trailer release the segments that they will be added to the set of free segments F. The design of minimal adaptive routing algorithms can lead to solutions where deadlock states can be reached. A deadlock state, in a warehouse distribution center, arise when a set of conveyor-loads are in transit to their respective destination workstations but all of them are stopped forever in intermediate workstations. They are waiting for the availability of output segments of

these intermediate workstations that have been previously assigned to conveyor-loads belonging to this set. Therefore, none of the implied conveyor-loads will reach their destination workstations. The minimal adaptive routing algorithm of our example presents this anomaly that we illustrate by means of the following deadlock state. We have four conveyor-loads, $\{cl_1, cl_2, cl_3, cl_4\}$, each one composed by more than one trailer. It is easy to verify that the state described in table 1, for the four conveyor-loads in transit, is reachable, where H and T represent the current workstations of the head and tail trailers, respectively. The rest of the columns in the table 1 represent: *Allocated segments*− segments assigned to the conveyor-load; *Destination workstation*− represents the destination workstation of the conveyor-load; *Next segment*− segment to be assigned to the head trailer according to the minimal adaptive routing algorithm. Observe that

Conveyor	**Trailers**		**Allocated**	**Destination**	**Next**
-loads	H	T	**Segments**	**Workstation**	**segment**
cl_1	w_0	w_3	sl_3	w_1	sa_0
cl_2	w_1	w_0	sa_0	w_2	sa_1
cl_3	w_2	w_1	sa_1	w_3	sa_2
cl_4	w_3	w_2	sa_2	w_1	sl_3

Table 1. Deadlock state reached in the example concerning four conveyor-loads.

all conveyor-loads are in intermediate workstations and in order to advance in the warehouse distribution center, all conveyor-loads need segments (those given by the minimal adaptive routing algorithm) that they are allocated by other conveyor-loads in the same set (compare the two columns "Allocated Segments" and "Next segment" in the table 1). On the other hand, none of the tail trailers can release segments because if some tail trailer moves ahead, it will be in the same workstation that the head trailer and this is not possible. Therefore, we have reached a deadlock state where the four classical necessary conditions for the existence of a deadlock are fulfilled. Finally, you can observe that although we are in a deadlock state, there exist two segments, sl_1 and sl_2, that they are free, and the minimal adaptive routing algorithm cannot assign these segments to the four conveyor-loads of our scenario.

3 The proposed methodology

In this paper we advocate for a methodology where, after an analysis phase of the model obtained from the framework (the interconnection network) and the minimal adaptive routing algorithm, a synthesis procedure transform the original routing algorithm to make it deadlock-free. In order to implement this methodology we will make use of Petri Net models. Therefore, the first task will be the construction of the Petri Net model that retains only those aspects related to the appearing of the deadlock states. Deadlocks appear as a consequence of the allocation of the segments by the conveyor-loads in transit in the warehouse distribution center. Therefore, we will adopt a *Resource Allocation* perspective to abstract the system *(RAS view of the warehouse distribution center)* where segments will be considered as resources, that they are used in a conservative way *(they are not created nor destroyed)* by the user processes that they are

the conveyor-load moving from a source workstation to a destination workstation. In next section, from the framework and the routing algorithm we will obtain a *Routing Graph* for each destination workstation. One of these Routing Graphs represents a transition graph where we present the reachable states of a conveyor-load, composed by more than one trailer, from a source workstation of the warehouse distribution center to the destination workstation corresponding to the Routing Graph. From these Routing Graph and the segments considered as resources, in section 5 we obtain a Petri Net that, in the case of minimal adaptive routing algorithms, belongs to the well known class of S^4PR nets. Now, using the known analysis results for this class of nets we can characterize the existence of deadlocks using a structural reasoning. The synthesis procedure is based on the methods for liveness enforcing developed by different authors [6] [7] [8]. The Fig. 2 presents in graphical form the methodology we propose for the design of deadlock-free minimal adaptive routing algorithms. In this methodology, the Petri Nets play a central role, because they are used to model the RAS view of the warehouse distribution center, and this is the reason of this paper: to present how to obtain these Petri Nets and to prove that they belong to a previously known class of Petri Nets (S^4PR), and so well studied.

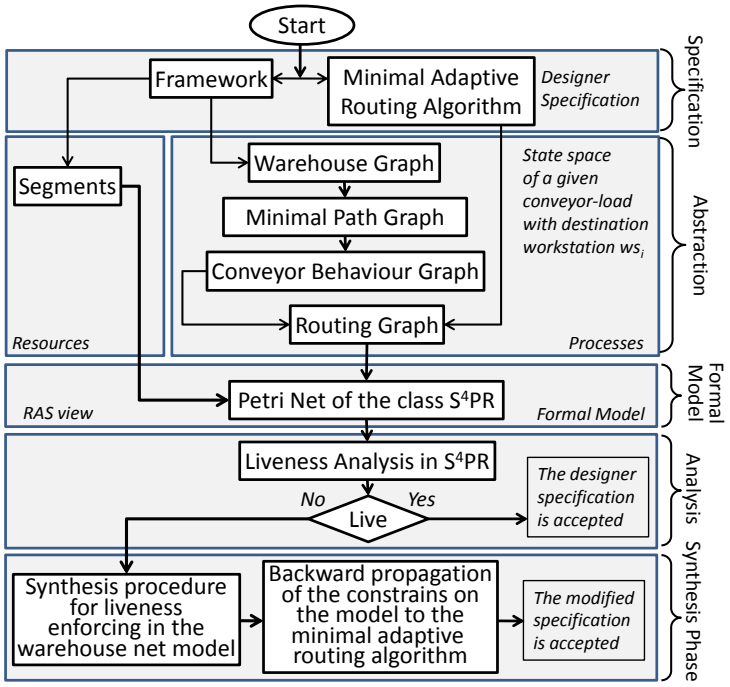

Fig. 2. Design Flow Methodology.

4 Construction of the Routing Graph

The goal of this section is to represent, step by step, the construction of the so called *Routing Graph* from the information about the framework of the warehouse distribution center and the minimal adaptive routing algorithm. This Routing Graph will represent the sequence of states that a conveyor-load must follow to reach a given destination workstation, w_i. The definition of the state concerns the set of segments that the conveyor-load is using each time. Therefore, *RG* give us the so called *Resource Allocation (RAS)* view of the warehouse distribution center. First, the framework is formalized through the *Warehouse Graph (WG)*. The WG is a labeled graph $WG{=}(W, S)$, where W is a set of vertices and S is a set of edges. The set W is equal to WS and $S{\subseteq}W{\times}2^{SG}{\times}W$, where WS is the set of vertices and SG the set of segments. An edge $(w_1, s, w_2){\in}S$ means that there exists a set of segments $s{\subseteq}SG$ from the workstation w_1 to the workstation w_2, as it is shown in the Fig. 1.b. We are considering the class of minimal adaptive routing algorithms. Therefore we will represent for each destination workstation w_i all the paths of minimal length from a workstation w_j to the destination workstation w_i. This information is captured into the *Minimal Path Graph (MPG_i)* with destination workstation w_i. Each one of these graphs is a subgraph of the $WG = (W, S)$ and it will be an acyclic directed labeled graph, $MPG_i{=}(V, E)$, where $V{=}W$, and $E{\subseteq}S$, verifying that:

1. All output arcs of w_i in WG do not belong to E.
2. The function $L_i{:}V{\rightarrow}\mathbb{N}$ is well defined: $L_i(w_i){=}0$ and $\forall w_j \neq w_i$, $L_i(w_j){=}k$, where k is the length of the minimal path from w_j to w_i in the WG.
3. All arcs $(w_1, s, w_2) \in S$ in WG, such that $L_i(w_i){+}1 \neq L_i(w_2)$, do not belong to E.

The graphs MGP_i for the example of Fig. 1 are depicted in the Fig. 3. Observe that we will have four of these graphs, one for each possible destination workstation. Each MPG_i can be seen as the set of paths that can follow a conveyor-load

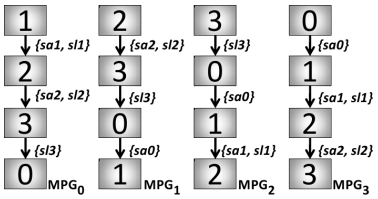

Fig. 3. Minimal Path Graph for all destination of our example.

originated in the workstation w_j with destination workstation w_i, an this path satisfy the minimality condition of the considered routing algorithm. Nevertheless, we are considering conveyor-loads with more than one trailer of length, because a conveyor-load with only one trailer cannot participate into a deadlock, since a deadlock must fulfill the *Hold and Wait* condition. Therefore in our model we must distinguish states according to the workstations where the head and tail trailers can be found. On the other hand, it is important to say that the

advancement of the head trailer from a workstation to another can be done if and only if there exists at least a segment that can be allocated for this movement. Segments, therefore are resources in our RAS view of the warehouse distribution center. If the head trailer allocates the needed resources for the movement of the full conveyor-load, the tail trailer take charge of the release operation after the use of a segment. In order to represent the states of a conveyor-load with destination workstation w_i we will construct, from the MPG_i, the so called *Conveyor Behaviour Graph (CBG)* for the destination workstation w_i, CBG_i=(Q, F), verifying that.

1. $Q \subseteq V \times V$, where $\forall w_h, w_t \in Q$, w_h=w_t or $L(w_h)$<$L(w_t)$. That is, the first component of the defined states corresponds to the workstation where the head trailer is, and the second to the workstation where the tail trailer can be found.
2. $F \subseteq Q \times \{A, R\} \times 2^{SG} \times Q$, where F will contain the following edges:
   (a) Allocation edges $((w_{h1}, w_t), A, S, (w_{h2}, w_t))$, $\forall\ w_t \in V, \forall((w_{h1}, s, w_{h2}) \in E$.
   (b) Release edges $((w_h, w_{t1}), R, S, (w_h, w_{t2}))$, $\forall\ w_h \in V, \forall((w_{t1}, s, w_{t2}) \in E$.

Obviously, CBG_i is a directed acyclic graph because MPG_i is also a directed acyclic graph. The Fig. 4 shows the Conveyor Behaviour Graph for destination workstation 0, CBG_0, corresponding to the MGP_0 of Fig. 3. Finally, to

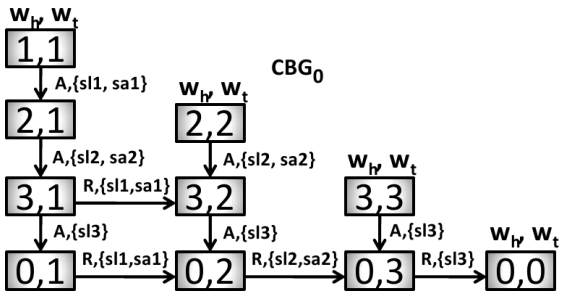

Fig. 4. Conveyor Behaviour Graph for destination workstation 0.

construct the announced Routing Graph of our warehouse distribution center we need to incorporate the information corresponding to the routing algorithm. The routing algorithm is a function R:$WS \times WS \rightarrow 2^{SG}$, such that if wc is the current workstation of the head trailer and wd the destination workstation of the conveyor-load $R((wc, wd))$ determines the output segments of wc to be allocated in order to reach the destination workstation. The model that we will construct is a possibilistic model, in the sense that from a current workstation we can have several alternative transitions, each one corresponding to a different allocated segment. Therefore in order to represent this information of the routing algorithm, from each CBG_i we will construct the so called *Routing Graph (RG)* to the destination workstation w_i, RG_i=(Q', F'), where $Q \subseteq V \times V \times SG^*$ represents the set of states in which a conveyor-load can be found.

ALGORITHM 2 *Construction of the* $RG_i = (Q', F')$
Input: $CBG_i = (Q, F)$
Output: $RG_i = (Q', F')$
begin
  next-level := $\{(w, w, \varepsilon) | (w, w) \in Q\}$
  Q' := *next-level*; $F' := \emptyset$;
  while *next-level* $\neq \emptyset$ **do**
    current-level := *next-level*; *next-level* := $\emptyset$;
    for each $(w_1, w_2, r) \in$ *current-level* **do**
      for each $((w_1, w_2), X, S, (w_3, w_4)) \in F$ **do**
        for each $c \in S$ **do**
          if $(c \in R(w_1, w_i))$ **and** $(X = A)$
            then *next-level* := *next-level* $\cup\{(w_3, w_4, r\&c)\}$;
                 $Q' := Q' \cup \{(w_3, w_4, r\&c)\}$;
                 $F' := F' \cup \{((w_1, w_2, r), X, c, (w_3, w_4, r\&c))\}$;
          endif
          if $(r = c\&t)$ **and** $(X = R)$
            then *next-level* := *next-level* $\cup\{(w_3, w_4, t)\}$;
                 $Q' := Q' \cup \{(w_3, w_4, t)\}$;
                 $F' := F' \cup \{((w_1, w_2, c\&t), X, c, (w_3, w_4, t))\}$;
          endif
        endfor
      endfor
    endfor
  endwhile
end

The state is characterized by the workstations of the head trailer and the tail trailer, respectively, and the sequence of segments that, in this state, the conveyor-load maintains allocated; $F' \subseteq Q' \times \{A, R\} \times SG \times Q'$ is the set of arcs that represents the transition from a state to another by the movement of the head trailer or tail trailer. The movement of the head trailer allocates (A) the segment specified in the arc. Observe, that now in the RG_i a path from a state (w, w, ϵ), $w \neq w_i$, that corresponds to the birth of a conveyor-load in the workstation w, to the state (w_i, w_i, ϵ), represents the routing of a conveyor-load in the warehouse distribution center from the source workstation w to the destination workstation w_i. So, in order to obtain this $RG_i = (Q', F')$ from the corresponding $CBG_i(Q, F)$, we apply the algorithm 2 *(Note: with the symbol &, in the algorithm, we denote the concatenation operation of two strings)* In the Fig. 5 the RG_0, obtained from the CBG_0 of the Fig. 4. Applying the previous algorithm, we use the solid arcs to represent the segment allocation, and the dashed arcs to represent the segment release.

5 The Petri Net Model

In the previous section we have obtained the *RAS abstraction* of the warehouse distribution center plus the considered path selection algorithm. This abstraction

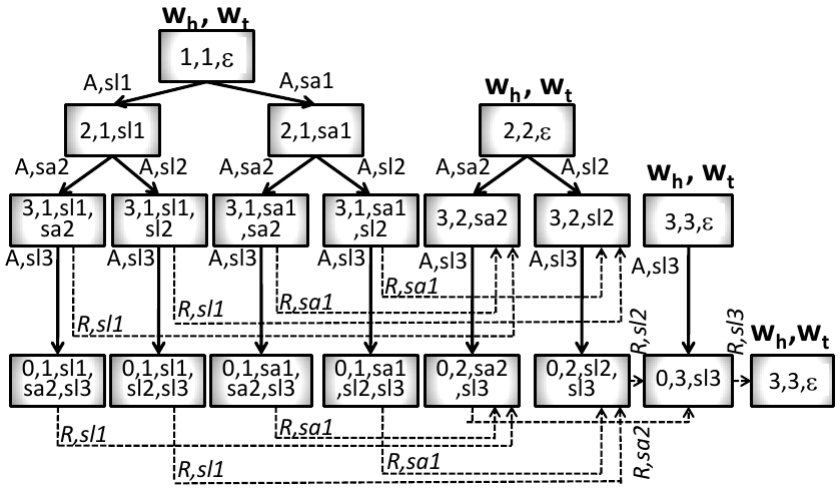

Fig. 5. Routing Graph for destination workstation 0.

is composed by the *resources*: The set SG of segments; and the set of *processes*: the set of routing processes to a destination workstation, each one represented by means of the corresponding Routing Graph. From these elements, in this section we proceed to the construction of a Petri Net integrating all processes and all resources.

First, from the $RG_i(Q', F')$, we construct the Petri Net $\mathcal{N}_i = \langle P_{0i} \cup Ps_i, T_i, F_i \rangle$ representing the state space of a conveyor-load born in the warehouse distribution center with destination workstation w_i. This construction proceeds according to the following rules.

1. Add a place to the set Ps_i for each vertex of the RG_i, $(w_1, w_2, s) \in Q'$ such that $w_1 \neq w_2$. The name of the place will be formed by the concatenation of identifiers of the workstations of the head and tail trailer, w_1 and w_2, respectively, and the sequence of segments that remain allocated for this conveyor-load and represented by s. All these places are unmarked at the initial marking M_0, because in the initial state there are not conveyor-loads in transit. We call these places *process places*.
2. Add a unique place po_i, $Po_i = \{po_i\}$, corresponding to the fusion of states of the form $(w, w, \epsilon) \in Q'$. The initial marking of this place will be equal to the maximum number of conveyor-loads that can be simultaneously in transit to the destination workstation w_i. If this number is not limited, or it its unknown, then we don't need to add a place po_i, i.e. the number of conveyor-loads with destination workstation w_i, in this network, is only limited by the available segments. These places will be called *idle places*.
3. Add a transition to the set T_i for each arc of the graph RG_i. For an arc $((w_1, w_2, s), X, c, (w_3, w_4, r)) \in F'$, the name of the transition will be $w_1\&w_2\&s\&w_3\&w_4\&r$. (The concatenation of this strings identifying the elements of the arcs).
4. For each arc $((w_1, w_2, s), X, c, (w_3, w_4, r)) \in F'$, $w_1 \neq w_2$, add an arc from the place $w_1\&w_2\&s$ to the transition $w_1\&w_2\&s\ w_3\&w_4\&r$, and an arc from this transition to the place $w_3\&w_4\&r$.

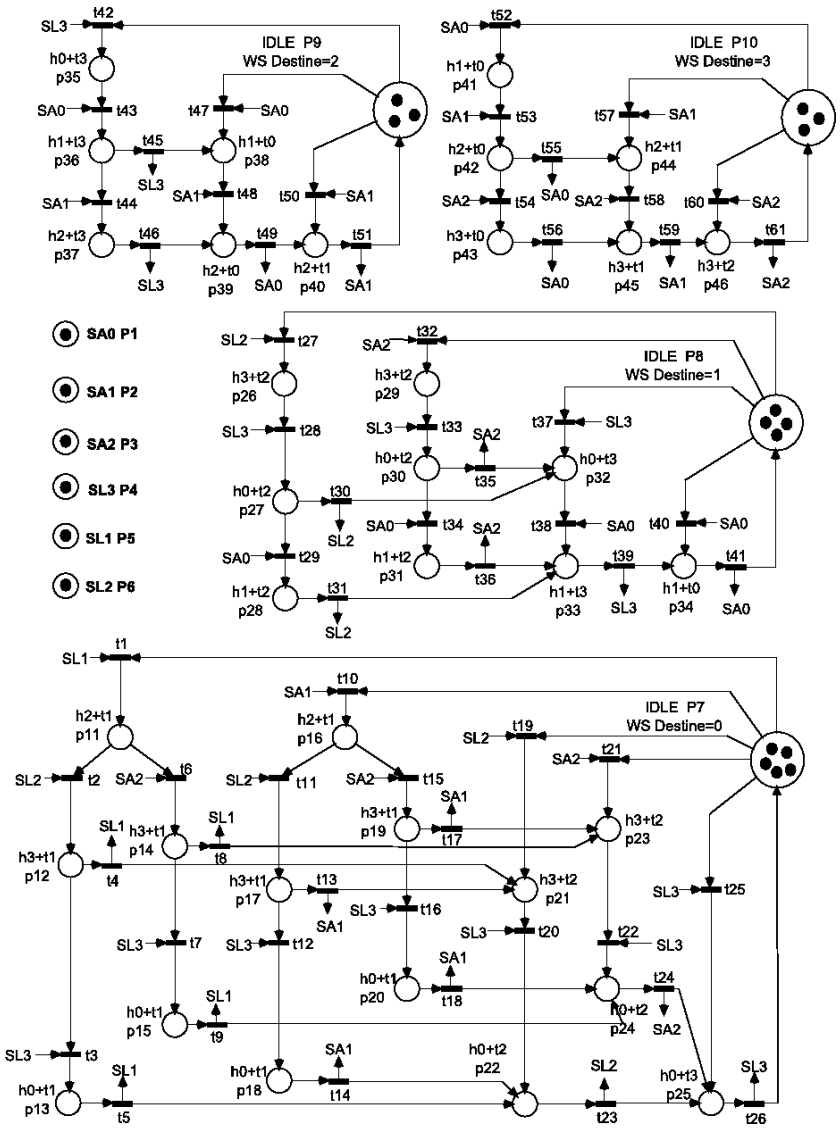

Fig. 6. Petri Net model for the example of the Fig. 1.

5. For each arc $((w, w, s), X, c, (w_3, w_4, r)) \in F'$ add an arc from the *idle* place $p0_i$ (if there exists) to the transition $w\&w\&s\&w_3\&w_4\&r$, and an arc from this transition to the place w_3, w_4, r.

Observe that the net $\mathcal{N}_i$, obtained following the rules of the preceding paragraphs, is a strongly connected state machine. In effect, by construction, each transition has only one input place and only one output place because a transition has been added for each arc in the graph RG_i, and the places correspond

to the both ends of the directed arc. Moreover, it is a strongly connected state machine because all vertex in RG_i, (w_1, w_2, s), is reachable by a path from a source vertex (w, w, ϵ), since the construction of RG_i requires that we can construct the sequence of allocated segments s from a source vertex; and from a vertex (w_1, w_2, s) always exists a path to the destination vertex (w, w, ϵ). Taking into account that place P_{0_i} represent the fusion of all vertices (w, w, ϵ) of the graph RG_i, we can conclude that the net $\mathcal{N}_i$ is strongly connected. Additionally, we can see that all circuits of $\mathcal{N}_i$ contain the place p_0, because the original RG_i is acyclic. After all these transformations we obtain a set of strongly connected state machines $\mathcal{N}_i$, each one corresponding to a different destination workstation in the warehouse distribution center. The last step to obtain the RAS view of the warehouse distribution center is the addition of the resources, that, in this case, they are the segments connecting the workstations, and their integration with to the state machines. This can be done state machine by state machine and constructing the full model by fusion of the resource places with the same name. That is, we are constructing the model in modular way. The two steps to be applied are:

1. Add a place p_c to the set P_R for each segment $c \in SG$ of in the warehouse distribution center. The initial marking of this place will be equal to the maximum number of trailers that can be in transit simultaneously in the segment. *(Normally. it will be equal to one representing tha availability of the segment).*
2. For each arc of the graph RG_i of the form $((w_1, w_2, s), A, c, (w_3, w_4, r)) \in F'$, add an arc from the place p_c, (resource place representing the segment c) to the transition $w_1\&w_2\&s\&w_3\&w_4\&r$. This arc, in the Petri Net, will represent the allocation of the segment c. For each arc of the graph RG_i of the form $((w_1, w_2, s), R, c, (w_3, w_4, r)) \in F'$ add an arc from the transition $w_1\&w_2\&s\&w_3\&w_4\&r$ to the place c. This arc in the Petri Net represents the release of the segment c.

We denote this net by $\mathcal{N}_i^R$, representing the routing of the conveyor-loads to the destination workstation w_i, and the competition for the resource/segments. The full model is obtained from the different $\mathcal{N}_i^R$ by the fusion of the resource places (*segments places*) with the same name that appear in different $\mathcal{N}_i^R$. The Fig. 6 represents, in an schematic way, the full Petri Net corresponding to the example in Fig. 2. In this Fig. 2 the names of places and transitions have been simplified in order to maintain readable. In order to identify the states of the conveyor-loads with the places that represent them, we have used the simplified notation h_i+t_j; that it means that the head trailer is in workstation i and the tail trailer is in workstation j.

The final part of this section is devoted to prove that the obtained Petri Nets for warehouse distribution centers with minimal path selection algorithms belong to the subclass of Petri Nets named S^4PR [6][9]. In order to do that we recall the basic definitions of this class of nets.

Definition 1 (The class of S^4PR nets) *Let $I_{\mathcal{N}}=\{1,2,...,m\}$ be a finite set of indices. An S^4PR net is a connected generalised self–loop free Petri net $\mathcal{N}=\langle P,T,\mathbf{C}\rangle$ where:*

1. *$P=P_0\cup P_S\cup P_R$ is a partition such that:*
   (a) $P_S=\bigcup_{i\in I_{\mathcal{N}}}P_{S_i}$, $P_{S_i}\neq\emptyset$ and $P_{S_i}\cap P_{S_j}=\emptyset$, for all $i\neq j$.
   (b) $P_0=\bigcup_{i\in I_{\mathcal{N}}}\{p_{0_i}\}$.
   (c) $P_R=\{r_1,r_2,\ldots,r_n\}$, $n>0$.
2. *$T=\bigcup_{i\in I_{\mathcal{N}}}T_i$, $T_i\neq\emptyset$, $T_i\cap T_j=\emptyset$, for all $i\neq j$*
3. *For all $i\in I_{\mathcal{N}}$, the subnet $\mathcal{N}_i$ generated by $P_{S_i}\cup\{p_{0_i}\}\cup T_i$ is a strongly connected state machine, such that every cycle contains p_{0_i}.*
4. *For each $r\in P_R$ there exists a minimal P–Semiflow, $\mathbf{y}_r\in\mathbb{N}^{|P|}$, such that $\{r\}=||\mathbf{y}_r||\cap P_R$, $\mathbf{y}_r[r]=1$, $P_0\cap||\mathbf{y}_r||=\emptyset$, and $P_S\cap||\mathbf{y}_r||\neq\emptyset$.*
5. *$P_S=\bigcup_{r\in P_R}(||\mathbf{y}_r||\setminus\{r\})$.*

Each place p_{0_i} is called *idle place*. Places of P_R are called *resource places* being unique for the whole model. The Places of P_S are called *process places*. This definition must be completed with the definition of the *acceptable initial markings*. Initial markings represent no activity in the system, allowing the routing of each conveyor-load in isolation.

Definition 2 *Let $\mathcal{N}=\langle P_0\cup P_S\cup P_R,T,\mathbf{C}\rangle$ be a S^4PR net. An initial marking $\mathbf{m_0}$ is acceptable for $\mathcal{N}$ if and only if: (1) $\forall i\in I_{\mathcal{N}}$, $\mathbf{m_0}[p_{0_i}]>0$. (2) $\forall p\in P_S$, $\mathbf{m_0}[p]=0$. (3) $\forall r\in P_R$, $\mathbf{m_0}[r]\geq\max_{p\in||\mathbf{y}_r||\setminus\{r\}}\mathbf{y}_r[p]$.*

From the previous definitions and the procedures described in the sections 4 and 5 to obtain the Petri Net model of an warehouse distribution center the following result can be easily verified.

Proposition 1 *Given an warehouse distribution center specified by means of a framework and a minimal adaptive routing algorithm, the Petri Net model obtained through the procedure described in sections 4 and 5, belongs to the class of S^4PR net systems.*

Proof (Sketch of the proof). In the section 5, after the rules to obtain the Petri Nets $\mathcal{N}_i$ from the corresponding Routing Graph RG_i, we have proven that each $\mathcal{N}_i$ is a strongly connected state machine, and for all $\mathcal{N}_i$,$\mathcal{N}_j$, $i\neq j$, they are disjoint net systems. We have also proven that every cycle of each strongly connected state machine $\mathcal{N}_i$ contains P_{0_i}. Therefore, to complete the proof we only need to prove the existence of a unique *p-semiflow* $\mathbf{y}_r\in\mathbb{N}^{|P|}$ for each resource r. But this is very easy to proof because from each transition where the resource place r inputs *(the resource is allocated)*, there exists a unique path, in the strongly connected state machine, to reach each transition where r outputs *(the resource is released)*. Moreover, all transitions where r is an output place in the state machine $\mathcal{N}_i$ are connected by means of a minimal path from some transition where r is an input place. Therefore, the resource r plus all the process places defining the minimal paths connecting the output transitions of r and the input transitions of r form the *p-semiflow* that it is unique because we are dealing with the nets $\mathcal{N}_i$ that they are state machines.

Observe that the previous result is also true for non-regular frameworks because we are considering in an explicit way the paths to a destination workstation. Therefore, non regularity does not affect the final Petri Net. Nevertheless, non-minimality of the path selection algorithms can lead to more general class of nets than the S^4PR in the case of existence of cycles in the followed route by some conveyor-load. Once we have characterized the type of nets we can obtain, we can use the developed theory for S^4PR, trying to interpret these results from the point of view of the warehouse distribution centers, in the next section. In some cases we will see that we arrive to some negative results.

6 The Analysis and synthesis phase

The Petri Net model obtained in the previous section belong to the S^4PR class. Therefore, we can take advantage of this property and use the theoretical results about the liveness characterization in S^4PR. One of this results is presented in the following theorem.

Theorem 2 ([6]) *An S^4PR, $\langle \mathcal{N}, \mathbf{m_0} \rangle$, is non–live if and only if there exists a marking $\mathbf{m} \in \mathrm{RS}(\mathcal{N}, \mathbf{m_0})$ such that the set of $\mathbf{m}$–process–enabled transitions is non–empty and each one of these transitions is $\mathbf{m}$–resource–disabled.*

This characterization is a state based characterization. The interpretation in terms of the warehouse distribution center is very easy. A token in a process place of the state machine $\mathcal{N}_i$ represent a conveyor-load in an intermediate workstation with destination workstation w_i. That is, is a conveyor-load in transit. The theorem 2 says that if all conveyor-loads in transit cannot advance because there is no an available segment to advance *(each one of these transitions is not enabled because an input resource place is empty)*, this situation characterizes a deadlock state: none of these conveyor-loads will arrive to its destination workstation because they are stopped forever in the current process places. In [6], verification procedures of the characterization stated in this theorem are presented. They are based in Integer Linear Programming Techniques.

An equivalent characterization to the previous one is based in the Petri Net concept of siphon. A siphon is a set of places that if they become a set of empty places, they remain empty forever *(these is a structural definition of siphon but we prefer to present the deep reason for the appearing of deadlocks in this class of nets)*. Therefore, all output transitions of the places of the empty siphon will be dead forever because at least an input place *(that belong to the siphon)* is empty forever. The presence of one of this siphons in the net is potentially bad because this siphon can become an empty siphon. The verification procedures search for a siphon and a reachable marking under which the siphon is empty. Empty siphons represent a generalization of the circular waits, because in a siphon we can find an intricate structure of superposed cycles of empty resources. For the Petri Net in Fig. 6, you can find the two following bad siphons D_i=$\{p_1, p_2, p_3, p_4, p_{13}, p_{15-to-20}, p_{22}, p_{24}, p_{25}, p_{28}, p_{30}, p_{31}, p_{33}, p_{34}, p_{36}, p_{37}, p_{39}, p_{40}, p_{42}, p_{43}, p_{45}, p_{46}\}$ and D_j=$\{p_1, p_2, p_3, p_4, p_6, p_{13}, p_{15}, p_{16}, p_{17}, p_{18}, p_{19}, p_{20}, p_{22}, p_{24}, p_{25}, p_{27}, p_{28}, p_{30}, p_{31}, p_{33}, p_{34}, p_{36}, p_{37}, p_{39}, p_{40}, p_{42}, p_{43}, p_{45}, p_{46}\}$. The deadlock state described in section 2 corresponds to the reachable marking written as a symbolic sum $m_r = p_5 + p_6 + 5 \cdot p_7 + 2 \cdot p_8 + \cdot p_9 + 2 \cdot p_{10} + p_{29} + p_{32} + p_{38} + p_{44}$. The

reader can easily verify that the siphon D_i is insufficiently marked or he/she can verify the m_r satisfy the conditions of the theorem 2. Therefore, we conclude that the proposed path selection algorithm is not deadlock-free. After the previous analysis phase, the theory of S^4PR nets gives results and methods to enforce the liveness in the case of nets presenting deadlock states. These techniques transform the initial Petri Net model in such a way that deadlock states become not reachable. In some sense, they correspond to deadlock prevention techniques. We can incorporate this phase because we are using Petri Nets as formal model and they belong to the subclass of S^4PR. The known synthesis approaches enforcing liveness work on the bad siphons that can be found in the Petri Net model. These techniques can be classified into two groups.

1. **Centralized Approach:** [6][9] These techniques compute a place for each bad siphon preventing that the siphon becomes empty. This new place is of the same category that the resource places, and so it is said that the synthesis problem is solved by using virtual resources that they are implemented as a centralized monitors in the central software. In the case of the Petri Net of the Fig. 6 we need three places to make live the net. In fact, in some cases, to take the decision to allocate the virtual resource/segment in a local workstation we can need coordinate the local path selection algorithm with other local routing algorithms.
2. **Distributed Approach:**[10]. Previous limitations are solved developing a distributed control policy using the so called *swap virtual segments*.

All these methods are iterative, but the performed transformations maintain the transformed Petri Net inside the class of S^4PR nets.

7 Conclusions

The design of deadlock-free minimal adaptive routing algorithms for warehouse distribution centers is a complex and tedious task, for which the current methodologies, in many cases, only supply trial and error procedures. The assistance to the designer is very small in order to fix the problem in the proposed algorithm. In this paper we propose a methodology oriented to the design of deadlock-free minimal adaptive routing algorithms trying to cope with all phases of the design. The first step in this methodology consists of the *abstraction* of the system in order to retain only the elements of the system allowing the study of the appearing of deadlocks. These elements are the segments of the warehouse distribution center, that they are seen as the resources for which the routing processes compete to send conveyor-loads to destination workstations. The other elements are the routing processes itself that represent the routing sequence through the framework according to the routing algorithm. The result of this abstraction process is formalized by means of a Routing Graph for each possible destination workstation. From the Routing Graphs and the segments we have obtained Petri Nets that, for the class of routing algorithms that we are considering, belong to the class of S^4PR. Therefore, we profit that the class of S^4PR is a well studied subclass of Petri Nets and using the known results we can proceed with the analysis and synthesis phases of our methodology. So, the deadlock-free property in the warehouse distribution center correspond to the liveness-property in our Petri

Net model. The analysis of this liveness property can be done by two alternative characterizations that have a good interpretation at the level of warehouse distribution center. Algorithms and methods to verify the property can be found in [6]. In the case of non-liveness, there exist methods to enforce the liveness property based in the addition of places that can be interpreted in terms of Petri Net model as centralized software monitors.

References

1. Fanti, M.: A deadlock avoidance strategy for AGV systems modelled by coloured Petri nets. In: Discrete Event Systems, 2002. Proc. Sixth Int. Workshop on. (2002) 61 – 66
2. Wu, N., Zhou, M.: Resource-oriented Petri nets in deadlock avoidance of AGV systems. In: Robotics and Autom., 2001. Proc. 2001 ICRA. IEEE Int. Conf. on. Volume 1. (2001) 64 – 69 vol.1
3. Wu, N., Zhou, M.: Modeling and deadlock control of automated guided vehicle systems. Mechatronics, IEEE/ASME Transactions on **9**(1) (2004) 50 –57
4. Wu, N., Zhou, M.: Modeling and deadlock avoidance of automated manufacturing systems with multiple automated guided vehicles. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on **35**(6) (2005) 1193 –1202
5. Wu, N., Zhou, M.: Shortest routing of bidirectional automated guided vehicles avoiding deadlock and blocking. Mechatronics, IEEE/ASME Transactions on **12**(1) (2007) 63 –72
6. Tricas, F.: Analysis, Prevention and Avoidance of Deadlocks in Sequential Resource Allocation Systems. PhD thesis, Zaragoza. España, Departamento de Ingeniería Eléctrica e Informática, Universidad de Zaragoza (2003)
7. Tricas, F., Colóm, J., Ezpeleta, J.: A Solution to the Problem of Deadlocks in Concurrent Systems Using Petri Nets and Integer Linear Programming. In Horton, G., Moller, D., Rude, U., eds.: Proc. of the 11th European Simulation Symp., Erlangen, Germany, The society for Computer Simulation Int. (1999) 542–546
8. Park, J., Reveliotis, S.A.: Enhancing the Flexibility of Algebraic Deadlock Avoidance Policies Through Petri Net Structural Analysis. In: Proc. of the 2000 IEEE Int. Conf. on Robotics and Autom., San, Francisco, USA (2000) 3371–3376
9. Tricas, F., García-Vallés, F., Colóm, J.M., Ezpeleta, J.: A Petri Net Structure-Based Deadlock Prevention Solution for Sequential Resource Allocation Systems. In: Proc. of the 2005 IEEE Int. Conf. on Robotics and Autom., Barcelona, Spain (2005) 272–278
10. Rovetto, C., Cano, E., Colóm, J.: Liveness Enforcing Methods for Resource Allocation Distributed Systems using Petri Nets, Research Report RR-056-09. Depto de Informática e Ingeniería de Sistemas., University of Zaragoza (2009)

Acknowledgement

This work has been partially supported by the European Community's Seventh Framework Programme under project DISC (Grant Agreement n. INFSO-ICT-224498). This work has been also partially supported by the project CICYT-FEDER DPI2006-15390.