Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Hybrid Model for Crane Scheduling

Carmen Del Vecchio * **Osvaldo Barbarisi** * **Alessandra Parisio** *

* *Dipartimento di Ingegneria, Università degli Studi del Sannio, P.za Roma 21, 82100 Benevento, Italy*
email: { c.delvecchio, barbarisi, aparisio }@unisannio.it

Abstract: In this paper an hybrid model for scheduling crane movements is presented. Cranes sharing the same track are considered. Cranes can reach any location along the track performing loading or unloading operations. As cranes share the same track their movements are constrained to avoid over crossing; tasks deadlines and priorities are also considered. Scheduler output are the assignment of the tasks to cranes and tasks starting time. Experimental data show the benefits deriving from the application of the proposed approach.

1. INTRODUCTION

A bridge crane (Figure 1) is a transportation equipment that moves horizontally and vertically along a track; it reaches a *loading or requesting point* (such as storing locations, ships or machines) to pick up an item and it reaches an *unloading or destination point* where the item has to be discharged. The destination point can be another movement equipment, a shipment/storing location or a machine. Cranes are usually found in chemical, steel and iron industries and in large storage sites (container terminals, ports, warehouses). In the following, a "job" is the request from the loading to the destination point (including loading and unloading operations). A job is specified through its requesting and destination point and, eventually, through some real time constraints such as the deadline for item discharge.

In this paper, we focus on cranes' scheduling problems. The goal of crane scheduling is to assign jobs to cranes so that a given plant performance function is optimized. This problem is similar to the m-parallel machine scheduling problem (e.g., Lee and Pinedo [1997], Hall et al. [2000]), however, crane scheduling has several unique characteristics. For example, if loading/unloading points are machines performing operations in a production process, cranes must serve the requests following the production process order. Similarly in port management, if unloading and loading operations must be performed at the same location, the discharging operation must precede the loading operation. Thus, there might be *precedence relationships* among jobs that must be respected when serving a task list.

Moreover cranes travel on the same track, and they can not cross each other; then if a crane precedes another one this precedence order has to be preserved (*e.g.,* in Figure 1 C_2 precedes C_1 when moving from left to right). This constraint limits crane positions; thus some pairs of jobs can not be performed simultaneously, this depending both on loading and unloading point of jobs, and on crane positions. In the following we will also call such jobs incompatible jobs.
The precise form of the scheduler depends on the operation mode of the system: in an *off-line* scheme, jobs to be served over the scheduling horizon are known in advance, whereas in the case of *on-line* control no such prior information is available. In heavy transport sights management, such as port container terminal, *off line* schemes are often applied (see Kim and Park [2004]) as in these contests work load is usually projected in advance; moreover since tasks are known the non crossing constraint can be modeled simply defining a set of incompatible jobs and avoiding their simultaneous execution. In warehouses managing problems (Lee and Di Cesare [1994], Amato and Basile [2001]) cranes are assigned to a predetermined subset of loading points, thus their movements do not interfere. Moreover cranes are allocated to a subset of locations then they are statically assigned to the requesting/destination point; this reduces problem complexity and dimension. Recent studies on cranes movements modeled with Petri Net allowed to obtain an exhaustive representation of crane movements and constraint. However the evolution of the system over time was complex and constrained by the Petri Net evolution itself. For instance cranes where constrained to serve tasks contemporarily (see Barbarisi et al. [2007]).

Literature on *on line* scheduling of crane movements is less numerous, usually referred to chemical industry where the job list can not be estimated due to unpredictable production events (Tang et al. [2000], Harjunkoski and Grossmann [2001]). Simulation is usually the proposed instrument to solve modeling issues such as the non crossing constraints (see Tamaki et al. [2004]); some plant model (see Tamaki et al. [2004]) allow 'escaping' positions along the track where a crane can temporarily wait until the other passes; in fact this is equivalent to admit over crossing.

Our goal is to present an hybrid model of crane movements. So far the model has been used for off line scheduling, where the task list is completely known in advance. Our main future goal will be to apply the same model for *on line* scheduling problems. The scheduling problem addressed in this paper allows tasks subject to precedence relationships and real time constraints (deadlines). Solving such problems requires a controller determining (i) a *dynamic assignment* of cranes to jobs (ii) the time instant when a crane starts to serve a job. For *dynamically*

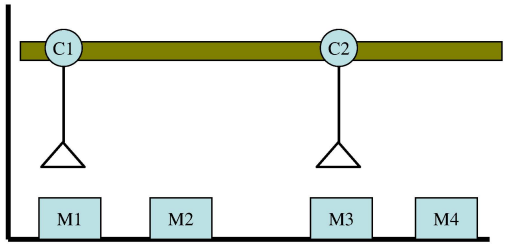

Fig. 1. Two cranes serving four locations and sharing the same track.

assigning cranes to jobs we mean that a requesting point can be served by any crane for any of its request.

Our main contributions are:

- **to develop a hybrid model of cranes movements**; we modeled crane movements and the real-time constraint with hybrid system. We obtained a mathematical formulation of cranes state and time constraints that we exploit to build our complete model, adding precedence relationship among cranes and the objective function formulation.
- **to develop a scheduler of crane movements**; exploiting the hybrid crane model, we built a scheduler of crane movements based on the assumption that a sequence of jobs over a significative time window is available or can be estimated with good accuracy. The resulting scheduling problem is also promising for an on line application.

The following sections describe the problem and the proposed approach.

2. PLANT MODEL - DEFINITIONS

This section introduces the basic variables' definitions for jobs and cranes.

We suppose that locations to be served are placed along a line and they are labeled with increasing numbers; cranes are also labeled in the direction of increasing locations numbers. Then the plant we consider is made of an ordered set $\mathcal{M} = \{m_1, \ldots, m_m\}$ of m locations, and an ordered set $\mathcal{C} = \{c_1, \ldots, c_{n_c}\}$ of n_c cranes moving on the same track. Figure 1 represents label for cranes and places in a four locations and two cranes site.

We start by describing jobs characteristics and main variables, then we introduce hybrid crane model.

Each job is specified by:

- *the requesting point*: any location along the track;
- *the destination point*: as for the requesting point can be any location along the track but different from the requesting point;
- *the deadline*: the time instant when the job must be accomplished;
- *a precedence relationship* with other jobs; jobs can be related to a production process, then a set of them could be constrained to be served following the production process order.

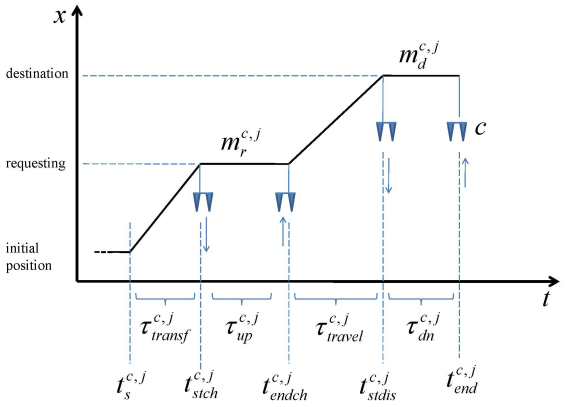

Fig. 2. Crane position vs. time.

All requests are collected in a job list; a location can appear in more than one request both as destination or requesting point. Jobs with the same origin and the same destination point are also allowed.

Several events describe the life of a job (see Figure 2). At the **start time**, $t_s^{c,j}$, the crane c starts to serve job $u_{c,j}$. The crane moves from the destination point of the previous job to the requesting point of job $u_{c,j}$; we denote this time interval as **transfer time** ($\tau_{transf}^{c,j}$). We suppose that all cranes move with equal transfer velocity, v_L.

Once reached the destination point, the crane c has to pick up the item: the **start placing time**, $t_{stch}^{c,j}$, denotes the instant when the placing operation starts and the **end placing time**, $t_{endch}^{c,j}$, the termination of this operation. The loading operation lasts for a time interval, the **pick up time**, denoted with $\tau_{up}^{c,j}$, which is supposed to be independent of the item and job.

When the item is loaded, crane moves along the track for a time interval ($\tau_{travel}^{c,j}$) depending on the destination point and on full charged velocity of crane, v_H; note that velocity is different if crane is empty (v_L) or charged (v_H).

At the **start discharging time**, $t_{stdis}^{c,j}$, crane reaches the destination point and the operations to release the item start. When the item is discharged the job finishes; this instant is named the **end time**, $t_{c,j}^{end}$. The duration of the discharging operation is denoted with $\tau_{dn}^{c,j}$.

2.1 Model of task list of job

We suppose that a list of job to be assigned to cranes is available: $\mathcal{L} = \{u_1, \ldots, u_j, \ldots, u_L\}$. The job u_j is defined through the following information

- $x_r^j \in \mathbb{R}$ the requesting point, where the item is picked up;
- $x_d^j \in \mathbb{R}$ the destination point;
- $t_d^j \in \mathbb{R}$ the deadline of job u_j.

3. HYBRID MODEL OF CRANE MOVEMENTS

We rely on hybrid system modeling, that is the model' state changes according to a continuous time and event

based scale. Crane position along the track being the *time-driven* component; reaching a position or getting-discharging an item being the *event-driven* component.

3.1 Hybrid system states and variables

According to the notation in Lygeros et al. [1999] we denote by Q_c the set of discrete variables corresponding to five discrete states, X_c the set of time driven variables and U_c the job list assigned to crane c.

$$Q_c := \{(q_c)|\ q_c \in \{0,1,2,3,4\}\} \tag{1}$$

$$X_c := \{(x_c, t_c, \delta_c)'|\ 0 \le x_c \le x_{c,max} \quad t_c, \delta_c \in \mathbb{R}\} \tag{2}$$

$$U_c := \{u_{c,1}, \ldots, u_{c,j}, \ldots, u_{c,n}\} \subseteq \mathcal{L} \tag{3}$$

The variables meaning is described next. x_c is the position of crane c along the track measured in the sense of positive x axis; its value ranges from 0, the extreme left position reachable along the track, to $x_{c,max}$ the extreme right. The time t_c, is the continuous time flowing from the beginning of the scheduling problem; its replication for each crane is redundant, but it will help in simplifying the notation, as it will be clear later in this section. δ_c stores the time instant when the discrete state changes.

We denote by $x_r^{c,j}$ and $x_d^{c,j}$, respectively, the position of the requesting and destination point of job $u_{c,j}$ served by crane c.

The mining of the discrete state, the corresponding value of the continuous variables and the condition allowing the transition towards the states will be explained next. Note that in order to simplify notation we will omit the dependence of job $u_{c,j}$ from crane c, when this does not generate ambiguity:

- **idle**: $(q_c = 0)$ crane is waiting to start a job. Crane position $x_c(t)$ corresponds to the initial crane position $x_{c,0}$ if the crane has not served any job, or to the destination point of last job served $x_d^{c,j-1}$. Crane leaves this state at the starting time $t_s^{c,j}$ of the job to be served. The time interval the crane spent into this state is denoted with $\tau_s^{c,j}$.
- **transfer**: $(q_c = 1)$ crane moves from the destination point of last job served to the requesting point of the job being served. Crane position $x_c(t)$ changes according to empty crane velocity v_L. State changes when crane reaches the requesting point $(x_c(t) = x_r^{c,j})$. Then the state transfer holds for a time interval $\tau_{transf}^{c,j}$ equal to:

$$\tau_{c,j}^{transf} = \frac{1}{v_L}|x_r^{c,j} - x_d^{c,j-1}| \tag{4}$$

- **up**: $(q_c = 2)$ crane stays at the requesting point $(x_c(t) = x_r^{c,j})$ and loads the item. Crane leaves this state when a time interval equal to $\tau_{up}^{c,j}$ elapses.
- **travel**: $(q_c = 3)$ crane leaves the requesting point and moves toward the destination point. Given crane velocity at full charge, v_H, and crane's initial and final positions, the time interval spent in this state $\tau_{c,j}^{travel}$ is equal to:

$$\tau_{c,j}^{travel} = \frac{1}{v_H}|x_d^{c,j} - x_r^{c,j}| \tag{5}$$

When crane reaches the destination $(x_c(t) = x_d^{c,j})$ it leaves this state.

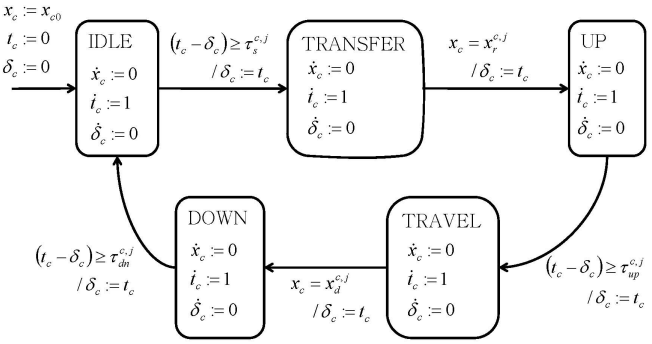

Fig. 3. Hybrid system for one crane.

- **down**: $(q_c = 4)$ crane fulfills discharging operations; crane position is equal to the destination point $x_c(t) = x_d^{c,j}$. This state holds for a time interval equal to $\tau_{dn}^{c,j}$.

The vector $\mu_c = [\mu_{c,1} \ldots \mu_{c,5}]$ will be used in what fallow. It denotes the discrete state occupied by the crane c during its evolution in time. As the crane can occupy only one state at time, vector μ_c is a binary vector.

3.2 Hybrid system dynamic

The *time driven* system component evolves according to the following differential equations:

$$\begin{bmatrix} \dot{x_c} \\ \dot{t_c} \\ \dot{\delta_c} \end{bmatrix} = \begin{bmatrix} f\,(x_c, q_c, u_{c,j}) \\ 1 \\ 0 \end{bmatrix}. \tag{6a}$$

In particular

$$f(x_c, q_c, u_{c,j})|_{q_c \in \{0,2,4\}} = 0 \tag{6b}$$

$$f(x_c, 1, u_{c,j}) = sign\left(x_r^{c,j} - x_c\right) \cdot v_L \tag{6c}$$

$$f(x_c, 3, u_{c,j}) = sign\left(x_d^{c,j} - x_c\right) \cdot v_H \tag{6d}$$

where the function $sign(a)$ is defined as follows:

$$sign(a) = \begin{cases} 1 & a > 0 \\ 0 & a = 0 \\ -1 & a < 0. \end{cases}$$

Defined implicitly from Figure 3 the transitions $e \in E = \{(0,1),(1,2),(2,3),(3,4),(4,0)\} \subseteq Q_c \times Q_c$, the guard conditions

$$G_c(q_c, \tilde{q}_c, u_{c,j}) : E_c \times U_c \to \mathcal{P}(X_c) \tag{6e}$$

are

$$G_c(0,1,u_{c,j}) = \left\{(x_c, t_c, \delta_c)' \in X_c | (t_c - \delta_c) \ge \tau_s^j\right\}$$
$$G_c(1,2,u_{c,j}) = \left\{(x_c, t_c, \delta_c)' \in X_c | x_c = x_r^{c,j}\right\}$$
$$G_c(2,3,u_{c,j}) = \left\{(x_c, t_c, \delta_c)' \in X_c | (t_c - \delta_c) \ge \tau_{up}^{c,j}\right\}$$
$$G_c(3,4,u_{c,j}) = \left\{(x_c, t_c, \delta_c)' \in X_c | x_c = x_d^{c,j}\right\}$$
$$G_c(4,0,u_{c,j}) = \left\{(x_c, t_c, \delta_c)' \in X_c | (t_c - \delta_c) \ge \tau_{dn}^{c,j}\right\}.$$

where the $\mathcal{P}(X_c)$ denotes the power set of the set X_c.

We also define the reset function $R_c(q_c, \tilde{q}_c, x_c, t_c, \delta_c) : E_c \times X_c \to \mathcal{P}(X_c)$ as

$$R_c\big(q_c, \tilde{q}_c, (x_c, t_c, \delta_c)'\big) = \{(x_c, t_c, t_c)'\}. \tag{6f}$$

And the initial condition is given by

$$Init_c = \{(q_c, x_c, t_c, \delta_c)' \in Q_c \times X_c | \\ q_c = 0, x_c = x_{c,0}, t_c = 0, \delta_c = 0\} \quad (6g)$$

3.3 Hybrid model of two cranes

The hybrid model can be extended to two cranes defining $Q = Q_1 \times Q_2$, $X = X_1 \times X_2$ and $U = U_1 \times U_2$. The guard conditions (6e) are extended as follows:

$$G\big((q_1, q_2)', (\tilde{q}_1, \tilde{q}_2)', (u_{1,j}, u_{2,j})'\big) = \\ = G_1(q_1, \tilde{q}_1, u_{1,j}) \times G_2(q_2, \tilde{q}_2, u_{2,j}) \quad (7)$$

Also the reset function is defined as

$$R\big((q_1, q_2)', (\tilde{q}_1, \tilde{q}_2)', (x_1, t_1, \delta_1, x_2, t_2, \delta_2)'\big) = \\ = R_1\big(q_1, \tilde{q}_1, (x_1, t_1, \delta_1)'\big) \times R_2\big(q_2, \tilde{q}_2, (x_2, t_2, \delta_2)'\big) \quad (8)$$

4. CRANES DYNAMICAL MODEL

The objective of this section is to construct a model describing the model variables evolution as a function of some asynchronous events.

Based on practical consideration, a discrete-time approach would lead to a computationally demanding model for long prediction horizons. An event-based modeling approach is used in this paper. This makes the work challenging and interesting. First, the job sequence on each crane is asynchronous and correlated through the non crossing constraints with job sequences on different cranes. Secondly, each crane evolves with different dynamics but their movements are not independent, it is necessary to describe the overall plant dynamic, where cranes move with a time varying speed at uncorrelated events.

For the proposed model, an event is a change of crane model discrete state. This will allow to have a "high resolution" model with a reduced computational complexity compared to a discrete time modeling.

We denote by $t(k)$ the instant of the k-th event. The state of the plant model at the event k is a snapshot at time $t(k)$ of (i) jobs being served by all cranes; (ii) the position of each crane; (iii) the discrete state of each crane (iv) the time elapsed in the discrete state. The plant inputs at the event k are the crane jobs schedule. Given the state and the input at $t(k)$, the model allows to predict the state at the event $k+1$ by modeling cranes' position, the assignment of scheduled jobs on cranes, the completion of jobs. Moreover since the events are asynchronous, a 'partial' completion of a job (i.e., a crane is moving from requesting to the destination point) on a crane has to be modeled. Objective of the next part is to formally define process states and inputs and describe their event-based evolution model.

The dynamic of the system described with (6) is rewritten in the following transition system. Note that $x(k), \delta(k)$ and $q(k)$ we denote vectors of dimension n_c, equal to the number of plant cranes.

$$x(k+1) = f_x\big(q(k), x(k), t(k), \delta(k), u(k)\big) \quad (9a)$$
$$t(k+1) = f_t\big(q(k), x(k), t(k), \delta(k), u(k)\big) \quad (9b)$$
$$\delta(k+1) = f_\delta\big(q(k), x(k), t(k), \delta(k), u(k)\big) \quad (9c)$$
$$q(k+1) = f_q\big(q(k), x(k), t(k), \delta(k), u(k)\big) \quad (9d)$$

where $u_{c,j}(k)$ is the job that crane c is serving at the instant $t(k)$.

The following equation describes the evolution of crane's position over time

$$x_c(k+1) = x_c(k)\big(\mu_{c,1}(k) + \mu_{c,2}(k) + \mu_{c,4}(k)\big) + \\ x_r^{c,j}\mu_{c,3}(k) + x_d^{c,j}\mu_{c,5}(k). \quad (10)$$

Crane changes position when it reaches the loading point ($\mu_{c,3}(k) = 1$) or the requesting point ($\mu_{c,5}(k) = 1$); it stays by the same position when it loads $\mu_{c,2}(k) = 1$, unloads ($\mu_{c,4}(k) = 1$) items, or when it is in idle ($\mu_{c,1}(k) = 1$). It results

$$q_c(k) = \sum_{i=1}^{5}(i-1)\mu_{c,i}(k).$$

with $\sum_{i=1}^{5}\mu_{c,i}(k) = 1$. The following equation describes the evolution of the variable $\Delta t_c(k)$ which is the time interval that crane c spends in a state q_c:

$$\Delta t_c(k) = \tau_s(u(k))\mu_{c,1}(k) + \\ \frac{1}{v_L}\big|x_r^{c,j}(u_c(k)) - x_c(k)\big|\mu_{c,2}(k) + \\ \tau_{UP}\mu_{c,3}(k) + \frac{1}{v_H}|x_d^{c,j}(u_c(k)) - x_c(k)|\mu_{c,4}(k) + \\ \tau_{DN}\mu_{c,5}(k) + \quad (11)$$

$$\sigma_c(k) = \begin{cases} 1 & if\ [\delta_c(k) + \Delta t_c(k) = \\ & \quad \min_i \{\delta_i(k) + \Delta t_i(k)\}] \\ 0 & otherwise \end{cases} \quad (12)$$

$$\delta_c(k+1) = (\delta_c(k) + \Delta t_c(k))\sigma_c(k) + \delta_c(k)(1 - \sigma_c(k)) \quad (13)$$

$$\mu_{c,1}(k+1) = \mu_{c,5}(k)\sigma_c(k) + \mu_{c,1}(k)(1 - \sigma_c(k)) \quad (14)$$
$$\mu_{c,2}(k+1) = \mu_{c,1}(k)\sigma_c(k) + \mu_{c,2}(k)(1 - \sigma_c(k)) \quad (15)$$
$$\vdots$$
$$\mu_{c,5}(k+1) = \mu_{c,4}(k)\sigma_c(k) + \mu_{c,5}(k)(1 - \sigma_c(k)) \quad (16)$$
$$u_c(k+1) = u_c(k)(\mu_{c,2}(k) + \mu_{c,3}(k) + \mu_{c,4}(k) + \mu_{c,5}(k)) \quad (17)$$
$$\mu_{c,2}(k) + \mu_{c,2}(k) + \mu_{c,3}(k) + \mu_{c,4}(k) + \mu_{c,5}(k) = 1 \quad (18)$$

The (18) guarantees that for each value of $q(k)$ it results that for only one i it results $\mu_{c,i}(k) = 1$. The binary variable $\sigma_c(k) \in \mathbb{B}$ is true ($\sigma_c(k)$=1) every time that a transition of crane c is active. This is active when the next event is occurs to the crane c (see (12)).

5. SCHEDULER MODEL

We denote the set of all task as $\mathcal{T} = \{J_1, J_2, \ldots, J_n\}$. In order we introduce the virtual job J_∞. This job has to be intended as the last job for each crane, so when it has been assigned to a crane, the crane is always in idle. We impose that $t_s(J_\infty) = \infty$.

First of all we introduce the following matrix $B = [b_{ij}] \in \mathbb{Z}^{m \times m}$. The element b_{ij} defined as equal to the order of the job that has requesting point m_i and destination m_j.

$$b_{ij} := ord(\{J \in \mathcal{T} | m_r(J) = M_i,\ m_d(J) = M_j\}), \quad (19)$$

then it results $\sum_{i=1}^{m}\sum_{j=1}^{m} b_{ij} = n$.

The scheduler on each event $t(k)$ has to define the input $u(k) \in (\mathcal{T} \cup J_\infty)^{n_c}$. The scheduler assigns the task to a crane c only when this is idle ($q_c = 0$), hence it holds

$$q_c(k) \in Q/\{0\} \Rightarrow u_{c,k+1} = u_{c,k}$$

The scheduler has to select the job $u \in \mathcal{T}$ and to define the time interval τ_s^j that the crane waits before starting to serve the task. The scheduler has also to ensure that task will be accomplished before the deadline $t_d(u)$. Moreover the scheduler has to verify that the two cranes will not cross each other. The position $\hat{x}_c(k)$ of each crane at the event k^{th} is given by

$$\hat{x}_c(k) := \begin{cases} x_c(k) & q_c(k) = 0 \\ x_c(k) + d_1 v_L \cdot (t(k) - \delta_c(k)) & q_c(k) = 1 \\ x_c(k) & q_c(k) = 2 \\ x_c(k) + d_2 v_H \cdot (t(k) - \delta_c(k)) & q_c(k) = 3 \\ x_c(k) & q_c(k) = 4 \end{cases} \quad (20)$$

where $d_1 = sgn(x_r(u(k)) - x(k))$ and $d_2 = sgn(x_d(u(k)) - x(k))$; hence that for any assignment of tasks to cranes, the scheduler verifies that the following condition is verified

$$\hat{x}_1(k) \leq \hat{x}_2(k) \quad (21)$$

Then we introduce the following matrix $A_{c,k}(u) = [a_{ij}^c] \in \mathbb{B}^{m \times m}$ of assigned job:$(x_{1,k} \leq x_{2,k} \leq \ldots \leq x_c(k) \leq \ldots \leq x_{nc,k})$

$$a_{ij}^c(u) = \begin{cases} 1 & q_c = 0 \wedge M_i = m_r(u_c), M_j = m_d(u_c) \\ 0 & otherwise \end{cases} \quad (22)$$

and the vector $t_s(u(k)) \in \mathbb{R}_+^c$ of the delay of each. In order it has to guarantee that each crane will not collide with the others . We define the matrix

$$L = \begin{bmatrix} 1 & -1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -1 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \ddots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 1 & -1 \end{bmatrix} \in \mathbb{Z}^{(nc-1) \times nc}$$

The scheduler will assign the task to the cranes by solving the following problem

$$\min_{[u_1, u_2, \ldots]} \sum_{k=1}^{\infty} \|t_s(u(k))\|_2 \quad (23a)$$

$$\text{s.a} \quad x(k+1) = f_x(q(k), x(k), t(k), \delta(k), u(k)) \quad (23b)$$

$$t(k+1) = f_t(q(k), x(k), t(k), \delta(k), u(k)) \quad (23c)$$

$$\delta(k+1) = f_\delta(q(k), x(k), t(k), \delta(k), u(k)) \quad (23d)$$

$$q(k+1) = f_q(q(k), x(k), t(k), \delta(k), u(k)) \quad (23e)$$

$$B_{k+1} = B_k - \sum_{c=1}^{nc} A_{c,k}(u(k)) \quad (23f)$$

$$B_0 = B \quad (23g)$$

$$B_\infty = 0 \quad (23h)$$

$$Lx_k \leq 0 \quad (23i)$$

$$\delta_c(k) \leq t_d(u(k)) \quad (23j)$$

6. COMPUTATIONAL RESULTS

This section deals with the application of the proposed crane scheduler to an operating plant. The plant produces four macro-families of products; within each family there are several different products ranging from ten to more than fifty. The production process consists of four phases one of which is not performed to some product. Each phase can be carried out by one or more machines.

At the end of each production phase two cranes move the items to the machine in the next phase. Then the required crane movements correspond to the transports from a production phase to the next one. Table 1 reports the list of crane movements.

The two cranes share the same track, than non crossing constraint should also be respected. The production process has severe timing constraint; if the product is not processed within the deadlines it deteriorates; then production deadlines should stringently be respected.

All remaining data about the plant cannot be disclosed. For this reason, the data presented in the next will be scaled. The crane parameters are all unit.

Non-crossing constraint is respected by checking at each intermediate machine the relative position of cranes. If non-crossing constraint would be violated, crane waits until all intermediate positions between requesting and destination are available.

Problem (23a) has been implemented using AMPL mathematical language and solved using MINLP solver, run on a 2.4 GHz Intel Xenon machine. The solver implements a branch-and-bound algorithm to solve quadratic and nonlinear programming problems.

Results for a representative production plan are reported; Table 2 reports the batch size for each typology in the sample production plan and the corresponding processing times. The set of all requests forming the task list is obtained from the production plan; for instance, one unit of product P1 corresponds to tasks M_1-M_3 and M_3-M_5. The precedence relationship among the production phases is modeled defining ordered subsets of tasks with precedence constraints. Hence in the above example, tasks M_1-M_3, M_3-M_5 and M_5-M_8 belong to the same subset. The deadlines are assigned according to production constraints. Table 3 reports the task list corresponding to the production plan in Table 2. Table 4 reports the scheduler

Table 1. Crane movements

Typology	Task
P1	$M_1 - M_3$
	$M_3 - M_5$
	$M_5 - M_8$
P2	$M_2 - M_5$
	$M_5 - M_8$
P3	$M_1 - M_4$
	$M_4 - M_6$
	$M_6 - M_7$
P4	$M_2 - M_4$
	$M_4 - M_6$
	$M_6 - M_8$

Table 2. Production Plan

Typology	Batch Size	Production time (min) Ph.1	Ph.2	Ph.3	Ph. 4
P1	2	60	70	38	56
P2	2	60	–	58	63
P3	1	65	23	60	64
P4	1	60	70	54	60

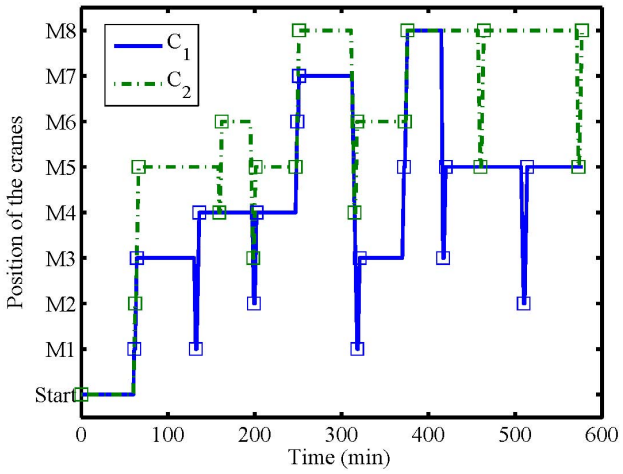

Fig. 4. Crane movements for task list in Table 3

results. Figure 4 depicts the crane movements of the two schedules in Table 3. Machines are shown on the vertical axis and the time on the horizontal axis. Clearly the non crossing constraint is respected.

Table 3. Task list

Task	**Load-Unload**	**Deadline**
Task1	$M_1 - M_3$	70
Task2	$M_2 - M_5$	70
Task3	$M_1 - M_4$	140
Task4	$M_4 - M_6$	170
Task5	$M_2 - M_4$	212
Task6	$M_3 - M_5$	210
Task7	$M_6 - M_7$	262
Task8	$M_5 - M_8$	260
Task9	$M_1 - M_3$	350
Task10	$M_4 - M_6$	330
Task11	$M_5 - M_8$	390
Task12	$M_6 - M_8$	390
Task13	$M_3 - M_5$	425
Task14	$M_5 - M_8$	470
Task15	$M_2 - M_5$	530
Task16	$M_5 - M_8$	590

Table 4. Scheduler results

$t(k)$	**Crane C_1 Task (From-To)**	**Crane C_2 Task (From-To)**
61	Task1 (M_1-M_3)	–
62	–	Task2 (M_2-M_5)
132	Task3 (M_1-M_4)	–
159	–	Task4 (M_4-M_6)
198	–	Task6 (M_3-M_5)
199	Task5 (M_2-M_4)	–
247	–	Task8 (M_5-M_8)
249	Task7 (M_6-M_7)	–
315	–	Task10 (M_4-M_6)
318	Task9 (M_1-M_3)	–
372	Task11 (M_5-M_8)	–
373	–	Task12 (M_6-M_8)
417	Task13 (M_3-M_5)	–
460	–	Task14 (M_5-M_8)
510	Task15 (M_2-M_5)	–
573	–	Task16 (M_5-M_8)
Completion time		$578min$

7. CONCLUSIONS AND FUTURE WORK

The problem of controlling crane movements in a multi cranes system has been studied. For this class of problems, a model of crane movements has been developed using an hybrid model and their scheduling to serve a set of requests has been solved. The proposed problem formulation allows the inclusion of constraints, arising from time-base constraints, as well as cranes over crossing avoiding. Results from some examples show that the proposed scheduler control could be used in working plant. Further steps will aim to study the application of the proposed model for an online receding horizon control scheme.

REFERENCES

F. Amato and F. Basile. Optimal control of warehousing systems with simultaneous crane and shuttle optimization. In *Proceedings of 8th IEEE International Conference on Emerging Technologies and Factory Automation.*, volume 2, pages 95–104, 2001.

O. Barbarisi, C. Del Vecchio, and A. Parisio. Cranes control with time-based and position constraints. In *Proceedings of IEEE Conference on Decision and Control*, New Orleans, Louisiana, USA, Dicember 2007.

N.G. Hall, C.N. Potts, and C. Sriskandarajah. Parallel machine scheduling with a common server. *Discrete Applied Mathematics*, 102:223–243, 2000.

I. Harjunkoski and I. E. Grossmann. A decomposition approach for the scheduling of a steel plant production. *Computers and Chemical Engineering*, 25:1647–1660, 2001.

K. H. Kim and Y. Park. A crane scheduling method for port container terminals. *European Journal of Operational Research*, 156:752–768, 2004.

Doo Yong Lee and Frank Di Cesare. Scheduling flexible manufacturing systems using petri nets and heuristic search. *IEEE Transactions on Robotics and Automation*, 10(2):123–133, 1994.

Y. H. Lee and M. Pinedo. Scheduling jobs on parallel machines with sequence-dependent setup times. *European Journal of Operational Research*, 100(3):464–474, 1997.

J. Lygeros, K. H. Johansson, S. Sastry, and Magnus Egerstedt. On the existence of executions of hybrid automata. In *Proceedings of the 38th IEEE Conference on Decision and Control*, volume 3, pages 2249–2254, 1999.

H. Tamaki, K. Sakakibara, H. Murao, and S. Kitamura. Optimization model and simulation-based solution for a class of crane scheduling problems. In *Proceedings of the 2004 IEEE lntemational Conference on Control Applications*, 2004.

L. Tang, J. L., A. Rong, and Z. Yang. A mathematical programming model for scheduling steelmaking-continuous casting production. *European Journal of Operational Research*, 120:423–435, 2000.