Deadlock- and Collision-Control in Robotic Flow Shops

Claudia Fiedler and Wolfgang Meyer

***Abstract*— We describe a resource oriented modelling method for robotic flow shops and exemplify it by a galvanic plant with 25 tanks and 2 transporting units. Deadlocks due to loops in the process plan and collisions among the two transporting units are the major problems which occur in this kind of discrete parts manufacturing. To avoid deadlocks we describe an event dependent and a capacity dependent strategy. We also describe a collision avoidance strategy for two robots with overlapping handover areas. Both strategies are implemented in a Petri net based simulation environment. The off line tool is used for the interactive design of schedules and routings for robotic flow shop transport systems.**

***Keywords*— Timed Petri Nets, Manufacturing, Hoist Scheduling**

I. Problem Statement

THE increasing use of flexible production environments poses high demands on production planners. Besides the necessity to optimize the stationary production process over a long period it is more and more important to be able to change quickly and efficiently between different production modes. For plants with automated transport systems we have to find optimum control sequences for the transporter to meet the requirements. Therefore we have developed a simulation model to find control sequences both for stationary and for flexible production environments. We extend the model described in [1] by a capacity dependent deadlock avoidance strategy and a collision avoidance strategy for two transport robots and discuss results of a real plant application.

The application considered is a line of basins containing chemical, electrolytic or rinsing bathes served by one or more transporters (Fig.1). The plant consists of m machines (or basins, or tanks) M_1, …, M_m, an input station M_0 and an output station M_{m+1} sometimes combined at the same place.

The input station houses a set of parts J. Each part has to be processed according to its process plan, the list of the operation times o_i, ($i \in M$) at the machines and the transport times t_{ij} ($i, j \in M$) between them. The operation times o_i of part J are kept in intervals $\left[l_i^J, u_i^J\right]$ with a lower bound l_i^J and an upper bound u_i^J. If the upper bound is equal to the lower bound, we speak of a no-wait condition. The upper bound can be infinity, too. There are one or more transporters T_n operating within defined areas on the same or on different tracks. The travel times δ_{ij} can be constant, additive or Euclidean. Additive travel times follow the triangle equality and Euclidean travel times follow the triangle inequality. They are symmetric ($\delta_{ij}=\delta_{ji}$) and zero from a machine to itself ($\delta_{ii}=0$). The transport times t_{ij} between the operations o_i are the sum of travel times δ_{ij} and a constant time needed for loading and unloading the part. The parts in the input station can be of the same type or of different types. The goal is to minimize the cycle time V for the transporter sequence which equals the job release time in case of identical parts (or products, or jobs), or to minimize the throughput time in case of different part types.

Claudia Fiedler and Wolfgang Meyer are with the Institute of Automation, Hamburg University of Technology, 21071 Hamburg, Germany
(e-mail: claudia.fiedler@tu-harburg.de; w.meyer@tu-harburg.de)

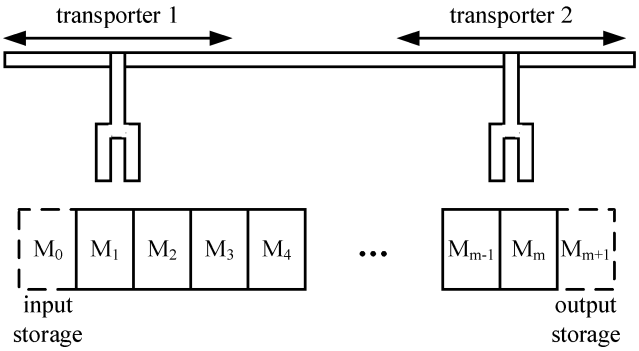

Fig. 1 Layout of the plant

II. Robotic Flow Shop Scheduling: State of the Art

We address the Hoist Scheduling Problem (HSP) here as a special case of a robotic flow shop [2] [3]. An overview of different kinds of HSPs is given by Manier and Bloch in [7]. They extend the Graham notation for job and flow shop scheduling [3][8] to HSPs.

In HSP, the operation times per machine are not fixed but decision variables whose values must be selected from a given range called the *interval processing time*. The transporters have Euclidean travel times [2], and loaded transporters are not allowed to wait (*no-wait condition)*. The NP-completeness is proven by Crama and Klundert [4]. Phillips and Unger [5] solved the monocyclic case with integer programming. Rodozek and Wallace used a hybrid constraint logic programming (CLP) and mixed integer programming (MIP) algorithm [6].

In [9] we presented a process centered modelling method for the HSP according to the A-path method of Zhou and DiCesare [10]. Additionally, we developed a resource centered model for one hoist in [1]. In this paper, we extend our previous investigations for multiple hoists. In such

applications, collisions among several hoists (or cranes, or transporting units) pose a major problem to system scheduling and deadlock control. Before we design suitable deadlock and collision avoidance algorithms in Sections IV and V, we summarize the fundamental properties of the used scheduling models in the next section.

III. Process vs. resource centered Modelling

The general structure of the scheduling model is shown in Fig. 2. The parts or jobs are modelled as process tokens with the processing times of operations as attributes. They are released to the request generator with constant release times V.

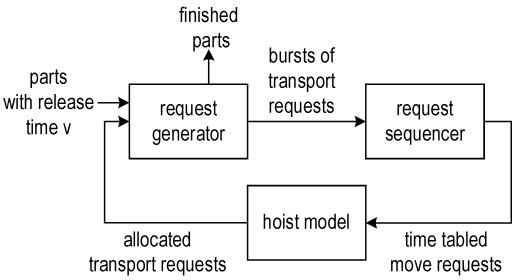

Fig. 2 Scheduling model

The request generator (which is the model of the plant) sends transport requests to the request sequencer if the state of the model has changed because of a finished transport operation and the starting of a tank operation. According to the predetermined priority, the sequencer decides which of the transport requests is fulfilled next as soon as the transporter is available. Then the time tabled request releases a transporter move if the transporter is not at the needed place. Finally, the allocated request causes a transport operation and a new transport request.

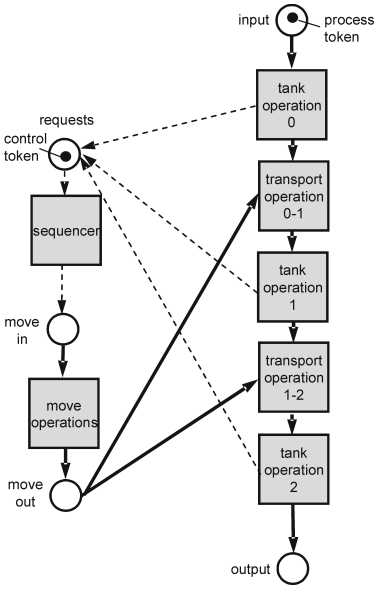

Fig. 3 Process centered model (A-path model)

The start value of V for the first simulation run is the sum of the maximum operation times of the bottleneck tank and the transport times to and from the tank. If a cyclic behavior can not be achieved or if the operation times exceed the upper bounds of the given intervals, the release time V will be increased by a small amount and the simulation starts again until the constraints as prescribed by the (Petri net) model are fulfilled.

In Fig.3 the simplified process centered Petri net model is shown. The A-path as the sequence of tank and transport operations for one process (or job) are on the right and the move operations and the sequencer are on the left side.The signal flow of the requests takes place along the dashed lines. The process flow occurs along the bold lines. Just the processing times of operations can be changed by input data not the sequence of operations.

In flexible manufacturing environments there is a fast change in product types, however. To find sequences for lot switching or for new products, the process centered model is unsuitable because for each new process plan (or product) a new A-path has to be implemented. As in improvement in modelling flexibility, the resource centered model is shown in Fig.4. Compared to the process centered model, now the resources are at the primary focus of interest instead of processes or A-paths. The signal flow is similar to the process centered model but the process flow is composed of single operation elements using the corresponding resources. Therewith flexible A-paths are possible. The flexibility is reflected in Fig.4 in the number of process flow connections, too. In the process centered model there is just one way for the parts whereas in the resource centered model the processes can be composed in any order. In [12] the Compact Modelling as a similar concept for the Job Shop Scheduling Problem (JSP) is described and the effects on the number of Petri net elements are calculated.

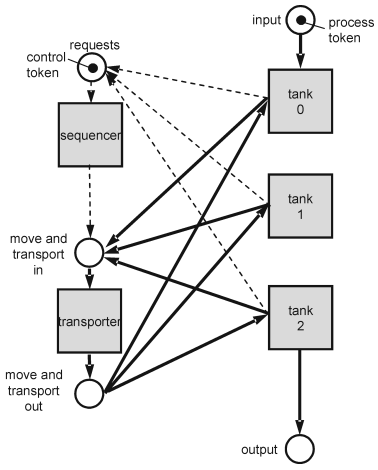

Fig. 4 Resource centered model

IV. Deadlock prevention

In Discrete Event Systems with loops in the process plans deadlocks may occur. We therefore need a deadlock prevention algorithm in the resource centered model to enable the model to simulate processes with loops in the process plan. The idea is to prevent the last change in the state of the plant which closes a deadlock. The following example may illustrate the algorithm:

Given are three resources a_1, a_2 and a_3. Each resource has capacity one, i.e., each resource can handle just one of the processes. For process P1 the actual resource may be a_1, for process P2 a_2 and for P3 a_3. Then these four combinations for the following two resources for the three processes are possible:

$$\text{P1}=\begin{pmatrix}(a_1a_2a_1)\\(a_1a_3a_1)\\(a_1a_2a_3)\\(a_1a_3a_2)\end{pmatrix};\ \text{P2}=\begin{pmatrix}(a_2a_1a_2)\\(a_2a_3a_2)\\(a_2a_1a_3)\\(a_2a_3a_1)\end{pmatrix}\ \text{and}\ \text{P3}=\begin{pmatrix}(a_3a_1a_3)\\(a_3a_2a_3)\\(a_3a_1a_2)\\(a_3a_2a_1)\end{pmatrix}.$$

Each combination of the three processes P1, P2 and P3 is a deadlock with either size 2 if two processes or resources are involved or size 3 for three involved processes. For example, if P1=(a_1 a_2 a_1) and P2=(a_2 a_1 a_2) then there is a deadlock because P1 blocks the next tank of P2 and P2 uses the next tank of P1. A deadlock with size 3 occurs if P1=(a_1 a_2 a_1), P2=(a_2 a_3 a_2) and P3=(a_3 a_1 a_3), because there are three involved processes and for each of the three processes there exists a process which uses the next resource. That means there is a deadlock if:

Let **P** be the set of processes in the plant

$$\mathbf{P}=\{\text{P1}\ldots\text{Pn}\};\quad n\in\mathbb{N} \tag{1}$$

and each P_i consists of the actual and the next operation,

$$P_i=\left(a^i_{\text{actual}}\ a^i_{\text{next}}\right);\ i=1\ldots n \tag{2}$$

then there exists a subset **Q** of **P**

$$\mathbf{Q}\subseteq\mathbf{P};\ \mathbf{Q}=\{Q_1\ldots Q_m\};\ m\in\mathbb{N} \tag{3}$$

with

$$\mathbf{A}^{\mathbf{Q}}_{\mathbf{actual}}=\mathbf{A}^{\mathbf{Q}}_{\mathbf{next}} \tag{4}$$

where $\mathbf{A}^{\mathbf{Q}}_{\mathbf{actual}}$ is the set of the actual resources occupied by the processes of **Q** and $\mathbf{A}^{\mathbf{Q}}_{\mathbf{next}}$ the set of the next resources used by the processes of **Q**.

In the implementation of the resource centered model we have to prohibit the transport of the last job to that resource which leads to Eqn.(4) and results in a deadlock. We call it an *event dependent deadlock prevention*. In the worst case it needs a lot of calculation time to decide if a transport operation results in a deadlock because we have to look ahead at least until each process in the plant took one step forward. Therefore we implemented *process dependent capacity restrictions*, instead. The prevention algorithm works as follows:

Table 1. Process dependent capacity

operation	0	1	2	3	4	5	6	7	8	9	10
tank	**1**	**8**	5	**3**	4	2	**8**	**3**	6	7	**1**
area	S1					S2			S3		
capacity	1					2			2		

In Table 1 a process with 11 operations and 8 resources (tanks) is given as an example. There are 3 loop resources in the process plan: tank 1, tank 3, and tank 8. The resources with loops are marked as bold numbers. The idea is to restrict the capacity of the area between two loop resources to the number of non-loop resources. If there is no non-loop resource between bold numbers, they are grouped together and considered as one loop resource. Table 1 shows 3 ranges S1 to S3 with the capacities allowed. With this entrance restriction for new tokens (or jobs) it is guaranteed that it is always possible to move a part (or job) from a loop resource to a non-loop resource.

In Fig. 5 the simplified model with deadlock prevention is displayed. The decision of the sequencer for one out of several concurrent processes is based on the due date of the actual tank operation. Other priority rules are possible as well. The timetabled request which is sent from the sequencer to the deadlock prevention module contains information about the operation number. The deadlock prevention algorithm then decides if the capacity of the area under consideration will be exceeded when the process will be transported to the next tank, or not. If so, the deadlock prevention module sends an inhibit signal to the sequencer for this process and tries the next one. Every time the state of the request generator changes as caused by a transport operation, all the inhibited requests stored in the sequencer module are tested whether the danger for deadlock still holds or not.

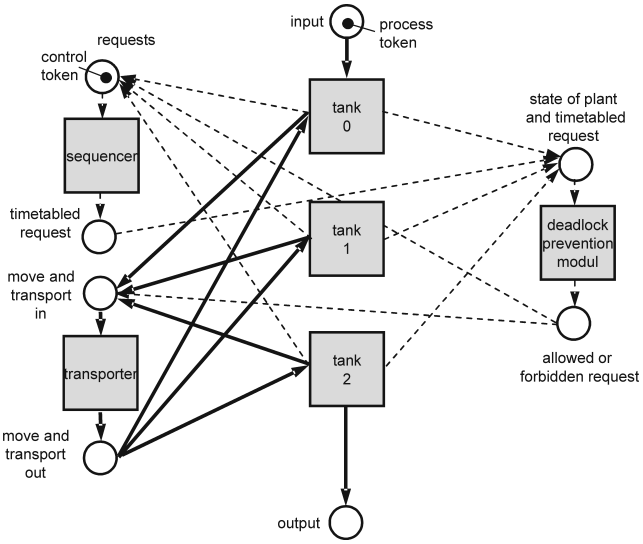

Fig. 5 Resource centered model with deadlock prevention module

V. Collision Avoidance

In our plant application there are two transporters on one track serving two overlapping areas. Because of the overlapping areas a collision avoidance strategy is a must. Fig. 6 shows the extended model.

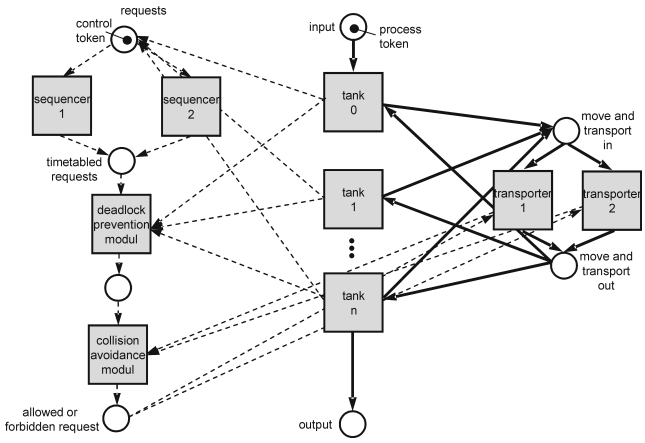

Fig. 6 Resource centered model with two transporters and one collision avoidance module

There are n tanks and two transporters. Each of the two transporters needs its own sequencer. After sequencing the transport requests, the deadlock prevention module checks the possibility of a deadlock. Then the collision avoidance module tests if the transporter intends to enter the collision endangered area. The size of the collision endangered area depends on the layout of the plant and therewith on the implemented allocation conditions. Fig.7 shows an example for a collision endangered area.

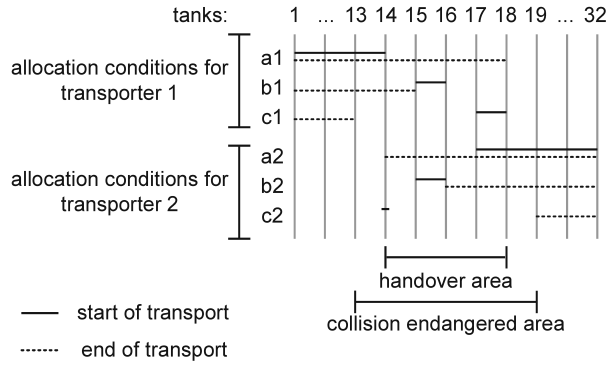

Fig. 7 Allocation conditions for the transporters

The solid lines mark possible start tanks s of a transport operation and dashed lines mark possible destination tanks e of the transport. There are three conditions a,b,c for each transporter. These conditions read as follows (compare Fig. 7):

1. The transport operation is conducted by transporter 1 if

condition $a1 \vee b1 \vee c1$ is true, $\begin{cases} a1=(s\in[1,14]\wedge e\in[1,18]), \\ b1=(s\in[15,16]\wedge e\in[1,15]), \\ c1=(s\in[17,18]\wedge e\in[1,13]), \end{cases}$

2. the transport operation is conducted by transporter 2 if

condition $a2 \vee b2 \vee c2$ is true, $\begin{cases} a2=(s\in[17,32]\wedge e\in[14,32]), \\ b2=(s\in[15,16]\wedge e\in[16,32]), \\ c2=(s\in[14,14]\wedge e\in[19,32]). \end{cases}$

The ranges of a1 and a2 define the handover area and b1, c1, and b2, c2 describe the behaviour of the transporters if start tanks are within the handover area. If the minimum distance between two transporters should always be larger than 1 tank then the collision endangered area results to [13,19] in the example of Fig. 7. There is only one transporter allowed in this area and it has to leave as soon as the transport has been finished. During this time period the second transporter can only perform transport operations outside that area.

The Petri net implementation of the collision avoidance algorithm for one transporter is shown in Fig. 8. There is a similar algorithm for the second transporter. If there is a transport request for transporter 1 (the place at the top right is marked by a token), it will be tested if either the start or the end tank is in the collision endangered area.

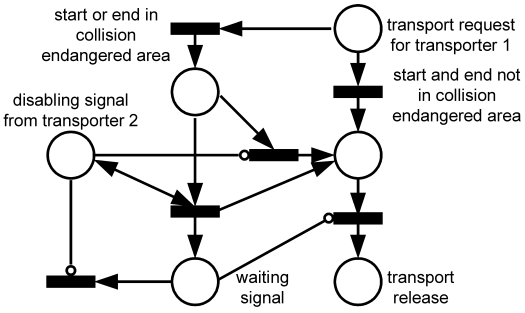

Fig. 8 Collision avoidance algorithm for transporter 1

If not, the token goes straight to the transport release place. If yes, it will be tested if there is a disabling signal from transporter 2. In case of a missing signal the transport request will be released. If there is a signal the transporter has to wait until transporter 2 has left the collision endangered area. Other strategies can be implemented as well.

VI. Factory Applications

The concepts of Sections IV and V have been implemented in a simulation tool being used for plant design and scheduling. The result of a simulation run is a transporter sequence in a text file which can be directly transferred into a PLC to control the transporters at the factory floor. The input is an Excel file with the process plan, the move times for the unloaded transporters, the information about the overlapping area, and the information about the process dependent capacity restrictions. The model is implemented in PACE 5.0, a simulation tool for coloured timed Petri nets [13]. Illustrative examples for plant design and transport scheduling for 1-hoist galvanic plants are given and described in detail in [1].

A complex real world problem for two transporting units T1, T2 is shown in Table 2. The table in the first two columns contains the process plan as the sequence of operations in the respective tanks and the process interval times. For each process, 25 tank resources are used, 6 of them more than once. Some of them are in parallel, for example tank 20 and tank 21 which are the bottleneck resources. The transporters follow Euclidean travel times between 0 and 35 seconds. For loading and unloading they need 10 seconds. Transporter T1 starts the process with the transport to tank 2 and tank 17. Transporter T2 serves tanks 1 to 14, and transporter T2 tanks 14 to 25, with a handover range from tank 14 to 18 (compare Fig.7).

Table 2. Process plan and simulation results

tank	Time intervals	Capacity. restrictions	V= 1245 cap.2	Capacity restrictions	V=1245 cap.1
1	[0;∞]		0/0		0/0
2	[5;∞]		0/0		8/61
17	[10;∞]	1	56/56		37/37
25	[235;245]		14/6		31/12
23	[60;∞]		604/536	1	20/20
24	[120;∞]		0/0		0/0
22	[55;65]	1	0/0		28/28
23	[60;∞]		109/109		86/88
24	[120;∞]		629/556		51/24
20/21	[2390;2410]		0/0		0/0
19	[60;∞]		86/83		25/25
18	[120;∞]		155/155		57/57
17	[120;∞]		8/0		25/6
15	[175;185]		0/0		30/30
16	[60;∞]		0/0		57/57
17	[120;∞]		0/0		0/0
13/14	[590;610]		0/0		0/5
12	[120;∞]		57/116		0/0
11	[120;∞]		20/0		22/47
10	[120;∞]		0/0		0/0
3	[175;185]		0/0		0/0
4/5	[2030;2050]		0/0		0/0
6	[60;90]		16/17		16/16
9	[120;∞]		9/9		9/9
10	[120;∞]		56/57		56/57
7/8	[60;∞]		0/0		0/0
2	[5;∞]		0/0		0/0
1	[0;∞]		0/0		0/0

Besides the 2 transporters, 6 loop resources exist (indicated grey in Table 2, first column). According to the deadlock prevention strategy in Section IV, there are 7 restricted areas S1 to S7 with maximum capacities of 1 to 4 (indicated white in Table 2, first column). Two types of simulations have been conducted to understand the functionalities of the deadlock prevention and collision avoidance modules. Table 2, columns 4 and 6 show the results for two different capacitiy restrictions, the second one more restrictive than the first. In both cases, we only need to restrict those areas S1 and S2 (respectively the relevant resources tank 25 and 22) which lie upstream of the bottleneck resources tank 21 and 20. For these two cases, columns 4 and 6 show the deviations from the lower boundary of the processing time interval for the first set of processes which uses one set of parallel resources (tanks 4, 13 and 20), and for the second set of processes (which uses tanks 5, 14 and 21). In this application, the less restricted deadlock prevention strategy (column 4) performs better as it results in a transport schedule which only slightly deviates from the prescribed process plan (tank 25, in grey). Finally, Figs. 9 and 10 show the transport schedules as the upper two lines of the resource Gantt charts. The chart displays 8 processes (or jobs) concurrently being processed in the plant in periodic (stationary) operation.

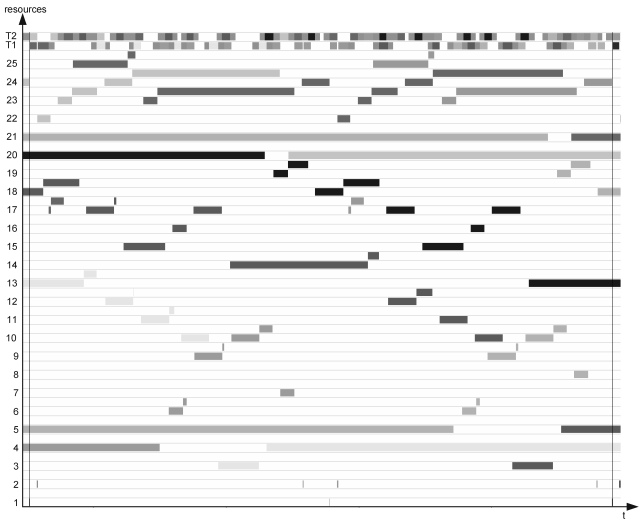

Fig. 9 Gantt chart for a release time V=1245 with two restricted areas

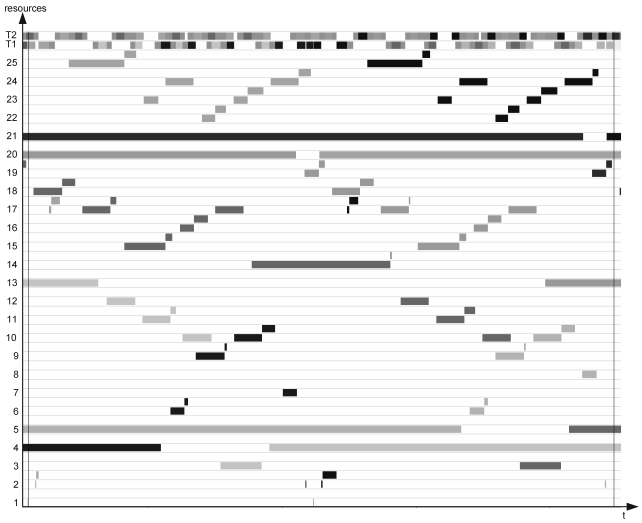

Fig. 10 Gantt chart for a release time of V=1245 with one restricted area

VI. Conclusions

We described a modelling method for a scheduler based on coloured timed Petri nets. It leads to stationary transporter sequences to feed them into a PLC to control the transporter. The process plans can be easily changed just by modifying an Excel file. If there are loops in the process plans, deadlocks are resolved with a deadlock prevention algorithm based on capacity restrictions. Collisions between transporters are avoided too. Lot switching and dynamic scheduling solutions can be obtained as well. The scheduler is implemented for a real 25 tank plant with two transporters in a factory for electronic devices.

References

[1] C. Fiedler and W. Meyer, "Process Centred versus Resource Centred Modelling for Flexible Production Lines", *Proc. of Int. MultiConference of Engineers and Computer Scientists*, Hongkong, March 2008.

[2] Y. Crama, V. Kats, J. van de Klundert and E. Levner, "Cyclic scheduling in robotic flowshops", *Annals of Operations Research*, vol. 96, pp. 97–124, 2000.

[3] M. Dawande, H.N. Geismar, S.P. Sethi and C. Sriskandarajah, "Sequencing and Scheduling in Robotic Cells: Recent Developments", *Journal of Scheduling*, vol. 8, pp. 387–426, 2005.

[4] Y. Crama and J. Klundert, "Robotic flowshop scheduling is strongly NP-complete", in *Ten Years LNMB (W.K.Klein Haneveld, O.J. Vrieze and L.C.M. Kallenberg, eds.)*, CWI Tract 122, Amsterdam, pp. 277-286, 1997.

[5] L. W. Phillips and P. S. Unger, "Mathematical Programming Solution of a Hoist Scheduling Program", *AIIE Transactions*, vol. 28, no. 2, pp. 219-225, June 1976.

[6] R. Rodošek and M. Wallace, "A Generic Model and Hybrid Algorithm for Hoist Scheduling Problems", in *Lecture Notes in Computer Science*, vol.1520, pp. 385-399, Springer Verlag, Berlin, 1998.

[7] M.-A. Manier and C. Bloch, "A Classification for Hoist Scheduling Problems", *International Journal of Flexible Manufacturing Systems*, vol. 15, no.1, pp. 37 – 55, Jan 2003.

[8] R.L.Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnoy Kan, "Optimization and approximation in deterministic sequencing and scheduling: A survey", *Annals of Discrete Mathematics*, 5, 287-326, 1979.

[9] C. Fiedler, "Event-driven Generation of Periodic Hoist Schedules", *Proc. IEEE Conf. Systems, Man, and Cybernetics*, Taipei, Oct. 2006.

[10] M. Zhou and F. DiCesare, *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems*, London: Kluwer Academic Publishers, 1993.

[11] M. Lawley, S. Reveliotis, and P. Ferreira, "Design Guidelines for Deadlock-Handling Startegies in Flexible Manufacturing Systems", *The International Journal of Flexible Manufacturing Systems*, Kluwer Academic Publishers, Boston, 9, pp. 5-30, 1997.

[12] A. v.Drathen, "Compact Modeling of Manufacturing Systems with Petri nets", *Proc. IEEE Conf. Systems, Man, and Cybernetics*, Montreal, Oct. 2007.

[13] www.ibepace.com