MECHANICAL ENGINEERING | RESEARCH ARTICLE

A livelock control policy for a flexible manufacturing system modeling with a subclass of generalized Petri nets

C.Q. Hou, S.Y. Li, Y. Cai, H.M. Wu, A.M. An and Y. Wang

Cogent Engineering (2014), 1: 944766
A livelock control policy for a flexible manufacturing system modeling with a subclass of generalized Petri nets

C.Q. Hou1, S.Y. Li1*, Y. Cai1, H.M. Wu2, A.M. An3 and Y. Wang1

Abstract: Livelocks, like deadlocks, can result in the serious problems in running process of flexible manufacturing systems (FMSs) as well. Current deadlock control policies based on the approaches of siphon detection and control, cannot cope with livelocks in a system of sequential systems with shared resources (S⁴R), a typical subclass of Petri nets that can model FMSs. On the basis of the mixed integer programming method, this study proposes a livelock control policy (LCP) that can not only solve the new smart siphons (NSSs) associated with livelocks or deadlocks in an S⁴R system directly, but also make the solved NSSs max⁴-controlled by adding the corresponding control places (CPs). As a result, an original S⁴R system with livelocks or deadlocks can be turned into the live controlled one in which no NSSs can be found. The related theoretical analysis and several examples are given to demonstrate the proposed LCP. Compared with the existing methods in the literature, the proposed one is more general and powerful.

Subjects: Algorithms & Complexity, Concurrent Engineering, Control Engineering

Keywords: flexible manufacturing system (FMS), Petri nets, mixed integer programming (MIP), livelocks or deadlocks, new smart siphon (NSS)
1. Introduction

A flexible manufacturing system (FMS) is a novel production mode in which there is some amount of flexibility that allows the system to respond to market requirement changes in as timely a manner as possible, no matter predicted or unpredicted. An FMS is also a computer controlled configuration in which different operations are executed concurrently, and therefore, have to compete a limited number of resources such as robots, machines, storage devices, fixtures, and conveyors (Li & Zhou, 2009; Wu & Zhou, 2009). This competition can cause deadlocks that are highly undesirable phenomena, in which a set of processes keeps waiting indefinitely for other processes in the set to release resources (Chao, 2006; Ezpeleta, Colom, & Martinez, 1995; Huang, Jeng, Xie, & Chung, 2001; Li, Li, & Hu, 2011; Li & Wang, 2011; Li & Zhou, 2004). In a deadlock state, the global or local FMS is blocked or crippled, causing unnecessary cost or catastrophic results. When an FMS is at a livelock state, a special case of deadlocks, some processes continuously change their state, while the other processes are deadlocked (Chao, 2007; Liu & Li, 2010). That is to say, a livelocked system still does some (although not useful) work in which some execution that never makes effective progress cycle infinitely. As a result, deadlocks and livelocks must be considered for designing an FMS.

In order to effectively handle the above problems in an FMS; digraphs, automata, and Petri nets are among the most popular mathematical tools available. Among them, Petri nets have been widely used to model an FMS, because of their own features (Li & Zhou, 2009; Wu & Zhou, 2009). They can be well used to describe behavioral properties such as reversibility, boundedness, and liveness. A siphon, as a structural object of a Petri net, is a set of places (where tokens can leak out or inject in) of a Petri net model. Once a siphon has lost all its tokens, i.e. it is emptiable at a reachable marking, output transitions of places in the siphon can never be executed and hence the net is not live. Previous research indicates that siphons are closely related to the deadlocks in Petri net models of FMSs. It is well known that the number of strict minimal siphons (SMSs) in a Petri net grows quickly or may grow exponentially with respect to its size (Li & Zhou, 2009) such that it is difficult and time consuming to find out these SMSs. Hence, how to design or select the proper method to solve the siphons is very important for developing new deadlock control policies (DCPs). Many scholars focus on solving and controlling siphons and have developed a large number of siphon-based controlled (SC) policies (Chao, 2006; Huang et al., 2001; Huang, Jeng, Xie, & Chung, 2006; Li, An, Cai, et al., 2013; Li, An, Wang, et al., 2013; Li & Li, 2012a, 2012b; Li, Li, & Al-Ahmari, 2012; Li, Liang, Lun, & Wang, 2004; Li, Hu, & Wang, 2007; Li, Wang, & Wei, 2006; Li & Zhao, 2008; Li & Zhou, 2004, 2006a, 2006b, 2008; Li, Zhou, & Jeng, 2008; Piroddi, Cordone, & Fumagalli, 2009; Uzam & Zhou, 2006; Zhong & Li, 2010).

However, the above DCPs still cannot detect the siphons that cause livelocks and then cannot deal with the livelock problems in OPNs or S\textsuperscript{4}R. When a system is at a livelock state, the overall states of the system continue to change, while parts of the system are deadlocked. In other words, once a system is at livelock, other processes cannot be processed smoothly. Actually, livelock is a special case of resource starvation although the system may be deadlock-free. As far as the authors know, Chao (2007) first points out that an S\textsuperscript{4}R without deadly marked siphons (DMSs) that contribute to deadlocks (Park & Reveliotis, 2001) is deadlock-free but may be in a livelock state where both live and dead transitions exist, implying that the representative mixed integer programming (MIP) methods in Chu and Xie (1997) and Park and Reveliotis (2001) are no longer valid to identify DMSs, since the net may have livelocks even though it is deadlock-free. Accordingly, the concept of max\textsuperscript{″} controlled siphons is proposed in Chao (2007), but the corresponding approach to detect siphons associated with livelocks is not given. Subsequently, based on the max\textsuperscript{″} controllability condition, Liu and Li (2010) present a more general MIP to detect deadlocks or livelocks caused by siphons that are called extended DMSs, and open a new path to analysis deadlocks and livelocks in view of siphons. The limits of the proposed MIP using max\textsuperscript{″} controlled siphons in Liu and Li (2010) are non-linear and more complicated, which increases the computational load. Similarly, no corresponding SC policy is presented in Liu and Li (2010).

Considering the shortcomings of the above SC policies, this paper proposes a novel MIP method to directly solve the smart siphons associated with livelocks or deadlocks in Petri net models. By means
of the structural analysis of Petri nets and the concept of smart siphons (Li & Li, 2012b), new con-
straints are generated and then added to the MIP method in Li and Li (2012b), called revised MIP
(RMIP). This RMIP can detect smart siphons that cause deadlocks or livelocks in Petri net models
directly. Accordingly, a proper control place (CP) (Li et al., 2011; Li & Li, 2012b) is added for each
solved smart siphon to make itself max-controlled (Chao, 2007; Zhong & Li, 2010) during an itera-
tive process. This research claims that if there is no feasible solution (NFS) of an RMIP problem, then
the corresponding Petri net system is live. That is to say, livelocks or deadlocks are eliminated in the
controlled Petri net system until no new smart siphons (SSSs) can be found by the proposed method
in this paper.

The rest of this work is organized as follows. Section 2 reviews preliminaries of Petri nets that are
used throughout this work. In Section 3, by adding new constraints to an MIP method in Li and Li
(2012b), we present an RMIP method to directly solve SSSs associated with livelocks or deadlocks.
By the proposed RMIP method and max-controlled SSSs, an iterative livelock control policy (LCP) is
developed in the same section. In Section 4, three examples are introduced to demonstrate the
proposed LCP. We have a discussion in Section 5. Finally, Section 6 concludes this paper and
discusses future work.

2. Preliminaries

2.1. Petri nets

A Petri net is a four-tuple \( N=(P, T, F, W) \), where \( P \) and \( T \) are finite and non-empty sets. \( P \) is a set of
places and \( T \) is a set of transitions with \( P \cup T \neq \emptyset \) and \( P \cap T = \emptyset \). \( F \subseteq (P \times T) \cup (T \times P) \) is called the flow
relation or the set of directed arcs. \( W : (P \times T) \cup (T \times P) \rightarrow \mathbb{N} \) is a mapping that assigns a weight to an
arc: \( W(x, y) > 0 \) if \( (x, y) \in F \), and \( W(x, y) = 0 \) otherwise, where \( x, y \in P \cup T \) and \( \mathbb{N} \) denotes the set of
non-negative integers. \( N=(P, T, F, W) \) is called an ordinary net, denoted as \( N=(P, T, F) \), if \( \forall f \in F, W(f)=1 \). \( N =
(P, T, F, W) \) is called a generalized net, if \( \exists f \in F, W(f) > 1 \). Given a node \( x \in P \cup T \), \( x = \{ y \in P \cup T \mid (y, x) \in F \} \) is
called the preset of \( x \), while \( x^+ = \{ y \in P \cup T \mid (x, y) \in F \} \) is called the postset of \( x \). A marking is a
mapping \( M : P \rightarrow \mathbb{N} \). \( N, M_0 \) is called a marked Petri net or a net system, where \( N \) is a net and \( M_0 \) is
an initial marking. The set of markings reachable from \( M \) in \( N \) is denoted as \( R(N, M) \). A string \( x_1, \ldots, x_n \) is
called a path of \( N \) if \( \forall i \in \{1, 2, \ldots, n \} \), \( x_i \in X_i \), where \( \forall X_i \in \{ x_1, x_2, \ldots, x_n \} , x \in P \cup T \). An elementary
diagram from \( x_1 \) to \( x_n \) is a path whose nodes are all different (except, perhaps, \( x_1 \) and \( x_n \)). A path \( x_1, \ldots, x_n \) is
called a circuit if it is an elementary path with \( x_1=x_n \). An ordinary Petri net \( N \) is called a state
machine if \( \forall t \in T, |t| = |t'| = 1 \). A state machine component \( N'=(P', T', F', W') \) of a Petri net \( N=(P, T, F,
W) \) is a state machine and is a subnet of \( N \) consisting of places in \( P' \), their presets and postsets, and
related arcs. A Petri net is said to be state machine decomposable if it is covered by state machine
components (Li & Zhou, 2009).

A \( P(T) \)-vector is a column vector \( I(J) : P(T) \rightarrow \mathbb{Z} \) indexed by \( P(T) \), where \( \mathbb{Z} \) is the set of integers. \( I \) is a
\( P \)-invariant (\( P \)-inv for short) if \( I 
\neq 0 \) and \( I[N] = 0 \). \( P \)-inv \( I \) is said to be a \( P \)-semiflow if every element of
\( I \) is non-negative. \(|I| = \{ p \in P \mid I(p) \neq 0 \} \) is called the support of \( I \). \(|I|^{+} = \{ p \mid I(p) > 0 \} \) denotes the
positive support of \( I \) and \(|I|^{-} = \{ p \mid I(p) < 0 \} \) denotes the negative support of \( I \). A non-empty set
\( S \subseteq P \) is a siphon (trap) if \( S \subseteq S' \) (\( S' \subseteq S \) ). A siphon is minimal if there is no siphon contained in it as a
proper subset. A minimal siphon is called a SMS if it does not contain a trap. \( \cap \) is used to denote the
set of SMSs in a net. Siphon \( S \) is inv-controlled by \( P \)-inv \( I \) under \( M_0 \) if \( I^T M_0 > 0 \) and \( \{ p \in P \mid I(p) > 0 \} \subset S \).

A transition \( t \in T \) is enabled at a marking \( M \) if \( \forall p \in \mathbb{P}(t), M(p) \geq W(p, t) \). This fact is denoted as \( M \ni t \). A transition \( t \in T \) is
live under \( M_0 \) if \( \forall M \in R(N, M_0), \exists M' \in R(N, M, M'[t]) \). \( N, M_0 \) is live if \( \forall t \in T, t \) is live under \( M_0 \), \( N, M_0 \) is deadlock-free (weakly live)
if \( \forall M \in R(N, M_0), \exists t \in T, M[t] \). A marking \( M \in R(N, M_0) \) is a livelock if \( \forall M' \in R(N, M_0), M' \) is neither a
deadlock nor a final marking, where \( M' \neq M_0 \) (Li & Zhou, 2009). \( N=(P, T, F, W) \) is said to be a subnet
of \( N=(P, T, F, W) \), which is generated by \( X = P \cup T \) if \( F = F \cap (X \times X), \forall f \in F, W'(f) = W(f) \), \( P \subseteq P \), and
T \subseteq T. 
2.2. Basic definitions and theorems

**Definition 2.1** (Li & Zhou, 2008) Let \( \{r_1, r_2, \ldots, r_m\} \subseteq P_R(\geq 2) \) be a set of resources in \( N \). An elementary circuit \( C(r_1, t_1, r_2, t_2, \ldots, r_m, t_m) \), denoted as \( C_p \), is called a resource circuit if the following conditions are satisfied: (1) \( \forall i \in N_m = \{1, 2, \ldots, m\}, r_i \in T_p \) (2) \( \forall i \in \{2, 3, \ldots, m\}, r_i \in T_p \) and (3) \( r_1 \in t_m \). Moreover, \( T_c \) is used to denote the set of transitions in \( C_p \).

**Definition 2.2** (Li & Zhou, 2009) A well-marked \( S'R \) (Systems of Sequential Systems with Shared Resources) net is a marked Petri net \( N = (P, T, F, W) \) with initial marking \( M_0 \) such that

\[
(1) \quad P = P_A \cup P^0 \cup P_R, \text{ where } P_A = \cup_{i \in N} P_{i_a} \text{ is called the set of operation places such that } P_A \cap P_A = \emptyset, \\
\forall i, j \in N_n, i \neq j, P^0 = \cup_{i \neq 1} \{p^0\} \text{ is called the set of idle places with } P^0 \cap P^0 = \emptyset, \text{ and } P_R = \{r_1, r_2, \ldots, r_m\} \text{ is called the set of resource places such that } (P^0 \cup P_A) \cap P_R = \emptyset.
\]

\[
(2) \quad T = \cup_{j \in N} T_j \text{ is called the set of transitions, where } \forall i, j \in N_n, i \neq j, T_j \neq \emptyset \text{ and } T_j \cap T_j = \emptyset.
\]

\[
(3) \quad W = W_A \cup W_R, \text{ where } W_A : (P_A \cup P^0) \times T \cup (T \times (P_A \cup P^0)) \rightarrow \{0, 1\} \text{ such that } \\
\forall i, j \in N_n, i \neq j, (P_A \cup \{p^0\}) \times T_j \cup (T_j \times (P_A \cup \{p^0\})) \rightarrow \{0\}, \text{ and } W_R : (P^0 \times T \cup (T \times P_R) \rightarrow \mathbb{N}.
\]

\[
(4) \quad \forall i \in N_n \text{ the subnet } N_j \text{ derived from } P_A \cup \{p^0\} \cup T_j \text{ is a strongly connected state machine such that } \\
\text{every circuit contains idle place } p^0_j.
\]

\[
(5) \quad \forall r \in P_R \text{ there exists a unique minimal } \gamma \text{-semiflow } I_r, \text{ such that } \\
||I_r|| \cap P_R = \{r\}, P^0 \cap ||I_r|| = \emptyset, P_A \cap ||I_r|| \neq \emptyset, \text{ and } \forall p \in ||I_r||, I_r(p) = 1. \text{ Furthermore, } \\
P_A = (\cup_{r \in P_R} ||I_r||) \cup \{P_A\}.
\]

\[
(6) \quad N \text{ is pure and strongly connected.}
\]

\[
(7) \quad \forall p \in P_A, M_0(p) = 0, \forall r \in P_R, M_0(r) \geq \max_{p \in ||I_r||} I_r(p); \text{ and } \forall p^0 \in P^0, M_0(p^0) \geq 1.
\]

**Definition 2.3** (Li & Zhou, 2009) A multiset \( \Omega \), over a non-empty set \( A \), is a mapping \( \Omega : A \rightarrow \mathbb{N} \) which we represent as a formal sum \( \sum_{a \in A} \Omega(a) \cdot a \). In multiset \( \Omega \), non-negative integer \( \Omega(a) \) is the coefficient of element \( a \in A \), indicating the number of occurrences of \( a \) in \( \Omega \). It is said that \( a \in A \) belongs to \( \Omega \), denoted by \( a \in \Omega \), if \( \Omega(a) > 0 \). It does not belong to \( \Omega \), denoted by \( a \not\in \Omega \), if \( \Omega(a) = 0 \).

**Definition 2.4** (Li & Zhou, 2009) \( \Omega_1 - \Omega_2 := \sum_{a \in A} (\Omega_1(a) - (\Omega_1 \cap \Omega_2)(a)) \cdot a \).

**Definition 2.5** (Li & Zhou, 2009) Let \( r \) be a resource place and \( S \) be an SMS in an \( S'R \) net. The operation places that use \( r \) is called the set of holders of \( r \), denoted as \( H(r) \). The holders of resource \( r \) is defined as the difference of two multisets \( I \) and \( r : H(r) = I - r \). As a multisets, \( Th(S) = \sum_{r \in S} H(r) - \sum_{p \in S} I(p) \cdot p \) is called the complementary set of siphon \( S \), where \( S = S_A \cup S_R \), \( S_R = S \cap P_R \) and \( S_A = S \cap P_A \). \( ||Th(S)|| = \{p | p \in \cup_{r \in S} H(r), p \not\in S\} \) is called the support of complementary set \( Th(S) \) of siphon \( S \).

Let \( \sum_{p \in ||Th(S)||} (h(p), p) \) denote \( Th(S) \). \( h(p) \) is called the risk coefficient of place \( p \). Clearly, \( h(p) \) indicates that siphon \( S \) loses \( h(p) \) tokens if the number of tokens in \( p \) increases by one.

**Definition 2.6** (Zhong & Li, 2010) Let \( S \) be a siphon of a well-marked \( S'R \) \( (N, M_0) \). \( S \) is said to be \( max' \)-marked at marking \( M \in (N, M_0) \) iff \( \exists p \in S_A \) such that \( M(p) \geq 1 \) or \( \exists p \in S_R \) such that \( M(p) \geq max_{r \in P_R \cap Th(S)} (W(p, r)) \).

**Definition 2.7** (Zhong & Li, 2010) Let \( S \) be a siphon of a well-marked \( S'R \) \( (N, M_0) \). \( S \) is said to be \( max' \)-controlled iff \( S \) is \( max' \)-marked at any reachable marking, that is, \( \forall M \in R(N, M_0), \exists p \in S_A \) such that \( M(p) \geq 1 \) or \( \exists p \in S_R \) such that \( M(p) \geq max_{r \in P_R \cap Th(S)} (W(p, r)) \).
Theorem 2.8 (Zhong & Li, 2010) Let \((N, M_0)\) be a well-marked \(S^4R\). If all SMSs of the net are max’-controlled, the net is live.

Note that an \(S^4R\) is more general than \(ES^{3PR}\), \(S^{3PR}\), and \(L-S^{3PR}\) (Li & Zhou, 2009), i.e. \(L-S^{3PR} \subseteq S^{3PR} \subseteq ES^{3PR} \subseteq S^4R\). Theorem 2.8 is hence suitable for verifying liveness of \(ES^{3PR}\), \(S^{3PR}\), and \(L-S^{3PR}\).

For example, an FMS layout is shown in Figure 1(a) (Chao, 2007). There are three loading buffers \(I_1–I_3\) and three unloading buffers \(O_1–O_3\) to load and unload the FMS corresponding to three raw product types, \(P_1–P_3\), to be processed by machine \(M_1\) and moved by robot \(R_1\).

According to the production cycles, a raw product \(P_1\) is taken from \(I_1\) by \(R_1\) and put in \(M_1\). After being processed by \(M_1\), it is output there as \(O_1\). A raw product \(P_2\) is taken from \(I_2\) and processed by \(M_1\) and then moved from \(M_1\) to \(O_2\) by \(R_1\). A raw product \(P_3\) is taken from \(I_3\), processed by \(M_1\) and output from \(M_1\) as \(O_3\). Figure 1(b) (Chao, 2007) shows its Petri net model \(S^4R (N, M_0)\), where \(P_0 = \{p_1, p_4, p_7\}\), \(P_A = \{p_2, p_3, p_5, p_6, p_8\}\), \(P_R = \{p_9, p_{10}\}\), \(|P| = |P_A| + |P_R| = 10\), and \(|T| = 8\).

By integrated net analyzer (INA, 2003), only an SMS, i.e. \(S = \{p_3, p_6, p_9, p_{10}\}\) can be found. From the above Definitions, \(Th(S) = \{p_2, p_5\}\), \(C_R = \{p_9, t_5, p_{10}, t_2\}\), and \(T_{c_i} = \{t_2, t_5\}\) can be obtained, respectively.

3. An iterative LCP using the methods of RMIP and max’-controlled NSSs

3.1. RMIP method

By analyzing the relationship between the states of deadlocks or livelocks and the presence of smart siphons, and then adding new constraints to an MIP method in Li and Li (2012b), the RMIP method is described below.

\[
G_{MIP}^{\text{RMIP}}(M) = \max \left( \sum_{p \in P} v_p + \sum_{p \in P_A} v_p \right)
\]

s. t.

\[
f_{pt} \geq \frac{M(p) - W(p, t) + 1}{SB(p)}, \quad \forall W(p, t) > 0
\]
\[ f_{pt} \geq v_p, \quad \forall W(p, t) > 0 \]  
\[ z_t \geq \sum_{p \in T} f_{pt} - |t| + 1, \quad \forall t \in T \]  
\[ v_p \geq z_t, \quad \forall W(t, p) > 0 \]  
\[ M = M_0 + |N|Y, M \geq 0, Y \geq 0 \]  
\[ \sum_{p \in P} v_p \leq |P| - 2 \]  
\[ v_p \geq M(p) \]  
\[ SB(p), \quad \forall p \in P \]  
\[ \nu_r \geq \sum_{t \in r \cap T_r} f_{rt} - |r \cap T_r| + 1, \quad \forall r \in P_R \]  
\[ v_p, z_t, f_{pt}, f_{rt} \in [0, 1], \quad \forall p \in P, \forall t \in T \]  

where \( SB(p) = \max \{|M = M_0 + |N|Y, M \geq 0, Y \geq 0\} \) is the structural bound of place \( p \), \( f_{rt} \) is the binary variable for arc \((r, t)\) that is enabled if \( M(r) \) is larger than \( W(r, t) \). In a live (resp. dead) subnet, all transitions are enabled (resp. disabled) whose \( z_t = 1 \) (resp. \( z_t = 0 \)). Note that if \( \nu_r = 1 \) for each input place of a transition \( t \) in an OPN, then \( z_t = 1 \). When \( \nu_r = 1 \) but \( p \) is unmarked, \( z_t \) keeps 1, \( t \) is said to be pseudo-enabled. For a GPN, arc \((p, t)\) from each input place of a transition \( t \) may not be enabled even though \( p \) is marked \( (\nu_p = 1) \). That is to say, it is need for \( p \) to be sufficiently marked and then \( \nu_r = 1 \) can lead to \( z_t = 1 \). In order to distinguish the solution of the MIP method in Li and Li (2012b), the solution of the above RMIP method is called NSS.

The RMIP method can be explained as follows: Equation 8 expresses the fact if the number of tokens at a place \( p \) is no less than the weight of arc \((p, t)\), then arc \((p, t)\) is enabled and \( v_p = 1 \). Physically, when a transition \( t \) is enabled and then fired, tokens can flow along arc \((t, p)\) and enter into each output place \( p \). This is consistent with \( f_{pt} \geq v_p \) (Equation 3) where \( f_{pt} = 1 \) if \( z_t = 1 \). In addition, if \( t \) is an output, rather than input, transition of a place \( p \), \( f_{pt} = 1 \) does not mean \( z_t = 1 \); \( f_{pt} = 0 \) means \( z_t = 0 \) since \( t \) is disabled if any input arc is disabled. Equation 9 expresses the fact if all output arcs of resource place \( r \) are enabled, then \( r \) must be max-\( \ast \)marked and \( v_r = 1 \) due to Definition 2.6. Otherwise \( r \) is not max-\( \ast \)marked and \( v_r = 0 \). The objective function and the other constraints Equations 2–7 and Equation 10 are the same as those in the MIP method in Li and Li (2012b). In summary, this RMIP method can find a set of non-max-\( \ast \)marked places that can form an NSS under a marking \( M \in R(N, M_0) \).

**Theorem 3.1** Let \((N, M_0)\) be a well-marked \( S' R \). If the proposed RMIP method applies to it, then its any feasible solution corresponds to an NSS.

**Proof** If the feasible solution \( G^{MIP}(M) \) of the proposed RMIP method exists, then we obtain an NSS under a certain reachable marking \( M \in R(N, M_0) \). By contradiction, suppose that this NSS is not minimal. This implies that the NSS contains a minimal siphon (denoted as \( S_{\text{min}} \)) as a proper subset. Hence, there exists a reachable dead marking \( M' \) such that \( v_p = 0 \) iff \( p \in S_{\text{min}} \). It is clear that there exists a new feasible solution \( G^{MIP}(M') \) in which \( G^{MIP}(M') > G^{MIP}(M) \), contradicting the fact that \( G^{MIP}(M) \) is maximal among all feasible solutions that meet constraints (Equations 2–10). As a result, Theorem 3.1 holds.

Comparing with the MIP method in Li and Li (2012b), main difference is that this RMIP method can detect a set of non-max-\( \ast \)marked but non-max-marked places that forms a siphon under a marking \( M \) there are both live and dead transitions. In other words, the former can only solve a smart siphon.
causing deadlocks, and the latter can solve an NSS associated with deadlocks and livelocks. In addition, NSF of the proposed RMIP method is obtained, which means no any NSS exists in an SIR. Furthermore, an important variable, denoted as $U_{z_i}$, with respect to the sum of the values of $z_i (i \in \mathbb{N}^+) = \{1, 2, \ldots, \}$) is introduced to identify the characteristic of the solved NSSs. After executing the above RMIP method, both $G_{\text{mp}}(M)$ and the value of each $z_i$ can be solved. Hence, $U_{z_i} = \sum_{i=1}^{n} z_i$ is obtained accordingly, where $|T|$ denotes the total number of transitions in an SIR. From Section 2.1, it is known that if an SIR is deadlocked under a marking $M$ in which all transitions are disabled, then $z_i = 0$, and $U_{z_i} = 0$ is obtained, implying that the characteristic of corresponding NSS is about deadlock. When an SIR is livelocked under a marking $M$ in which some transitions are disabled and the rest of transitions are fired, $z_i = 1$ and $U_{z_i} = |T|$ are true when an SIR is live, implying that no any NSS or NSS* exists in an SIR.

3.2. An iterative LCP combing the RMIP method with max’-controlled NSSs

After solving an NSS (resp. NSS*) with its complementary set (denoted as ThNSS (resp. ThNSS*) can be found by Definition 2.5. Depending on ThNSS (resp. ThNSS*), two types of CPs in Li and Li (2012b) and Li, An, Wang, et al. (2013) are properly selected and then added for the solved NSS (resp. NSS*). From the above discussion, we add CP for the solved NSS or NSS* to make it max’-not max-controlled. Thus, the key is to determine the initial marking of added CPs (denoted as $M_0(V)$). From Li and Zhou (2009), it is known that the value of $M_0(V)$ is closely related to the value of the control depth variable with respect to the solved NSS (denoted as $\xi_{\text{NSS}}$, i.e. $M_0(V) = M_0(\text{NSS}) - \xi_{\text{NSS}}$, where $\xi_{\text{NSS}} \in \mathbb{N}_m$.

Assume that $\text{NSS} = \{p_1, p_2, \ldots, p_k\}$ is a siphon of original net $(N, M_0)$, where $N_0 = (P_0, T_0, F_0)$. Add a CP (denoted as V) to make $P$-vector $I$ be a $P$-invariant of the augmented net $(N', M')$, where $V \in \text{NSS}$, $I(p) = 1, \forall p \in P_0, M'(p) = M_0(p)$ and $I(V) = -1$. Thus, $I'^{M_0} = I'M_0 = M_0(\text{NSS}) - M'(V) > 0$ holds. In order to make the solved NSS max’-controlled, $\xi_{\text{NSS}} \geq \max_{V \in \text{NSS}^* \cap \text{ThNSS}} \{|W(p, t)|\}$ is determined due to Definitions 2.6 and 2.7. Hence, the definition of $\xi_{\text{NSS}}$ is given as follows.

**Definition 3.2** Let $(N, M_j)$ be a well-marked SIR. $\xi_{\text{NSS}}$ is called the control depth variable with respect to an NSS if $\xi_{\text{NSS}} \in \mathbb{N}_m$ and $\xi_{\text{NSS}} \geq \max_{V \in \text{NSS}^* \cap \text{ThNSS}} \{|W(p, t)|\}$, where $V \in \text{NSS}^*$, $\text{NSS} = \text{NSS}_A \cup \text{NSS}_R$.

Similarly, $\xi_{\text{NSS}^*}$ is used to denote the control depth variable with respect to an NSS*, where $\xi_{\text{NSS}^*} \in \mathbb{N}_1$ and $\xi_{\text{NSS}^*} \geq \max_{V \in \text{NSS}^* \cap \text{ThNSS}^*} \{|W(p, t)|\}$. In general, decreasing $\xi_{\text{NSS}}$ (resp. NSS*) intends to relax the control of NSS (resp. NSS*), which may reduce the restriction of the behavioral permissiveness of the controlled system. Moreover, the max’ but not max-controlled condition for NSS (resp. NSS*) relaxes the controllability condition and may lead to a live controlled system with more behavioral permissiveness (Liu & Li, 2010; Zhong & Li, 2010).

An NSS or NSS* in a plant net model $(N, M_j)$ can be found by the aforementioned RMIP approach. However, a control policy needs to find all NSSs and NSS* and then adds the proper CPs for them in the augmented net $(N', M^*)$ such that the final controlled system is live. Hence, an iterative LCP using the proposed RMIP method, which can find a An NSS or NSS* and then add a proper CP for it at each iteration until a live controlled system is obtained, is developed in this subsection and described below, where $|P| = |P_A| + |P^D| + |P_R|$, $j \in \mathbb{N}_+$, $\Pi$, and $|V|$ denote the total number of the places in an original net, the iteration step number, the set of solved NSSs and NSS* in the augmented net $(N, M_j)$, and the number of added CPs in $(N, M_j)$, respectively.

The proposed LCP: An iterative LCP combing the RMIP method with max’-controlled NSSs (NSSs)

**Input:** a marked SIR $(N, M_0)$, where $N = (P_A \cup P_D \cup P_R, T, F)$

**Output:** a live controlled net system $(N', M^*)$

1. preset $\Pi := \emptyset$, $V := \emptyset$, and $j := 1$
2. solve an NSS* or NSS by the proposed RMIP method in Section 3.1
3: if NSF for \( G^{MIP}(M_1) \) exists then
4: execute NSS\( j \) or NSS\( j := \emptyset \), \( N^* := N \), and \( M^* := M_0 \)
5: end if
6: while any feasible solution for \( G^{MIP}(M_j) \) exists do
7: according to the adding CPs method stated in Li and Li (2012b), Li, An, Wang, et al. (2013), add proper \( V_j \) to NSS, or NSS\( j \) in the augmented net \( (N_j, M_j) \)
8: execute \( \Pi_j := \Pi \cup \{ \text{NSS}^*_j \} \cup \{ \text{NSS}_j \}, V := V \cup \{ V_j \}, N^* := N_j, M^* := M_j \) and \( j := j + 1 \)
9: solve the next \( \text{NSS}_j \) or NSS, by the proposed RMIP method in Section 3.1
10: end while
11: Output \( (N^*, M^*) \)

At the first iteration, the proposed LCP solves \( \text{NSS}^*_1 \) or NSS\( _1 \) and computes \( G^{MIP}(M_1) \). \( \text{NSS}^*_1 \) or NSS\( _1 := \emptyset \) if NSF for \( G^{MIP}(M_j) \) exists, implying that the original Petri net is live. Otherwise, the proposed ICP can obtain \( \text{NSS}^*_1 \) or NSS\( _1 \) and then add a proper \( V_j \) to it.

At the \( j \)th iteration, the proposed LCP also solves \( \text{NSS}^*_j \) or NSS\( _j \) and then computes \( G^{MIP}(M_j) \). Similarly, \( \text{NSS}^*_j \) or NSS\( _j := \emptyset \) if NSF for \( G^{MIP}(M_j) \) holds. Therefore, the corresponding controlled system is live and Policy 1 terminates at this iteration. Otherwise, the proposed LCP solves \( \text{NSS}^*_j \) or NSS\( _j \) and then adds a proper \( V_j \) for it. The iteration proceeds until NSF for \( G^{MIP}(M_j) \) holds, implying that the corresponding controlled system is live.

**Theorem 3.3** Let \( (N, M) \) be a well-marked S\( ^*R \) in which deadlocks and livelocks exist. The proposed LCP is applied to it, which leads to a live augmented net \( (N^*, M^*) \).

**Proof** Suppose that an NSS\( j \) is solved at the \( j \)th iteration and the augmented net after the \( j \)th iteration is denoted by \( (N_j, M_j) \), where \( \forall p \in P \cup P^0 \cup P_R, M_{\emptyset}(p) = M_{\emptyset}(p), M_{\emptyset}(V) = M_{\emptyset}(\text{NSS}) - \varepsilon_{\text{NSS}}, \) and \( \varepsilon_{\text{NSS}} \leq \max_{t \in \text{R} \cap h_{\text{NSS}}} \{ W(p, t) \} \). By Definition 2.5, Th( NSS\( _j \)) is then computed. Depending on \( \text{h}_{\text{NSS}}(p) \), a proper \( V_j \) is added for NSS\( _j \). As stated in Li and Li (2012b) and Li, An, Wang, et al. (2013), the addition of both ordinary and general CPs to the original net can make the solved NSS\( _j \) or NSS\( _j := \emptyset \) if NSF for \( G^{MIP}(M_j) \) holds, which means that no NSS\( _j \) and NSS\( _j := \emptyset \) can be found in the finally augmented net \( (N^*, M^*) \). As a result, the final controlled system \( (N^*, M^*) \) is live owing to Theorem 2.8.

From Chao (2006), Ezpeleta et al. (1995), Huang et al. (2001), Li et al. (2011), Li and Zhou (2004), Li and Zhou (2009), and Zhong and Li (2009), it is known that the number of SMS\( s \) in a Petri net grows quickly fast, in the worst case, grows exponentially with respect to its size. Moreover, the computational complexity, structural complexity, and behavior permissiveness are usually used to criticize the different DCPs. Since the proposed LCP can iteratively solve NSS\( ^* \) (NSS\( s \)) that are belonged to a class of SMS\( s \), this LCP is NP-hard in theory. Due to the fact that any siphon that has more resource places can be composed by those containing less ones (Chao, 2006; Li & Li, 2012b; Li & Zhou, 2008), the solved NSS\( ^* \) (NSS\( s \)) contain the minimal number of resource places. Hence, this LCP may reduce the possibility of adding redundant CPs to some degree. In summary, the proposed LCP has high computational efficiency and can generally obtain a live controlled Petri net system with a simple structure and more behavior permissiveness, which can be verified via two examples in Section 4.

Take an S\( ^*R \) shown in Figure 1(b) as an illustrative example, where \( |P| = |P^0| + |P_R| = 10 \) and \( |T| = 8 \). By Definition 2.2, \( C_P = \{ p_0, t_2, p_{10}, t_7 \} \) and \( T_C = \{ t_7, t_2 \} \) can be obtained, and INA (2003) indicates that this S\( ^*R \) has 29\( ^* \) reachable states, where \( ^* \) denotes maximally permissive behavior.
At the first iteration, $\text{NSS}^*_1 = \{p_5, p_9, p_8, p_9, p_{10}\}$ with $U_{s_1} = 2 < |T| = 8$ is solved by Policy 1. By Definition 2.5, $\text{Th}(\text{NSS}_1)^* = \{p_2, p_3\}$ with $h_{\text{NSS}_1}(p_2) = 4$ and $h_{\text{NSS}_1}(p_3) = 2$ are computed.

Due to $\zeta_{\text{NSS}_1}^* \geq \max_{1 \leq i, \tau \in \text{Th}(\text{NSS}_1)^*} W(p, \tau) = \max\{4, 1\} = 4, M_{\text{NSS}_1}(V_1) = 4 + 4 - 4 = 4$ can be determined. Depending on $h_{\text{NSS}_1}(p_2) = 4$ and $h_{\text{NSS}_1}(p_3) = 2$, a GCP $V_1$ with $V_1 = \{4t_2, 2t_5\}, V_1^* = \{4t_1, 2t_4\}$, and $M_{\text{NSS}_1}(V_1) = 4$ is then added.

At the second iteration, Policy 1 obtains NFS for $G_{\text{MIP}}(M_2)$, which means that no NSSs or NSS$^*$s can be found and Policy 1 terminates at this iteration. As a result, the finally controlled system is live. The results of Policy 1 are briefly shown in Table 1.

From Table 1, the proposed LCP solves one NSS$^*$ and accordingly adds a GCP for it. Policy 1 terminates in the second iteration until NFS for $G_{\text{MIP}}(M_2)$ is declared. Finally, the controlled system $(N^*, M^*)$ is live due to Theorems 2.2 and 3.2. In addition, by INA (2003), $(N^*, M^*)$ has 29 reachable states and $R = 100\%$, where $R$ denotes the ratio of the number of reachable states of the live controlled system $(N^*, M^*)$ to that of maximally permissive behavior of the original $S^R (N, M_0)$.

### 4 Examples

In this section, two examples are presented to demonstrate the proposed ICP in this work. Moreover, its control performance comparison with the existing approaches presented in Zhong and Li (2009) (denoted as ZL), Li et al. (2004) (denoted as LLW), and Li et al. (2006) (denoted as LWW) is also given, where the number of redundant added CPs is obtained by executing the method in Li et al. (2012) and the comparison is carried out on a 2.8 GHz Pentium computer with 512 MB of RAM under a Windows XP operating system.

Example 1 Figure 2(a) and (b) shows an FMS layout and its Petri net model $S^R (N, M_0)$ taken from Zhong and Li (2009), respectively. There are two uploading buffers I1 and I2 and three downloading buffers O1 and O2 to upload and download the FMS corresponding to two raw product types, P1 and P2, to be processed by machine M1 and moved by robot R1 (resp. R2) and R2 (resp. R1).

According to two production cycles, a raw product P1 is taken from I1 by R2 and put in M1. After being processed by M1, it is downloaded by R1 and output to O1. A raw product P2 is taken from I2 by R1 and put in M1 to be processed, and then moved from M1 to O2 by R2. For Figure 2(b), we have

| Figure 2. (a) An FMS layout; and (b) The corresponding Petri net model $S^R (N, M_0)$. |
|---|
| ![Figure 2](image) |

| Table 1. Results of the proposed LCP for a marked $S^R$ shown in Figure 1(b) |
|---|---|---|---|---|---|---|
| $j$ | $\text{NSS}/\text{NSS}^*$ | $\text{Th}(\text{NSS})/\text{Th}(\text{NSS})^*$ | $U_{s,j}$ | $V_j$ | $V_j^*$ | $M_{\text{NSS}}(V)$ |
|---|---|---|---|---|---|---|
| 1 | $\{p_5, p_9, p_8, p_9, p_{10}\}$ | $\{p_2, p_3\}$ | 2 | $4t_2, 2t_5$ | $4t_1, 2t_4$ | 5 | 10 |
| 2 | - | - | 8 | NFS | 11 |

At the second iteration, Policy 1 obtains NFS for $G_{\text{MIP}}(M_2)$, which means that no NSSs or NSS$^*$s can be found and Policy 1 terminates at this iteration. As a result, the finally controlled system is live. The results of Policy 1 are briefly shown in Table 1.
\[ P^0 = \{p_1, p_5\}, P_A = \{p_2, p_3, p_4, p_6, p_7\}, P_B = \{p_9, p_{10}, p_{11}\}, |P| = |P^0| + |P_A| + |P_B| = 11, \text{ and } |T| = 8. \]

By Definition 2.1, \( C_0 = \{p_9, t_7, p_{10}, t_2\}, T_{C_0} = \{t_7, t_2\}, C_R = \{p_{10}, t_6, p_{11}, t_3\}, \text{ and } T_{C_R} = \{t_3, t_6\} \) can be obtained. The number of its maximally permissive states is 194\( \times \) by executing INA (2003). Tables 2 and 3 show the results of the proposed LCP and its control performance comparison with the existing approaches, respectively.

**Example 2** Considering another FMS layout and its Petri net model \( S' R (N, M_0) \) depicted in Figure 3(a) and (b) (Zhong & Li, 2009), respectively. This FMS consists of four robots R1–R4, each of which can hold one or three products every time, and three machines M1–M3, each of which can process two or three products every time. There are three loading buffers I1–I3 and three unloading buffers O1–O3 to load and unload the FMS. There are three raw product types, namely P1, P2, and P3, to be processed.

According to three production cycles, a raw product J1 is taken from I1 by R1 and R2 and put in M1. After being processed by M1 it is then moved to M3 by R4. Finally, after being processed by M3, it is processed by M2 and R3 and then moved to O1. A raw product J2 is taken from 12 and processed by M2 and R4, and then processed by M2 only. After being processed by M2 it is then moved from M2 to O2 by R4. A raw product J3 is taken from I3, processed by M3 and R3, and then by M3 and R4. After that, it is then processed by M1. Finally, after being processed by M1, it is then moved from M1 to O3 by R1 and R2 sequentially. For Figure 2(b), we have \( P^0 = \{p_8, p_{12}, p_{20}\} \).

\[ P_A = \{p_1, p_3, p_4, p_7, p_9, p_{10}, p_{11}, p_{13}, p_{14}, p_{15}, p_{16}, p_{17}\}, P_B = \{p_{18}, p_{19}, p_{21}, p_{22}, p_{23}, p_{24}, p_{25}\}, |P| = |P^0| + |P_A| + |P_B| = 25, \text{ and } |T| = 19. \]

By Definition 2.1, \( C_0 = \{p_{21}, t_{16}, p_{22}, t_9, p_{24}, t_8\}, T_{C_0} = \{t_8, t_9, t_{16}\}, C_R = \{p_{22}, t_{16}, p_{23}, t_5\}, T_{C_R} = \{t_5, t_{14}\}, C_R = \{p_{24}, t_{15}, p_{23}, t_5, p_{22}, t_9\}, T_{C_R} = \{t_5, t_9, t_{15}\} \).

From Tables 3 and 5, the number of necessary (redundant) added CPs obtained by this LCP is fewer (equal to zero), leading to the live controlled \( S' R \) with simple structures. Moreover, this LCP achieves 97.4% compared to ZL 97.4%.

---

**Table 2. Results of the proposed LCP for a non-live marked \( S' R \) shown in Figure 2(b)**

| \( j \) | \( NSS/NSS^* \) | \( Th(NSS)/Th(NSS)^* \) | \( U_{kl} \) | \( V_i \) | \( V_j \) | \( M_0(V) \) | \( G^{MAX}(M) \) | \( |P| + |V| \) |
|---|---|---|---|---|---|---|---|---|
| 1 | \( p_1, p_3, p_{10}, p_5 \) | \( p_2, 3 p_1, p_1 \) | 4 | \( 2 t_0, t_0 \) | \( t_0, t_0, t_0 \) | 5 | 10 | 12 |
| 2 | \( p_5, p_9, p_1, p_2 \) | \( 2 p_5, p_5 \) | 4 | \( 2 t_0, t_0 \) | \( t_0, t_0, t_0 \) | 5 | 10 | 12 |
| 3 | - | - | 4 | \( 2 t_0, t_0 \) | \( t_0, t_0, t_0 \) | 5 | 10 | 12 |

**Table 3. Control performance comparison**

| Control criteria | The proposed LCP | ZL |
|---|---|---|
| No. of added CPs | 2 | 3 |
| No. of reachable states | 189 | 189 |
| \( R \) | 97.4% | 97.4% |
| Executing time | \( \leq 1s \) | \( \leq 1s \) |
| Memory occupancy | 1.24 M | 1.28 M |
| No. of redundant added CPs | 0 | 1 |
more permissive states compared with the existing methods. That is to say, the proposed LCP can result in a liveness-enforcing Petri net supervisor with a simple structure and more permissive behavior in terms of structural complexity and behavior permissiveness.

5. Discussion

Different types of multiple resources can be requested by different processes in an FMS. As a typical class of generalized Petri nets, an S^4R can model more complicated resource allocation systems with multiple concurrent processes. Thus, an S^4R has better modeling power than ES^3PR, S^3PR, and L-S^3PR (Li & Zhou, 2009). An insufficiently marked siphon in an S^4R can lead to a dead state. Many DCPs (Huang et al., 2006; Li & Li, 2012b; Li et al., 2011, 2006; Piroddi et al., 2009; Zhong & Li, 2009) for S^4R are developed based on the concept of max-controlled siphons. However, these policies cannot solve livelock problems that may lead to undesirable results for in running process of an FMS.

Table 4. Results of the proposed LCP for another non-live marked S^4R shown in Figure 3(b)

| j | NSS_j/NSS^* | Th(NSS_j)/Th(NSS)^* | U_{i,j} | V_j | V_j^-1 | M_j(V_j) | G_H(M_j) | |P| + |V|
|---|---|---|---|---|---|---|---|---|
| 1 | (p_1, p_6, p_16, p_18, p_21)^* | (p_1, p_6)^* | 13 | t_2, t_7, t_17 | t_1, t_16 | 2 | 25 | 25 |
| 2 | (p_1, p_6, p_16, p_18, p_21)^* | (p_1, p_6, 2p_6)^* | 9 | t_2, t_7, t_17 | t_1, t_16 | 2 | 26 | 26 |
| 3 | (p_1, p_6, p_16, p_18, p_21, p_3, p_20)^* | (p_1, p_6, p_16, 2p_6, 3p_6)^* | 11 | t_2, t_7, 2t_17 | t_1, 2t_16 | 4 | 23 | 27 |
| 4 | | | 19 | t_2, t_7, 3t_17 | t_1, 2t_16 | 6 | NFS | 28 |

Table 5. Control performance comparison

| Control criteria | The proposed LCP | ZL | LLW^* | LWW |
|---|---|---|---|---|
| No. of added CPs | 3 | 3 | 6 | 2 |
| No. of reachable states | 7622 | 4116 | 2570 | 2654 |
| R | 89.1% | 48.1% | 30% | 31% |
| Executing time | ≤3 s | ≤3 s | ≤2 s | ≤2 s |
| Memory occupancy | 2.74 M | 2.38 M | 2.14 M | 1.97 M |
| No. of redundant added CPs | 0 | 1 | 4 | 0 |
As far as the authors know, only limited work of (Chao, 2007; Liu & Li, 2010; Zhong & Li, 2010) is reported on the siphon controllability conditions causing livelock states, i.e. max′ or max″-controlled siphons. But no corresponding SC policies are given and the approach in Liu and Li (2010) is non-linear and more complicated. By means of the study of Chao (2007), Zhong and Li (2010) and the structural analysis of Petri nets, the presence of a non-max-marked siphon is not a sufficient condition for an S4R to be not live. Actually, a set of non-max-marked places under a marking \( M \in \mathbb{R}^N \) does form a problematic siphon to make an S4R be not live. Thus, we add new constrains to the MIP method in Li and Li (2012b) to create a RMIP. The proposed RMIP method in Section 3.1 can solve NSSs in an S4R directly and then identify the solved ones causing deadlocks or livelocks depending on the value of \( U^*_z \). Compared with the method in Liu and Li (2010), this RMIP method is relatively simple and its computation is lower. Moreover, an iterative LCP using the RMIP method and max″-controlled siphon controllability is designed in Section 3.2. Section 4 shows two examples to illustrate that the proposed RMIP method and corresponding LCP are feasible in theory and have potential for development of the theory to cope with both livelocks and deadlocks than the prior theory based on siphon controllability.

6. Conclusions
On the basis of the structural analysis of Petri nets and the concept of a set of non-max″-marked places, this research presents an RMIP approach to directly solve NSSs associated with deadlocks or livelocks in an S4R. Moreover, an LCP adopting the RMIP method and max″-controlled siphon controllability is designed to eliminate deadlocks and livelocks in a non-live S4R. The results in Section 4 show that the proposed LCP has a certain advantage over the existing policies in terms of behavioral permissiveness and structural complexity of the final live controlled system. The extension of the proposed RMIP and the corresponding LCP to more general classes of Petri nets is the focus of our future research.

Acknowledgments
The authors would like to thank the editor and anonymous reviewers whose comments and suggestions greatly helped us improve the quality and presentation of this paper.

Funding
This work was supported by the National Nature Science Foundation of China under [grant number 61364004], the Natural Technology and Development Program of the 12th Five-Year Plan of China under [grant number 2011BA03B08], the Doctoral Research Funds of Lanzhou University of Technology under [grant number 04-237], and Alumni Foundation of Civil Engineering 77, Lanzhou University of Technology under [grant number TMQK-1301].

Author details
C.Q. Hou1
E-mail: AHL8XCM@lut.cn
S.Y. Li1
E-mail: lishaoyong99@163.com
Y. Cai1
E-mail: blue9801@163.com
H.M. Wu2
E-mail: jhjys@lut.cn
A.M. An3
E-mail: anaiminli@163.com
Y. Wang4
E-mail: Wangying@lut.cn
1 School of Civil Engineering, Lanzhou University of Technology, Lanzhou 730050, People’s Republic of China.
2 School of Science, Lanzhou University of Technology, Lanzhou 730050, People’s Republic of China.
3 College of Electrical and Information Engineering, Lanzhou University of Technology, Lanzhou 730050, People’s Republic of China.
4 E-mail: Wangying@lut.cn

References
Chao, D. Y. (2006). Computation of elementary siphons for deadlock control. The Computer Journal, 49, 470–479. http://dx.doi.org/10.1093/comjnl/bxl019
Chao, D. Y. (2007). Max′-controlled siphons for liveness of S′∗PG′R. IET Control Theory and Applications, 1, 933–936. http://dx.doi.org/10.1049/iet-cta:20060275
Chu, F., & Xie, X. L. (1997). Deadlock analysis of Petri nets using siphons and mathematical programming. IEEE Transactions on Robotics and Automation, Part A, 13, 793–804.
Ezepeleta, J., Colom, J. M., & Martínez, J. (1995). A Petri net based deadlock prevention policy for flexible manufacturing systems. IEEE Transactions on Robotics and Automation, 11, 173–184. http://dx.doi.org/10.1109/70.370500
Huang, Y. S., Jeng, M. D., Xie, X. L., & Chung, D. H. (2006). Siphon-based deadlock prevention policy for flexible manufacturing systems. IEEE Transactions on Systems, Man, and Cybernetics, Part A, 36, 1248–1256. http://dx.doi.org/10.1109/TSMCA.2006.878953
Huang, Y. S., Jeng, M. D., Xie, X. L., & Chung, S. L. (2001). Deadlock prevention policy based on Petri nets and siphons. International Journal of Production Research, 39, 283–305. http://dx.doi.org/10.1080/00207540010002405
Integrated Net Analyzer. (2003). A software tool for analysis of Petri nets (Version 2.2) [online]. Retrieved July 31, 2003, from http://www.informatik.hu-berlin.de/starke/ina.html
Li, S. Y., & Li, Z. W. (2012a). Structure reduction of liveness-enforcing Petri nets using mixed integer programming. Asian Journal of Control, 14, 384–399. http://dx.doi.org/10.1002/acs.342

Li, S. Y., & Li, Z. W. (2012b). Solving siphonility with the minimal cardinality in Petri nets and its applications to deadlock control. International Journal of Production Research, 50, 6203–6218. http://dx.doi.org/10.1080/00207543.2011.590540

Li, S. Y., Li, Z. W., Al-Ahmari, A., & An, A. M. (2012). An extraction algorithm for a set of elementary siphons based on mixed integer programming. Journal of Systems Science and Systems Engineering, 21, 106–125. http://dx.doi.org/10.1007/s11518-012-5188-z

Li, Y. S., An, A. M., Cai, Y., Hong, C. Q., Han, X. L., & Wang, Y. (2013). A deadlock control policy for a subclass of Petri nets G-system. Control Theory & Applications, 30, 1429–1436. http://dx.doi.org/10.1080/0951192X.2011.575182

Li, Z. W., & Wang, A. R. (2011). Design of liveness-enforcing supervisors with necessity siphons for deadlock prevention in generalized Petri nets. IEEE Transactions on Systems, Man, and Cybernetics, Part A, 34, 38–51. http://dx.doi.org/10.1109/TSMA.2003.820576

Li, Z. W., & Zhou, M. C. (2006a). Two-stage method for synthesizing liveness-enforcing supervisors for flexible manufacturing systems using Petri nets. IEEE Transactions on Industrial Informatics, 2, 313–325. http://dx.doi.org/10.1109/TII.2006.885185

Li, Z. W., & Zhou, M. C. (2006b). Clarifications on the definition of elementary siphons in Petri nets. IEEE Transactions on Systems, Man, and Cybernetics, Part A, 36, 1227–1229. http://dx.doi.org/10.1109/09.2006.878956

Li, Z. W., & Zhou, M. C. (2008). On siphon computation for deadlock control in a class of Petri nets. IEEE Transactions on Systems, Man, and Cybernetics, Part A, 38, 667–679. http://dx.doi.org/10.1109/09.2008.968413

Li, Z. W., & Zhou, M. C. (Eds.). (2009). Proceedings of 8th International Conference on Networking, Sensing and Control. New York, NY: CRC Press.

Park, J., & Reveliotis, S. A. (2001). Deadlock avoidance in flexible manufacturing systems. International Journal of Production Research, 39, 1987–2030. http://dx.doi.org/10.1080/09511920110086661

Uzam, M., & Zhou, M. C. (2006). Clarifications on the definitions of elementary siphons in Petri nets. IEEE Transactions on Systems, Man, and Cybernetics, Part A, 36, 1227–1229. http://dx.doi.org/10.1109/TSMCA.2009.2013189

Zhong, C. F., & Li, Z. W. (2009). A deadlock prevention approach for flexible manufacturing systems using necessary siphons. IEEE Transactions on Systems, Man, and Cybernetics, Part A, 39, 650–661. http://dx.doi.org/10.1109/TSMCA.2009.2013189

Uzam, M., & Zhou, M. C. (2006). An improved iterative synthesis method for liveness enforcing supervisors of flexible manufacturing systems. International Journal of Production Research, 44, 1987–2030. http://dx.doi.org/10.1080/00207540500431321